Visio Automation Reference

Object Reference

<Global>
AccelItem
AccelItems
AccelTable
AccelTables
Addon
Addons
Application
Attribute
Attributes
Cell
Characters
Color
Colors
Connect
Connects
Curve
Document
Documents
Entities
Entity
EntityApp
EntityApps
Event
EventList
Font
Fonts
Hyperlink
Layer
Layers
Master
Masters
Menu
MenuItem
MenuItems
Menus
MenuSet
MenuSets
OLEObject
OLEObjects
Page
Pages
Path
Paths
Selection
Shape
ShapeData
Shapes
StatusBar
StatusBarItem
StatusBarItems
StatusBars
Style
Styles
Toolbar
ToolbarItem
ToolbarItems
Toolbars
ToolbarSet
ToolbarSets
UI Object
Window
Windows

List of Properties
List of Methods
List of Events
List of Shapesheet Cells
Syntax Conventions
ThisDocument Object
Visio Type Library

<Global> Object

<Global> object

The Visio global object is automatically available to VBA code that is part of the VBA project of a Visio document. The Visio global object is not available to code in other contexts.

Members of the global object can be accessed without qualification. For example, to access the ActivePage member of the global object:

```
Set pageObj = ActivePage
```

The above syntax is different than the syntax you would use for accessing members of non-global objects. For example:

```
Set pageObj = AppObj.ActivePage
```

Related to the Global object is the ThisDocument object. The VBA project of every Visio document has a class module called ThisDocument. When referenced from code in the VBA project, ThisDocument returns a reference to the project's Document object.

Version added: VISIO 4.5

Properties

ActiveDocument
ActivePage
ActiveWindow
Addons
Application
Documents
VBE
Windows

AccelItem Object

AccelItem object

An AccelItem object represents a single accelerator used by Visio. An AccelItem consists of a key, modifiers to the key, and the Visio command that the accelerator will execute when it is pressed by the user. A key is any ASCII key code, and is not case-sensitive. The modifiers are Alt, Control, and Shift. Command identifiers are declared by the Visio type library (and visconst.bas). They are prefixed with "visCmd."

Version added: VISIO 4.0

Properties
Alt
CmdNum
Control
Key
Parent
Shift

Methods
Delete

AccelItems Object

AccelItems object

The AccelItems collection includes an AccelItem object for each accelerator in a Visio window context. Unlike other Visio collections, the AccelItems collection is indexed starting with 0 rather than 1.

Use the AccelItems property of an AccelTable object to retrieve its AccelItems collection. The default property of AccelItems is Item.

Version added: VISIO 4.0

Properties
Count
Item
Parent

Methods
Add

AccelTable Object

AccelTable object

An AccelTable object represents a Windows accelerator table. There can be one AccelTable object for each Visio window context (drawing window, stencil window, ShapeSheet window, and so forth).

Version added: VISIO 4.0

Properties
AccelItems
Parent
SetID
TableName

Methods
Delete

AccelTables Object

AccelTables object

The AccelTables collection includes an AccelTable object for each Visio window context that has accelerators. Unlike other Visio collections, the AccelTables collection is indexed starting with 0 rather than 1.

Use the AccelTables property of a UI object to retrieve its AccelTables collection. The default property of AccelTables is Item.

An AccelTable object is identified in the AccelTables collection by its SetID, which corresponds to a Visio window context. The following are valid SetIDs for AccelTable objects:

visUIObjSetNoDocument
visUIObjSetDrawing
visUIObjSetStencil
visUIObjSetShapeSheet
visUIObjSetIcon
visUIObjSetInPlace
visUIObjSetPrintPreview
visUIObjSetBinderInPlace
visUIObjSetHostingInPlace

Version added: VISIO 4.0

Properties
Count
Item
ItemAtID
Parent

Methods
Add
AddAtID

Addon Object

Addon object

An Addon object represents an installed Visio add-on. A Visio add-on is a program that can be launched from an instance of Visio and that typically interacts with the instance through Automation. An add-on can be implemented by an executable (.EXE) file. A Visio Library (.VSL) file can implement several add-ons.

Use the Addons collection of an Application object to retrieve an Addon object. The default property of Addon is Name.

Version added: VISIO 4.0

Properties
Application
Enabled
Index
Name
ObjectType

Methods
Run

Addons Object

Addons object

An Addons collection represents the set of installed add-ons known to an Application object. Installed add-ons are those Visio finds in its Addons or StartUp paths, or those that other add-ons have dynamically installed using the Add method of the Addons collection.

Use the Addons property of an Application object to retrieve its Addons collection. The default property of Addons is Item.

Version added: VISIO 4.0

Properties
Application
Count
Item
ObjectType

Methods
Add
GetNames

Application Object

Application object

An Application object represents an instance of Visio. An external program must typically create or retrieve an Application object before it can retrieve other Visio objects from that instance. Use the Visual Basic CreateObject function to run a new instance, or use the GetObject function to retrieve an instance that is already running.

Code in the VBA project of a Visio document can use the Visio global object instead of a Visio Application object to retrieve other objects.

Use the Documents, Windows, and Addons properties of an Application object to retrieve the Document, Window, and Addon collections of the instance.

Use the ActiveDocument, ActivePage, or ActiveWindow properties to retrieve the currently active Document, Page, or Window object. The Application object's menus and toolbars can be accessed using the BuiltInMenus, BuiltInToolbars, CustomMenus, or CustomToolbars properties. ActiveDocument is the default property of an Application object.

Version added: VISIO 4.1

Properties

Active
ActiveDocument
ActivePage
ActiveWindow
AddonPaths
Addons
AlertResponse
Application
BuiltInMenus
BuiltInToolbars
CustomMenus
CustomMenusFile
CustomToolbars
CustomToolbarsFile
DeferRecalc
Documents
DrawingPaths
EventInfo
EventList
EventsEnabled
FilterPaths
HelpPaths
InstanceHandle
InstanceHandle32
IsVisio16
IsVisio32
Language
ObjectType
OnDataChangeDelay
Path
PersistsEvents
ProcessID
ProfileName

PromptForSummary
ScreenUpdating
ShowMenus
ShowProgress
ShowStatusBar
ShowToolbar
StartupPaths
Stat
StencilPaths
TemplatePaths
ToolbarStyle
TraceFlags
UserName
VBE
Version
WindowHandle
WindowHandle32
Windows

Methods
ClearCustomMenus
ClearCustomToolbars
ConvertResult
DoCmd
EnumDirectories
FormatResult
PurgeUndo
QueueMarkerEvent
Quit
Redo
SaveWorkspaceAs
SetCustomMenus
SetCustomToolbars
Undo

Events
AfterModal
AppActivated
AppDeactivated
AppObjectActivated
AppObjectDeactivated
BeforeDocumentClose
BeforeDocumentSave
BeforeDocumentSaveAs
BeforeMasterDelete
BeforeModal
BeforePageDelete
BeforeQuit
BeforeSelectionDelete
BeforeShapeDelete
BeforeStyleDelete
BeforeWindowClose
BeforeWindowPageTurn
BeforeWindowSelDelete
CellChanged
ConnectionsAdded

ConnectionsDeleted
DesignModeEntered
DocumentAdded
DocumentChanged
DocumentCreated
DocumentOpened
DocumentSaved
DocumentSavedAs
FormulaChanged
MarkerEvent
MasterAdded
MasterChanged
PageAdded
PageChanged
RunModeEntered
SelectionAdded
SelectionChanged
ShapeAdded
ShapeChanged
StyleAdded
StyleChanged
TextChanged
WindowActivated
WindowOpened
WindowTurnedToPage

Attribute Object

Attribute object

An Attribute object represents a single attribute of a given shape. Attributes are unique by their Name property for a particular shape. Each Attribute object has a value string associated with it.

Versions 4.0 and beyond of Visio store a shape's attributes in its Custom Properties section, rather than using Attribute and Attributes objects. The Attribute and Attributes objects are retained for compatibility with applications developed for earlier versions of Visio.

[Beginning with version 4.5, support for this object is included only with Visio Technical. Other versions of Visio, such as Visio Professional, do not include support for this object.]

Version added: VISIO 3.0 TECH

Properties
DefaultValue
Name
Prompt
Value

Methods
Delete

Attributes Object

Attributes object

The Attributes collection includes an Attribute object for each attribute of a shape. Use the Attributes property of a ShapeData object to retrieve its Attributes collection. The default property of Attributes is Item.

Versions 4.0 and beyond of Visio store a shape's attributes in its Custom Properties section, rather than using Attribute and Attributes objects. The Attribute and Attributes objects are retained for compatibility with applications developed for earlier versions of Visio.

[Beginning with version 4.5, support for this object is included only with Visio Technical. Other versions of Visio, such as Visio Professional, do not include support for this object.]

Version added:VISIO 3.0 TECH

Properties
Count
Item

Methods
Add

Cell Object

Cell object

A Cell object has a formula that evaluates to some value. You can get or set a cell's formula, or you can get or set its value. A cell belongs to a Shape or Style object and represents a property of the shape or style. For example, the height of a shape equals the value of the shape's height cell.

A program controls much of a shape's appearance and behavior by working with the formulas of the shape's cells. You can visually inspect most of a shape's cells by opening a ShapeSheet window showing that shape. Use the Cells or CellsSRC property of a Shape object to retrieve a Cell object. To retrieve a cell in a style, use the Cells property of a Style object. The default property of a Cell object is ResultIU.

Version added: VISIO 2.0

Properties
Application
Column
Document
Error
EventList
Formula
FormulaForce
IsConstant
IsInherited
LocalName
Name
ObjectType
PersistsEvents
Result
ResultForce
ResultFromInt
ResultFromIntForce
ResultInt
ResultIU
ResultIUForce
ResultStr
Row
RowName
Section
Shape
Stat
Style
Units

Methods
GlueTo
GlueToPos
Trigger

Events
CellChanged
FormulaChanged

Characters Object

Characters object

A Characters object represents a shape's text with text fields expanded to the number of characters they display in a drawing window. You retrieve a Characters object by getting the Characters property of a Shape object. The Begin and End properties of a Characters object determine the range of the shape's text that is represented by the Characters object. Initially, the range contains all of the shape's text; you can set Begin and End to specify a subrange of the text.

After you retrieve a Characters object, you can use its Text property to retrieve or set the shape's text. Use the Copy, Cut, and Paste methods to copy, cut, or paste the Character object's text to or from the Clipboard, or use the CharProps or ParaProps property to change its formatting. The default property of a Characters object is Text.

Version added:VISIO 3.0

Properties
Application
Begin
CharCount
CharProps
CharPropsRow
Document
End
EventList
FieldCategory
FieldCode
FieldFormat
FieldFormula
IsField
ObjectType
ParaProps
ParaPropsRow
PersistsEvents
RunBegin
RunEnd
Shape
Stat
TabPropsRow
Text
TextAsString

Methods
AddCustomField
AddField
Copy
Cut
Paste

Events
TextChanged

Color Object

Color object

A Color object represents a color in the color palette for a Visio document. The default property of a Color object is PaletteEntry.

Version added: VISIO 4.0

Properties

Application
Blue
Document
Flags
Green
Index
ObjectType
PaletteEntry
Red
Stat

Colors Object

Colors object

A Colors collection includes a Color object for each color in the palette for a Visio document. Use the Colors property of a Document object to retrieve its Colors collection. The default property of Colors is Item.

Version added: VISIO 4.0

Properties
Application
Count
Document
Item
ObjectType
Stat

Connect Object

Connect object

A Connect object represents a connection between two shapes in a drawing, such as a line and a box in an organization chart. You retrieve a particular Connect object from the Connects or FromConnects collection of a Shape object, or the Connects collection of a Page or Master object. Use the GlueTo or GlueToPos method of a Cell object to connect one shape to another in a drawing. The default property of a Connect object is FromSheet.

Version added: VISIO 2.0

Properties
Application
Document
FromCell
FromPart
FromSheet
Index
ObjectType
Stat
ToCell
ToPart
ToSheet

Connects Object

Connects object

A Connects collection is a collection of Connect objects. A Connect object represents a connection between two shapes in a drawing, such as a line and a box in an organization chart.

Use the Connects property of a Shape object to retrieve a Connects collection with a Connect object for every Shape to which the indicated Shape is connected (glued) .

Use the FromConnects property of a Shape object to retrieve a Connects collection with a Connect object for every Shape that is connected (glued) to the indicated Shape.

Use the Connects property of a Page object to retrieve a Connects collection with an entry for every connection on the Page.

Use the Connects property of a Master object to retrieve a Connects collection with an entry for every connection in the Master.

The default property of a Connects collection is Item.

Version added:VISIO 2.0

Properties
Application
Count
Document
FromSheet
Item
ObjectType
Stat
ToSheet

Curve Object

Curve object

A Curve object is an item in a Path object representing a consecutive sequence of rows in the Geometry section of its Path object.

If the Curve is in a collection from Shape.Paths, its coordinates are expressed in the shape's parent coordinate system. If the Curve is in a collection from Shapes.PathsLocal, its coordinates are expressed in the shape's local coordinate system. In both cases, coordinates are expressed in internal drawing units (inches).

A Curve describes itself in terms of its parameter domain, which is the range [Start(),End()]. Use the Start property of a Curve object to obtain the curve's starting point and the End property of a Curve object to obtain the curve's ending point.

Use the Point method of a curve object to extrapolate a point along the curve's path. Use the PointAndDerivatives method of a Curve to determine a point along the curve's path and, optionally, its first and second derivatives.

Use the Points property of a Curve to obtain a stream of points that approximate the curve's path.

The default property of Curve is Point.

Version added: VISIO 5.0

Properties
Application
Closed
End
ObjectType
Points
Start
Stat

Methods
Point
PointAndDerivatives

Document Object

Document object

A Document object represents a drawing file (.VSD), stencil file (.VSS), or template file (.VST) that is open in an instance of Visio. A Document object is a member of the Documents collection of an Application object. Use the Open method of a Documents collection to open an existing document. Use the Add method of a Documents collection to create a new document. Use the ActiveDocument property of an Application object to retrieve the active document in an instance. Use the Pages, Masters, and Styles properties of a Document object to retrieve Page objects, Master objects, and Style objects, respectively. The default property of a Document object is Name.

The VBA project of every Visio document possesses a class module called ThisDocument. When referenced from code in the project, ThisDocument returns a reference to the project's Document object. For example, the code in a document's project can display the name of the project's document in a message box with this statement:

```
MsgBox ThisDocument.Name
```

Version added: VISIO 2.0

Properties

Application
BottomMargin
Category
Colors
Company
Creator
CustomMenus
CustomMenusFile
CustomToolbars
CustomToolbarsFile
DefaultFillStyle
DefaultLineStyle
DefaultStyle
DefaultTextStyle
Description
EventList
Fonts
FullName
HyperlinkBase
Index
InPlace
Keywords
LeftMargin
Manager
Masters
Mode
Name
ObjectType
OLEObjects
Pages
PaperHeight
PaperSize
PaperWidth
Path

PersistsEvents
PrintCenteredH
PrintCenteredV
PrintFitOnPages
PrintLandscape
PrintPagesAcross
PrintPagesDown
PrintScale
ReadOnly
RightMargin
Saved
SavePreviewMode
Stat
Styles
Subject
Template
Title
TopMargin
VBProject
Version

Methods
ClearCustomMenus
ClearCustomToolbars
Close
Drop
ExecuteLine
FollowHyperlink
OpenStencilWindow
ParseLine
Print
Save
SaveAs
SaveAsEx
SetCustomMenus
SetCustomToolbars

Events
BeforeDocumentClose
BeforeDocumentSave
BeforeDocumentSaveAs
BeforeMasterDelete
BeforePageDelete
BeforeSelectionDelete
BeforeShapeDelete
BeforeStyleDelete
CellChanged
ConnectionsAdded
ConnectionsDeleted
DesignModeEntered
DocumentAdded
DocumentChanged
DocumentCreated
DocumentOpened
DocumentSaved
DocumentSavedAs

FormulaChanged
MasterAdded
MasterChanged
PageAdded
PageChanged
RunModeEntered
SelectionAdded
ShapeAdded
ShapeChanged
ShapesDeleted
StyleAdded
StyleChanged
TextChanged

Documents Object

Documents object

A Documents collection includes a Document object for each open document in an instance of Visio. Use the Documents property of an Application object to retrieve its Documents collection. The default property of a Documents collection is Item.

Version added: VISIO 2.0

Properties
Application
Count
EventList
Item
ObjectType
PersistsEvents

Methods
Add
GetNames
Open
OpenEx

Events
BeforeDocumentClose
BeforeDocumentSave
BeforeDocumentSaveAs
BeforeMasterDelete
BeforePageDelete
BeforeSelectionDelete
BeforeShapeDelete
BeforeStyleDelete
CellChanged
ConnectionsAdded
ConnectionsDeleted
DesignModeEntered
DocumentAdded
DocumentChanged
DocumentCreated
DocumentOpened
DocumentSaved
DocumentSavedAs
FormulaChanged
MasterAdded
MasterChanged
PageAdded
PageChanged
RunModeEntered
SelectionAdded
ShapeAdded
ShapeChanged
StyleAdded
StyleChanged
TextChanged

Entities Object

Entities object

The Entities collection includes Entity objects that represent AutoCAD extended entity data. Use the Entities property of a ShapeData object to retrieve its Entities collection. The default property of Entities is Item.

[Beginning with version 4.5, support for this object is included only with Visio Technical. Other versions of Visio, such as Visio Professional, do not include support for this object.]

Version added: VISIO 3.0 TECH

Properties
Count
Item
Name

Methods
Add

Entity Object

Entity object

An Entity object represents AutoCAD extended entity data. To determine the type of data stored in an Entity, use its Group property. The Entity object has no default property.

[Beginning with version 4.5, support for this object is included only with Visio Technical. Other versions of Visio, such as Visio Professional, do not include support for this object.]

Version added: VISIO 3.0 TECH

Properties
BinaryData
BinaryLength
Control
Group
Handle
Index
LayerName
LongValue
Name
RealValue
ShortValue
String
VectorX
VectorY
VectorZ

Methods
Delete

EntityApp Object

EntityApp object

An EntityApp object represents an application that has registered its name with Visio for storing extended entity data. To store extended entity data inside Visio shapes, first create an EntityApp object for your application, then add Entity objects to its Entities collection. The default property of EntityApp is Name.

Deleting an EntityApp object deletes all extended entity data for that application in a given shape.

[Beginning with version 4.5, support for this object is included only with Visio Technical. Other versions of Visio, such as Visio Professional, do not include support for this object.]

Version added: VISIO 3.0 TECH

Properties
Entities
Name

Methods
Delete

EntityApps Object

EntityApps object

The EntityApps collection includes an EntityApp object for each application that has extended entity data in a given shape. Use the EntityApps property of a ShapeData object to retrieve its EntityApps collection. The default property of EntityApps is Item.

[Beginning with version 4.5, support for this object is included only with Visio Technical. Other versions of Visio, such as Visio Professional, do not include support for this object.]

Version added: VISIO 3.0 TECH

Properties
Count
Item

Methods
Add

Event Object

Event object

An Event object is a member of the EventList collection of a source object such as a Document. An Event encapsulates an event code, action code pair. When the event occurs to the source object or one of its subobjects, the action is performed.

The Event property of the Event object establishes the event that triggers the action, and its Action property indicates the action that will be performed. An Event object can trigger two kinds of actions: it can run an add-on, or it can send a notification of the event to the calling program. To create an Event object, use the Add or AddAdvise method of an EventList object.

Use the Persistable property to find out if the event can be stored with a Visio document, or the Persistent property to find out if the event will be stored. Use the Trigger method to trigger an Event object's action without waiting for the event to occur. Use the Enabled property to temporarily disable an event.

The default property of an Event object is Event.

Version added: VISIO 4.0

Properties
Action
Application
Enabled
Event
EventList
ID
Index
ObjectType
Persistable
Persistent
Target
TargetArgs

Methods
Delete
Trigger

EventList Object

EventList object

The EventList collection includes an Event object for each event an object should respond to. The object that possesses the EventList is sometimes called the source object. Use the EventList property of the source object to retrieve its EventList collection.

In general, the level of the source object in Visio's object hierarchy determines the scope of its response. For example, if an Event object for the DocumentOpened event is in the EventList of a Document object, that event's action is triggered only when that document is opened. If the same Event object is in the EventList of an Application object, the event's action is triggered whenever any document is opened in that instance of Visio.

Use the Add method of an EventList object to create an Event object that runs an add-on. Use the AddAdvise method to create an Event object that sends a notification.

The default property of EventList is Item.

Version added: VISIO 4.0

Properties
Application
Count
Item
ItemFromID
ObjectType

Methods
Add
AddAdvise

Font Object

Font object

A Font object represents a typeface that is either applied to text in a document or available for use on the system on which the document is open. A Font object maps its name (for example, "Arial") to the font ID (for example, 3) that Visio stores in a Font cell in a Character Properties section of a shape whose text is formatted with that font.

Note that font IDs can change when a document is opened on different systems or when fonts are installed or removed.

The default property of a Font object is Name.

Version added: VISIO 4.0

Properties
Application
Attributes
CharSet
Document
ID
Index
Name
ObjectType
PitchAndFamily
Stat

Fonts Object

Fonts object

The Fonts collection includes a Font object for each font applied to text in a document or available to be applied. Use the Fonts property of a Document object to retrieve its Fonts collection. The default property of a Fonts collection is Item.

Use the ItemFromID property to retrieve a Font object by its font ID, which is the value shown in the Font cell in a shape's Character Properties section.

Version added: VISIO 4.0

Properties
Application
Count
Document
Item
ItemFromID
ObjectType
Stat

Hyperlink Object

Hyperlink object

A Hyperlink object completely encapsulates the properties and behavior of a hyperlink. A Visio shape can have one hyperlink that navigates to any named location, such as another page, a local document, or a URL. A Hyperlink object enables you to access and manipulate the shape's hyperlink row.

Use the AddHyperlink method to add a Hyperlink object to a shape.

Use the Follow method of a Hyperlink object to navigate to the named location.

The default property of a Hyperlink object is Description.

Version added: VISIO 5.0

Properties
Address
Application
Description
ExtraInfo
Frame
NewWindow
ObjectType
Shape
Stat
SubAddress

Methods
AddToFavorites
Copy
CreateURL
Delete
Follow

Layer Object

Layer object

A Layer object represents a layer of a page or master. You can assign shapes to or remove them from the layer. Use the CellsC property of the Layer object to access cells whose values define layer attributes such as whether it is visible or printable.

Note that a layer's Index and Row properties will typically have different values. The Index property indicates the layer's ordinal position in its Layers collection. The layer's Row property indicates the index of the row in the Layers section where the layer's attributes are defined, in the page sheet of the master or page to which the layer belongs.

The default property of a Layer object is Name.

Version added: VISIO 4.0

Properties
Application
CellsC
Document
EventList
Index
Master
Name
ObjectType
Page
PersistsEvents
Row
Stat

Methods
Add
Delete
Remove

Layers Object

Layers object

The Layers collection includes a Layer object for each layer defined for a page or master. Use the Layers property of a Page object or a Master object to retrieve its Layers collection. The default property of Layers is Item.

Version added: VISIO 4.0

Properties
Application
Count
Document
EventList
Item
Master
ObjectType
Page
PersistsEvents
Stat

Methods
Add

Master Object

Master object

A Master object represents a master in a stencil. You retrieve a particular Master object from the Masters collection of a Document object whose stencil contains that master. To create an instance of a master in a drawing, use the Drop method of a Page object that represents a drawing page. The default property of a Master object is Name.

Version added: VISIO 2.0

Properties
AlignName
Application
Connects
Document
EventList
IconSize
IconUpdate
ID
Index
Layers
MatchByName
Name
ObjectType
OLEObjects
OneD
PageSheet
PatternFlags
PersistsEvents
Prompt
Shapes
Stat
UniqueID

Methods
BoundingBox
CenterDrawing
Close
Delete
DrawBezier
DrawLine
DrawOval
DrawPolyline
DrawRectangle
DrawSpline
DropMany
Export
ExportIcon
GetFormulas
GetResults
Import
ImportIcon
InsertFromFile
InsertObject
Layout

Open
OpenDrawWindow
OpenIconWindow
SetFormulas
SetResults

Events
BeforeMasterDelete
BeforeSelectionDelete
BeforeShapeDelete
CellChanged
ConnectionsAdded
ConnectionsDeleted
FormulaChanged
MasterChanged
SelectionAdded
ShapeAdded
ShapeChanged
TextChanged

Masters Object

Masters object

A Masters collection includes a Master object for each master in a document's stencil. Use the Masters property of a Document object to retrieve its Masters collection. The default property of a Masters collection is Item.

Version added: VISIO 2.0

Properties
Application
Count
Document
EventList
Item
ObjectType
PersistsEvents
Stat

Methods
Add
GetNames

Events
BeforeMasterDelete
BeforeSelectionDelete
BeforeShapeDelete
CellChanged
ConnectionsAdded
ConnectionsDeleted
FormulaChanged
MasterAdded
MasterChanged
SelectionAdded
ShapeAdded
ShapeChanged
TextChanged

Menu Object

Menu object

A Menu object represents a single menu on a Visio menu bar, such as the File menu or the Edit menu. The index of a Menu object within the Menus collection corresponds to the menu's position from left to right on the menu bar, starting with 0 for the menu farthest to the left.

Version added: VISIO 4.0

Properties
Caption
Index
MDIWindowMenu
MenuItems
Parent

Methods
Delete

MenuItem Object

MenuItem object

A MenuItem object represents a single menu item on a Visio menu, such as the Copy menu item on the Edit menu. A MenuItem object contains all the information it needs to display the menu item in the menu and launch the appropriate Visio command or add-on. It also contains text for the Undo, Redo, and Repeat menu items and error messages, plus a string to be displayed in the status bar when the menu item is highlighted.

If the menu item displays a submenu, the MenuItem object has a MenuItems collection that represents items on the submenu. In this case, the MenuItem object's Caption property contains the submenu title and its CmdNum property is set to 200. Most of the MenuItem's other properties are ignored, because this object serves much the same role as a Menu object.

The index of a MenuItem object within the MenuItems collection corresponds to the menu item's position from top to bottom on the menu or submenu, starting with 0 for the first menu item.

The default property of MenuItem is Caption.

Version added: VISIO 4.0

Properties
ActionText
AddonArgs
AddonName
Caption
CmdNum
HelpContextID
HelpFile
Index
IsHierarchical
IsSeparator
MenuItems
MiniHelp
Parent

Methods
Delete

MenuItems Object

MenuItems object

The MenuItems collection contains a MenuItem object for each command on a Visio menu. Unlike other Visio collections, the MenuItems collection is indexed starting with 0 rather than 1.

Use the MenuItems property of a Menu object or a MenuItem object to retrieve its MenuItems collection. The default property of MenuItems is Item.

Version added: VISIO 4.0

Properties
Count
Item
Parent

Methods
Add
AddAt

Menus Object

Menus object

The Menus collection includes a Menu object for each menu in a Visio menu set. Unlike other Visio collections, the Menus collection is indexed starting with 0 rather than 1.

Use the Menus property of a MenuSet object to retrieve its Menus collection. The default property of Menus is Item.

Version added: VISIO 4.0

Properties
Count
Item
Parent

Methods
Add
AddAt

MenuSet Object

MenuSet object

A MenuSet object represents an entire menu set used by a Visio window context. A shortcut menu (which appears when the right mouse button is pressed) is represented by a MenuSet object that has a single untitled Menu object in its Menus collection.

Version added: VISIO 4.0

Properties
Caption
Menus
Parent
SetID

Methods
Delete

MenuSets Object

MenuSets object

A MenuSets collection includes a MenuSet object for each Visio window context that has menus. Unlike other Visio collections, the MenuSets collection is indexed starting with 0 rather than 1.

Use the MenuSets property of a UI object to retrieve its MenuSets collection. The default property of MenuSets is Item.

A MenuSet object is identified in the MenuSets collection by its SetID, which corresponds to a Visio window context. The following are valid SetIDs for MenuSet objects:

visUIObjSetNoDocument
visUIObjSetDrawing
visUIObjSetStencil
visUIObjSetShapeSheet
visUIObjSetIcon
visUIObjSetInPlace
visUIObjSetPrintPreview
visUIObjSetCntx_DrawObjSel
visUIObjSetCntx_DrawOleObjSel
visUIObjSetCntx_TextEdit
visUIObjSetCntx_StencilRO
visUIObjSetCntx_ShapeSheet
visUIObjSetCntx_Toolbar
visUIObjSetBinderInPlace
visUIObjSetCntx_StencilRW
visUIObjSetCntx_StencilDocked
visUIObjSetCntx_FullScreen
visUIObjSetCntx_Hyperlink
visUIObjSetHostingInPlace

Version added: VISIO 4.0

Properties
Count
Item
ItemAtID
Parent

Methods
Add
AddAtID

OLEObject Object

OLEObject object

An OLEObject object represents an OLE 2.0 linked or embedded object or an ActiveX control in a Visio document, page, or master. Use the Object property of OLEObject to obtain the IDispatch interface on the ActiveX control or embedded or linked OLE 2.0 object represented by a shape.

The default property of OLEObject is Object.

Version added: VISIO 5.0

Properties
Application
ClassID
ForeignType
Object
ObjectType
ProgID
Shape
Stat

OLEObjects Object

OLEObjects object

An OLEObjects collection includes an OLEObject object for each OLE 2.0 linked or embedded object or ActiveX control contained in a document, page, or master. Each member of an OLEObjects collection is an OLEObject object, which represents an OLE 2.0 linked or embedded object or an ActiveX control in a Visio document.

Use the OLEObjects property of a Document, Page, or Master to obtain an OLEObjects collection.

The default property of OLEObjects is Item.

Version added: VISIO 5.0

Properties
Application
Count
Item
ObjectType
Stat

Page Object

Page object

A Page object represents a drawing page, which can be either a foreground page or a background page. Use the ActivePage property of an Application object to retrieve the active page in an instance. The members of a Document object's Pages collection represent the pages in that document. Use the Shapes property of a Page object to retrieve the page's shapes. The default property of a Page object is Name.

Version added: VISIO 2.0

Properties
Application
Background
BackPage
BackPageAsObj
BackPageFromName
Connects
Document
EventList
ID
Index
Layers
Name
ObjectType
OLEObjects
PageSheet
PersistsEvents
Shapes
Stat

Methods
AddGuide
BoundingBox
CenterDrawing
Delete
DrawBezier
DrawLine
DrawOval
DrawPolyline
DrawRectangle
DrawSpline
Drop
DropMany
Export
GetFormulas
GetResults
Import
InsertFromFile
InsertObject
Layout
OpenDrawWindow
Paste
Print
SetFormulas
SetResults

Events
BeforePageDelete
BeforeSelectionDelete
BeforeShapeDelete
CellChanged
ConnectionsAdded
ConnectionsDeleted
FormulaChanged
PageChanged
SelectionAdded
ShapeAdded
ShapeChanged
TextChanged

Pages Object

Pages object

A Pages collection includes a Page object for each drawing page in a document. The order of items in a Pages collection is significant: If there are n foreground pages in a document, then the first n pages in its Pages collection will be the foreground pages. They will be sequenced in the same order as they will print. The remaining pages in the collection are the background pages of the document. These are in no particular order. Use the Pages property of a Document object to retrieve its Pages collection. The default property of a Pages collection is Item.

Version added: VISIO 2.0

Properties
Application
Count
Document
EventList
Item
ObjectType
PersistsEvents
Stat

Methods
Add
GetNames

Events
BeforePageDelete
BeforeSelectionDelete
BeforeShapeDelete
CellChanged
ConnectionsAdded
ConnectionsDeleted
FormulaChanged
PageAdded
PageChanged
SelectionAdded
ShapeAdded
ShapeChanged
TextChanged

Path Object

Path object

A Path object is an item in a Paths collection that represents one Geometry section in a shape's ShapeSheet as a sequence of one or more segments whose ends abut.

A Curve is an item in a Path object that is any linear or curved segment representing a consecutive sequence of rows in the Geometry section that the Path object represents. The number of Curve objects in a Path object is not necessarily the same as the number of rows in its Geometry section.

The Path object is conceptually of zero width. Line weights, patterns, and ends are ignored. Corner rounding is included. A Path object may or may not be closed (start of the first Curve is coincident with the end of the last Curve). A Path may intersect itself 0 or more times. For example, a Path may describe a figure 8.

If a Path object is from a collection obtained by the Paths property of a shape, its coordinates will be expressed in the shape's parent coordinate system. If a Path object is from a collection obtained by the PathsLocal property of a shape, its coordinates will be expressed in the shape's local coordinate system. In both cases, coordinates are expressed in internal drawing units (inches).

The default property of a Path object is Item.

Version added: VISIO 5.0

Properties
Application
Closed
Count
Item
ObjectType
Points
Stat

Paths Object

Paths object

A Paths collection includes a Path object for each Geometry section in a shape's ShapeSheet. Each Path object represents a single Geometry section as a sequence of one or more segments whose ends abut. The Paths collection of a group or a shape with multiple Geometry sections will have multiple Path objects.

Use the Paths property of a shape to obtain a Paths collection expressed in the shape's parent coordinate system. Use the PathsLocal property of a shape to obtain a Paths collection expressed in the shape's local coordinate system. In both cases, coordinates are expressed in internal drawing units (inches).

If a shape object is of type Page, Foreign, or Guide, then its Paths and PathsLocal properties will contain zero items.

If a shape object is of type Group, then its Paths and PathsLocal properties will be the union of the paths of its component shapes.

If a shape object is of type Shape, then its Paths and PathsLocal properties will include one item for each Geometry section that defines a stroke of positive length.

The default property of a Paths collection is Item.

Version added: VISIO 5.0

Properties
Application
Count
Item
ObjectType
Stat

Selection Object

Selection object

A Selection object represents a set of Shape objects to which an operation can be applied. The Selection property of a Window object returns a Selection object that corresponds to the set of shapes selected in that window.

After a Selection object is retrieved, you can add or remove shapes by using the Select method of the Selection object. A Selection object can represent shapes from only one Shapes collection at a time, so you cannot base a Selection object on one Shapes collection and add shapes to it from another.

The default property of a Selection object is Item.

Version added: VISIO 2.0

Properties
Application
ContainingMaster
ContainingPage
ContainingShape
Count
Document
EventList
FillStyle
FillStyleKeepFmt
Item
LineStyle
LineStyleKeepFmt
ObjectType
PersistsEvents
Stat
Style
StyleKeepFmt
TextStyle
TextStyleKeepFmt

Methods
BoundingBox
BringForward
BringToFront
Combine
ConvertToGroup
Copy
Cut
Delete
DeselectAll
Duplicate
Export
FitCurve
FlipHorizontal
FlipVertical
Fragment
Group
Intersect
Join

Layout
ReverseEnds
Rotate90
Select
SelectAll
SendBackward
SendToBack
Subtract
Trim
Ungroup
Union

Shape Object

Shape object

A Shape object represents anything you can select with the pointer tool in a drawing window: a basic shape, a group, a guide or guide point, or an object from another application. You can retrieve a particular shape from the Shapes collection of a Page object, or a Master object or from the Shapes collection of a Shape object that represents a group. Use the Cells and Connects properties of a Shape object to retrieve Cell objects and Connect objects, respectively. The default property of a Shape object is Name.

Version added: VISIO 2.0

Properties

Application
AreaIU
CellExists
Cells
CellsSRC
CellsSRCExists
Characters
CharCount
ClassID
Connects
ContainingMaster
ContainingPage
ContainingShape
Data1
Data2
Data3
Document
EventList
FillStyle
FillStyleKeepFmt
ForeignType
FromConnects
GeometryCount
Help
HitTest
Hyperlink
ID
Index
Layer
LayerCount
LengthIU
LineStyle
LineStyleKeepFmt
Master
Name
NameID
Object
ObjectIsInherited
ObjectType
OneD
Parent
Paths
PathsLocal

PersistsEvents
ProgID
RowCount
RowExists
RowsCellCount
RowType
SectionExists
Shapes
Stat
Style
StyleKeepFmt
Text
TextStyle
TextStyleKeepFmt
Type
UniqueID

Methods
AddHyperlink
AddNamedRow
AddRow
AddRows
AddSection
BoundingBox
BringForward
BringToFront
CenterDrawing
ConvertToGroup
Copy
Cut
Delete
DeleteRow
DeleteSection
DrawBezier
DrawLine
DrawOval
DrawPolyline
DrawRectangle
DrawSpline
Drop
DropMany
Duplicate
Export
FitCurve
FlipHorizontal
FlipVertical
GetFormulas
GetResults
Group
Import
InsertFromFile
InsertObject
Layout
OpenDrawWindow
OpenSheetWindow
ReverseEnds

Rotate90
SendBackward
SendToBack
SetBegin
SetCenter
SetEnd
SetFormulas
SetResults
Ungroup

Events
BeforeSelectionDelete
BeforeShapeDelete
CellChanged
FormulaChanged
SelectionAdded
ShapeAdded
ShapeChanged
TextChanged

ShapeData Object

ShapeData object

The ShapeData object provides database-like features for a Visio shape. No property or method of any Visio object returns a ShapeData object. Instead, you use the Visual Basic CreateObject function to create the object, passing "Visio.ShapeDatabase" as the object name. To use the database with a particular shape, you first get the shape from Visio and then pass it to the database using the PutShape method of the ShapeData object. After that, you can access the shape's attribute and AutoCAD-compatible extended entity data through Attribute and Entity objects of the ShapeData object.

In Visio, attribute data is stored in a shape's custom properties, so you can access them via the Cells or CellsSRC property of the shape. Extended entity data is stored in the Data2 and Data3 fields of a shape, flagged with an ASCII 1 character as the first character of the field. If you are using the ShapeData object and don't want to overwrite this data, use the Visual Basic Chr$() function to check for an ASCII 1 at the beginning of these fields.

The ShapeData object has no default property.

[Beginning with version 4.5, support for this object is included only with Visio Technical. Other versions of Visio, such as Visio Professional, do not include support for this object.]

Version added: VISIO 3.0 TECH

Properties
Attributes
EntityApps
Shape

Methods
BeginTransaction
EndTransaction
PutShape

Shapes Object

Shapes object

A Shapes collection includes a Shape object for each basic shape, group, guide or guide point, or object from another application on a drawing page, master, or group. The order of items in a Shapes collection corresponds to the stacking (drawing) order of the shapes, from backmost to frontmost. Use the Shapes property of a Page, Master, or Shape object to retrieve its Shapes collection. The default property of a Shapes collection is Item.

Version added:VISIO 2.0

Properties
Application
ContainingMaster
ContainingPage
ContainingShape
Count
Document
EventList
Item
ItemFromID
ObjectType
PersistsEvents
Stat

StatusBar Object

StatusBar object

A StatusBar object represents a Visio status bar, which is shown at the bottom of a Visio window.

Version added: VISIO 4.0

Properties
Caption
Parent
SetID
StatusBarItems

Methods
Delete

StatusBarItem Object

StatusBarItem object

A StatusBarItem object represents a single item (button, message, and so forth) on a status bar. The index of a StatusBarItem object within its collection corresponds to the position of the item on the status bar, starting with 0 for the item farthest to the left.

Version added: VISIO 4.0

Properties
ActionText
AddonArgs
AddonName
CmdNum
CntrlID
CntrlType
HelpContextID
HelpFile
Index
Parent
Priority
Spacing
TypeSpecific1
TypeSpecific2

Methods
Delete
IconFileName

StatusBarItems Object

StatusBarItems object

The StatusBarItems collection includes a StatusBarItem object for each Visio window context. Unlike other Visio collections, the StatusBarItems collection is indexed starting with 0 rather than 1.

Use the StatusBarItems property of a StatusBar object to retrieve its StatusBarItems collection. The default property of StatusBarItems is Item.

Version added: VISIO 4.0

Properties
Count
Item
Parent

Methods
Add
AddAt

StatusBars Object

StatusBars object

The StatusBars collection includes a StatusBar object for each Visio window context that can display a status bar. Unlike other Visio collections, the StatusBars collection is indexed starting with 0 rather than 1.

Use the StatusBars property of a UI object to retrieve its StatusBars collection. The default property of StatusBars is Item.

A StatusBar object is identified in the StatusBars collection by its SetID, which corresponds to a Visio window context. The following are valid SetIDs for StatusBar objects:

visUIObjSetNoDocument
visUIObjSetDrawing
visUIObjSetStencil
visUIObjSetShapeSheet
visUIObjSetIcon
visUIObjSetPrintPreview

Version added: VISIO 4.0

Properties
Count
Item
ItemAtID
Parent

Methods
Add
AddAtID

Style Object

Style object

A Style object represents a style defined in a document. You retrieve a particular style from the Styles collection of a Document object. A style defines some combination of line, fill and text attributes as indicated by the values of its IncludesFill, IncludesLine, and IncludesText properties. For example, if IncludesFill is non-zero, the style defines fill attributes. A style can also inherit attributes from another style, as indicated by its FillBasedOn, LineBasedOn, and TextBasedOn properties.

Like a Shape object, a Style object has cells whose formulas define the values of the style's attributes. To retrieve one of these cells, use the Cells property of the Style object.

Any Shape object to which a style is applied inherits the attributes defined by the style. Use the LineStyle, FillStyle, TextStyle, or Style properties of a Shape object to apply a style to a shape or to determine what style is applied to a shape.

The default property of a Style object is Name.

Version added: VISIO 2.0

Properties
Application
BasedOn
Cells
Document
EventList
FillBasedOn
ID
IncludesFill
IncludesLine
IncludesText
Index
LineBasedOn
Name
ObjectType
PersistsEvents
Stat
TextBasedOn

Methods
Delete
GetFormulas
GetResults
SetFormulas
SetResults

Events
BeforeStyleDelete
StyleChanged

Styles Object

Styles object

A Styles collection includes a Style object for each style defined in a document. Use the Styles property of a Document object to retrieve its Styles collection. The default property of a Styles collection is Item.

Version added: VISIO 2.0

Properties
Application
Count
Document
EventList
Item
ItemFromID
ObjectType
PersistsEvents
Stat

Methods
Add
GetNames

Events
BeforeStyleDelete
StyleAdded
StyleChanged

Toolbar Object

Toolbar object

A Toolbar object represents a group of toolbar items in a Visio window. The index of the Toolbar object within the Toolbars collection corresponds to its order in the Visio window, starting with 0 for the toolbar closest to the top. Up to ten toolbars can be displayed in a Visio window at one time.

The default property of Toolbar is Caption.

Version added:VISIO 4.0

Properties
Caption
Index
Parent
ToolbarItems

Methods
Delete

ToolbarItem Object

ToolbarItem object

A ToolbarItem object represents one item in a Toolbar object. A ToolbarItem object can represent a button, space, combo box, or any other item in the Visio toolbars. The index of the ToolbarItem object within the ToolbarItems collection corresponds to its position on the toolbar, starting with 0 for the item farthest to the left.

Version added: VISIO 4.0

Properties
ActionText
AddonArgs
AddonName
CmdNum
CntrlID
CntrlType
HelpContextID
HelpFile
Index
Parent
Priority
Spacing
TypeSpecific1
TypeSpecific2

Methods
Delete
IconFileName

ToolbarItems Object

ToolbarItems object

The ToolbarItems collection includes a ToolbarItem object for each item on a Visio toolbar. Unlike other Visio collections, the ToolbarItems collection is indexed starting with 0 rather than 1.

Use the ToolbarItems property of a Toolbar object to retrieve its ToolbarItems collection. The default property of ToolbarItems is Item.

Version added: VISIO 4.0

Properties
Count
Item
Parent

Methods
Add
AddAt

Toolbars Object

Toolbars object

A Toolbars collection includes a Toolbar object for each toolbar in a Visio window context. Unlike other Visio collections, the Toolbars collection is indexed starting with 0 rather than 1.

Use the Toolbars property of a ToolbarSet object to retrieve its Toolbars collection. The default property of Toolbars is Item.

A Toolbars collection can include a maximum of ten Toolbar objects.

Version added: VISIO 4.0

Properties
Count
Item
Parent

Methods
Add
AddAt

ToolbarSet Object

ToolbarSet object

A ToolbarSet object represents the set of toolbars for a Visio window context.

Version added: VISIO 4.0

Properties
Caption
Parent
SetID
Toolbars

Methods
Delete

ToolbarSets Object

ToolbarSets object

A ToolbarSets collection includes a ToolbarSet object for each Visio window context that can display toolbars. Unlike other Visio collections, the ToolbarSets collection is indexed starting with 0 rather than 1.

Use the ToolbarSets property of a UI object to retrieve its ToolbarSets collection. The default property of ToolbarSets is Item.

A ToolbarSet object is identified in the ToolbarSets collection by its SetID, which corresponds to a Visio window context. The following are valid SetIDs for ToolbarSet objects:

visUIObjSetNoDocument
visUIObjSetDrawing
visUIObjSetStencil
visUIObjSetShapeSheet
visUIObjSetIcon
visUIObjSetPrintPreview
visUIObjSetText

Version added: VISIO 4.0

Properties
Count
Item
ItemAtID
Parent

Methods
Add
AddAtID

UI Object Object

UI Object object

The UI object represents Visio's menus, toolbars, accelerators, and status bars, from either the built-in Visio user interface or a customized version of it. Use the BuiltInMenus property of an Application object to retrieve a UI object that contains Visio's menus and accelerators. Use the BuiltInToolbars property of an Application object to retrieve a UI object that contains Visio's toolbars and status bars.

If an Application object or Document object has a customized user interface, use the CustomMenus or CustomToolbars properties to retrieve UI objects that represent these.

A UI object can be stored in a file and loaded into Visio. Use the SaveToFile method to save the object and LoadFromFile to load it, or set the CustomMenusFile or CustomToolbarsFile of an Application object or Document object to the name of the stored UI file.

Version added: VISIO 4.0

Properties
AccelTables
Flavor
MenuSets
Name
StatusBars
ToolbarSets

Methods
LoadFromFile
SaveToFile
UpdateUI

Window Object

Window object

A Window object represents an open document window in an instance of Visio. Use the ActiveWindow property of an Application object to retrieve the active window in an instance of Visio.

Use the Page property of a Window object to retrieve a Page object that represents the page shown in the window. Use the Document property to retrieve a Document object that represents the document displayed in that window. Use the Selection property to retrieve a Selection object that represents the shapes selected in that window.

The default property of a Window object is Application.

Version added: VISIO 2.0

Properties
Application
Document
EventList
Index
Master
ObjectType
Page
PageAsObj
PageFromName
PersistsEvents
Selection
ShowConnectPoints
ShowGrid
ShowGuides
ShowPageBreaks
ShowRulers
Stat
SubType
Type
WindowHandle
WindowHandle32
Zoom

Methods
Activate
AddToGroup
Close
Combine
Copy
Cut
Delete
DeselectAll
DockedStencils
Duplicate
Fragment
Group
Intersect
Join
Paste

RemoveFromGroup
Select
SelectAll
Subtract
Trim
Union

Events
BeforeWindowClose
BeforeWindowPageTurn
BeforeWindowSelDelete
SelectionChanged
WindowActivated
WindowTurnedToPage

Windows Object

Windows object

The Windows collection includes a Window object for each drawing window, docked or floating stencil window, ShapeSheet window, or edit icon window that is open in an instance of Visio. If a docked stencil window contains more than one stencil, only one window is counted.

Use the Windows property of an Application object to retrieve its Windows collection. The default property of a Windows collection is Item.

Version added: VISIO 2.0

Properties
Application
Count
EventList
Item
ObjectType
PersistsEvents

Methods
Arrange

Events
BeforeWindowClose
BeforeWindowPageTurn
BeforeWindowSelDelete
SelectionChanged
WindowActivated
WindowOpened
WindowTurnedToPage

ShapeSheet Cells

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

See the Cells property.

EventXFMod
FillBkgnd
FillForegnd
FillPattern
FlipX
FlipY
Font
Format
Glue
GlueType
HAlign
Height
HideText
ImgHeight
ImgOffsetX
ImgOffsetY
ImgWidth
IndFirst
IndLeft
IndRight
Invisible
Label
LayerProperties
LeftMargin
LineCap
LineColor
LinePattern
LineTo
LineWeight
Lock
LockAspect
LockBegin
LockCalcWH
LockCrop
LockDelete
LockEnd
LockFormat
LockGroup
LockHeight
LockMoveX
LockMoveY
LockRotate
LockSelect
LockTextEdit
LockVtxEdit
LockWidth
LocPinX
LocPinY
Menu
NoAlignBox
NoCtlHandles
NoFill
NonPrinting
NoObjHandles
NoShow
PageHeight

PageScale
PageWidth
PinX (Guide Info section)
PinX (Shape Transform section)
PinY (Guide Info section)
PinY (Shape Transform section)
Pos
Print
Prompt (Action section)
Prompt (Custom Properties section)
Prompt (User-defined Cells section)
PropRow
ResizeMode
RightMargin
Rounding
ShdwBkgnd
ShdwForegnd
ShdwOffsetX
ShdwOffsetY
ShdwPattern
Size
Snap
SortKey
SpAfter
SpBefore
SpLine
SplineKnot
SplineStart
Start
Style
TextBkgnd
TheData
TheText
Tip
TopMargin
TxtAngle
TxtHeight
TxtLocPinX
TxtLocPinY
TxtPinX
TxtPinY
TxtWidth
Type
UpdateAlignBox
UserRow
Value (Custom Properties section)
Value (User-defined Cells section)
VerticalAlign
Visible
WalkPreference
Width
X
XBehavior
XDynamics
XGridDensity
XGridOrigin

XGridSpacing
XRulerDensity
XRulerOrigin
Y
YBehavior
YDynamics
YGridDensity
YGridOrigin
YGridSpacing
YRulerDensity
YRulerOrigin

Properties

AccelItems
AccelTables
Action
ActionText
Active
ActiveDocument
ActivePage
ActiveWindow
AddonArgs
AddonName
AddonPaths
Addons
Address
AlertResponse
AlignName
Alt
Application
AreaIU
Attributes
Background
BackPage
BackPageAsObj
BackPageFromName
BasedOn
Begin
BinaryData

BinaryLength
Blue
BottomMargin
BuiltInMenus
BuiltInToolbars
Caption
Category
CellExists
Cells
CellsC
CellsSRC
CellsSRCExists
Characters
CharCount
CharProps
CharPropsRow
CharSet
ClassID
Closed
CmdNum
CntrlID
CntrlType
Colors
Column
Company
Connects
ContainingMaster
ContainingPage
ContainingShape
Control
Count
Creator
CustomMenus
CustomMenusFile
CustomToolbars
CustomToolbarsFile
Data1
Data2
Data3
DefaultFillStyle
DefaultLineStyle
DefaultStyle
DefaultTextStyle
DefaultValue
DeferRecalc
Description
Document
Documents
DrawingPaths
Enabled
End
Entities
EntityApps
Error
Event
EventInfo

EventList
EventsEnabled
ExtraInfo
FieldCategory
FieldCode
FieldFormat
FieldFormula
FillBasedOn
FillStyle
FillStyleKeepFmt
FilterPaths
Flags
Flavor
Fonts
ForeignType
Formula
FormulaForce
Frame
FromCell
FromConnects
FromPart
FromSheet
FullName
GeometryCount
Green
Group
Handle
Help
HelpContextID
HelpFile
HelpPaths
HitTest
Hyperlink
HyperlinkBase
IconSize
IconUpdate
ID
IncludesFill
IncludesLine
IncludesText
Index
InPlace
InstanceHandle
InstanceHandle32
IsConstant
IsField
IsHierarchical
IsInherited
IsSeparator
IsVisio16
IsVisio32
Item
ItemAtID
ItemFromID
Key
Keywords

Language
Layer
LayerCount
LayerName
Layers
LeftMargin
LengthIU
LineBasedOn
LineStyle
LineStyleKeepFmt
LocalName
LongValue
Manager
Master
Masters
MatchByName
MDIWindowMenu
MenuItems
Menus
MenuSets
MiniHelp
Mode
Name
NameID
NewWindow
Object
ObjectIsInherited
ObjectType
OLEObjects
OnDataChangeDelay
OneD
Page
PageAsObj
PageFromName
Pages
PageSheet
PaletteEntry
PaperHeight
PaperSize
PaperWidth
ParaProps
ParaPropsRow
Parent
Path
Paths
PathsLocal
PatternFlags
Persistable
Persistent
PersistsEvents
PitchAndFamily
Points
PrintCenteredH
PrintCenteredV
PrintFitOnPages
PrintLandscape

PrintPagesAcross
PrintPagesDown
PrintScale
Priority
ProcessID
ProfileName
ProgID
Prompt
PromptForSummary
ReadOnly
RealValue
Red
Result
ResultForce
ResultFromInt
ResultFromIntForce
ResultInt
ResultIU
ResultIUForce
ResultStr
RightMargin
Row
RowCount
RowExists
RowName
RowsCellCount
RowType
RunBegin
RunEnd
Saved
SavePreviewMode
ScreenUpdating
Section
SectionExists
Selection
SetID
Shape
Shapes
Shift
ShortValue
ShowConnectPoints
ShowGrid
ShowGuides
ShowMenus
ShowPageBreaks
ShowProgress
ShowRulers
ShowStatusBar
ShowToolbar
Spacing
Start
StartupPaths
Stat
StatusBarItems
StatusBars
StencilPaths

String
Style
StyleKeepFmt
Styles
SubAddress
Subject
SubType
TableName
TabPropsRow
Target
TargetArgs
Template
TemplatePaths
Text
TextAsString
TextBasedOn
TextStyle
TextStyleKeepFmt
Title
ToCell
ToolbarItems
Toolbars
ToolbarSets
ToolbarStyle
ToPart
TopMargin
ToSheet
TraceFlags
Type
TypeSpecific1
TypeSpecific2
UniqueID
Units
UserName
Value
VBE
VBProject
VectorX
VectorY
VectorZ
Version
WindowHandle
WindowHandle32
Windows
Zoom

Methods

Activate
Add
AddAdvise
AddAt
AddAtID
AddCustomField
AddField
AddGuide
AddHyperlink
AddNamedRow
AddRow
AddRows
AddSection
AddToFavorites
AddToGroup
Arrange
BeginTransaction
BoundingBox
BringForward
BringToFront
CenterDrawing
ClearCustomMenus
ClearCustomToolbars
Close
Combine
ConvertResult

ConvertToGroup
Copy
CreateURL
Cut
Delete
DeleteRow
DeleteSection
DeselectAll
DockedStencils
DoCmd
DrawBezier
DrawLine
DrawOval
DrawPolyline
DrawRectangle
DrawSpline
Drop
DropMany
Duplicate
EndTransaction
EnumDirectories
ExecuteLine
Export
ExportIcon
FitCurve
FlipHorizontal
FlipVertical
Follow
FollowHyperlink
FormatResult
Fragment
GetFormulas
GetNames
GetResults
GlueTo
GlueToPos
Group
IconFileName
Import
ImportIcon
InsertFromFile
InsertObject
Intersect
Join
Layout
LoadFromFile
Open
OpenDrawWindow
OpenEx
OpenIconWindow
OpenSheetWindow
OpenStencilWindow
ParseLine
Paste
Point
PointAndDerivatives

Print
PurgeUndo
PutShape
QueueMarkerEvent
Quit
Redo
Remove
RemoveFromGroup
ReverseEnds
Rotate90
Run
Save
SaveAs
SaveAsEx
SaveToFile
SaveWorkspaceAs
Select
SelectAll
SendBackward
SendToBack
SetBegin
SetCenter
SetCustomMenus
SetCustomToolbars
SetEnd
SetFormulas
SetResults
Subtract
Trigger
Trim
Undo
Ungroup
Union
UpdateUI

Events

AfterModal
AppActivated
AppDeactivated
AppObjectActivated
AppObjectDeactivated
BeforeDocumentClose
BeforeDocumentSave
BeforeDocumentSaveAs
BeforeMasterDelete
BeforeModal
BeforePageDelete
BeforeQuit
BeforeSelectionDelete
BeforeShapeDelete
BeforeStyleDelete
BeforeWindowClose
BeforeWindowPageTurn
BeforeWindowSelDelete
CellChanged
ConnectionsAdded
ConnectionsDeleted
DesignModeEntered
DocumentAdded
DocumentChanged
DocumentCreated
DocumentOpened

DocumentSaved
DocumentSavedAs
FormulaChanged
MarkerEvent
MasterAdded
MasterChanged
PageAdded
PageChanged
RunModeEntered
SelectionAdded
SelectionChanged
ShapeAdded
ShapeChanged
ShapesDeleted
StyleAdded
StyleChanged
TextChanged
WindowActivated
WindowOpened
WindowTurnedToPage

Example

AccelItems property

Applies to: AccelTable

Summary: Returns the AccelItems collection of an AccelTable object.

Version: VISIO 4.0

Syntax: objRet = object.**AccelItems**

Element	Description
objRet	An AccelItems collection
object	The AccelTable object that owns the collection

See also: AccelItems object

for AccelItems, AccelTables

```
'This VBA macro demonstrates deleting an accelerator.

Public Sub DeleteAccelItem_Example ()

  Dim uiObj As Visio.UIObject
  Dim accelTableObj As Visio.AccelTable
  Dim accelItemsObj As Visio.AccelItems
  Dim accelItemObj As Visio.AccelItem
  Dim i As Integer

    'Retrieve the UIObject for the copy of the BuiltInMenus
    Set uiObj = Visio.Application.BuiltInMenus

    'Set accelTableObj to the drawing menu set
    Set accelTableObj = uiObj.AccelTables.ItemAtID(visUIObjSetDrawing)

    'Retrieve the accelerator items collection
    Set accelItemsObj = accelTableObj.AccelItems

    ' Retrieve the accelerator item for the Visual Basic Editor.
    ' To do this you must iterate through the collection and locate the item
    ' you want to manipulate.
    ' The item can be identified either by checking the CmdNum or by checking
for
    ' the specific key. Because checking for the key requires looking at the
Alt,
    ' Control, Shift, and Key properties it is better to use the CmdNum.
    ' Because you retrieved the builtin menus, you know that you can find the
    ' accelerator.
    For i = 0 To accelItemsObj.Count - 1
        Set accelItemObj = accelItemsObj.Item(i)
        If accelItemObj.CmdNum = Visio.visCmdToolsRunVBE Then
            Exit For
        End If
    Next i

    ' Delete the accelerator.
    accelItemObj.Delete

    ' Tell Visio to use the new UI.
    ThisDocument.SetCustomMenus uiObj

End Sub
```

Example

AccelTables property

Applies to: UI Object

Summary: Returns the AccelTables collection of a UI object.

Version: VISIO 4.0

Syntax: objRet = object.**AccelTables**

Element	Description
objRet	An AccelTables collection
object	The UI object that owns the collection

Remarks: If a UI object represents menu items and accelerators (for example, if the object was retrieved using the BuiltInMenus property of an Application object), its AccelTables collection represents tables of accelerator keys for that UI object.

To retrieve accelerators for a particular window context, for example, the drawing window, use the ItemAtID property of an AccelTables collection. If a context does not include accelerators, it has no AccelTables collection. For a list of valid window contexts, see the AccelTables object.

See also: AccelTables object

for AccelTables

`*AccelItems Property`

Action property

Applies to: Event

Summary: Gets or sets the action code of an Event object.

Version: VISIO 4.0

Syntax: intRet = object.**Action**
object.**Action** = actionCode

Element	Description
intRet	The Event object's action code
object	The Event object
actionCode	The new action code to assign

Remarks: An Event object consists of an event-action pair. When the event occurs, the action is performed.

An action code is a numeric constant that specifies the action that is triggered when the event occurs. Visio supports the following action codes:

visActCodeRunAddon = 1
visActCodeAdvise = 2

Other properties of an Event object specify the target of the action and any arguments to be sent to the target. For example, if the action is visActCodeRunAddon, the target is the name of the add-on to run, and the arguments are sent to the add-on when it is run.

To create an Event object whose action is visActCodeRunAddon, use the Add method of an EventList object. To create an Event object whose action is visActCodeAdvise, use the AddAdvise method.

See also: Add method, AddAdvise method, Event property, EventInfo property, EventList object, Target property, TargetArgs property

for Action

Example

ActionText property

Applies to: MenuItem, StatusBarItem, ToolbarItem

Summary: Gets or sets the action text for a menu item, toolbar item, or status bar item.

Version: VISIO 4.0

Syntax: object.**ActionText** = actionStr
actionStr = object.**ActionText**

Element	Description
object	The object that owns the action text
actionStr	A string that describes the action

Remarks: The ActionText property determines the string that is displayed with the Undo, Redo, and Repeat menu items on Visio's Edit menu. This string is inserted into error messages and is used as the tool tip for StatusBarItem and ToolbarItem objects.

If ActionText is null and the object's CmdNum property is set to one of Visio's command IDs, the object uses the default action text from Visio's built-in user interface.

See also: CmdNum property, MiniHelp property

for ActionText, BuiltInMenus, MenuItems, Menus, MenuSets, MiniHelp, SetCustomMenus

```
'This VBA macro demonstrates adding a menu and menu item to the drawing window
'menu set. It also sets the menu item's properties such as: Caption,
'AddOnName, AddOnArgs, ActionText, and MiniHelp.

Public Sub AddMenuItem_Example ()

  Dim UIObj As Visio.UIObject
  Dim menuSetsObj As Visio.MenuSets
  Dim menuSetObj As Visio.MenuSet
  Dim menusObj as Visio.Menus
  Dim menuObj As Visio.Menu
  Dim menuItemsObj as Visio.MenuItems
  Dim menuItemObj As Visio.MenuItem

  'Get a UI object that represents Visio's built-in menus
  Set UIObj = Visio.Application.BuiltInMenus

  'Get the MenuSets collection
  Set menuSetsObj = UIObj.MenuSets

  'Get the drawing window menu set
  Set menuSetObj=menuSetsObj.ItemAtId(visUIObjSetDrawing)

  'Get the menus collection
  Set menusObj = menuSetObj.Menus

  'Add a Demo menu before the Window menu.
  Set menuObj = menusObj.AddAt(7)
  menuObj.Caption = "Demo"

  'Get the menuItems collection
  Set menuItemsObj = menuObj.MenuItems

  'Add a menu item to the new Demo menu
  Set menuItemObj = menuItemsObj.Add

  'Set the properties for the new menu item
  menuItemObj.Caption = "Run &ShowArgs"
  menuItemObj.AddOnName = "ShowArgs.EXE"
  menuItemObj.AddOnArgs = "/DVS=Fun"
  menuItemObj.ActionText = "Run ShowArgs"
  menuItemObj.MiniHelp = "Run the ShowArgs application"

  'Tell Visio to use the new UI when the document is active
  ThisDocument.SetCustomMenus uiObj

End Sub
```

Example

Activate method

Applies to: Window

Summary: Activates the indicated window.

Version: VISIO 2.0

Syntax: object.**Activate**

Element	Description
object	The Window object to activate

Remarks: Visio can have more than one window open at a time, but only one window is active. Activating a window can change the objects returned by the ActiveWindow, ActivePage, and ActiveDocument properties.

See also: ActiveWindow property

for Activate, Arrange

```
'This VBA macro demonstrates activating and arranging windows.

Public Sub Activate_Example()

  Dim docObj As Visio.Document
  Dim winObj As Visio.Window
  Dim winObj2 As Visio.Window

  'Create 2 new windows by adding documents.
  Set docObj = Documents.Add("")
  Set winObj = ActiveWindow
  Set docObj = Documents.Add("")
  Set winObj2 = ActiveWindow

  'Tile the windows using the arrange method
  '(currently the last opened window is active).
  Windows.Arrange

  'Make the other window the active window using the Activate method.
  winObj.Activate

End Sub
```

Example

Active property

Applies to: Application

Summary: Indicates whether the instance of Visio represented by the Application object is the active application on the Windows desktop.

Version: VISIO 4.1

Syntax: intRet = object.**Active**

Element	Description
intRet	False (0) if the application is not active; True (-1) it if is active
object	The Application object to check

Remarks: The Active property returns the value True (non-zero) if the instance of Visio represented by the Application object is the active application on the Windows desktop (the application with the highlighted title bar) among all applications the user has open.

Note that the active application in Windows is distinct from the active Visio instance, which is returned by a call to the OLE GetActiveObject API (GetObject in Visual Basic). GetObject retrieves the instance of Visio that was most recently activated, which may or may not be the active application on the desktop at that moment. Of all instances of Visio that are currently running, only one is the active Visio instance.

For example, suppose the user has opened one instance of Visio and one instance of another application, such as Excel.

* If the instance of Visio is the active application on the user's desktop, GetObject(, "visio.application") retrieves that instance and its Active property will be True.

* If the user activates the instance of Excel, GetObject(, "visio.application") retrieves the same instance of Visio, but its Active property will be False.

If an Application object's Active property is True, you can assume that the corresponding instance of Visio is the active instance of Visio unless the InPlace property is also True. If an instance of Visio is activated for visual (in-place) editing in a container application, that instance may not necessarily report itself as the active Visio.

for Active

```
'This VB program demonstrates getting the active instance of Visio.

Public Sub Active_Prop_Example()

  Dim appVisio1 As Visio.Application
  Dim appVisio2 As Visio.Application

  'Create 2 new instances of Visio.
  Set appVisio1 = CreateObject("visio.application")
  Set appVisio2 = CreateObject("visio.application")

  'Use the Active property to verify whether or not the instance of Visio is
  'active.
  Debug.Print appVisio1.Active  ' Result = False
  Debug.Print appVisio2.Active  ' Result = True

End Sub
```

Example

ActiveDocument property

Applies to: <Global>, Application

Summary: Returns the active Document object.

Version: VISIO 2.0

Syntax: objRet = object.**ActiveDocument**

Element	Description
objRet	A Document object that represents the active document
object	The Application object that owns the document

Remarks: The active document is the document shown in the active window.

There is no active document when no documents are open. When no document is active, ActiveDocument returns Nothing and does not raise an exception. The result returned by ActiveDocument should be compared with Nothing to determine if there really is an active document.

Alternatively, you can infer there is an active document if Application.Documents.Count is greater than zero.

If your code is in the VBA project of a Visio document, ActiveDocument will often, but not necessarily, return a reference to the ThisDocument object. ThisDocument is a class module in the VBA project of every Visio document. When referenced from code in a project, ThisDocument returns a reference to the project's Document object.

If the ThisDocument object is presently being shown in the active window, then ActiveDocument and ThisDocument will refer to the same document. Whether your code uses ActiveDocument or ThisDocument depends on its purpose. Your code shouldn't assume they're the same document.

See also: ActivePage property, ActiveWindow property, WindowActivated event

for ActiveDocument

```
'This VBA macro demonstrates two methods of getting an active document.

Public Sub GetActiveDocument_Example ()

  Dim docObj As Visio.Document

  'The two 'If' statements below demonstrate two ways of safely
  'retrieving the active document if one exists.

  If Documents.Count > 0 Then              'Iterate through documents
    Set docObj = ActiveDocument            'Get document
  End If

  'Alternate Example

  If Not(ActiveDocument Is Nothing) Then   'Find out if a document exists
    Set docObj = ActiveDocument            'Retrieve the document
  End If

End Sub
```

Example

ActivePage property

Applies to: <Global>, Application

Summary: Returns the active Page object.

Version: VISIO 2.0

Syntax: objRet = object.**ActivePage**

Element	Description
objRet	A Page object that represents the active page
object	The Application object that owns the page

Remarks: The ActivePage property returns a Page object only when the active window displays a drawing page; otherwise, it returns Nothing. To verify that a page is active, use the Is operator to compare ActivePage with Nothing.

See also: ActiveDocument property, ActiveWindow property, WindowActivated event

for ActivePage

```
'This VBA macro demonstrates getting an active page.

Public Sub GetActivePage_Example ()

  Dim pagObj As Visio.Page

  If Not(ActivePage Is Nothing) Then               'Find out if a page exists
    Set pagObj = Visio.Application.ActivePage      'Retrieve the page
  End If

End Sub
```

Example

ActiveWindow property

Applies to: <Global>, Application

Summary: Returns the active Window object.

Version: VISIO 2.0

Syntax: objRet = object.**ActiveWindow**

Element	Description
objRet	A Window object that represents the active window
object	The Application object that owns the window

Remarks: Visio has four types of windows: stencil, drawing, ShapeSheet, and edit icon. The active Window object may change as a result of other actions. If a window in an instance of Visio is not active, ActiveWindow returns Nothing.

See also: Activate method, ActiveDocument property, ActivePage property, WindowActivated event

for ActiveWindow

```
*Page Property
```

Example

Add method

Applies to: AccelItems, AccelTables, Addons, Attributes, Documents, Entities, EntityApps, EventList, Layer, Layers, Masters, MenuItems, Menus, MenuSets, Pages, StatusBarItems, StatusBars, Styles, ToolbarItems, Toolbars, ToolbarSets

Summary: Adds a new object to the indicated collection.

Version: VISIO 2.0

Syntax: objRet = object.**Add**
objRet = object.**Add** (arguments)

Element	Description
objRet	The new object added to the collection
object	The collection to receive the new object
arguments	How to add a particular kind of object

Remarks: The Add method takes no arguments for the Entities and Masters collections.

The Add method also takes no arguments for the AccelItems, AccelTables, Menus, MenuSets, StatusBarItems, StatusBars, ToolbarItems, Toolbars, and ToolbarSets collections. All properties of the new object are initialized to zero, so you need to set only the properties that you wish to change from the defaults. Note that you can add up to 10 toolbars to the ToolBars collection. Attempting to add more will cause an error (E_FAIL).

The Add method for the Addons, Attributes, Documents, EntityApps, EventList, Layers, and Styles collection, and the Add method for a Layer object, take arguments that control how the object is added.

Addons collection: Adds the specified .EXE or .VSL file to the collection and returns an Addon object if the string expression specifies an .EXE file, or nothing if it specifies a .VSL file.

Attributes, EntityApps: Takes a string argument giving the tag of the item to add.

Documents collection:
Add("") creates a new drawing based on no template. Add("somefile.vst") creates a new drawing based on the specified .VST file. Visio also opens stencils that are part of the template's workspace and copies styles and other settings associated with the template to the new document. If the template filename is invalid, no document is returned and an error is generated. "Somefile" can include a path or not. If no path is indicated, Visio searches directories designated in the application's TemplatePath.

Add("vss") creates a new stencil based on no stencil. Add("somefile.vss") opens a copy of the specified .VSS file. Add("somefile.vsd") opens a copy of the specified .VSD file. These latter two options are equivalent to selecting Copy in the Open section of the Open dialog box or using the OpenEx method with a visOpenCopy flag.

EventList collection: The Add method has the following syntax and returns the new Event object:

Set eventObj = eventList.Add(eventCode,actionCode,target,targetArgs)

The arguments set the initial values of the Event object's Event, Action, Target, and TargetArgs properties. Event codes are declared by the Visio type library (and visconst.bas) and have the prefix "visEvt." The actionCode argument should be visActCodeRunAddon. To create an Event object with the actionCode visActCodeAdvise, use the AddAdvise method instead of Add.

Layers collection: Creates a layer with the specified name and returns a Layer object that represents the new layer.

Layer object: Assigns the specified Shape object to the layer. Use the following syntax:

layerObj.Add(shapeObj,fPreserveMembers)

If the shape is a group and fPreserveMembers is zero, the component shapes of the group are also added to the layer. If fPreserveMembers is non-zero, the component shapes are not also added to the layer.

Styles collection: The Add method has the following syntax and returns the new Style object:

Set styleObj = stylesObj.Add(newStyleName, BasedOnName, fIncludesText, fIncludesLine, fIncludesFill)

The arguments set the initial values of the Style object's Name, BasedOn, IncludesText, IncludesLine, and IncludesFill properties. Pass a null string ("") for the basedOnName argument to base the new style on no style.

See also: Action property, AddAdvise method, AddAt method, AddAtID method, BasedOn property, Event object, Event property, IncludesFill property, IncludesLine property, IncludesText property, Layer property, Name property, Open method, OpenEx method, Target property, TargetArgs property

for Add, Addons, EventList, SelectAll, Selection, Styles

```
'This VBA macro demonstrates adding Document objects such as templates,
'stencils, and drawings to the Documents collection.

'It demonstrates adding pages, masters, layers, styles, events, and
'add-ons to their corresponding collection.

'It also demonstrates drawing a rectangle and grouping, selecting, and
'duplicating shapes.

'Before running this macro, replace "Myfile.vsd" with a valid .vsd file
'and "c:\My Documents\MyAddon.exe" with a valid path and filename

Public Sub Add_Example()

  Dim mastersObj As Visio.Masters
  Dim addonsObj As Visio.Addons
  Dim pagesObj As Visio.Pages
  Dim eventListObj As Visio.EventList

  Dim layersObj As Visio.Layers
  Dim layerObj As Visio.Layer
  Dim stylesObj As Visio.Styles

  Dim documentObj As Visio.Document
  Dim windowObj As Visio.Window
  Dim eventObj As Visio.Event
  Dim masterObj As Visio.Master
  Dim pageObj As Visio.Page
  Dim shapeObj As Visio.Shape
  Dim shapeGrpObj As Visio.Shape, shapeGrpObj2 As Visio.Shape
  Dim styleObj As Visio.Style
  Dim addonObj As Visio.Addon

  'Add a document based on the Basic Template.

  Set documentObj = Documents.Add("Basic Diagram.vst")

  'Add a document based on a drawing (creates a copy of drawing).
  Set documentObj = Documents.Add("Myfile.vsd")

  'Add a document based on a stencil (creates a copy of the stencil).
  Set documentObj = Documents.Add("Basic Shapes.vss")

  'Add a document object based on no template.
  Set documentObj = Documents.Add("")

  Set pagesObj = documentObj.Pages
  'Add a page to the pages collection.
  Set pageObj = pagesObj.Add


  Set mastersObj = documentObj.Masters
  'Add a master to the masters collection.
  Set masterObj = mastersObj.Add
```

```
  Set layersObj = pageObj.Layers
  'Add a layer named "MyLayer" to the page's layers collection.
  Set layerObj = layersObj.Add("MyLayer")

  'Draw 2 rectangles.
  Set shapeObj = pageObj.DrawRectangle(3, 3, 5, 6)
  Set shapeObj = pageObj.DrawRectangle(4, 4, 6, 7)

  'Select the 2 rectangles and group them.
  ActiveWindow.SelectAll
  ActiveWindow.Selection.Group


  'Duplicate the group and set each group as a shape object.
  Set shapeGrpObj = pageObj.Shapes(1)
  shapeGrpObj.Duplicate
  Set shapeGrpObj2 = pageObj.Shapes(2)

  'Add a shape to the layer.
  'This group's component shapes are added to the layer.
  layerObj.Add shapeGrpObj, False

  'Add a shape to the layer.
  'This group's component shapes are added to the layer.
  layerObj.Add shapeGrpObj2, True

  Set stylesObj = documentObj.Styles
  'Add a style named "My FillStyle" to the styles collection.

  'This style is based on the style Red Fill and includes only a Fill style.
  Set styleObj = stylesObj.Add("My FillStyle", "Red Fill", False, False, True)

  'Add a style named "My NoStyle" to the styles collection.
  'This style is based on no style and includes Text, Line, and Fill styles.
  Set styleObj = stylesObj.Add("My NoStyle", "", True, True, True)

  Set eventListObj = documentObj.EventList
  'Add a Before Delete Selection event to the eventList collection.

  'The event will start the Page Layout Wizard.  The wizard takes no
arguments.
  Set eventObj = eventListObj.Add(visEvtCodeBefSelDel, visActCodeRunAddon, _
  "Page Layout Wizard.exe", "")

  'Add the addon "MyAddon.exe" to the Addons collection.
   Set addonsObj = Visio.Addons
   Set addonObj = addonsObj.Add("c:\My Documents\MyAddon.exe")

End Sub
```

Example

AddAdvise method

Applies to: EventList

Summary: Adds an Event object whose action is visActCodeAdvise to the EventList of a source object.

Version: VISIO 4.1

Syntax: objRet = object.**AddAdvise** (eventCode, eventSink, IIDSink, targetArgs)

Element	Description
objRet	The newly added Event object
object	The EventList collection to receive the new Event
eventCode	Which event(s) should generate notifications
eventSink	A reference to an OLE interface on the object to receive event notifications
IIDSink	Reserved for future use. Must be null or "".
targetArgs	The string Visio is to pass to methods of the eventSink interface when it calls them

Remarks: An Event object whose action is visActCodeAdvise causes Visio to send a notification to the calling program when the event occurs. Such events are not persistent--they cannot be stored with a Visio document and must be re-created at run time.

Alternately, an Event object can have the action visActCodeRunAddon, which causes Visio to run the specified add-on when the event occurs. Such events are persistent--they can be stored with a Visio document. To create this kind of event, use the Add method instead of AddAdvise.

In an Event object created with AddAdvise, the source object establishes the scope in which the events will be reported. In general, events are reported for the source object and its subobjects, if any. For example, to receive notification when a particular document is opened, add an Event object for the DocumentOpened event to the EventList of that document. To receive notification when any document is opened in an instance of Visio, add the Event object to the EventList of the Application object.

EventCode identifies the event to report. This is typically a combination of constants. For example, visEvtMod+visEvtCell is the event code for the CellChanged event. Event constants are declared by the Visio type library (and visconst.bas). They are prefixed with "visEvt" and are also listed in event topics in this help file.

EventSink is a reference to an object defined in the calling program. The purpose of the event sink object is to receive event notifications from Visio.

TargetArgs is a string of additional information to be passed back when the event occurs. The calling program can obtain this string by getting the TargetArgs property of the Event object that sent the notification.

To handle notifications, the event sink object must define a method called visEventProc with arguments that correspond to the parameters passed by Visio when it sends the notification. All event notifications from Visio have the following format:

object.visEventProc(eventCode, source, id, sequence, subject, moreInfo)

EventCode indicates the event(s) that occurred. If you prefer, you can provide a visEventProc method for each event or provide a single visEventProc method that receives all notifications and switches internally based on eventCode.

Source is the object whose EventList contains the Event object that triggered the notification.

Id is the unique identifier of the Event object within the EventList (unlike its index, which can change as Events are added or deleted from the EventList). VisEventProc can access the Event object by using source.EventList.ItemFromID(id).

Sequence is the ordinal position of the event with respect to the sequence of events that have occurred in the calling instance of Visio. The first event that occurs in an instance of Visio has a sequence number of 1, the second event 2, and so forth. In some cases sequence can be used in conjunction with the EventInfo property to obtain more information about the event.

Subject, if provided, identifies the object most directly affected by the event. For example, the subject of a ShapeAdded event is a Shape object representing the just added shape, while the subject of a BeforeSelectionDelete event is a Selection object in which the about to be deleted are selected. If the notification does not include a subject, this parameter is set to Nothing.

MoreInfo, if provided, gives additional information about the subject of the event. For many events, it will be a string similar to the command line Visio passes the Add-ons it executes. If the notification does not include additional information, this parameter is set to Nothing.

For details about notification parameters for a particular event, see the particular event topic in this help file.

See also: Add method, Event object, EventInfo property, ItemFromID property, Persistent property, Persistable property, PersistsEvents property

for AddAdvise

```
'This VB program demonstrates creating an instance of the sink object class
'CEventSamp.

  Dim g_Sink As CEventSamp

Private Sub Form_Load()

  Dim docObj As Visio.Document
  Dim eventsObj As Visio.EventList

  ...

  'Create an instance of the CEventSamp class
  'g_Sink is global to the form.
  Set g_Sink = New CEventSamp

  'Create a new drawing
  'An instance of Visio has already been assigned to g_appVisio
  Set docObj = g_appVisio.Documents.Add("")

  'Get the EventList collection of this document.
  Set eventsObj = docObj.EventList

  'Add Event objects that will send notifications.
  'Add an Event object for the DocumentSaved event.
  eventsObj.AddAdvise visEvtCodeDocSave, g_Sink, "", "Document Saved..."

  'Add an Event object for the ShapeDeleted event.
  eventsObj.AddAdvise visEvtCodeShapeDelete, g_Sink, "", "Shape Deleted..."

  'Add an Event object for the PageAdded event.
  eventsObj.AddAdvise (visEvtPage + visEvtAdd), g_Sink, "", "Page Added..."

End Sub
```

Example

AddAt method

Applies to: MenuItems, Menus, StatusBarItems, ToolbarItems, Toolbars

Summary: Creates a new object at the specified index in a collection.

Version: VISIO 4.0

Syntax: objRet = object.**AddAt**(index)

Element	Description
objRet	The new object added to the collection
object	The collection to receive the new object
index	The index at which to add the object

Remarks: If the index is 0, the object is added at the beginning of the collection.

The beginning of a Menus collection is the left-most menu. For example, the File menu is the first menu in the Menus collection for the drawing window context.
The beginning of a MenuItems collection is the top-most menu. For example, the New Window menu item is the first menu item in the MenuItems collection for the Window Menu object.

The beginning of a Toolbars collection is the top-most toolbar.
The beginning of a ToolbarItems collection is the left-most item.

See also: Add method, AddAtID method

for AddAt

```
'This VBA macro demonstrates adding a menu and menu item to the drawing window
'menu set.
'This macro uses the AddAt method to add a menu before the Visio Window menu.

Public Sub AddMenuItem_Example ()

  Dim UIObj As Visio.UIObject
  Dim menuSetsObj As Visio.MenuSets
  Dim menuSetObj As Visio.MenuSet
  Dim menusObj as Visio.Menus
  Dim menuObj As Visio.Menu
  Dim menuItemsObj as Visio.MenuItems
  Dim menuItemObj As Visio.MenuItem

  'Get a UI object that represents Visio's built-in menus
  Set UIObj = Visio.Application.BuiltInMenus

  'Get the MenuSets collection
  Set menuSetsObj = UIObj.MenuSets

  'Get the drawing window menu set
  Set menuSetObj=menuSetsObj.ItemAtId(visUIObjSetDrawing)

  'Get the menus collection
  Set menusObj = menuSetObj.Menus

  'Add a Demo menu before the Window menu.
  Set menuObj = menusObj.AddAt(7)
  menuObj.Caption = "Demo"

  'Get the menuItems collection
  Set menuItemsObj = menuObj.MenuItems

  'Add a menu item to the new Demo menu
  Set menuItemObj = menuItemsObj.Add

  'Set the properties for the new menu item
  menuItemObj.Caption = "Run &ShowArgs"
  menuItemObj.AddOnName = "ShowArgs.EXE"
  menuItemObj.AddOnArgs = "/DVS=Fun"
  menuItemObj.ActionText = "Run ShowArgs"
  menuItemObj.MiniHelp = "Run the ShowArgs application"

  'Tell Visio to use the new UI when the document is active
  ThisDocument.SetCustomMenus uiObj

End Sub
```

Example

AddAtID method

Applies to: AccelTables, MenuSets, StatusBars, ToolbarSets

Summary: Creates a new object for the specified ID in a collection.

Version: VISIO 4.0

Syntax: objRet = object.**AddAtID**(id)

Element	Description
objRet	The new object added to the collection
object	The collection to receive the new object
id	The window context for the new object

Remarks: The ID corresponds to a window or context menu. If the collection already contains an object at the specified ID, AddAtID returns an error.

The following IDs are declared by the Visio type library (and visconst.bas). Not all collections include an object for every possible ID. For a list of valid contexts for a particular collection, see the help topics for that collection.

visUIObjSetNoDocument = 1
visUIObjSetDrawing = 2
visUIObjSetStencil = 3
visUIObjSetShapeSheet = 4
visUIObjSetIcon = 5
visUIObjSetInPlace = 6
visUIObjSetPrintPreview = 7
visUIObjSetText = 8
visUIObjSetCntx_DrawObjSel = 9
visUIObjSetCntx_DrawOleObjSel = 10
visUIObjSetCntx_DrawNoObjSel = 11
visUIObjSetCntx_InPlaceNoObj = 12
visUIObjSetCntx_TextEdit = 13
visUIObjSetCntx_StencilRO = 14
visUIObjSetCntx_ShapeSheet = 15
visUIObjSetCntx_Toolbar = 16
visUIObjSetCntx_Icon = 17
visUIObjSetBinderInPlace = 18
visUIObjSetCntx_Debug = 19
visUIObjSetCntx_StencilRW = 20
visUIObjSetCntx_StencilDocked = 21

See also: Add method, AddAt method

for AddAtID

```
'This VBA macro demonstrates adding a menu and menu item.
'It also uses the visUIObjSetDrawing constant to specify the drawing
'window context.

Public Sub AddMenuItem_Example ()

  Dim UIObj As Visio.UIObject
  Dim menuSetsObj As Visio.MenuSets
  Dim menuSetObj As Visio.MenuSet
  Dim menusObj as Visio.Menus
  Dim menuObj As Visio.Menu
  Dim menuItemsObj as Visio.MenuItems
  Dim menuItemObj As Visio.MenuItem

  'Get a UI object that represents Visio's built-in menus
  Set UIObj = Visio.Application.BuiltInMenus

  'Get the MenuSets collection
  Set menuSetsObj = UIObj.MenuSets

  'Get the drawing window menu set
  Set menuSetObj=menuSetsObj.ItemAtId(visUIObjSetDrawing)

  'Get the menus collection
  Set menusObj = menuSetObj.Menus

  'Add a Demo menu before the Window menu.
  Set menuObj = menusObj.AddAt(7)
  menuObj.Caption = "Demo"

  'Get the menuItems collection
  Set menuItemsObj = menuObj.MenuItems

  'Add a menu item to the new Demo menu
  Set menuItemObj = menuItemsObj.Add

  'Set the properties for the new menu item
  menuItemObj.Caption = "Run &ShowArgs"
  menuItemObj.AddOnName = "ShowArgs.EXE"
  menuItemObj.AddOnArgs = "/DVS=Fun"
  menuItemObj.ActionText = "Run ShowArgs"
  menuItemObj.MiniHelp = "Run the ShowArgs application"

  'Tell Visio to use the new UI when the document is active
  ThisDocument.SetCustomMenus uiObj

End Sub
```

AddCustomField method

Applies to: Characters

Summary: Replaces the text represented by a Characters object with a custom formula field.

Version: VISIO 3.0

Syntax: object.**AddCustomField** strFormula, intFormat

Element	Description
object	The Characters object to receive the new field
strFormula	The formula of the new field
intFormat	The format of the new field

Remarks: The AddCustomField method is similar to using Visio's Field command located on the Insert menu to insert a custom formula field in text. To add any other type of field (not custom), use the AddField method.

The following constants for field formats are declared by the Visio type library (and visconst.bas):

visFmtNumGenNoUnits = 0
visFmtNumGenDefUnits = 1

visFmt0PlNoUnits = 2
visFmt0PlDefUnits = 3
visFmt1PlNoUnits = 4
visFmt1PlDefUnits = 5
visFmt2PlNoUnits = 6
visFmt2PlDefUnits = 7
visFmt3PlNoUnits = 8
visFmt3PlDefUnits = 9

visFmtFeetAndInches = 10
visFmtRadians = 11
visFmtDegrees = 12
visFmtFeetAndInches1Pl = 13
visFmtFeetAndInches2Pl = 14

visFmtFraction1PlNoUnits = 15
visFmtFraction1PlDefUnits = 16
visFmtFraction2PlNoUnits = 17
visFmtFraction2PlDefUnits = 18

visFmtDateShort = 20
visFmtDateLong = 21
visFmtDateMDYY = 22
visFmtDateMMDDYY = 23
visFmtDateMmmDYYYY = 24
visFmtDateMmmmDYYYY = 25
visFmtDateDMYY = 26
visFmtDateDDMMYY = 27
visFmtDateDMMMYYYY = 28

visFmtDateDMMMMYYYY = 29

visFmtTimeGen = 30
visFmtTimeHMM = 31
visFmtTimeHHMM = 32
visFmtTimeHMM24 = 33
visFmtTimeHHMM24 = 34
visFmtTimeHMMAMPM = 35
visFmtTimeHHMMAMPM = 36

visFmtStrNormal = 37
visFmtStrLower = 38
visFmtStrUpper = 39

visFmtJDategggeXmXdXww = 40
visFmtJDateyyyyXmXdXww = 41
visFmtJDategggeXmXdX = 42
visFmtJDateyyyyXmXdX = 43
visFmtJDateyyyymd = 44
visFmtJDateyymmdd = 45
visFmtJDatehmmaxpx = 46
visFmtJDateaxpxhXmmX = 47
visFmtJDatehXmmX = 48

See also: AddField method

for AddCustomField

AddField method

Applies to: Characters

Summary: Replaces the text represented by a Characters object with a new field.

Version: VISIO 3.0

Syntax: object.**AddField** intCategory, intCode, intFormat

Element	Description
object	The Characters object to receive the new field
intCategory	The category for the new field
intCode	The code for the new field
intFormat	The format for the new field

Remarks: Use the AddField method to replace the text represented by a Characters object with a new field of the category, code, and format you specify. The AddField method is similar to using Visio's Field command from the Insert menu to insert a field in text.

You can use AddField to add the following categories of fields:

Date/Time
Document Info
Geometry
Object Info
Page Info

To add a Custom Formula field, use the AddCustomField method.

The following constants for field categories, field codes, and field formats are declared by the Visio type library (and visconst.bas):

visFCatCustom = 0
visFCatDateTime = 1
visFCatDocument = 2
visFCatGeometry = 3
visFCatObject = 4
visFCatPage = 5
visFCatNotes = 6

visFCodeCreateDate = 0
visFCodeCreateTime = 1
visFCodeCurrentDate = 2
visFCodeCurrentTime = 3
visFCodeEditDate = 4
visFCodeEditTime = 5
visFCodePrintDate = 6
visFCodePrintTime = 7

visFCodeCreator = 0
visFCodeDescription = 1
visFCodeDirectory = 2
visFCodeFileName = 3

visFCodeKeyWords = 4
visFCodeSubject = 5
visFCodeTitle = 6
visFCodeManager = 7
visFCideCompany = 8
visFCodeCategory = 9
visFCodeHyperLinkBase = 10

visFCodeWidth = 0
visFCodeHeight = 1
visFCodeAngle = 2

visFCodeData1 = 0
visFCodeData2 = 1
visFCodeData3 = 2
visFCodeObjectID = 3
visFCodeMasterName = 4
visFCodeObjectName = 5
visFCodeObjectType = 6

visFCodeBackgroundName = 0
visFCodePageName = 1
visFCodeNumberOfPages = 2
visFCodePageNumber = 3

visFmtNumGenNoUnits = 0
visFmtNumGenDefUnits = 1

visFmt0PlNoUnits = 2
visFmt0PlDefUnits = 3
visFmt1PlNoUnits = 4
visFmt1PlDefUnits = 5
visFmt2PlNoUnits = 6
visFmt2PlDefUnits = 7
visFmt3PlNoUnits = 8
visFmt3PlDefUnits = 9

visFmtFeetAndInches = 10
visFmtRadians = 11
visFmtDegrees = 12
visFmtFeetAndInches1Pl = 13
visFmtFeetAndInches2Pl = 14

visFmtFraction1PlNoUnits = 15
visFmtFraction1PlDefUnits = 16
visFmtFraction2PlNoUnits = 17
visFmtFraction2PlDefUnits = 18

visFmtDateShort = 20
visFmtDateLong = 21
visFmtDateMDYY = 22
visFmtDateMMDDYY = 23
visFmtDateMmmDYYYY = 24
visFmtDateMmmmDYYYY = 25
visFmtDateDMYY = 26
visFmtDateDDMMYY = 27

visFmtDateDMMMYYYY = 28
visFmtDateDMMMMYYYY = 29

visFmtTimeGen = 30
visFmtTimeHMM = 31
visFmtTimeHHMM = 32
visFmtTimeHMM24 = 33
visFmtTimeHHMM24 = 34
visFmtTimeHMMAMPM = 35
visFmtTimeHHMMAMPM = 36

visFmtStrNormal = 37
visFmtStrLower = 38
visFmtStrUpper = 39

See also: AddCustomField method

for AddField

Example

AddGuide method

Applies to: Page

Summary: Adds a guide to a drawing page.

Version: VISIO 2.0

Syntax: objRet = object.**AddGuide** (guideType, x, y)

Element	Description
objRet	A Shape object that represents the new guide
object	The Page object to receive the new guide
guideType	The type of guide to add
x	The x-coordinate of the guide (for a horizontal guide, this argument is ignored)
y	The y-coordinate of the guide (for a vertical guide, this argument is ignored)

Remarks: The following constants declared by the Visio type library (and visconst.bas) are valid values for the type of guide to add:

visPoint = 1 (Guide point)
visHorz = 2 (Horizontal guide)
visVert = 3 (Vertical guide)

for AddGuide

```
'This VBA macro demonstrates adding a horizontal guide to a page.

Public Sub AddGuide_Example ()

  Dim pagsObj as Visio.Pages
  Dim pagObj as Visio.Page
  Dim shpsObj as Visio.Shapes
  Dim shpObj as Visio.Shape

  'Gets the Pages collection of the ThisDocument object in a VBA project
  Set pagsObj = ThisDocument.Pages

  'Sets the pagObj to the first page
  Set pagObj = pagsObj(1)

  Set shpsObj = pagObj.Shapes

  'Adds a guide to the Shapes collection and sets it as the shpObj
  'Adds a horizontal guide running through the middle of an 8.5x11 page
  Set shpObj = pagObj.AddGuide(visHorz,0,5.5)

End Sub
```

Example

AddHyperlink method

Applies to: Shape

Summary: Adds a Hyperlink object to a Visio shape.

Version: VISIO 5.0

Syntax: objRet = object.**AddHyperlink**

Element	Description
objRet	The Hyperlink object that is returned
object	The shape object to examine

Remarks: The AddHyperlink method is the equivalent of adding a hyperlink to a Shape object from the Insert menu. If a Hyperlink object already exists for the shape then a reference to the existing Hyperlink object is returned.

AddHyperlink will fail if the Shape is a guide; that is, Shape.Type = visTypeGuide.

See also: Hyperlink object, Hyperlink property

for AddHyperlink, CreateURL, Hyperlink, SubAddress

```
' You can paste this routine into a Visio VBA project and step through it
' to see how the Hyperlink object works along with several of its properties
and
' methods.
'
Sub HlinkDemo()

    Dim shpobj As Visio.Shape
    Dim hlinkobj As Visio.Hyperlink
    Dim fCaught As Boolean

    Set shpobj = ActivePage.DrawRectangle(1, 2, 2, 1)

    ' Step 1 - A shape with no Hyperlink should raise an exception when
    '          the Hyperlink property is called.
    '
    On Error GoTo lblCatch
    fCaught = False

    Set hlinkobj = shpobj.Hyperlink

    If Not fCaught Then
        Debug.Print "ERROR - Hyperlink didn't throw an exception"
    End If


    ' Step 2 - Add a hyperlink to a shape.  Do a subsequent add will
    '          return the existing Hyperlink.
    '
    Set hlinkobj = shpobj.AddHyperlink


    ' Step 3 - Demonstrates how to use the Document.HyperlinkBase
    '          to allow relative Hyperlinks.  We print the resulting
    '          URLs to the debug window to show how the relative path is
    '          composed against the base path and the difference
    '          between canonical and non-canonical form.
    '
    ActiveDocument.HyperlinkBase = "C:\My Documents\Visio"
    hlinkobj.Address = "..\Drawing.VSD"
    hlinkobj.SubAddress = "ANCHOR"

    Debug.Print hlinkobj.CreateURL(False)
    Debug.Print hlinkobj.CreateURL(True)


    hlinkobj.Address = "http://www.abc.com/index.htm"
    hlinkobj.SubAddress = "ANCHOR"

    Debug.Print hlinkobj.CreateURL(False)
    Debug.Print hlinkobj.CreateURL(True)


    ' Step 4 - Use Hyperlink.Delete to remove the Hyperlink from the
```

```
    '           Shape.  Hyperlink.Stat will change to reflect the object's
    '           new state.  Note that subsequent requests for the Hyperlink
    '           will fail.
    '
    '           Note - It's always a good idea to release all references
    '           to an object after it's been deleted.
    '
    hlinkobj.Delete

    If visStatDeleted <> hlinkobj.Stat Then
        Debug.Print "ERROR:Hyperlink.Delete didn't change stat to
visShapeDeleted
    End If

    On Error GoTo lblCatch
    fCaught = False

    Set hlinkobj = shpobj.Hyperlink

    If Not fCaught Then
        Debug.Print "ERROR:Calling Shape.Hyperlink on deleted link didn't
raise an exception."
    End If


    ' Step 5 - Deleted Hyperlink will raise an exception if any properties
    '           are accessed.
    '
    On Error GoTo 0
    hlinkobj.Address = "http://www.abc.com"

    Exit Sub

lblCatch:
    Debug.Print "Error was thrown : " & Err.Description
    fCaught = True
    Resume Next
End Sub
```

AddNamedRow method

Applies to: Shape

Summary: Adds a row with the indicated name to the indicated ShapeSheet section at the specified position.

Version: VISIO 4.0

Syntax: retVal = object.**AddNamedRow** (section, rowName, rowTag)

Element	Description
retVal	The row number of the new row
object	The Shape object to receive the new row
section	The section in which the row is to be added
rowName	The name of the new row
rowTag	The type of row to be added

Remarks: Named rows can only be added to the Custom Properties (visSectionProp) and User-defined Cells sections (visSectionUser). The row tag should always be 0.

The cells in the newly added row are accessed by passing the row number returned by AddNamedRow to CellsSRC. Cells in the row can also be accessed using the row's name with the Cells property. For details about cell references and cells in named rows, search Visio online Help for "User.Row" or "Prop.Row."

See also: Cells property, CellsSRC property, RowName property

for AddNamedRow

Example

AddonArgs property

Applies to: MenuItem, StatusBarItem, ToolbarItem

Summary: Gets or sets the argument string to be sent to the add-on associated with a MenuItem, StatusBarItem, or ToolbarItem object.

Version: VISIO 4.0

Syntax: object.**AddonArgs** = argsStr
argsStr = object.**AddonArgs**

Element	Description
object	The object that starts the add-on
argsStr	The argument string to be passed to the add-on

Remarks: The arguments string can be anything appropriate for the add-on. Note, however, that the arguments are packaged together with other information into a command string, which cannot exceed 127 characters. For best results, limit arguments to 50 characters.

The object's AddOnName property indicates the add-on to which the arguments are sent.

See also: AddonName property

for AddonArgs

```
'This VBA macro demonstrates adding a menu and menu item to the drawing window
'menu set.
'It also sets the menu item's properties such as Caption,
'AddOnName, AddOnArgs, ActionText, and MiniHelp.

Public Sub AddMenuItem_Example ()

  Dim UIObj As Visio.UIObject
  Dim menuSetsObj As Visio.MenuSets
  Dim menuSetObj As Visio.MenuSet
  Dim menusObj as Visio.Menus
  Dim menuObj As Visio.Menu
  Dim menuItemsObj as Visio.MenuItems
  Dim menuItemObj As Visio.MenuItem

  'Get a UI object that represents Visio's built-in menus
  Set UIObj = Visio.Application.BuiltInMenus

  'Get the MenuSets collection
  Set menuSetsObj = UIObj.MenuSets

  'Get the drawing window menu set
  Set menuSetObj=menuSetsObj.ItemAtId(visUIObjSetDrawing)

  'Get the menus collection
  Set menusObj = menuSetObj.Menus

  'Add a Demo menu before the Window menu.
  Set menuObj = menusObj.AddAt(7)
  menuObj.Caption = "Demo"

  'Get the menuItems collection
  Set menuItemsObj = menuObj.MenuItems

  'Add a menu item to the new Demo menu
  Set menuItemObj = menuItemsObj.Add

  'Set the properties for the new menu item
  menuItemObj.Caption = "Run &ShowArgs"
  menuItemObj.AddOnName = "ShowArgs.EXE"
  menuItemObj.AddOnArgs = "/DVS=Fun"
  menuItemObj.ActionText = "Run ShowArgs"
  menuItemObj.MiniHelp = "Run the ShowArgs application"

  'Tell Visio to use the new UI when the document is active
  ThisDocument.SetCustomMenus uiObj

End Sub
```

Example

AddonName property

Applies to: MenuItem, StatusBarItem, ToolbarItem

Summary: Gets or sets the name of an add-on associated with a MenuItem, StatusBarItem, or ToolbarItem object.

Version: VISIO 4.0

Syntax: object.**AddonName** = addonStr
addonStr = object.**AddonName**

Element	Description
object	The object that runs the add-on or executes the code
addonStr	The name of the add-on to be run or VBA code to be executed

Remarks: If the AddonName property is set, Visio ignores the object's CmdNum property.

Use the AddonArgs property to specify arguments to send to the add-on when it is run.

Using AddonName, you can also assign VBA code to a MenuItem, StatusBarItem or ToolbarItem that will run when the item is selected. When an item whose AddonName is set is selected, Visio will first ask the VBA project of the active document to parse the string that AddonName equals. If VBA successfully parses the string, Visio will tell VBA to execute the string. Using this technique, you can cause a menu, statusbar or toolbar item to run a VBA macro or procedure, show a VBA form, log information to VBA's immediate window, and so forth. See ExecuteLine for examples.

If VBA says the string does not parse, then Visio will run the add-on named by AddonName. If there is no such add-on, Visio does nothing.

See also: AddonArgs property, CmdNum property, ExecuteLine method, ParseLine method

for AddonName

```
'This VBA macro demonstrates adding a menu and menu item to the drawing window
'menu set.
'It also sets the menu item's properties such as Caption,
'AddOnName, AddOnArgs, ActionText, and MiniHelp.

Public Sub AddMenuItem_Example ()

  Dim UIObj As Visio.UIObject
  Dim menuSetsObj As Visio.MenuSets
  Dim menuSetObj As Visio.MenuSet
  Dim menusObj as Visio.Menus
  Dim menuObj As Visio.Menu
  Dim menuItemsObj as Visio.MenuItems
  Dim menuItemObj As Visio.MenuItem

  'Get a UI object that represents Visio's built-in menus
  Set UIObj = Visio.Application.BuiltInMenus

  'Get the MenuSets collection
  Set menuSetsObj = UIObj.MenuSets

  'Get the drawing window menu set
  Set menuSetObj=menuSetsObj.ItemAtId(visUIObjSetDrawing)

  'Get the menus collection
  Set menusObj = menuSetObj.Menus

  'Add a Demo menu before the Window menu.
  Set menuObj = menusObj.AddAt(7)
  menuObj.Caption = "Demo"

  'Get the menuItems collection
  Set menuItemsObj = menuObj.MenuItems

  'Add a menu item to the new Demo menu
  Set menuItemObj = menuItemsObj.Add

  'Set the properties for the new menu item
  menuItemObj.Caption = "Run &ShowArgs"
  menuItemObj.AddOnName = "ShowArgs.EXE"
  menuItemObj.AddOnArgs = "/DVS=Fun"
  menuItemObj.ActionText = "Run ShowArgs"
  menuItemObj.MiniHelp = "Run the ShowArgs application"

  'Tell Visio to use the new UI when the document is active
  ThisDocument.SetCustomMenus uiObj

End Sub
```

AddonPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for add-ons.

Version: VISIO 4.0

Syntax: strRet = object.**AddonPaths**
object.**AddonPaths** = pathsStr

Element	Description
strRet	A text string containing a list of folders
object	An Application object
pathsStr	A text string containing a list of folders

Remarks: The string passed to and received from AddonPaths is the same string shown in Visio's File Paths dialog (accessible from the Tools/Options dialog). This same string is stored in Visio's profile (.ini) file (appObj.ProfileName) in the entry whose key is "AddonsPath."

To indicate more than one folder, separate individual items in the path string with semicolons. If a path is not fully qualified, Visio looks for the folder in the folder that contains the Visio program files (appObj.Path).

For example, if Visio's executable file is installed in c:\Visio, and AddonPaths is "Add-ons;d:\Add-ons", Visio looks for add-ons in both c:\Visio\Add-ons and d:\Add-ons.

When Visio looks for add-ons, it will look in all paths named in AddonPaths plus in all sub-folders of those paths. Also, the fact that a path is named in AddonPaths does not imply the path actually exists. If you pass AddonPaths to the EnumDirectories method, it will return a complete list of fully qualified paths that Visio will actually look in.

See also: DrawingPaths property, FilterPaths property, HelpPaths property, StartupPaths property, StencilPaths property, TemplatePaths property, ProfileName property, Path property, EnumDirectories method

for AddonPaths

Example

Addons property

Applies to: <Global>, Application

Summary: Returns the Addons collection of an Application object.

Version: VISIO 4.0

Syntax: objRet = object.**Addons**

Element	Description
objRet	The Addons collection of the Application object
object	The Application object that owns the collection

Remarks: The Addons collection includes an Addon object for each add-on in the folders specified by the AddonPaths property and for each add-on that is added dynamically to the collection by other add-ons.

See also: Add method, Addons object, AddonPaths property

for Addons

*Add Method

Example

Address property

Applies to: Hyperlink

Summary: Returns or sets the Address in a shape's Hyperlink object that represents the address to which a shape's hyperlink will navigate.

Version: VISIO 5.0

Syntax: strRet = object.**Address**
object.**Address** = stringExpression

Element	Description
strRet	The current value of the field
object	The object that has or gets the value
stringExpression	The new value for the field

Remarks: Setting a Hyperlink object's Address is equivalent to entering information in the Link To File or URL field of the Hyperlink dialog box, accessed from the Insert menu. This is also equivalent to setting the result of the Address cell in the shape's hyperlink row.

A shape's hyperlink Address can be a DOS, UNC, or URL path, for example, C:\Drawings\MyDrawing.VSD, \\Server\Shared\MyDrawing.VSD, or http://www.visio.com, respectively.

If the Address is relative, for example, "..\Drawing.vsd", then it is composed against the HyperlinkBase property, if supplied, or the hyperlink's document path. If the document is not saved, the hyperlink is undefined.

If the Address is empty, then it is assumed to be the containing document. In this case, the SubAddress property should contain the name of the drawing page where the hyperlink will navigate.

See also: ExtraInfo property, SubAddress property, Frame property, HyperlinkBase property

for Address

*AddToFavorites method

Example

AddRow method

Applies to: Shape

Summary: Adds a row to the indicated ShapeSheet section at the specified position.

Version: VISIO 2.0

Syntax: retVal = object.**AddRow** (section, row, tag)

Element	Description
retVal	The row number of the row that was added
object	The Shape object to receive the new row
section	The section in which to add the row
row	The position at which to add the row
tag	The type of row to add

Remarks: If the section does not exist, the section is created with a blank row. New cells in new rows are initialized with default formulas if applicable. Otherwise, a program must include statements to set the formulas for the new cells. If the row cannot be added, retVal is set to visRowNone. If a row exists at that position, a blank row is inserted. The row constants declared by the Visio type library (and visconst.bas) serve as base positions at which a section's rows begin. Add offsets to these constants to specify the first row and beyond, for example, visRowFirst + 0, visRowFirst + 1, and so on.

The tag argument specifies the type of row to add to a Geometry section. For all other rows, use 0 (zero) for the tag argument.

To add rows at the end of a section, pass the constant visRowLast for the row argument. The value returned is the actual row index, or visRowNone if an error occurs.

If you try to add a row to a character, tab, or paragraph properties section, an error is generated.

Visio declares the following row constants:

```
visRowFirst = 0                 'first logical row in any section
visRowLast = -2                 'last logical row in any section
visRowNone = -1                 'unspecified row
visRowXFormOut = 1
visRowXFormIn = 1
visRowLine = 2
visRowFill = 3
visRowXForm1D = 4
visRowEvent = 5
visRowLayerMem = 6
visRowGuide = 7
visRowStyle = 8
visRowForeign = 9
visRowPage = 10
visRowText = 11
visRowTextXForm = 12
```

visRowAlign = 14
visRowLock = 15
visRowData123 = 16
visRowMisc = 17
visRowRulerGrid = 18
visRowComponent = 0
visRowVertex = 1
visRowMember = 0
visRowCharacter = 0
visRowParagraph = 0
visRowTab = 0
visRowScratch = 0
visRowExport = 0
visRowField = 0
visRowControl = 0
visRowAction = 0
visRowLayer = 0
visRowUser = 0
visRowProp = 0
visRowFormat = 0

Visio declares the following constants for Geometry section row tags:

visTagBase = 130
visTagComponent = 137
visTagMoveTo = 138
visTagLineTo = 139
visTagArcTo = 140
visTagEllipticalArcTo = 144
visTagSplineBegin = 165
visTagSplineSpan = 166
visTagInvalid = -1

See also: AddRows method, AddSection method, CellsSRC property, DeleteRow method

for AddRow

*AddSection Method

AddRows method

Applies to: Shape

Summary: Adds the indicated number of rows to the indicated ShapeSheet section at the specified position.

Version: VISIO 4.0

Syntax: retVal = object.**AddRows** (section, row, tag, count)

Element	Description
retVal	The row number of the first row that was added
object	The Shape object to receive the new rows
section	The section in which to add the rows
row	The position at which to add the rows
tag	The type of rows to add
count	The number of rows to add

Remarks: If the section does not exist, it is created with blank rows. New cells in new rows are initialized with default formulas, if applicable. Otherwise, a program must include statements to set the formulas for the new cells. If the rows cannot be added, retVal is set to visRowNone. If a row exists at the indicated position, blank rows are inserted. The row constants declared by the Visio type library (and visconst.bas) serve as base positions at which a section's rows begin. Add offsets to these constants to specify the first row and beyond, for example, visRowFirst + 0, visRowFirst + 1, and so on.

The tag argument specifies the type of rows to add to a Geometry section. For all other rows, use 0 (zero) for the tag argument.

To add rows at the end of a section, pass the constant visRowLast for the row argument. The value returned is the actual row index, or visRowNone if an error occurs.

If you try to add rows to a character, tab, or paragraph properties section, an error is generated.

See also: AddRow method

for AddRows

Example

AddSection method

Applies to: Shape

Summary: Adds a new section to a ShapeSheet.

Version: VISIO 2.0

Syntax: intRet = object.**AddSection** (section)

Element	Description
intRet	The index of the section that was added
object	The Shape object to receive the new section
section	The type of section to add

Remarks: The AddSection method is typically used to add Geometry sections to a shape. You can also use AddSection to add Scratch, Control, and Connection sections. If the section cannot be added, intRet is set to visSectionNone.

A new section has no rows. Use the AddRow method to add rows to the new section.

There are two ways to specify a Geometry section's index. The first is to use the visSectionFirstComponent or visSectionLastComponent constants, which specify the first and last Geometry sections. The second is to use the visSectionFirstComponent constant plus an index. For example, a shape with three Geometry sections would be represented as follows:

Section	Constant index
1	visSectionFirstComponent + 0
2	visSectionFirstComponent + 1
3	visSectionFirstComponent + 2

Notice that the index you are adding is always one less than the Geometry section's number (for example, you add 2 to get the 3rd section). To insert a Geometry section between two others, specify the higher section's index (for example, to insert a new section between sections 2 and 3, specify visSectionFirstComponent + 2.) Specifying an index that doesn't exist is the same as specifying visSectionLastComponent.

The returned value is the zero-based index of the added section added to visSectionFirstComponent. In the example above, AddSection(visSectionFirstComponent +1) would return visSectionFirstComponent + 1.

If you attempt to add a non-Geometry section that already exists for the shape, an error is generated.

The following constants for sections are declared by the Visio type libary (and visconst.bas):

```
visSectionFirst = 0    'first logical section
visSectionLast = 252    'last logical section
visSectionObject = 1
visSectionMember = 2
```

visSectionCharacter = 3
visSectionParagraph = 4
visSectionTab = 5
visSectionScratch = 6
visSectionExport = 7
visSectionTextField = 8
visSectionControls = 9
visSectionFirstComponent = 10
visSectionLastComponent = 239
visSectionAction = 240
visSectionLayer = 241
visSectionUser = 242
visSectionProp = 243
visSectionNone = 255 'unspecified logical section

See also: AddRow method, CellsSRC property, DeleteSection method

for AddRow, AddSection, Cells, CellsSRC, DeleteRow, DeleteSection, Formula, RowType

```
'This VBA macro demonstrates working with ShapeSheet sections and rows.
'It bows, or curves, the lines of a rectangle by changing the lines to
'arcs.

Public Sub BuildShape_Example ()

  Dim DrawPageObj As Visio.Page
  Dim shpObj as Visio.Shape
  Dim cellObj As Visio.Cell
  Dim BowCell As String, BowFormula As String
  Dim idxInnerRect As Integer, I As Integer

  'Set the value of the BowCell string
  BowCell = "Scratch.X1"

  'Set the value of the BowFormula string
  BowFormula = "=Min(Width, Height) / 5"

  Set DrawPageObj = ActivePage

  'If there isn't an active page, set the page object to the first page of
  'the active document.
  If DrawPageObj Is Nothing Then
    Set DrawPageObj = ActiveDocument.Pages(1)
  End If

  'Draw a rectangle on the active page.
  Set shpObj = DrawPageObj.DrawRectangle(1, 5, 5, 1)

  'Add a scratch section, add a row to the scratch, and then
  'place the value of BowFormula into Scratch.X1.

  shpObj.AddSection visSectionScratch                 'Add scratch section
  shpObj.AddRow visSectionScratch, visRowScratch, 0   'Insert a new row

  'Set the cell object to the Scratch.X1 and set formula
  Set cellObj = shpObj.Cells(BowCell)                 'Get Scratch.X1
  cellObj.Formula = BowFormula                        'Set up offset for the
arc

  'Bow in or curve the original rectangle's lines by changing each row
  'to an arc and entering the bow value.

  For I = 1 To 4
    shpObj.RowType(visSectionFirstComponent, visRowVertex + I) = visTagArcTo
    Set cellObj = shpObj.CellsSRC(visSectionFirstComponent, visRowVertex + I,
2)
    cellObj.Formula = "-" & BowCell
  Next I

  'Create an inner rectangle. Add a new geometry (component)
  'section and four line segments within it.  Then draw the rectangle
  'inside the now-bowed edges of the previous rectangle.
```

```
  idxInnerRect = visSectionFirstComponent + 1    'Inner rectangle section
index
  shpObj.AddSection idxInnerRect                 'Add inner rectangle section
  shpObj.AddSection idxInnerRect + 1             'Add another rectangle
section
  shpObj.DeleteSection idxInnerRect + 1          'Delete the previous section

  shpObj.AddRow idxInnerRect, visRowVertex, visTagComponent     'Add row
  shpObj.AddRow idxInnerRect, visRowVertex + 1, visTagMoveTo    'Add row

  For I = 1 To 4
    shpObj.AddRow idxInnerRect, visRowLast, visTagLineTo        'Add 4 rows
  Next I

  'Draw rectangle
  Set cellObj = shpObj.CellsSRC(idxInnerRect, 1, 0)   'Start
    cellObj.Formula = "Width * 0 + " & BowCell        'X

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 1, 1)
    cellObj.Formula = "Height * 0 + " & BowCell       'Y

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 2, 0)   'Bottom line
    cellObj.Formula = "Width * 1 - " & BowCell        'X

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 2, 1)
    cellObj.Formula = "Height * 0 + " & BowCell       'Y

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 3, 0)   'Right line
    cellObj.Formula = "Width * 1 - " & BowCell        'X

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 3, 1)
    cellObj.Formula = "Height * 1 - " & BowCell       'Y

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 4, 0)   'Top line
    cellObj.Formula = "Width * 0 + " & BowCell        'X

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 4, 1)
    cellObj.Formula = "Height * 1 - " & BowCell       'Y

  Set cellObj = shpObj.CellsSRC(idxInnerRect, 5, 0)   'Left line
    cellObj.Formula = "Geometry2.X1"                  'X
  Set cellObj = shpObj.CellsSRC(idxInnerRect, 5, 1)
    cellObj.Formula = "Geometry2.Y1"                  'Y

End Sub
```

Example

AddToFavorites method

Applies to: Hyperlink

Summary: Adds a shortcut to the hyperlink address in the presently registered Favorites folder.

Version: VISIO 5.0

Syntax: object.**AddToFavorites** [stringExpression]

Element	Description
object	The object to make a shortcut for
stringExpression	The title to assign to the new shortcut; optional

Remarks: The argument to AddToFavorites is optional.

If a string is not supplied, the Hyperlink's Description property will be used as the new favorite's title. If the Description property is empty, the shortcut will be given a generic title such as Favorite1.

The optional stringExpression argument can specify the full path for the favorites file, for example, "C:\TEMP\My Favorite.URL", or a path relative to the favorites folder. See the example for more information.

From VB or VBA, a call to AddToFavorites can take either of these two forms:

```
object.AddToFavorites "SomeString"
object.AddToFavorites
```

From C/C++, if the string is supplied, pass a variant of type VT_BSTR. Visio will assign that string as the title of the shortcut. If the string is not supplied, pass a variant of type VT_EMPTY, or of type VT_ERROR and HRESULT DISP_E_PARAMNOTFOUND.

for Address, AddToFavorites

```
Sub TestAddToFavorites()
    ' Create a blank Visio document and paste this function into
    ' ThisDocument object via the Visual Basic Editor. Both Example 3 and
    ' Example 4 require that you replace the path names with existing
    ' path names
    '

    Dim shp As Visio.Shape
    Dim hlink As Visio.Hyperlink

    ' Create a new shape to add the hyperlink to.
    '
    Set shp = ActivePage.DrawRectangle(1, 2, 2, 1)
    Set hlink = shp.AddHyperlink

    hlink.Description = "Visio Home Page"
    hlink.Address = "http://www.visio.com"

    ' Example 1 - Default Name
    '
    hlink.AddToFavorites

    ' Example 2 - Specify a different favorites name.  Note
    '             you don't need to specify the URL extension.
    '
    hlink.AddToFavorites "New Favorite Name"

    ' Example 3 - Specify a different favorites path.
    '
    hlink.AddToFavorites "C:\TEMP\My Favorite.URL"

    ' Example 4 - Relative path to favorites folder.  Note that
    '             URL extension is added automatically.
    '
    hlink.AddToFavorites ".\Companies\Visio Home Page"
End Sub
```

Example

AddToGroup method

Applies to: Window

Summary: Adds the selected shapes to the selected group.

Version: VISIO 2.0

Syntax: object.**AddToGroup**

Element	Description
object	The Window object that owns the selected group and shapes

Remarks: The current selection must contain both the shapes to add and the group to add them to. The group must be the primary selection or the only group in the selection.

See also: Group method, RemoveFromGroup method, Ungroup method

for AddToGroup

```
'This VBA macro demonstrates adding shapes to a group. It assumes that
'at least one shape and only one group is selected on the page in the active
'window.

Public Sub Grouping_Example ()

  'Adds the individual selected shapes to the selected group
  ActiveWindow.AddToGroup

End Sub
```

AfterModal event

Applies to: Application

Summary: The event that occurs after Visio leaves a modal state.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtAfterModal (&H1040)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events in the instance
subject	The Application object in which this event occurred
moreInfo	Nothing for this event

Remarks: Visio becomes modal when it displays a dialog box. A modal instance of Visio does not handle Automation calls. The BeforeModal event indicates that the instance is about to become modal, and the AfterModal event indicates that the instance is no longer modal.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, BeforeModal event, Event object, EventList object

for AfterModal

AlertResponse property

Applies to: Application

Summary: Determines whether Visio shows alerts and dialogs to the user.

Version: VISIO 4.1

Syntax: intRet = object.**AlertResponse**
object.**AlertResponse** = intExpression

Element	Description
intRet	0 to display alerts to the user, or the value of the default response to supply
object	The object that has or gets the setting
intExpression	The value of the default response, or 0 to allow the user to respond

Remarks: Certain operations, such as closing a document with unsaved modifications, cause Visio to display an alert or dialog requesting the user to supply a response such as OK, Yes, No or Cancel. To prevent Visio from displaying alerts or dialogs when a program performs such actions, set AlertResponse to the response you want from the alert or dialog. Visio will not display the alert or dialog; instead, Visio will behave as if the user responded to the alert or dialog with the value of AlertResponse.

If AlertResponse is 0 (its default value), alerts and dialogs are displayed.

The values you supply for AlertResponse correspond to the standard Windows values IDOK, IDCANCEL, and so forth. Common values you might supply include:

IDOK (1)
IDCANCEL (2)
IDABORT (3)
IDRETRY (4)
IDIGNORE (5)
IDYES (6)
IDNO (7)

In most cases you should restore AlertResponse to its previous value when you've completed the operation.

See also: ShowProgress property

for AlertResponse

AlignName property

Applies to: Master

Summary: Returns or sets the position of a master name in a stencil window.

Version: VISIO 2.0

Syntax: intRet = object.**AlignName**
object.**AlignName** = intNewAlignment

Element	Description
intRet	Returns the current alignment of the master's name
object	The Master object whose name is to be aligned
intNewAlignment	The new alignment for the master's name

Remarks: Use AlignName to change the alignment of the master name in relation to the icon. The following constants declared by the Visio type library (and visconst.bas) show the possible alignment values:

visLeft = 1
visCenter = 2
visRight = 3

for AlignName

Alt property

Applies to: AccelItem

Summary: Gets or sets whether the Alt key is a modifier for the AccelItem object.

Version: VISIO 4.0

Syntax: object.**Alt** = intExpression
intRet = object.**Alt**

Element	Description
intRet	True (-1) if Alt modifies Key in the accelerator; otherwise False (0)
object	An AccelItem object
intExpression	True (non-zero) if Alt modifies Key in the accelerator; otherwise False (0)

See also: Control property, Key property, Shift property

for Alt

AppActivated event

Applies to: Application

Summary: The event that occurs after an instance of Visio becomes active.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtAppActivate (&H1001)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Application object in which this event occurred
moreInfo	Nothing for this event

Remarks: The AppActivated event indicates that the instance of Visio has become the active application on the Windows desktop--the instance is now the frontmost application. AppActivated is a different event than AppObjectActivated, which occurs after an instance of Visio becomes the active Visio--the instance of Visio that is retrieved by the GetObject function in a Visual Basic program.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Active property, Add method, AddAdvise method, AppDeactivated event, AppObjectActivated event, Event object, EventList object

for AppActivated

AppDeactivated event

Applies to: Application

Summary: The event that occurs after an instance of Visio becomes inactive.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtAppDectivate (&H1002)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Application object in which this event occurred
moreInfo	Nothing for this event

Remarks: The AppDeactivated event indicates that the instance of Visio is no longer the active application on the Windows desktop--the instance is no longer the frontmost application. AppDeactivated is a different event than AppObjectDeactivated, which occurs after an instance of Visio ceases to be the active Visio--the instance of Visio that is retrieved by GetObject.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Active property, Add method, AddAdvise method, AppActivated event, AppObjectDeactivated event, Event object, EventList object

for AppDeactivated

Example

Application property

Applies to: <Global>, Addon, Addons, Application, Cell, Characters, Color, Colors, Connect, Connects, Curve, Document, Documents, Event, EventList, Font, Fonts, Hyperlink, Layer, Layers, Master, Masters, OLEObject, OLEObjects, Page, Pages, Path, Paths, Selection, Shape, Shapes, Style, Styles, Window, Windows

Summary: Returns the instance of Visio that contains the indicated object.

Version: VISIO 2.0

Syntax: objRet = object.**Application**

Element	Description
objRet	The Application object that contains the indicated object
object	The object for which to retrieve the Application object

Remarks: Every Visio object is associated with a running instance of Visio. To retrieve the instance of Visio that is associated with a particular object, use the Application property.

for Application

*SaveToFile Method

AppObjectActivated event

Applies to: Application

Summary: The event that occurs after an instance of Visio becomes the active Visio.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtObjActivate (&H1004)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Application object in which this event occurred
moreInfo	Nothing for this event

Remarks: The AppObjectActivated event indicates that the instance of Visio has become the active Visio--the instance that will be retrieved by the GetObject function in a Visual Basic program. AppObjectActivated is a different event than AppActivated, which occurs after an instance of Visio becomes the active application on the Windows desktop.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, AppObjectDeactivated event, AppActivated event, Event object, EventList object

for AppObjectActivated

AppObjectDeactivated event

Applies to: Application

Summary: The event that occurs after an instance of Visio ceases to be the active Visio.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtObjDeactivate (&H1008)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Application object in which this event occurred
moreInfo	Nothing for this event

Remarks: The AppObjectDeactivated event indicates that the instance of Visio is no longer the active Visio--the instance of Visio that is retrieved by the GetObject function in a Visual Basic program. AppObjectDeactivated is a different event than AppDeactivated, which occurs after an instance of Visio ceases to be the active application--the frontmost application on the Windows desktop.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, AppObjectActivated event, AppDeactivated event, Event object, EventList object

for AppObjectDeactivated

AreaIU property

Applies to: Shape

Summary: Returns the area of the object in internal units (square inches).

Version: VISIO 4.0

Syntax: retVal = object.**AreaIU**

Element	Description
retVal	The area of the object in internal units
object	The Shape object to examine

See also: LengthIU property, BoundingBox method

for ArealU

Example

Arrange method

Applies to: Windows

Summary: Arranges the windows in the indicated Windows collection.

Version: VISIO 2.0

Syntax: object.**Arrange**

Element	Description
object	The collection of windows to arrange

Remarks: This method is equivalent to choosing the Tile command located on the Window menu in Visio. The active window remains active.

for Arrange

```
*Activate Method
```

Example

Attributes property

Applies to: Font, ShapeData

Summary: For a ShapeData object, returns the Attributes collection of the shape associated with that object. For a Font object, returns the attributes of the font.

Version: VISIO 3.0 TECH

Syntax: objRet = object.**Attributes**
intRet = object.**Attributes**

Element	Description
objRet	The Attributes collection of a ShapeData object
intRet	The attributes of a Font object
object	The ShapeData or Font object to examine

Remarks: Use the Attributes property to retrieve a list of attributes defined for a given Visio shape. Before retrieving the Attributes collection, you must first use the PutShape method to associate the Shape object with a ShapeData object.

The value returned for a Font object will be a combination of the following:

visFontRaster = 16
visFontDevice = 32
visFontScalable = 64
visFont0Alias = 128

If a font is marked as the font 0 alias, it is used instead of font 0 (the default font). This is used in some localized versions of Visio and is controlled through entries in Visio's initialization settings.

See also: Attributes object, PutShape method

for Attributes

*PutShape Method

Example

Background property

Applies to: Page

Summary: Determines whether the indicated page is a background page.

Version: VISIO 2.0

Syntax: retVal = object.**Background**
object.**Background** = intExpression

Element	Description
retVal	TRUE if the page is a background page; otherwise FALSE
object	The Page object to examine
intExpression	0 to declare page as a forground page; non-zero to declare it a background page

See also: BackPage property

Example for Background

```
'This VBA macro demonstrates iterating through a document's pages and
'determining whether a page is a foreground or background page. It displays
'the foreground pages in a list box.

'To run this macro, first insert a user form with a list box.

Public Sub IteratePages()

Dim pagsObj As Visio.Pages      ' Pages collection
Dim pagObj As Visio.Page        ' current page in collection
Dim i As Integer                ' current index into collection

    ' Retrieve the pages collection
    Set pagsObj = ThisDocument.Pages

    ' Make sure the listbox is cleared
    UserForm1.ListBox1.Clear

    ' Iterate through the collection
    For i = 1 To pagsObj.Count
        ' Retrieve the page object at the current index
        Set pagObj = pagsObj(i)

        ' Check whether the current page is a background page
        ' Display the names of all the foreground pages
        If pagObj.Background = False Then
            UserForm1.ListBox1.AddItem pagObj.Name
        End If
    Next i

    ' Display the user form
    UserForm1.Show

End Sub
```

BackPage property

Applies to: Page

Summary: Returns or sets the background page of a page.

Version: VISIO 2.0

Syntax: objVariantRet = object.**BackPage**
object.**BackPage** = stringVariant

Element	Description
objVariantRet	A Page object that represents the background page returned in a variant
object	The Page object that has or gets the background page
stringVariant	A variant to which is assigned a string that names the new background page

Remarks: If the indicated page has no background, BackPage returns an empty variant. Otherwise the returned variant refers to the Page object that is the background page of the indicated page.

To assign a background page to a page, set that page's BackPage property to the name of the page you want to assign as a background. To cause a page to have no background page, pass an empty string to BackPage.

[Note: In earlier versions of Visio (through version 4.1), BackPage returned an object (as opposed to a variant of type object) and BackPage accepted a string (as opposed to a variant of type string). Due to changes in Automation support tools, it became necessary to change the property to accept and return variants. For backward compatibility, BackPageAsObj and BackPageFromName were added. BackPageAsObj and BackPageFromName have the same signatures and occupy the same vtble slots as did the prior version of BackPage.]

See also: Background property, BackPageAsObj property, BackPageFromName property

Example for BackPage

BackPageAsObj property

Applies to: Page

Summary: Returns the background page of a page.

Version: VISIO 4.5

Syntax: objRet = object.**BackPageAsObj**

Element	Description
objRet	A Page object that represents the background page or nothing
object	The Page object that has the background page

Remarks: If the indicated page has no background, BackPageAsObj returns nothing. Otherwise the Page object that is the background page of the indicated page is returned.

[Note: In earlier versions of Visio (through version 4.1), PageObj.BackPage returned an object. Due to changes in Automation support tools, it became necessary to change the BackPage property to return a variant of type object. For backwards compatibility, BackPageAsObj was added. It behaves like the BackPage property used to, and occupies the same slot in the vtble as the old property. If you're developing new code, you'll likely find very few occasions when you must use BackPageAsObj.]

See also: Background property, BackPage property, BackPageFromName property

Example for BackPageAsObj

BackPageFromName property

Applies to: Page

Summary: Sets the background page of a page.

Version: VISIO 4.5

Syntax: object.**BackPageFromName** = stringExpression

Element	Description
object	The Page object that gets the background page
stringExpression	The name of the new background page

Remarks: To assign a background page to a page, set that page's BackPageFromName property to the name of the page you want to assign as a background. To cause a page to have no background page, pass an empty string to BackPageFromName.

[Note: In earlier versions of Visio (through version 4.1), PageObj.BackPage accepted a string. Due to changes in Automation support tools, it became necessary to change the BackPage property to return a variant of type string. For backwards compatibility, BackPageFromName was added. It behaves like the BackPage property used to, and occupies the same slot in the vtble as the old property. If you're developing new code, you'll likely find very few occasions when you must use BackPageFromName.]

See also: Background property, BackPage property, BackPageAsObj property

Example for BackPageFromName

BasedOn property

Applies to: Style

Summary: Gets or sets the style that the indicated Style object is based on.

Version: VISIO 4.0

Syntax: strVal = object.**BasedOn**
object.**BasedOn** = styleName

Element	Description
strVal	The name of the current style
object	The Style object that has or gets the style
styleName	The name of the new style

Remarks: To base a style on no style, set BasedOn to a null string ("").

See also: FillBasedOn property, LineBasedOn property, TextBasedOn property

Example for BasedOn

BeforeDocumentClose event

Applies to: Application, Document, Documents

Summary: The event that occurs before a document is closed.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtDel+visEvtDoc (&H4002)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that is about to close
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for BeforeDocumentClose

BeforeDocumentSave event

Applies to: Application, Document, Documents

Summary: The event that occurs just before a Visio document is saved.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeBefDocSave (7)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document about to be saved
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, DocumentSaved event, DocumentSavedAs event, BeforeDocumentSaveAs event, Event object, EventList object

Example for BeforeDocumentSave

BeforeDocumentSaveAs event

Applies to: Application, Document, Documents

Summary: The event that occurs just before a Visio document is saved with Save As.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeBefDocSaveAs (8)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document about to be saved
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, DocumentSaved event, DocumentSavedAs event, BeforeDocumentSave event, Event object, EventList object

Example for BeforeDocumentSaveAs

BeforeMasterDelete event

Applies to: Application, Document, Documents, Master, Masters

Summary: The event that occurs before a master is deleted from a document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtDel+visEvtMaster (&H4008)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Master that is about to be deleted
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for BeforeMasterDelete

BeforeModal event

Applies to: Application

Summary: The event that occurs before an instance of Visio enters a modal state.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtBeforeModal (&H1020)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Application object in which this event is about to occur
moreInfo	Nothing for this event

Remarks: Visio becomes modal when it displays a dialog box. A modal instance of Visio does not handle Automation calls. The BeforeModal event indicates that the instance is about to become modal, and the AfterModal event indicates that the instance is no longer modal.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, AfterModal event, Application object, Event object, EventList object

Example for BeforeModal

BeforePageDelete event

Applies to: Application, Document, Documents, Page, Pages

Summary: The event that occurs before a page is deleted.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtDel+visEvtPage (&H4010)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Page that is about to be deleted
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for BeforePageDelete

BeforeQuit event

Applies to: Application

Summary: The event that occurs before an instance of Visio terminates.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtBeforeQuit (&H1010)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Application object in which this event is about to occur
moreInfo	Nothing for this event

Remarks: Note: Code in the VBA project of a Visio document will never see BeforeQuit. This is because the project is a property of a document, and all documents get closed before BeforeQuit is sent. VBA projects should clean up in response to BeforeDocumentClose rather than in response to BeforeQuit.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for BeforeQuit

BeforeSelectionDelete event

Applies to: Application, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs before the entries in a selection are deleted.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeBefSelDel (901)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Selection whose entries are about to be deleted
moreInfo	Nothing for this event

Remarks: A Shape object can serve as the source object for BeforeSelectionDelete iff the shape's Type property is visTypeGroup (2) or visTypePage(1).

The BeforeSelectionDelete event indicates that selected shapes are about to be deleted. This notification is sent whether or not any of the shapes are locked; however, locked shapes will not actually be deleted. To find out if a shape is locked against deletion, check the value of its LockDelete cell.

The BeforeSelectionDelete and BeforeShapeDelete events are similar in that they both fire before shape(s) are deleted. They differ in how they behave when a single operation deletes several shapes. Suppose a cut operation deletes 3 shapes. BeforeShapeDelete will fire 3 times and the respective subject objects will be the 3 to be deleted shapes. BeforeSelectionDelete will fire once and its subject object will be a Selection object in which the 3 to be deleted shapes are selected.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, BeforeShapeDelete event, BeforeWindowSelDelete event, ShapesDeleted event, Event object, EventList object

Example for BeforeSelectionDelete

BeforeShapeDelete event

Applies to: Application, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs before a shape is deleted.

Version: VISIO 4.5

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtDel+visEvtShape (&H4040)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Shape that is about to be deleted
moreInfo	Nothing for this event

Remarks: A Shape object can serve as the source object for BeforeShapeDelete iff the shape's Type property is visTypeGroup (2) or visTypePage(1).

The BeforeSelectionDelete and BeforeShapeDelete events are similar in that they both fire before shape(s) are deleted. They differ in how they behave when a single operation deletes several shapes. Suppose a cut operation deletes 3 shapes. BeforeShapeDelete will fire 3 times and the respective subject objects will be the 3 to be deleted shapes. BeforeSelectionDelete will fire once and its subject object will be a Selection object in which the 3 to be deleted shapes are selected.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source BeforeShapeDelete using the AddAdvise method. In addition, BeforeShapeDelete is included in the event set of all the objects in the Applies to list except the Document object. For those objects you can use VBA Dim WithEvents variables to sink BeforeShapeDelete. For performance considerations, the Document object's event set does not include BeforeShapeDelete. To sink BeforeShapeDelete from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, BeforeSelectionDelete event, BeforeWindowSelDelete event, ShapesDeleted event, Event object, EventList object

Example for BeforeShapeDelete

BeforeStyleDelete event

Applies to: Application, Document, Documents, Style, Styles

Summary: The event that occurs before a style is deleted.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtDel+visEvtStyle (&H4004)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Style that is about to be deleted
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for BeforeStyleDelete

BeforeWindowClose event

Applies to: Application, Window, Windows

Summary: The event that occurs before a window is closed.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtDel+visEvtWindow (&H4001)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Window that is about to close
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for BeforeWindowClose

BeforeWindowPageTurn event

Applies to: Application, Window, Windows

Summary: The event that occurs before a window is about to show a different page in itself.

Version: VISIO 4.5

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeBefWinPageTurn (703)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Window that is about to show a different page
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, WindowTurnedToPage event

Example for BeforeWindowPageTurn

BeforeWindowSelDelete event

Applies to: Application, Window, Windows

Summary: The event that occurs before the shapes in the selection of a window are deleted.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeBefWinSelDel (702)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Window of the selection whose entries are about to be deleted
moreInfo	Nothing for this event

Remarks: This event will fire if user interactions cause shapes in a window to be deleted. This event will not fire if a program causes shapes in a window to be deleted using methods such as winobj.cut.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, BeforeShapeDelete event, BeforeSelectionDelete event, ShapesDeleted event, Event object, EventList object

Example for BeforeWindowSelDelete

Begin property

Applies to: Characters

Summary: Returns or sets the beginning index of the indicated Characters object, which represents a range of text in a shape.

Version: VISIO 3.0

Syntax: intRet = object.**Begin**
object.**Begin** = intExpression

Element	Description
intRet	The current beginning index of the Characters object
object	The Characters object that has or gets the index
intExpression	The new beginning index of the Characters object

Remarks: The Begin property determines the beginning of the text range represented by a Characters object. The value of the Begin property is an index that represents the boundary between two characters, similar to an insertion point in text. Like selected text in a drawing window, a Characters object represents the sequence of characters that are affected by subsequent actions, such as the Cut or Copy method. When you first retrieve a Characters object, its current text range includes all of the shape's text. You can change the text range by setting the Character object's Begin and End properties. Changing the text range of a Characters object has no effect on the text of the corresponding shape.

The Begin property can have a value from 0 to the value of CharCount for the corresponding shape. An index of 0 places Begin before the first character in the shape's text. An index of CharCount places Begin after the last character in the shape's text. If you specify a value less than 0, Visio sets Begin to 0. If you specify a value that would place Begin inside the expanded characters of a field, Visio sets Begin to the start of the field.

The value of Begin must always be less than or equal to the value of End. If you attempt to set Begin to a value greater than End, Visio sets both Begin and End to the value specified for Begin.

See also: End property

Example for Begin

BeginTransaction method

Applies to: ShapeData

Summary: Begins a transaction for the ShapeData object.

Version: VISIO 3.0 TECH

Syntax: object.**BeginTransaction**

Element	Description
object	The ShapeData object that owns the transaction

Remarks: If you need to perform multiple operations on the ShapeData object and the objects it provides (Attribute and Entity objects), you can increase the speed of the operations by using BeginTransaction and EndTransaction. Call BeginTransaction before you start your work, and EndTransaction when all operations are complete. This forces all database operations to be performed only when the EndTransaction call is received.

See also: EndTransaction method

Example for BeginTransaction

BinaryData property

Applies to: Entity

Summary: Specifies the binary data contained in an Entity object.

Version: VISIO 3.0 TECH

Syntax: strRet = object.**BinaryData**
object.**BinaryData** = expression

Element	Description
strRet	The current binary data returned as a string
object	The Entity object that owns the binary data
expression	The new binary data

Remarks: If an Entity has a Group of 1004, then it contains binary data. This data is set and retrieved using the string data type. In Visual Basic you must use fixed length strings so that null terminators are ignored (ASCII 0). An Entity containing binary information is limited to a buffer of 127 bytes.

Example for BinaryData

BinaryLength property

Applies to: Entity

Summary: Specifies the length of the binary data contained in an Entity object.

Version: VISIO 3.0 TECH

Syntax: retVal = object.**BinaryLength**

Element	Description
retVal	The number of bytes of data contained in the Entity
object	The Entity object to examine

Remarks: If an Entity object's Group property is set to 1004, it contains binary data. When allocating a buffer for this data, use the BinaryLength property to determine the size of buffer to allocate.

Example for BinaryLength

Blue property

Applies to: Color

Summary: Gets or sets the intensity of the blue component of a Color object.

Version: VISIO 4.0

Syntax: intRet = object.**Blue**
object.**Blue** = intVal

Element	Description
intRet	The current value of the color's blue component
object	The Color object that has or gets the component
intVal	The new value of the color's blue component

Remarks: The Blue property can be a value from 0 to 255.

A color is represented by red, green, and blue components. It also has a flag that indicates how the color is to be used. These correspond to members of the Windows PALETTEENTRY data structure. For details, search the Windows SDK online help for "PALETTEENTRY."

See also: Flags property, Green property, PaletteEntry property, Red property

Example for Blue

BottomMargin property

Applies to: Document

Summary: Specifies the bottom margin for printing a document's pages.

Version: VISIO 4.0

Syntax: retVal = object.**BottomMargin**(units)
object.**BottomMargin**(units) = newValue

Element	Description
retVal	The margin value expressed in the given units
object	The Document object that has or gets the margin value
units	The units to use when retrieving or setting the margin value
newValue	The new margin value

Remarks: This property corresponds to the Bottom Margin control in Visio's Page Setup dialog box. To see the Page Setup dialog box, choose Page Setup from the File menu.

Units can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property.

See also: LeftMargin property, RightMargin property, TopMargin property, Result property

Example for BottomMargin

BoundingBox method

Applies to: Master, Page, Selection, Shape

Summary: Returns a rectangle that tightly encloses a shape, or the shapes of a page, master or selection.

Version: VISIO 4.5

Syntax: object.**BoundingBox** flags, left, bottom, right, top

Element	Description
object	The page, master, group or selection whose bounding box is to be retrieved
flags	Flags that influence the bounding box that is returned
left	Returns x-coordinate of left edge of bounding box
bottom	Returns y-coordinate of bottom edge of bounding box
right	Returns x-coordinate of right edge of bounding box
top	Returns y-coordinate of top edge of bounding box

Remarks: ShapeObj.BoundingBox returns a rectangle that tightly encloses the shape.

Obj.BoundingBox, where obj is a Page, Master or Selection object, returns a rectangle that tightly encloses the page's, master's or selection's shapes (and their sub-shapes).

The bounding rectangle determined for an individual shape depends on its Type property:
- visTypePage: Equivalent to Page.BoundingBox or Master.BoundingBox.
- visTypeGroup: Rectangle that tightly encloses the group's shapes (and the shapes within the group).
- visTypeShape: Determined rectangle depends on flags. See below.
- visTypeForeignObject: Determined rectangle depends on flags. See below.
- visTypeGuide: See below.

If BoundingBox returns an error, or if it is asked to return the rectangle enclosing zero shapes, the rectangle returned will be { left: 0, bottom: 0, right: -1, top: -1 }.

BoundingBox will ignore the extents of shapes of type visTypeGuide in the result it returns unless flags includes visBBoxIncludeGuides (&H1000). If guide extents are requested, then only the x positions of vertical guides and the y positions of horizontal guides will contribute to the rectangle that is returned. If only vertical guides are reported on, the returned y extent will be { 0, -1 }. If only horizontal guides are reported on, the reported x extent will be { 0, -1 }.

In all other cases the rectangle returned will have left <= right and bottom <= top. The numbers returned are in internal units (inches). If flags includes visBBoxDrawingCoords (&H2000), the returned numbers will be in the drawing coordinate system of the page or master whose shapes are being considered. Otherwise the returned numbers will be drawing units in the local coordinate system of the parent of the considered shapes.

Flags has several bits that control exactly which bounding box (or boxes) BoundingBox will retrieve for each shape of type visTypeShape or visTypeForeignObject it considers. If more than one of the bits described below is set, the rectangle determined for the shape will cover all rectangles implied by the bits.

- visBBoxUprightWH (&H1): The "upright width/height" box of a shape is the smallest rectangle parallel to the local coordinate system of the shape's parent that encloses the shape's width/height box. If the shape is not rotated, its upright width/height box and its width/height box are the same. Note that paths in the shape's geometry needn't and often don't lie entirely within the shape's width/height box.

- visBBoxUprightText (&H2): The "upright text" box of a shape is the smallest rectangle parallel to the local coordinate system of the shape's parent that encloses the shape's text.

- visBBoxExtents (&H4): The "extents" box of a shape is the smallest rectangle parallel to the local coordinate system of the shape's parent that encloses the paths stroked by the shape's geometry. This may be larger or smaller than the shape's upright width/height box. The extents box determined for a shape of type visTypeForeignObject will equal that shape's upright width/height box.

Note that the extents rectangle is with respect to the center of the shape's strokes. It does not take into account the width of the strokes. Nor does the rectangle include any area covered by shadows or line end markers. Visio does not at present expose a means to determine a shape's "black bits" box, i.e. the extents box adjusted to account for stroke widths, shadows and line ends.

Note also that a shape may have control points or connection points that lie outside any of the bounding rectangles reported by the shape. Positions of control points and connection points can be determined by querying results of the shape's cells.

See also: AreaIU property, LengthIU property, HitTest property

Example for BoundingBox

Example

BringForward method

Applies to: Selection, Shape

Summary: Brings the shape or selected shapes forward one position in the z-order.

Version: VISIO 2.0

Syntax: object.**BringForward**

Element	Description
object	The Shape or Selection object to bring forward

See also: BringToFront method, SendBackward method, SendToBack method

Example for BringForward

```
*SendBackward Method
```

Example

BringToFront method

Applies to: Selection, Shape

Summary: Brings the shape or selected shapes to the front of the z-order.

Version: VISIO 2.0

Syntax: object.**BringToFront**

Element	Description
object	The Shape or Selection object to bring to the front

See also: BringForward method, SendBackward method, SendToBack method

Example for BringToFront

*SendBackward Method

Example

BuiltInMenus property

Applies to: Application

Summary: Returns a UI object that represents a copy of the built-in Visio menus and accelerators.

Version: VISIO 4.0

Syntax: objRet = object.**BuiltInMenus**

Element	Description
objRet	A UI object that represents Visio's built-in menus and accelerators
object	An Application object

Remarks: You can use the BuiltInMenus property to obtain a UI object and modify its menus and accelerators. You can then use the SetCustomMenus method of an Application or Document object to substitute your customized menus and accelerators for the built-in Visio menus and accelerators.

You can also use the SaveToFile method of the UI object to store its menus in a file and reload them as custom menus by setting the CustomMenusFile property of an Application or Document object.

See also: BuiltInToolbars property, CustomMenusFile property, SaveToFile method, SetCustomMenus method, UI Object object

Example for BuiltInMenus

```
*ActionText Property
```

Example

BuiltInToolbars property

Applies to: Application

Summary: Returns a UI object that represents a copy of the built-in Visio toolbars and status bars.

Version: VISIO 4.0

Syntax: objRet = object.**BuiltInToolbars**(fWhichToolbars)

Element	Description
objRet	A UI object that represents Visio's built-in toolbars and status bars
object	An Application object
fWhichToolbars	Ignored as of Visio 5.0

Remarks: You can use the BuiltInToolbars property to obtain a UI object and modify its toolbars and status bars. You can then use the SetCustomToolbars method of an Application or Document object to substitute your customized toolbars and status bars for the built-in Visio toolbars and status bars.

You can also use the SaveToFile method of the UI object to store its toolbars in a file and reload them as custom toolbars by setting the CustomToolbarsFile property of an Application or Document object.

Prior to Visio 5.0, the argument to this method designated which type ("flavor") of toolbar to get (MSOffice or LotusSS). Visio 5.0 no longer supports the notion of toolbar flavors. The argument is now ignored.

See also: BuiltInMenus property, CustomToolbarsFile property, Flavor property, SaveToFile method, SetCustomToolbars method, UI Object object

Example for BuiltInToolbars, CntrlType, IconFileName, ItemAtID, Priority, SetCustomToolbars, ToolbarItems, Toolbars, ToolbarSets

```
'This VBA macro demonstrates retrieving a copy of the built-in Visio
'toolbars, adding a toolbar button, and setting the button properties.

Public Sub AddToolbarButton_Example()

  Dim uiObj As Visio.UIObject
  Dim toolbarSetObj As Visio.ToolbarSet
  Dim toolbarItemsObj As Visio.ToolbarItems
  Dim objNewToolbarItem As Visio.ToolbarItem

  'Get the UI object for the copy of the MSOffice toolbars
  Set uiObj = Visio.Application.BuiltInToolbars(visToolBarMSOffice)

  'Get the Drawing Window toolbarsets
  'NOTE: Use ItemAtID to get the toolbarset.
  'Using uiObj.ToolbarSets(visUIObjSetDrawing) will not work.
  Set toolbarSetObj = uiObj.ToolbarSets.ItemAtID(visUIObjSetDrawing)

  'Get the ToolbarItems collection
  Set toolbarItemsObj = toolbarSetObj.Toolbars(0).ToolbarItems

  'Add a new button in the first position
  Set objNewToolbarItem = toolbarItemsObj.AddAt(0)

  'Set the properties for the new toolbar button
  objNewToolbarItem.ActionText = "Run Chart Shape Wizard"
  objNewToolbarItem.AddOnName = "Chart Shape Wizard.exe"
  objNewToolbarItem.CntrlType = visCtrlTypeBUTTON
  objNewToolbarItem.Priority = 1

  'Set the toolbar button icon
  objNewToolbarItem.IconFileName "dvs.ico"

  'Tell Visio to actually use the new custom UI
  ThisDocument.SetCustomToolbars uiObj

End Sub
```

Example

Caption property

Applies to: Menu, MenuItem, MenuSet, StatusBar, Toolbar, ToolbarSet

Summary: Gets or sets the caption for the indicated object.

Version: VISIO 4.0

Syntax: object.**Caption** = stringVal
stringVal = object.**Caption**

Element	Description
object	The object that has or gets the caption
stringVal	The caption string of the object

Remarks: The Caption property of a Menu object determines the menu title, including the & that indicates a hotkey. For example: "&File". The Caption property of a MenuItem object determines the menu text for that item, including the hotkey and accelerator key. For example: "&New...Ctrl+N".

The stringVal argument can include the escape characters \t and \a. For example:
To insert a tab in the string and align text in columns on menus, use the \t character.
To align the text that follows it flush right on the menu or menu bar, use the \a character.
To display a double quotation mark on the menu, use two in the string: "".
To display an ampersand on the menu, use two in the string: &&.

Note that the accelerator key in the Caption property is part of the menu item's text. To define an accelerator, you set properties of an AccelItem object whose CmdNum property value is the same as that of the MenuItem object.

Visio does not use the Caption property of a MenuSet, StatusBar, or ToolbarSet object.

See also: AccelItem object, ActionText property, CmdNum property

Example for Caption

```
'This VBA macro demonstrates adding a menu and menu item to the drawing window
'menu set.
'It also sets the menu item's properties such as Caption,
'AddOnName, AddOnArgs, ActionText, and MiniHelp.

Public Sub AddMenuItem_Example ()

  Dim UIObj As Visio.UIObject
  Dim menuSetsObj As Visio.MenuSets
  Dim menuSetObj As Visio.MenuSet
  Dim menusObj as Visio.Menus
  Dim menuObj As Visio.Menu
  Dim menuItemsObj as Visio.MenuItems
  Dim menuItemObj As Visio.MenuItem

  'Get a UI object that represents Visio's built-in menus
  Set UIObj = Visio.Application.BuiltInMenus

  'Get the MenuSets collection
  Set menuSetsObj = UIObj.MenuSets

  'Get the drawing window menu set
  Set menuSetObj=menuSetsObj.ItemAtId(visUIObjSetDrawing)

  'Get the menus collection
  Set menusObj = menuSetObj.Menus

  'Add a Demo menu before the Window menu.
  Set menuObj = menusObj.AddAt(7)
  menuObj.Caption = "Demo"

  'Get the menuItems collection
  Set menuItemsObj = menuObj.MenuItems

  'Add a menu item to the new Demo menu
  Set menuItemObj = menuItemsObj.Add

  'Set the properties for the new menu item
  menuItemObj.Caption = "Run &ShowArgs"
  menuItemObj.AddOnName = "ShowArgs.EXE"
  menuItemObj.AddOnArgs = "/DVS=Fun"
  menuItemObj.ActionText = "Run ShowArgs"
  menuItemObj.MiniHelp = "Run the ShowArgs application"

  'Tell Visio to use the new UI when the document is active
  ThisDocument.SetCustomMenus uiObj

End Sub
```

Example

Category property

Applies to: Document

Summary: Returns or sets the value of the Category field in a document's properties.

Version: VISIO 5.0

Syntax: strRet = object.**Category**
object.**Category** = stringExpression

Element	Description
strRet	The current value of the field
object	The document object that has or gets the value
stringExpression	The new value of the field

Remarks: Setting the Category property is equivalent to entering information in the Category field in the Properties dialog box, accessed from the File menu.

See also: Description property, Keywords property, Subject property, Title property, Company property, Manager property, HyperlinkBase property

Example for Category

*Document Property

Example

CellChanged event

Applies to: Application, Cell, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs after the value changes in a cell in a Visio document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtCell (&H2800)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Cell whose result just changed
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source CellChanged using the AddAdvise method. In addition, CellChanged is included in the event set of all the objects in the Applies to list except the Document object. For those objects, you can use VBA Dim WithEvents variables to sink CellChanged. For performance considerations, the Document object's event set does not include CellChanged. To sink CellChanged from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, FormulaChanged event

Example for CellChanged

```
'This class module demonstrates defining a sink class called ShapeSink that
'declares the object variable m_shpObj using the WithEvents keyword. It
contains
'a procedure, InitWith, that assigns a particular Shape object, aShape, to
'm_shpObj. The class module also contains an event handler for the CellChanged
'event, which can be fired by a Shape object—in this case, the Shape object
'represented by aShape.

Dim WithEvents m_shpObj As Visio.Shape

Public Sub InitWith(ByVal aShape As Visio.Shape)
  Set m_shpObj = aShape
End Sub

Private Sub m_shpObj_CellChanged(ByVal Cell As Visio.IVCell)
  Debug.Print Cell.Shape.Name & " " & Cell.Name & " changed to =" &
Cell.Formula

End Sub
```

CellExists property

Applies to: Shape

Summary: Returns TRUE if the indicated ShapeSheet cell exists in the scope of the search.

Version: VISIO 4.0

Syntax: intRet = object.**CellExists**(stringExpression, fExistsLocally)

Element	Description
intRet	0 if cell doesn't exist, -1 if it does
object	The Shape object to examine
stringExpression	The name of the ShapeSheet cell to search for
fExistsLocally	The scope of the search

Remarks: The stringExpression argument must specify a cell name. To search for a cell by section, row, and cell index, use the CellsSRCExists property.

If fExistsLocally is FALSE (0), CellExists returns TRUE if the object either contains or inherits the cell. If fExistsLocally is TRUE (non-zero), CellExists returns TRUE only if the object contains the cell locally; if the cell is inherited, CellExists returns FALSE.

See also: Cells property, CellsSRC property, CellsSRCExists property, RowExists property, SectionExists property

Example for CellExists

Example

Cells property

Applies to: Shape, Style

Summary: Returns a Cell object that represents the specified ShapeSheet cell.

Version: VISIO 2.0

Syntax: objRet = object.**Cells** (stringExpression)

Element	Description
objRet	A Cell object that represents the requested cell
object	The Shape or Style object that owns the cell
stringExpression	The name of a cell in a ShapeSheet

Remarks: Cells("somestring") does not raise an exception if "somestring" does not name an actual cell. Subsequent methods invoked on the returned object will fail. You can determine if a cell with name "somestring" exists using CellExists.

Click on the ShapeSheet cells See also entry below for information about particular cells.

One technique that can be used to determine which name to pass to Cells is to use Visio's ShapeSheet window. Open a ShapeSheet window showing the cell you want the name of. Click on another cell, then click in the formula box under the toolbar, then click in the cell whose name you want. The name will appear in the formula box. The Developing Visio Solutions book also has information about cell naming conventions.

The cells in a shape's User-defined Cells and Custom Properties sections belong to rows whose names have been assigned by the user or a program. Cells in named rows can be accessed using the Cells property.

For example, if "MyRowsName" is the name of a row in shape's User-defined Cells section, the zero'th (value) cell in this row can be accessed using this statement:

```
cellobj = shpobj.cells("User.MyRowsName")
```

The prompt cell in MyRowsName could be accessed using this statement:

```
cellobj = shpobj.cells("User.MyRowsName.Prompt")
```

Next, assume that MyRowsName is in the Custom Properties section instead of the User-defined Cells section. The zero'th (value) cell would be accessed using this statement:

```
cellobj = shpobj.cells("Prop.MyRowsName")
```

Other cells in the row could be accessed using this statement:

```
cellobj = shpobj.cells("Prop.MyRowsName.xxx")
```

where xxx is one of: Label, Prompt, SortKey, Type, Format, Invisible, or Ask.

See also: AddNamedRow method, CellExists property, CellsSRC property, RowName property,

ShapeSheet Cells

Example for Cells

```
*AddSection Method
```

CellsC property

Applies to: Layer

Summary: Returns a Cell object that represents the specified ShapeSheet cell.

Version: VISIO 4.0

Syntax: objRet = object.**CellsC**(column)

Element	Description
objRet	A Cell object that represents the requested cell
object	The Layer object that owns the cell
column	The cell index of the cell to get

Remarks: Column can be one of the following values:

visLayerName = 0
visLayerPassword = 1
visLayerColor = 2
visLayerStatus = 3
visLayerVisible = 4
visLayerPrint = 5
visLayerActive = 6
visLayerLock = 7
visLayerSnap = 8
visLayerGlue = 9

Example for CellsC

Example

CellsSRC property

Applies to: Shape

Summary: Returns a Cell object that represents the ShapeSheet cell identified by section, row, and column indices.

Version: VISIO 2.0

Syntax: objRet = object.**CellsSRC** (section, row, column)

Element	Description
objRet	A Cell object that represents the requested cell
object	The Shape object that owns the cell
section	The cell's section index
row	The cell's row index
column	The cell's column index

Remarks: To access any shape formula by its section, row, and column indices, use the CellsSRC property. Constants for section, row, and column indices are declared by the Visio type library (and visconst.bas).

CellsSRC(s,r,c) does not raise an exception if indices s, r and c do not identify an actual cell. Subsequent methods invoked on the returned object will fail. You can determine if a cell with indices s, r and c exists using CellsSRCExists.

CellsSRC is typically used to iterate through the cells in a section or row. To retrieve a single cell, use the Cells property and specify a cell name. For example: Set celObj = Cells("PinX").

See also: AddRow method, AddSection method, Cells property, CellsSRCExists property

Example for CellsSRC

```
*AddSection Method
```

CellsSRCExists property

Applies to: Shape

Summary: Returns TRUE if the indicated ShapeSheet cell exists in the scope of the search.

Version: VISIO 4.0

Syntax: intRet = object.**CellsSRCExists**(section,row,column,fExistsLocally)

Element	Description
intRet	0 if cell doesn't exist, -1 if it does
object	The Shape object to examine
section	The cell's section index
row	The cell's row index
column	The cell's column index
fExistsLocally	The scope of the search

Remarks: Constants for section, row, and column indices are declared by the Visio type library (and visconst.bas).

If fExistsLocally is FALSE (0), CellsSRCExists returns TRUE if the object either contains or inherits the cell. If fExistsLocally is TRUE (non-zero), CellsSRCExists returns TRUE only if the object contains the cell locally; if the cell is inherited, CellsSRCExists returns FALSE.

To search for a cell by name, use the CellExists property.

See also: Cells property, CellExists property, CellsSRC property, RowExists property, SectionExists property

Example for CellsSRCExists

CenterDrawing method

Applies to: Master, Page, Shape

Summary: Centers a page's, master's or group's shapes with respect to the extent of the page, master or group.

Version: VISIO 4.0

Syntax: object.**CenterDrawing**

Element	Description
object	The page, master, or group that contains the shapes to center

Remarks: Centering shapes does not change their positions relative to each other.

Example for CenterDrawing

Example

Characters property

Applies to: Shape

Summary: Returns a Characters object that represents the text of the indicated shape.

Version: VISIO 3.0

Syntax: objRet = object.**Characters**

Element	Description
objRet	A Characters object that represents the shape's text
object	The Shape object that owns the text

See also: Characters object, Text property

Example for Characters

```
*Text Property
```

CharCount property

Applies to: Characters, Shape

Summary: Returns the number of characters in the indicated object.

Version: VISIO 3.0

Syntax: intRet = object.**CharCount**

Element	Description
intRet	The number of characters in the object's text
object	The Characters or Shape object that contains the text

Remarks: For a Shape object, CharCount returns the number of characters in the shape's text. For a Characters object, CharCount returns the number of characters in the text range represented by that object.

The value returned by CharCount includes the expanded number of characters for any fields in the object's text. For example, if the text contains a field that displays the filename of a drawing, CharCount includes the number of characters in the filename, rather than the 4-character escape sequence used to represent a field in the Text property of a Shape object.

See also: Text property

Example for CharCount

CharProps property

Applies to: Characters

Summary: Sets the indicated character property of a Characters object to a new value.

Version: VISIO 3.0

Syntax: object.**CharProps**(intWhichProp) = intExpression

Element	Description
object	The Characters object that gets the new value
intWhichProp	The property to set
intExpression	The new value for the property

Remarks: Depending on the extent of the text range and the format, setting the CharProps property may cause rows to be added or removed from the Character section of the ShapeSheet. To retrieve information about existing formats, use the CharPropsRow property.

The values of the intWhichProp argument correspond to named cells in the Character section of the ShapeSheet. Constants for intWhichProp are declared by the Visio type library (and visconst.bas):

visCharacterFont = 0
visCharacterColor = 1
visCharacterStyle = 2
visCharacterCase = 3
visCharacterPos = 4
visCharacterSize = 7

For information about types of formatting, see the corresponding Character section cell in Visio online help.

See also: CharPropsRow property, ParaProps property, ShapeSheet Cells

Example for CharProps

CharPropsRow property

Applies to: Characters

Summary: Returns the index of the row in the Character section of a ShapeSheet that contains character formatting information for a Characters object.

Version: VISIO 3.0

Syntax: intRet = object.**CharPropsRow**(bias)

Element	Description
intRet	The index of the row that defines the Characters object's format
object	The Characters object to examine
bias	The direction of the search

Remarks: If the formatting of the Characters object is represented by more than one row in the Character section of the ShapeSheet, CharPropsRow returns -1. If the Characters object represents an insertion point rather than a sequence of characters (that is, if its Begin and End properties return the same value), use the bias argument to determine which row index to return:

visBiasLeft = 1
visBiasRight = 2
visBiasLetVisioChoose = 0

Specify visBiasLeft for the row that covers character formatting for the character to the left of the insertion point, or visBiasRight for the row that covers character formatting for the character to the right of the insertion point.

See also: CharProps property, ParaPropsRow property, TabPropsRow property

Example for CharPropsRow

CharSet property

Applies to: Font

Summary: Returns the Windows character set for a Font object.

Version: VISIO 4.0

Syntax: intRet = object.**CharSet**

Element	Description
intRet	The character set code for the object
object	The Font object to examine

Remarks: The Windows character set specifies character mapping for a font. The possible values of the CharSet property correspond to those of the lfCharSet member of the Windows LOGFONT data structure. For details, search the Windows SDK online help for LOGFONT.

See also: PitchAndFamily property

Example for CharSet

Example

ClassID property

Applies to: OLEObject, Shape

Summary: Returns the class ID string of a shape representing an ActiveX control or an embedded or linked OLE object.

Version: VISIO 4.5

Syntax: strRet = object.**ClassID**

Element	Description
strRet	The class id of the OLE object represented by the shape
object	The Shape object to examine

Remarks: ClassID will raise an exception if the shape doesn't represent an ActiveX control or OLE 2.0 embedded or linked object. A shape represents an ActiveX control or an OLE 2.0 embedded or linked object if the visTypeIsOLE2 bit (&H8000) is set in the value returned by shpObj.ForeignType.

ClassID returns a string of the form:

{2287DC42-B167-11CE-88E9-002AFDDD917}

This identifies the application that services the object. It might, for example, identify an embedded object on a Visio page as being an Excel object.

After using a shape's Object property to obtain an Automation interface on the object the shape represents, you might want to obtain the shape's ClassID or ProgID in order to determine the methods and properties provided by the interface.

See also: ForeignType property, Object property, ProgID property

Example for ClassID

```
*ProgID property
```

Example

ClearCustomMenus method

Applies to: Application, Document

Summary: Restores the built-in Visio menus in place of any custom menus that are in effect.

Version: VISIO 4.0

Syntax: object.**ClearCustomMenus**

Element	Description
object	The Application or Document object that is using the custom menus

Remarks: Calling ClearCustomMenus on an object without custom menus has no effect.

Example for ClearCustomMenus, ClearCustomToolbars

```
'This VBA macro demonstrates clearing custom menus and toolbars for the
'ThisDocument and Application object.

Public Sub RestoreBuiltInUI_Example()

    ' Tell Visio to use the built-in menus
    ThisDocument.ClearCustomMenus
    Visio.Application.ClearCustomMenus

    ' Tell Visio to use the built-in toolbars
    ThisDocument.ClearCustomToolbars
    Visio.Application.ClearCustomToolbars

End Sub
```

Example

ClearCustomToolbars method

Applies to: Application, Document

Summary: Restores the built-in Visio toolbars in place of any custom toolbars that are in effect.

Version: VISIO 4.0

Syntax: object.**ClearCustomToolbars**

Element	Description
object	The Application or Document object that is using the custom toolbars

Remarks: Calling ClearCustomToolbars on an object without custom toolbars has no effect.

Example for ClearCustomToolbars

```
*ClearCustomMenus Method
```

Example

Close method

Applies to: Document, Master, Window

Summary: Closes the indicated window, document or master.

Version: VISIO 2.0

Syntax: object.**Close**

Element	Description
object	The Window, Document or Master object to close

Remarks: If the indicated window is the only window open for a document and the document contains unsaved changes, an alert appears asking if you want to save the document. AlertResponse can be used to prevent the alert from appearing.

If you close a docked stencil window, only that window is closed. However, if you close a drawing window that contains docked stencils, the docked stencil window is also closed.

MasterObj.Close should be used after opening a master for editing using masterObj.Open. Close will push any changes made to the master while it was open to instances of the master. For more details, see the Open method remarks.

See also: Open method, AlertResponse property

Example for Close

```
'This VBA macro demonstrates closing a window.

Public Sub CloseShapeSheets_Example ()

  Dim I As Integer
  I = Windows.Count

  While I <> 0
    If Windows(I).Type = visSheet Then
      Windows(I).Close
      I = Windows.Count
    Else
      I = I - 1
    End If
  Wend

End Sub
```

Closed property

Applies to: Curve, Path

Summary: Returns true if the object's starting point is coincident with its ending point.

Version: VISIO 5.0

Syntax: intRet = object.**Closed**

Element	Description
intRet	True (-1) if Path or Curve is closed; otherwise False (0)
object	The Path or Curve object to examine

Remarks: Use the Closed property of a Path or Curve object to test for equality (Visio uses 10E-6 as its "fuzz" factor) of the object's starting and ending points. A closed Curve object can be in a non-closed Path object and a non-closed Curve object can be in a closed Path object.

Path.Closed is unrelated to a Path's fill. A Path will be filled if its section's Geometryn.NoFill cell is 0. If Visio is told to fill an unclosed Path, it pretends there is a LineTo from the Path's end to its start. When filling a Path, Visio considers a point to be inside the Path if a ray drawn from the point in any direction crosses the Path or any of the shape's other Paths an odd number of times.

Example for Closed

Example

CmdNum property

Applies to: AccelItem, MenuItem, StatusBarItem, ToolbarItem

Summary: Gets or sets the command ID associated with an AccelItem, MenuItem, StatusBarItem, or ToolbarItem object.

Version: VISIO 4.0

Syntax: object.**CmdNum** = intVal
intVal = object.**CmdNum**

Element	Description
object	The object that has or gets the command ID
intVal	The command ID of the object

Remarks: When the AddOnName property of a MenuItem, StatusBarItem, or ToolbarItem object indicates an add-on to run, Visio automatically assigns a CmdNum.

Set CmdNum to 0 for a MenuItem that represents a separator in a menu or a ToolbarItem that represents a spacer in a toolbar.

The CmdNum property for a MenuItem that represents a submenu should be visCmdHierarchical.

CmdNum should never be 0 for an AccelItem object.

Valid command IDs are declared by the Visio type library (and visconst.bas). They have the prefix "visCmd."

See also: AddonName property, IsHierarchical property

Example for CmdNum

```
'This VBA macro demonstrates changing a built-in Visio toolbar button icon.
'The new icon persists as long as the document is active. This macro assumes
'you aren't using a custom user interface.
'Make sure the "dvs.ico" is in a folder along the Visio add-ons path.

Public Sub ChangeToolbarButtonIcon_Example()

  Dim uiObj As Visio.UIObject
  Dim toolbarSetObj As Visio.ToolbarSet
  Dim toolbarItemsObj As Visio.ToolbarItems
  Dim toolbarItemObj As Visio.ToolbarItem
  Dim i As Integer                          ' Loop variable
  Dim bFound As Boolean                     ' Found flag

  'Get the UI object for the copy of the MSOffice toolbars
  Set uiObj = Visio.Application.BuiltInToolbars(visToolBarMSOffice)

  'Get the Drawing Window toolbarsets
  'NOTE: Use ItemAtID to get the toolbarset
  'Using uiObj.ToolbarSets(visUIObjSetDrawing) will not work
  Set toolbarSetObj = uiObj.ToolbarSets.ItemAtID(visUIObjSetDrawing)

  'Get the ToolbarItems collection
  Set toolbarItemsObj = toolbarSetObj.Toolbars(0).ToolbarItems

  'Get the ToolbarItem for the NextPage toolbarbutton
  bFound = False
  For i = 0 To toolbarItemsObj.Count - 1
    Set toolbarItemObj = toolbarItemsObj(i)
    If toolbarItemObj.CmdNum = visCmdTurnToNextPage Then
      bFound = True
      Exit For
    End If
  Next i

  If bFound Then
    'Set the icon
    toolbarItemObj.IconFileName "dvs.ico"

    'Tell Visio to actually use the new custom UI
    ThisDocument.SetCustomToolbars uiObj
  End If

End Sub
```

CntrlID property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the control ID for a ToolbarItem or StatusBarItem object.

Version: VISIO 4.0

Syntax: object.**CntrlID** = intVal
intVal = object.**CntrlID**

Element	Description
object	The object that has or gets the control ID
intVal	The control ID of the object

Remarks: The control ID uniquely identifies a toolbar or status bar button.

Constants for Visio's standard control IDs are declared by the Visio type library (and visconst.bas). They have the prefix "visCtrlID."

CntrlID is a unique identifier for an item on the toolbar or status bar. Every unique item (except spaces) must have a unique ID. If you are creating a custom button on the toolbar, Visio will assign a unique control ID at runtime. Unique control IDs start at 1000. The CntrlID property for Visio standard toolbar items have values less than 1000 except for visCtrlIDNew, which has a special ID of 8383.

See also: CntrlType property, TypeSpecific1 property, TypeSpecific2 property

Example for CntrlID

Example

CntrlType property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the control type of an item in a toolbar or status bar.

Version: VISIO 4.0

Syntax: object.**CntrlType** = intVal
intVal = object.**CntrlType**

Element	Description
object	The object that has or gets the control type
intVal	The control type of the object

Remarks: If you are adding a custom toolbar or status bar button, set CntrlType to visCtrlTypeBUTTON. The following control types are declared by the Visio type library (and visconst.bas):

visCtrlTypeEND
visCtrlTypeSTATE
visCtrlTypeBUTTON
visCtrlTypeSTATE_BUTTON
visCtrlTypeHIERBUTTON
visCtrlTypeSTATE_HIERBUTTON
visCtrlTypeDROPBUTTON
visCtrlTypeSTATE_DROPBUTTON
visCtrlTypeSPINBUTTON
visCtrlTypePUSHBUTTON
visCtrlTypeEDITBOX
visCtrlTypeCOMBOBOX
visCtrlTypeCOMBODRAW
visCtrlTypeLISTBOX
visCtrlTypeLISTBOXDRAW
visCtrlTypeCOLORBOX
visCtrlTypeLABEL
visCtrlTypeMESSAGE
visCtrlTypeSPACER

See also: CntrlID property, TypeSpecific1 property, TypeSpecific2 property

Example for CntrlType

*BuiltInToolbars Property

Colors property

Applies to: Document

Summary: Returns the Colors collection of a Document object.

Version: VISIO 4.0

Syntax: objRet = object.**Colors**

Element	Description
objRet	The Colors collection of the object
object	The Document object that owns the collection

See also: Colors object, Document object

Example for Colors

Column property

Applies to: Cell

Summary: Returns the column index of a cell.

Version: VISIO 4.0

Syntax: intRet = object.**Column**

Element	Description
intRet	The column index of the Cell object
object	The Cell object to examine

See also: Cell object, CellsSRC property, Row property, Section property

Example for Column

Combine method

Applies to: Selection, Window

Summary: Creates a new shape by combining selected shapes.

Version: VISIO 2.0

Syntax: object.**Combine**

Element	Description
object	The Window or Selection object that contains the shapes to combine

Remarks: The Combine method is equivalent to choosing the Combine command from the Operation submenu on the Shape menu in Visio. The produced shape will be the topmost shape in its ContainingShape and will inherit the text and formatting of the first selected shape. The original shapes are deleted.

If the object being operated on is a Selection object, it will have no shapes selected in it when the operation is complete.

Combine and Join are similar. Combine will produce a shape with one geometry section for each original shape. The resultant shape will have holes in regions where the original shapes overlapped. Join differs from Combine in that it will coalesce abutting line and curve segments in the original shapes into a single geometry section in the resultant shape.

See also: Fragment method, Intersect method, Join method, Subtract method, Trim method, Union method, ContainingShape property

Example for Combine

Example

Company property

Applies to: Document

Summary: Returns or sets the value of the Company field in a Document's properties.

Version: VISIO 5.0

Syntax: strRet = object.**Company**
object.**Company** = stringExpression

Element	Description
strRet	The current value of the field
object	The Document object that has or gets the value
stringExpression	The new value of the field

Remarks: Setting the Company property is equivalent to entering information in the Company field in the Properties dialog box, accessed from the File menu.

See also: Description property, Keywords property, Subject property, Title property, Manager property, Category property, HyperlinkBase property

Example for Company

*Document Property

ConnectionsAdded event

Applies to: Application, Document, Documents, Master, Masters, Page, Pages

Summary: The event that occurs after connections have been established between Visio shapes.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtAdd+visEvtConnect (&H8100)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	A Connects object identifying the added connections
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source ConnectionsAdded using the AddAdvise method. In addition, ConnectionsAdded is included in the event set of all the objects in the Applies to list except the Document object. For those objects, you can use VBA Dim WithEvents variables to sink ConnectionsAdded. For performance considerations, the Document object's event set does not include ConnectionsAdded. To sink ConnectionsAdded from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, ConnectionsDeleted event

Example for ConnectionsAdded

ConnectionsDeleted event

Applies to: Application, Document, Documents, Master, Masters, Page, Pages

Summary: The event that occurs after connections have been removed between Visio shapes.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtDel+visEvtConnect (&H4100)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	A Connects object identifying the deleted connections
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source ConnectionsDeleted using the AddAdvise method. In addition, ConnectionsDeleted is included in the event set of all the objects in the Applies to list except the Document object. For those objects, you can use VBA Dim WithEvents variables to sink ConnectionsDeleted. For performance considerations, the Document object's event set does not include ConnectionsDeleted. To sink ConnectionsDeleted from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, ConnectionsAdded event

Example for ConnectionsDeleted

Connects property

Applies to: Master, Page, Shape

Summary: Returns a Connects collection for the indicated shape, page or master.

Version: VISIO 2.0

Syntax: objRet = object.**Connects**

Element	Description
objRet	The Connects collection of the Shape, Page or Master object
object	The Shape, Page or Master object that owns the collection

Remarks: The Connects collection of a shape contains every Connect for which the shape is the FromSheet. This tells you all the shapes the shape is connected to.

To obtain a Connects collection that contains every Connect for which the shape is the ToSheet, use the shape's FromConnects property. This tells you all the shapes that are connected to this shape.

The Connects collection of a page contains a Connect for every connection on the page.

The Connects collection of a master contains a Connect for every connection in the master.

See also: Connect object, Connects object, FromSheet property, ToSheet property, FromConnects property

Example for Connects

ContainingMaster property

Applies to: Selection, Shape, Shapes

Summary: Returns the Master object that contains the indicated object.

Version: VISIO 4.0

Syntax: objRet = object.**ContainingMaster**

Element	Description
objRet	The Master object that contains the object or collection
object	The object or collection to examine

Remarks: Use the ContainingMaster property to get the Master object that contains an object. If the object isn't in a Master, ContainingMaster returns Nothing. For example, if a Shape object belongs to the Shapes collection of a Page object, ContainingMaster returns Nothing.

See also: ContainingPage property, ContainingShape property, Master object

Example for ContainingMaster

ContainingPage property

Applies to: Selection, Shape, Shapes

Summary: Returns the Page object that contains the indicated object.

Version: VISIO 4.0

Syntax: objRet = object.**ContainingPage**

Element	Description
objRet	The Page object that contains the object or collection
object	The object or collection to examine

Remarks: Use the ContainingPage property to get the page that contains an object. If the object isn't in a Page object, ContainingPage returns Nothing. For example, if a Shape object belongs to a Master's Shapes collection, ContainingPage returns Nothing.

See also: ContainingMaster property, ContainingShape property, Page object

Example for ContainingPage

ContainingShape property

Applies to: Selection, Shape, Shapes

Summary: Returns the Shape object that contains the indicated object.

Version: VISIO 4.0

Syntax: objRet = object.**ContainingShape**

Element	Description
objRet	The Shape object that contains the object or collection
object	The object or collection to examine

Remarks: Use the ContainingShape property to get the Shape object that contains an object. The possible cases are as follows:

If the Shape object is the member of a group, ContainingShape returns that group.

If the Shape object is a top level shape in its Page or Master object (it is not a member of a group), ContainingShape returns the page sheet of its page or master.

If the ShapeObject is the page sheet of a page or master, ContainingShape returns Nothing.

See also: ContainingMaster property, ContainingPage property, PageSheet property, Shape object, Shapes property

Example for ContainingShape

Control property

Applies to: AccelItem, Entity

Summary: Gets or sets the Control value of an Entity object or whether the Control key modifies the key in an AccelItem object.

Version: VISIO 3.0 TECH

Syntax:
intRet = object.**Control**
object.**Control** = intExpression
strRet = object.**Control**
object.**Control** = stringExpression

Element	Description
strRet	The current control value for an Entity object
intRet	True (-1) if the Control key modifies the key in an AccelItem object; otherwise False (0)
object	The Entity or AccelItem object that has or gets the control value
intExpression	True (non-zero) if the Control key modifies the key in an AccelItem object; otherwise False (0)
stringExpression	The new control value for an Entity object

Remarks: An Entity whose Group property is set to 1002 represents a control value that may be either "{" or "}". The left brace begins a list and the right terminates the most recent list. It is up to the user to create and manage such lists, including adding the proper number of matching braces.

For an AccelItem object, set the Control property to TRUE to use the Control key as a modifier for an accelerator. For example, Control+Backspace.

See also: Alt property, Group property, Key property, Shift property

Example for Control

ConvertResult method

Applies to: Application

Summary: Converts a string or number into an equivalent number in different measurement units

Version: VISIO 4.5

Syntax: retVal = object.**ConvertResult**(stringOrNumber, unitsIn, unitsOut)

Element	Description
retVal	The result of the conversion
object	The Application object that is to perform the conversion operation
stringOrNumber	String or number to be converted
unitsIn	Measurement units to attribute to stringOrNumber
unitsOut	Measurement units to express result in

Remarks: StringOrNumber specifies the value to be converted. StringOrNumber can be passed as a string, floating point number or integer.

If passed as a string, stringOrNumber might be the formula or prospective formula of a cell (what you'd see in a ShapeSheet window with View Formulas selected), or the result or prospective result of a cell (what you'd see with View Values selected) expressed as a string. ConvertResult will evaluate the string and convert the result into the units designated by UnitsOut. Since the string is being evaluated outside the context of it's being the formula of a particular cell, ConvertResult will return an error if the string contains any cell references.

Possible values for stringOrNumber include:

1.7
3
"2.5"
"4.1 cm"
"12 ft - 17 in + (12 cm / SQRT(7))"

UnitsIn and UnitsOut can be strings such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property.

If StringOrNumber is a floating point number or integer, UnitsIn declares what unit of measure ConvertResult should construe the number to be in. Pass "" to indicate internal Visio units.

If stringOrNumber is a string, UnitsIn specifies how to interpret the evaluated result and is only used if the result is a scalar. For example, the expression "4 * 5 cm" evaluates to 20 cm, which is not a scalar so unitsIn is ignored. The expression "4 * 5" evaluates to 20 which is a scalar and is interpreted using the specified unitsIn.

UnitsOut specifies what units the returned number should be expressed in. If you want the results expressed in the same units as the evaluated expression, pass "NOCAST" or visNoCast.

Examples where string is specified:

Debug.Print application.ConvertResult("0.5 * 2", "ft", "ft") >>> 1.0
Debug.Print application.ConvertResult("0.5 * 2", "ft", "in") >>> 12.0
Debug.Print application.ConvertResult("1 cm", "ft", "in") >>> 0.39
Debug.Print application.ConvertResult("1 cm", "ft", "NOCAST") >>> 1.0
Debug.Print application.ConvertResult("1 cm", "ft", "") >>> 0.39
Debug.Print application.ConvertResult("1 cm", "ft", "bozo") >>> exception: Bad measurement unit.

Examples where number is specified:
Debug.Print application.ConvertResult(1, "ft", "ft") >>> 1
Debug.Print application.ConvertResult(1, "ft", "in") >>> 12
Debug.Print application.ConvertResult(1.0, "in", "ft") >>> 8.33333333333333E-02
Debug.Print application.ConvertResult(1.0, visFeet, "") >>> 12
Debug.Print application.ConvertResult(1, "bozo", "in") >>> exception: Bad measurement unit.

See also: FormatResult method, Result property

Example for ConvertResult

ConvertToGroup method

Applies to: Selection, Shape

Summary: Converts a selection or an object from another application (a linked or embedded object) to a group.

Version: VISIO 2.0

Syntax: object.**ConvertToGroup**

Element	Description
object	The Shape or Selection object to convert

Example for ConvertToGroup

Example

Copy method

Applies to: Characters, Hyperlink, Selection, Shape, Window

Summary: Copies the specified object, selection, or text range to the Windows Clipboard.

Version: VISIO 2.0

Syntax: object.**Copy**

Element	Description
object	The object to copy

Remarks: Use the Copy method to copy a shape, selection, text range, or Hyperlink object to the Clipboard.

When used with a Window object, Copy copies the shapes that are selected in that window (or all the shapes on the page displayed in that window if no shapes are selected). When used with a Characters object, Copy places the text range represented by that object on the Clipboard.

When used with a Hyperlink object, Copy places a copy of the Hyperlink object on the Clipboard to allow pasting into other Hyperlink-enabled applications or back into Visio itself.

To make copies of a shape without using the Clipboard, use the Duplicate method.

See also: Cut method, Delete method, Duplicate method, Paste method, Hyperlink object

Example for Copy, Cut, Paste

```
'This VBA macro demonstrates using the Copy, Paste, and Cut methods. It copies
a
'rectangle and moves it to the bottom-left corner of the page by setting the
'rectangle's PinX and PinY formulas.

Public Sub CopyPasteCut_Example ()

  Dim shpObj As Visio.Shape

  Set shpObj = ActivePage.DrawRectangle(1, 5, 5, 1)

  shpObj.Copy                              'Copy shape to the Clipboard
  ActivePage.Paste                         'Paste copy into drawing page
  shpObj.Cut                               'Remove original

  'Since the copied shape was the last one drawn, it can be retrieved from the
  'end of the shape collection.
  Set shpObj = ActivePage.Shapes.Item(ActivePage.Shapes.Count)

  'Move to the bottom left corner of the page by setting its
  'PinX and PinY formulas.
  shpObj.Cells("PinX").Formula = "Width * 0.5"    'Set new PinX and PinY
  shpObj.Cells("PinY").Formula = "Height * 0.5"

End Sub
```

Count property

Applies to: AccelItems, AccelTables, Addons, Attributes, Colors, Connects, Documents, Entities, EntityApps, EventList, Fonts, Layers, Masters, MenuItems, Menus, MenuSets, OLEObjects, Pages, Path, Paths, Selection, Shapes, StatusBarItems, StatusBars, Styles, ToolbarItems, Toolbars, ToolbarSets, Windows

Summary: Returns the number of objects in a collection.

Version: VISIO 2.0

Syntax: intRet = object.**Count**

Element	Description
intRet	The number of objects in the collection
object	The collection to examine

Example for Count, Documents

```
'This VBA macro demonstrates iterating through a Documents collection.
'It displays the names of all the open Visio documents in a list box.

Public Sub UpdateForm_Example ()

  Dim I As Integer
  Dim docObj As Visio.Document

  'ctlDocList is the name of the list box that receives the document names
  ctlDocList.Clear

  For I = 1 To Documents.Count
    Set docObj = Documents.Item(I)        'Get next open document
    ctlDocList.AddItem docObj.Name        'Add its name to the list box
  Next I

End Sub
```

CreateURL method

Applies to: Hyperlink

Summary: Returns a fully qualified and, optionally canonicalized, representation of the hyperlink's absolute address.

Version: VISIO 5.0

Syntax: strRet = object.**CreateURL**(intExpression)

Element	Description
strRet	A fully qualified URL representation of a hyperlink
object	The Hyperlink object to be acted upon
intExpression	True (non-zero) if canonical form; otherwise False (0)

Remarks: The CreateURL method of the Hyperlink object can be used to resolve relative URLs against a hyperlink's base address.

Setting CanonicalForm to True will apply URL canonicalization rules to the hyperlink. Only spaces are URL encoded during canonicalization. Port 80 is assumed for http URLs and will be removed during canonicalization. The URL "http://www.visio.com:80/" would be returned as "http://www.visio.com/", whereas http://www.visio.com:1000/" would be unchanged.

Following are some examples of results of the CreateURL method:

Address = "http://www.visio.com/"
CreateURL(False) returns "http://www.visio.com/"

Address = "C:\My Documents\Spreadsheet.XLS"
CreateURL(False) returns "file://C:\My Documents\Spreadsheet.XLS"
CreateURL(True) returns "file://C:\My%20Documents\Spreadsheet.XLS"

Relative Path Example :
Assume : Document.HyperlinkBase = "http://www.abc.com/bar/"
Address = "../file.htm"
CreateURL(False) returns "http://www.abc.com/file.htm"

See also: Address property, HyperlinkBase property, SubAddress property, ExtraInfo property

Example for CreateURL

```
*AddHyperlink Method
```

Creator property

Applies to: Document

Summary: Returns or sets the value of the Creator field in a document's properties.

Version: VISIO 2.0

Syntax: strRet = object.**Creator**
object.**Creator** = stringExpression

Element	Description
strRet	The current value of the field
object	The Document object that has or gets the value
stringExpression	The new value of the field

Remarks: Setting the Creator property is equivalent to entering information in the Creator field in the Properties dialog box located on the File menu.

See also: Description property, Keywords property, Subject property, Title property, Manager property, Company property, Category property, HyperlinkBase property

Example for Creator

*Document Property

CustomMenus property

Applies to: Application, Document

Summary: Returns a UI object that represents the current custom menus and accelerators of an Application object or a Document object.

Version: VISIO 4.0

Syntax: objRet = object.**CustomMenus**

Element	Description
objRet	A UI object that represents the object's current custom menus
object	The Application or Document object to examine

Remarks: If the object is not using custom menus, the CustomMenus property returns Nothing.

See also: CustomMenusFile property, CustomToolbars property, CustomToolbarsFile property, SetCustomMenus method, UI Object object

Example for CustomMenus

CustomMenusFile property

Applies to: Application, Document

Summary: Returns or sets the name of the file that defines custom menus and accelerators for an Application object or a Document object.

Version: VISIO 4.0

Syntax: strRet = object.**CustomMenusFile**
object.**CustomMenusFile** = fileStr

Element	Description
strRet	The name of the file that defines the current custom menus for the object
object	The Application or Document object that has or gets the custom menus
fileStr	The name of the file that defines new custom menus for the object

Remarks: If the object is not using custom menus, the CustomMenusFile property returns Nothing.

See also: CustomMenus property, CustomToolbars property, CustomToolbarsFile property

Example for CustomMenusFile

CustomToolbars property

Applies to: Application, Document

Summary: Returns a UI object that represents the current custom toolbars and status bars of an Application or Document object.

Version: VISIO 4.0

Syntax: objRet = object.**CustomToolbars**

Element	Description
objRet	A UI object that represents the object's current custom toolbars
object	The Application or Document object to examine

Remarks: If the object is not using custom toolbars, the CustomToolbars property returns Nothing.

See also: SetCustomToolbars method, CustomMenus property, CustomMenusFile property, CustomToolbarsFile property, UI Object object

Example for CustomToolbars

CustomToolbarsFile property

Applies to: Application, Document

Summary: Returns or sets the name of the file that defines custom toolbars and status bars for an Application or Document object.

Version: VISIO 4.0

Syntax: strRet = object.**CustomToolbarsFile**
object.**CustomToolbarsFile** = fileStr

Element	Description
strRet	The name of the file that defines the current custom toolbars for the object
object	The Application or Document object that has or gets the custom toolbars
fileStr	The name of the file that defines new custom toolbars for the object

Remarks: If the object is not using custom toolbars, the CustomToolbarsFile property returns Nothing.

See also: CustomMenus property, CustomMenusFile property, CustomToolbars property

Example for CustomToolbarsFile

Cut method

Applies to: Characters, Selection, Shape, Window

Summary: Deletes the indicated object, selection, or text range and places it on the Windows Clipboard.

Version: VISIO 2.0

Syntax: object.**Cut**

Element	Description
object	The object to cut

Remarks: When used with a Window object, Cut deletes the shapes that are selected in that window. When used with a Characters object, Cut places the text range represented by that object onto the Clipboard.

See also: Copy method, Delete method, Duplicate method, Paste method

Example for Cut

```
*Copy Method
```

Data1 property

Applies to: Shape

Summary: Returns or sets the value of the Data1 field for the indicated shape.

Version: VISIO 2.0

Syntax: strRet = object.**Data1**
object.**Data1** = stringExpression

Element	Description
strRet	The current value of the field
object	The Shape object that has or gets the value
stringExpression	The new value for the field

Remarks: Use the Data1 property to supply additional information about a shape. The property can contain up to 64 KB of characters. Text controls should be used with care with a string that is greater than three or four thousand characters.

Setting the Data1 property is equivalent to entering information in the Data 1 field in the Special dialog box located on the Format menu.

See also: Data2 property, Data3 property

Example for Data1, Data2, Data3

```
'This VBA macro demonstrates entering values in the Data1, Data2, and Data3
'fields.

Public Sub Data123_Example()

  Dim pageObj As Visio.Page
  Dim shapeObj As Visio.Shape

  Set pageObj = Documents.Add("").Pages(1)
  Set shapeObj = pageObj.DrawRectangle(3, 3, 5, 5)

  'Use the Data1, Data2, and Data3 properties to set the shape's Data fields.
  shapeObj.Data1 = "Data1 String"
  shapeObj.Data2 = "Data2 String"
  shapeObj.Data3 = "Data3 String"

  'Use the Data1, Data2, and Data3 properties to verify the shape's Data
  'field values.
  Debug.Print shapeObj.Data1
  Debug.Print shapeObj.Data2
  Debug.Print shapeObj.Data3

End Sub
```

Data2 property

Applies to: Shape

Summary: Returns or sets the value of the Data2 field for the indicated shape.

Version: VISIO 2.0

Syntax: strRet = object.**Data2**
object.**Data2** = stringExpression

Element	Description
strRet	The current value of the field
object	The Shape object that has or gets the value
stringExpression	The new value for the field

Remarks: Use the Data2 property to supply additional information about a shape. The property can contain up to 64 KB of characters. Text controls should be used with care with a string that is greater than three or four thousand characters.

Setting the Data2 property is equivalent to entering information in the Data 2 field in the Special dialog box located on the Format menu.

If the Shape object is associated with a ShapeData object, the Data 2 field may contain extended entity data flagged with an ASCII 1 as the first character of the field. To access this data, use an Entity object.

See also: Data1 property, Data3 property, Entity object, ShapeData object

Example for Data2

```
*Data1 Property
```

Data3 property

Applies to: Shape

Summary: Returns or sets the value of the Data3 field for the indicated shape.

Version: VISIO 2.0

Syntax: strRet = object.**Data3**
object.**Data3** = stringExpression

Element	Description
strRet	The current value of the field
object	The Shape object that has or gets the value
stringExpression	The new value for the field

Remarks: Use the Data3 property to supply additional information about a shape. The property can contain up to 64 KB of characters. Text controls should be used with care with a string that is greater than three or four thousand characters.

Setting the Data3 property is equivalent to entering information in the Data 3 field in the Special dialog box located on the Format menu.

If the Shape object is associated with a ShapeData object, the Data 3 field may contain extended entity data flagged with an ASCII 1 as the first character of the field. To access this data, use an Entity object.

See also: Data1 property, Data2 property, Entity object, ShapeData object

Example for Data3

```
*Data1 Property
```

DefaultFillStyle property

Applies to: Document

Summary: Returns or sets the default fill style of a document.

Version: VISIO 4.0

Syntax: strRet = object.**DefaultFillStyle**
object.**DefaultFillStyle** = stringExpression

Element	Description
strRet	The default fill style of the document
object	The Document object that has or gets the default fill style
stringExpression	The name of the default fill style to assign to the document

Remarks: This property corresponds to the value shown in the fill style control in Visio's toolbar when nothing is selected on the drawing page. The document's default fill style is applied to new shapes created with Visio's drawing tools or with the Draw methods via Automation.

See also: DefaultLineStyle property, DefaultTextStyle property, DefaultStyle property

Example for DefaultFillStyle

DefaultLineStyle property

Applies to: Document

Summary: Returns or sets the default line style of a document.

Version: VISIO 4.0

Syntax: strRet = object.**DefaultLineStyle**
object.**DefaultLineStyle** = stringExpression

Element	Description
strRet	The default line style of the document
object	The Document object that has or gets the default line style
stringExpression	The name of the default line style to assign to the document

Remarks: This property corresponds to the value shown in the line style control in Visio's toolbar when nothing is selected on the drawing page. The document's default line style is applied to new shapes created with Visio's drawing tools or with the Draw methods via Automation.

See also: DefaultFillStyle property, DefaultTextStyle property, DefaultStyle property

Example for DefaultLineStyle

DefaultStyle property

Applies to: Document

Summary: Returns the default fill style of a document or sets the default fill, line, and text styles of a document.

Version: VISIO 4.0

Syntax: strRet = object.**DefaultStyle**
object.**DefaultStyle** = stringExpression

Element	Description
strRet	The default fill style of the document
object	The Document object that has or gets the default style
stringExpression	The name of the default style to assign to the document

Remarks: A document's DefaultStyle property returns the same value as its DefaultFillStyle property. Setting DefaultStyle is equivalent to setting DefaultFillStyle, DefaultLineStyle, and DefaultTextStyle individually to the same multiple-attribute style. The fill, line, and text attributes of the document's default style is applied to new shapes created with Visio's drawing tools or with the Draw methods via Automation.

See also: DefaultFillStyle property, DefaultLineStyle property, DefaultTextStyle property

Example for DefaultStyle

DefaultTextStyle property

Applies to: Document

Summary: Returns or sets the default text style of a document.

Version: VISIO 4.0

Syntax: strRet = object.**DefaultTextStyle**
object.**DefaultTextStyle** = stringExpression

Element	Description
strRet	The default text style of the document
object	The Document object that has or gets the default text style
stringExpression	The name of the default text style to assign to the document

Remarks: This property corresponds to the value shown in the text style control in Visio's toolbar when nothing is selected on the drawing page. The document's default text style is applied to new shapes created with Visio's drawing tools or with the Draw methods via Automation.

See also: DefaultFillStyle property, DefaultLineStyle property, DefaultStyle property

Example for DefaultTextStyle

DefaultValue property

Applies to: Attribute

Summary: Returns or sets the default value of an Attribute object.

Version: VISIO 3.0 TECH

Syntax: strRet = object.**DefaultValue**
object.**DefaultValue** = strValue

Element	Description
strRet	The current default value
object	The Attribute object that has or gets the value
strValue	The new default value

Remarks: The DefaultValue and Prompt properties of an Attribute object make up what is called the attribute's definition. The DefaultValue property can be used when initializing an Attribute object for a given shape.

See also: Prompt property

Example for DefaultValue

```
*PutShape Method
```

DeferRecalc property

Applies to: Application

Summary: Determines whether Visio recalculates cell formulas during a series of actions.

Version: VISIO 4.1

Syntax: intRet = object.**DeferRecalc**
object.**DeferRecalc** = intExpression

Element	Description
intRet	False (0) if formulas are recalculated as needed; True (-1) if recalculation is deferred
object	The object that has or gets the setting
intExpression	False (0) to recalculate formulas as needed; True (non-zero) to defer recalculation

Remarks: Use the DeferRecalc property to improve performance during a series of actions. For example, you can defer formula recalculation while changing the formulas or values of several cells.

If a program neglects to turn DeferRecalc off after turning it on, Visio turn will it off when the user performs an operation.

If you release objects or send a large number of commands to Visio while recalculation is deferred, Visio may at times need to process its queue of pending recalculations. Because of this, you may want to be careful regarding the code sequences you use inside a scope where you want recalculation deferred. For example, consider the following Visual Basic sequence:

```
    visObj.DeferRecalc = True
    shpObj.Cells("height").ResultIU = 12 ' Causes deferred recalculations to be
processed.
    shpObj.Cells("width").ResultIU = 14
    visObj.DeferRecalc = False
```

Because Visual Basic makes and releases a temporary shape object in the indicated statement, Visio will process its queue at that point. In the following sequence, Visio will not process the recalculation queue until the program turns DeferRecalc off (or the user performs some operation).

```
    visObj.DeferRecalc = True
    Set cellObj1 = shpObj.Cells("height")
    Set cellObj2 = shpObj.Cells("Width")
    cellObj1.ResultIU = 12
    cellObj1.ResultIU = 14
    visObj.DeferRecalc = False
```

See also: ScreenUpdating property

Example for DeferRecalc

Delete method

Applies to: AccelItem, AccelTable, Attribute, Entity, EntityApp, Event, Hyperlink, Layer, Master, Menu, MenuItem, MenuSet, Page, Selection, Shape, StatusBar, StatusBarItem, Style, Toolbar, ToolbarItem, ToolbarSet, Window

Summary: Deletes the indicated object or selection.

Version: VISIO 2.0

Syntax:

object.**Delete**
object.**Delete** fDeleteShapes
object.**Delete** fRenumberPages

Element	Description
object	The object to delete
fDeleteShapes	1 (TRUE) to delete shapes assigned to the layer; otherwise 0 (FALSE)
fRenumberPages	1 (TRUE) to renumber remaining pages; otherwise 0 (FALSE)

Remarks: When used with a Window object, Delete deletes the selected shapes in that window. The Delete method is equivalent to choosing the Clear command located on the Edit menu in Visio.

When used with a Layer object, if fDeleteShapes is non-zero, shapes assigned only to the deleted layer are deleted. Otherwise, the shapes are simply no longer assigned to that layer.

When used with a Page object, if fRenumberPages is non-zero, the remaining pages' default page names are renumbered after the page is deleted, otherwise, the pages retain their names. This is equivalent to checking or unchecking Update Page Names in the Delete Pages dialog box.

See also: Copy method, Cut method, Duplicate method, Paste method

Example for Delete, Duplicate

```
'This VBA macro demonstrates deleting and duplicating objects.

Public Sub DuplicateAndDelete_Example ()

  Dim shpOriginal As Visio.Shape
  Dim shpDuplicate As Visio.Shape

  Set shpOriginal = ActivePage.DrawLine(1, 1, 5, 5)

  shpOriginal.Duplicate

  Set shpDuplicate = ActivePage.Shapes(2)

  shpDuplicate.Cells("BeginY") = "2"

  shpOriginal.Delete

End Sub
```

DeleteRow method

Applies to: Shape

Summary: Deletes a row from a section in a ShapeSheet.

Version: VISIO 2.0

Syntax: object.**DeleteRow** section, row

Element	Description
object	The Shape object that owns the row
section	The index of the section that contains the row
row	The index of the row to delete

Remarks: To remove one row at a time from a ShapeSheet section, use the DeleteRow method. If the section has indexed rows, the rows following the deleted row shift position. If the row does not exist, nothing is deleted.

You should not delete rows that define fundamental characteristics of a shape, such as the 1-D Endpoints row (visRowXForm1D) or the component row (visRowComponent) or the MoveTo row (visRowVertex + 0) in a Geometry section. You cannot delete rows from sections represented by visSectionCharacter, visSectionParagraph, and visSectionTab.

See also: AddRow method, DeleteSection method

Example for DeleteRow

```
*AddSection Method
```

DeleteSection method

Applies to: Shape

Summary: Deletes a section from a ShapeSheet.

Version: VISIO 2.0

Syntax: object.**DeleteSection** section

Element	Description
object	The Shape object that owns the section
section	The index of the section to delete

Remarks: When you delete a ShapeSheet section, all rows in the section are automatically deleted. If the specified section does not exist, nothing is deleted and no error is generated.

If a Geometry section is deleted, any following geometry sections shift up because they are indexed and no gaps can exist in an indexed range.

You can delete any section except the section represented by visSectionObject (although you can delete rows within that section).

See also: AddSection method, DeleteRow method

Example for DeleteSection

```
*AddSection Method
```

Description property

Applies to: Document, Hyperlink

Summary: Returns or sets the value of the Description field in a document's properties or the Description of a shape's Hyperlink object.

Version: VISIO 2.0

Syntax: strRet = object.**Description**
object.**Description** = stringExpression

Element	Description
strRet	The current value of the field
object	The object that has or gets the value
stringExpression	The new value for the field

Remarks: Setting a document's Description property is equivalent to entering information in the Description field in the Properties dialog box, accessed from the File menu.

Setting a hyperlink's Description property is equivalent to entering information in the optional Descriptive name of link field in the Hyperlink dialog box, accessed from the Insert menu. This is also equivalent to setting the result of the Description cell of the shape's hyperlink row.

See also: Creator property, Keywords property, Subject property, Title property, Manager property, Company property, Category property, HyperlinkBase property

Example for Description

*Document Property

DeselectAll method

Applies to: Selection, Window

Summary: Deselects all shapes in a window or selection.

Version: VISIO 2.0

Syntax: object.**DeselectAll**

Element	Description
object	The Window or Selection object that contains the shapes to deselect

See also: Select method, SelectAll method, Selection object, Selection property

Example for DeselectAll

DesignModeEntered event

Applies to: Application, Document, Documents

Summary: The event that occurs before a document enters design mode.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeDocDesign (6)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that is about to leave run mode
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, EventsEnabled property, RunModeEntered event, Mode property

Example for DesignModeEntered

DockedStencils method

Applies to: Window

Summary: Returns the document names of all stencils docked in a Visio drawing window.

Version: VISIO 4.5

Syntax: object.**DockedStencils** NameArray

Element	Description
object	The Window that contains the stencils whose names you want to get
NameArray	Array that receives document names of stencils docked in indicated window

Remarks: DockedStencils returns an array of strings that are the names of the documents being shown in the docked stencil panes, if any, of the indicated window.

If the indicated window is a drawing window, let n>=0 be the number of docked stencil panes presently being shown in it. If the window isn't a drawing window let n be 0.

If DockedStencils succeeds, NameArray returns a one-dimensional array of n strings indexed from 0 to n-1. NameArray is an out argument that is allocated by DockedStencils and ownership of which is passed back to the caller. The caller should eventually perform SafeArrayDestroy on the returned array. Note that SafeArrayDestroy has the side effect of freeing the strings referenced by the array's entries. DockedStencils fails if called with ! NameArray or *NameArray. (VB and VBA take care of all this for you. See the example for treatment in VBA.)

Let si be the string returned in NameArray(i). Suppose NameArray(i) represents pane pi and that pane pi is showing document di.

Visio presently operates such that the document associated with a docked stencil pane may or may not actually be open (in the Application object's Documents collection) at the moment. Typically the document being shown in the topmost pane is open and the documents shown in other panes aren't.

DockedStencil guarantees:

- if di is currently open, then Documents.Item(si) will succeed and return a document object representing di.

- if di is not open, then Documents.Open(si) will return a document object representing di.

See also: Item property, Open method

Example for DockedStencils

```
'This VBA macro demonstrates the DockedStencils method.
'A C++ program demonstrating the same method follows the VBA macro.

Public Sub GetDockedStencilNames()

    ' This gets the document names of all stencils docked
    ' in the active window and lists the names in the
    ' Immediate window.
    Dim stencilnames() As String
    ActiveWindow.DockedStencils stencilnames
    Dim lb As Integer, ub As Integer
    lb = LBound(stencilnames)
    ub = UBound(stencilnames)
    Debug.Print ActiveWindow.Document; " lb:"; lb; "ub:"; ub
    While lb <= ub
        Debug.Print stencilnames(lb)
        lb = lb + 1
    Wend

End Sub


------------------------------------------------------------
'This C++ program demonstrates the DockedStencils method.

extern "C" int RunDemo(void)
{
    // This gets the document names of the stencils docked
    // in the active window.
    //
    HRESULT hr;
    CVisioApplication app;
    CVisioWindow win;
    SAFEARRAY FAR* psaNames = NULL;

    if ( VAO_SUCCESS != vaoGetObjectWrap(app) )
        goto CU;

    hr= app.ActiveWindow(win);
    check_valid(hr, win);

    if ( NOERROR == win.DockedStencils(&psaNames) )
        {
        // Got the names, which will be an array of BSTRs.
        // We just iterate through them and you can verify
        // they're the right names using the debugger.

        long rgIndex[1];
        LONG llb, lub;
        if ( psaNames &&
             SafeArrayGetDim(psaNames) == 1 &&
             SafeArrayGetElemsize(psaNames) == sizeof(BSTR) &&
             (NOERROR == SafeArrayGetLBound(psaNames, 1, &llb)) &&
             (NOERROR == SafeArrayGetUBound(psaNames, 1, &lub)) )
```

```
            {
            BSTR bstr;
            for ( *rgIndex = llb; llb <= lub; llb++, (*rgIndex)++ )
                {
                SafeArrayGetElement(psaNames,rgIndex,&bstr);
                // SafeArrayElement makes copy of string.
                SysFreeString(bstr);
                }
            }
        SafeArrayDestroy(psaNames);
        }
CU:
    return 0;
}
```

DoCmd method

Applies to: Application

Summary: Performs the command with the indicated command ID.

Version: VISIO 4.0

Syntax: object.**DoCmd** (intExpression)

Element	Description
object	The Application object to perform the command
intExpression	The command to perform

Remarks: Constants for Visio command IDs are declared by the Visio type library (and visconst.bas). They have the prefix "visCmd." DoCmd works best with commands that don't display dialog boxes.

Example for DoCmd

```
'This VBA macro demonstrates using constants with the DoCmd method.

Public Sub DoCmd_Example()

  Dim docObj As Visio.Document

  Set docObj = Documents.Add("")

  Visio.Application.DoCmd (visCmdWindowShowMasterObjects)
  Visio.Application.DoCmd (visCmdNewMaster)

End Sub
```

Document property

Applies to: Cell, Characters, Color, Colors, Connect, Connects, Font, Fonts, Layer, Layers, Master, Masters, Page, Pages, Selection, Shape, Shapes, Style, Styles, Window

Summary: Returns the document that is associated with the object.

Version: VISIO 2.0

Syntax: objRet = object.**Document**

Element	Description
objRet	The Document object that contains the object
object	The object to examine

Remarks: The Document property of a docked stencil window returns a Document object for the stencil that is currently at the top of the window. If another stencil replaces the first in the top position, however, the first stencil's document is closed so the reference to it becomes invalid. For best results, assume that document references to docked stencils are not persistent.

Example for Category, Company, Creator, Description, Document, HyperlinkBase, Keywords, Manager, Subject, Title

```
'This VBA macro demonstrates using the Document object and its properties such
as
'Title, Creator, Description, Keywords, and Subject.


Public Sub Document_Prop_Example()

  Dim documentObj As Visio.Document
  Dim tempDocumentObj As Visio.Document
  Dim pageObj As Visio.Page
  Dim shapeObj As Visio.Shape
  Dim windowObj As Visio.Window
  Dim masterObj As Visio.Master

  Set documentObj = Documents.Add("")

  'Set the properties of the document.
  documentObj.Title = "My Document"
  documentObj.Creator = "Judy Lemke"
  documentObj.Description = "This is an office layout drawing."
  documentObj.Keywords = "office, chairs, desk"
  documentObj.Subject = "Office Layout"

  Set windowObj = ActiveWindow
  Set pageObj = ActivePage
  Set shapeObj = pageObj.DrawRectangle(2, 2, 5, 5)
  Set masterObj = documentObj.Masters.Add

  'Use the Document property and a window object to set the document object.
  Set tempDocumentObj = windowObj.Document
  Debug.Print tempDocumentObj.Title         'Verify using the Title property.

  'Use the Document property and a page object to set the document object.
  Set tempDocumentObj = pageObj.Document
  Debug.Print tempDocumentObj.Title

  'Use the Document property and a shape object to set the document object.
  Set tempDocumentObj = shapeObj.Document
  Debug.Print tempDocumentObj.Title

  'Use the Document property and a master object to set the document object.
  Set tempDocumentObj = masterObj.Document
  Debug.Print tempDocumentObj.Title

End Sub
```

DocumentAdded event

Applies to: Application, Document, Documents

Summary: The event that occurs after a Visio document is opened or created.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtAdd+visEvtDoc (&H8002)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that was just opened or created
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

DocumentAdded events can be added to the EventLists of Application, Documents or Document objects. The first two are straightforward--if a document is opened or created in the scope of the Application or its Documents collection, the DocumentAdded event occurs.

However, a DocumentAdded event added to the EventList of a Document object makes sense only if the event's action is visActCodeRunAddon. In this case, the event is persistable--it can be stored with the document. If the document that contains the persistent event is opened, its action is triggered. If a new document is based on or copied from the document that contains the persistent event, the DocumentAdded event is copied to the new document and its action is triggered. However, if the event's action is visActCodeAdvise, that event is not persistable and therefore is not stored with the document; hence it will never be triggered.

If you have a document that you suspect causes ill-behaved code to run in response to DocumentCreated, DocumentOpened or DocumentAdded, you can prevent these (and all) events from firing by setting App.EventsEnabled to false, or by adding EventsEnabled=0 to Visio's initialization file (visio.ini).

See also: Action property, Add method, AddAdvise method, Event object, EventList object, EventsEnabled property, DocumentOpened event, DocumentCreated event

Example for DocumentAdded

DocumentChanged event

Applies to: Application, Document, Documents

Summary: The event that occurs after certain properties of a Visio document are changed.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtDoc (&H2002)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that just changed
moreInfo	Nothing for this event

Remarks: The DocumentChanged event indicates that a document summary property, such as Creator or Description, has changed.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Creator property, Description property, Event object, EventList object, Keywords property, Subject property, Title property

Example for DocumentChanged

DocumentCreated event

Applies to: Application, Document, Documents

Summary: The event that occurs after a Visio document is created.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeDocCreate (1)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that was just created
moreInfo	Nothing for this event

Remarks: The DocumentCreated event is often added to the EventList of a Visio template file (.VST). The event's action is triggered whenever a new document is created based on that template.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

DocumentCreated events can be added to the EventLists of Application, Documents or Document objects. The first two are straightforward--if a new document is created in the scope of the Application or its Documents collection, the DocumentCreated event occurs.

However, a DocumentCreated event added to the EventList of a Document object makes sense only if the event's action is visActCodeRunAddon. In this case, the event is persistable--it can be stored with the document. If a new document is based on or copied from the document that contains the persistent event, the DocumentCreated event is copied to the new document and its action is triggered. However, if the event's action is visActCodeAdvise, that event is not persistable and therefore is not stored with the document; hence it will never be triggered.

If you have a document that you suspect causes ill-behaved code to run in response to DocumentCreated, DocumentOpened or DocumentAdded, you can prevent these (and all) events from firing by setting App.EventsEnabled to false, or by adding EventsEnabled=0 to Visio's initialization file (visio.ini).

See also: Action property, Add method, AddAdvise method, Event object, EventList object, EventsEnabled property, DocumentOpened event, DocumentAdded event

Example for DocumentCreated, ShapeAdded

```
'This VBA macro demonstrates counting shapes added to a drawing that are
'based on a master called Square.

'The DocumentCreated event handler runs when a new drawing based on the
template
'that contains this code is created. The handler initializes an integer
variable,
'nSquares, which is used to store the count.

'The ShapeAdded event handler runs each time a shape is added to the drawing
'page, whether the shape is dropped from a stencil, drawn with a drawing tool,
or
'pasted from the Clipboard. The handler checks the Master property of the new
'shape and, if the shape is based on the Square master, increments nSquares.

  ' Number of squares added to drawing
  Dim nSquares As Integer

Private Sub Document_DocumentCreated(ByVal doc as Visio.IVDocument)
  ' Initialize number of squares added
  nSquares = 0

End Sub

Private Sub Document_ShapeAdded( ByVal Shape As Visio.IVShape )

  Dim mastObj As Visio.Master

  ' Get the Master property of the shape
  Set mastObj = Shape.Master

  ' Check whether the shape has a master. If not, the shape was created
locally.
  If Not ( mastObj Is Nothing ) Then

    ' Check whether the master is "Square"
    If mastObj.Name = "Square" Then

    ' Increment the count for the number of squares added
    nSquares = nSquares + 1
    End If

  End If
  MsgBox "Number of squares: " & nSquares, vbInformation, _
  "Developing Visio Solutions"

End Sub
```

DocumentOpened event

Applies to: Application, Document, Documents

Summary: The event that occurs after a Visio document is opened.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeDocOpen (2)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that was just opened
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

DocumentOpened events can be added to the EventLists of Application, Documents or Document objects. The first two are straightforward--if a document is opened in the scope of the Application or its Documents collection, the DocumentOpened event occurs.

However, a DocumentOpened event added to the EventList of a Document object makes sense only if the event's action is visActCodeRunAddon. In this case, the event is persistable--it can be stored with the document. If the document that contains the persistent event is opened, its action is triggered. However, if the event's action is visActCodeAdvise, that event is not persistable and therefore is not stored with the document; hence it will never be triggered.

If you have a document that you suspect causes ill-behaved code to run in response to DocumentCreated, DocumentOpened or DocumentAdded, you can prevent these (and all) events from firing by setting App.EventsEnabled to false, or by adding EventsEnabled=0 to Visio's initialization file (visio.ini).

See also: Action property, Add method, AddAdvise method, Event object, EventList object, EventsEnabled property, DocumentCreated event, DocumentAdded event

Example for DocumentOpened

Documents property

Applies to: <Global>, Application

Summary: Returns the Documents collection for an instance of Visio.

Version: VISIO 2.0

Syntax: objsRet = object.**Documents**

Element	Description
objsRet	The Documents collection of the Application object
object	The Application object that owns the collection

Remarks: You can iterate through a Documents collection by using the Count property to retrieve the number of documents in the collection. You can use the Item property to retrieve individual elements from a collection.

See also: Documents object

Example for Documents

```
*Count Property
```

DocumentSaved event

Applies to: Application, Document, Documents

Summary: The event that occurs after a Visio document is saved.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeDocSave (3)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that was just saved
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, DocumentSavedAs event, BeforeDocumentSave event, BeforeDocumentSaveAs event, Event object, EventList object

Example for DocumentSaved, PageAdded, ShapesDeleted

```
'This procedure, visEventProc, demonstrates using a Select Case block to check
'for three events: DocumentSaved, PageAdded, and ShapeDeleted. Other events
fall
'under the default case (Case Else). Each Case block constructs a string
'(strDumpMsg) that contains the name and event code of the event that fired.
'Finally, the procedure displays the string in a message box.

' VisEventProc - Handles Visio events
'
' Parameters:
' eventCode    The event code of the event that fired.
' sourceObj    A reference to the source object whose EventList contains the
'              Event object.
' eventID      The unique ID of the Event object in its EventList collection.
' seqNum       The sequence of this event among events fired in the instance
of
'              Visio.
' subjectObj   A reference to the object that is subject of the event.
' moreInfo     A string that contains additional information, defined when the
'              Event object was created.
'
Public Sub VisEventProc(eventCode As Integer, sourceObj As Object, eventID As
_ Long, seqNum As Long, subjectObj As Object, moreInfo As Variant)

  Dim strDumpMsg As String

  ' Find out which event fired.
  Select Case eventCode
    Case visEvtCodeDocSave
    strDumpMsg = "Save(" & eventCode & ")"
    Case (visEvtPage + visEvtAdd)
    strDumpMsg = "Page Added(" & eventCode & ")"
    Case visEvtCodeShapeDelete
    strDumpMsg = "Shape Deleted(" & eventCode & ")"
    Case Else
    strDumpMsg = "Other(" & eventCode & ")"
  End Select

  ' Display the event name and code
  frmEventDisplay.EventText.Text = strDumpMsg

End Sub
```

DocumentSavedAs event

Applies to: Application, Document, Documents

Summary: The event that occurs after a Visio document is saved with Save As.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeDocSaveAs (4)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that was just Saved As
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, DocumentSaved event, BeforeDocumentSave event, BeforeDocumentSaveAs event, Event object, EventList object

Example for DocumentSavedAs

DrawBezier method

Applies to: Master, Page, Shape

Summary: Creates a new shape whose path is defined by the supplied sequence of Bezier control points.

Version: VISIO 4.1

Syntax: objRet = object.**DrawBezier**(xyArray, degree, flags)

Element	Description
objRet	The new Shape object
object	The page, master or group in which to draw the shape
xyArray	An array of alternating x and y values that define the Bezier control points for the new shape.
degree	The degree of the Bezier curve
flags	Flags that influence how the shape is drawn

Remarks: The xyArray and degree parameters must meet the following conditions:

1 <= degree <= 9
The number of points must be k*degree + 1, where k is a positive integer.

If the first point is called p0, then for any integer m between 1 and k, p(m*degree) is assumed to be the last control point of a Bezier segment, as well as the first control point of the next.

The result is a composite curve that consists of k Bezier segments. The input points from xyArray define the curve's control points. If you want the curve to be smooth, make sure that the points p(n-1), pn and p(n+1) are co-linear whenever n=m*degree with an integer m. The composite Bezier curve is represented in Visio as a B-spline with integer knots of multiplicity=degree.

The points should be in internal drawing units (inches). The passed array should be a type SAFEARRAY of 8-byte floating point values passed by reference (VT_R8| VT_ARRAY|VT_BYREF). This is how Visual Basic 4.0 passes arrays to Automation objects.

The flags argument is a bit mask that specifies options for drawing the new shape. Its value should be either 0 or visSpline1D (8). If flags is visSpline1D and if the first and last points in xyArray don't coincide, DrawBezier produces a shape with 1D behavior; otherwise, it produces a shape with 2D behavior.

If the first and last points in xyArray do coincide, DrawBezier produces a filled shape.

See also: DrawLine method, DrawOval method, DrawRectangle method, DrawPolyline method, DrawSpline method, FitCurve method

Example for DrawBezier

```
*DrawSpline Method
```

DrawingPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for drawings.

Version: VISIO 4.0

Syntax: strRet = object.**DrawingPaths**
object.**DrawingPaths** = pathsStr

Element	Description
strRet	A text string containing a list of folders
object	An Application object
pathsStr	A text string containing a list of folders

Remarks: The string passed to and received from DrawingPaths is the same string shown in Visio's File Paths dialog (accessible from the Tools/Options dialog). This same string is stored in Visio's profile (.ini) file (appObj.ProfileName) in the entry whose key is "DrawingsPath."

To indicate more than one folder, separate individual items in the path string with semicolons. If a path is not fully qualified, Visio looks for the folder in the folder that contains the Visio program files (appObj.Path).

For example, if Visio's executable file is installed in c:\Visio, and DrawingPaths is "Drawings;d:\Drawings", Visio looks for drawings in both c:\Visio\Drawings and d:\Drawings.

When Visio looks for drawings, it will look in all paths named in DrawingPaths plus in all sub-folders of those paths. Also, the fact that a path is named in DrawingPaths does not imply the path actually exists. If you pass DrawingPaths to the EnumDirectories method, it will return a complete list of fully qualified paths that Visio will actually look in.

See also: AddonPaths property, FilterPaths property, HelpPaths property, StartupPaths property, StencilPaths property, TemplatePaths property, ProfileName property, Path property, EnumDirectories method

Example for DrawingPaths

DrawLine method

Applies to: Master, Page, Shape

Summary: Adds a line to the shapes collection of a page, master or group.

Version: VISIO 2.0

Syntax: objRet = object.**DrawLine** (x1, y1, x2, y2)

Element	Description
objRet	A Shape object that represents the new line
object	The page, master or group on which to draw the line
x1	The x-coordinate of the line's begin point
y1	The y-coordinate of the line's begin point
x2	The x-coordinate of the line's end point
y2	The y-coordinate of the line's end point

Remarks: Using this method is equivalent to using the line tool in Visio. The arguments are in internal drawing units.

See also: DrawBezier method, DrawOval method, DrawPolyline method, DrawRectangle method, DrawSpline method

Example for DrawLine

*DrawRectangle Method

DrawOval method

Applies to: Master, Page, Shape

Summary: Adds an ellipse to the shapes collection of a page, master or group.

Version: VISIO 2.0

Syntax: retVal = object.**DrawOval** (x1, y1, x2, y2)

Element	Description
retVal	A Shape object that represents the new ellipse
object	The page, master or group on which to draw the ellipse
x1	The left side of the ellipse's width-height box
y1	The top of the ellipse's width-height box
x2	The right side of the ellipse's width-height box
y2	The bottom of the ellipse's width-height box

Remarks: Using this method is equivalent to using the ellipse tool in Visio. The arguments are in internal drawing units.

See also: DrawBezier method, DrawLine method, DrawPolyline method, DrawRectangle method, DrawSpline method

Example for DrawOval

*DrawRectangle Method

DrawPolyline method

Applies to: Master, Page, Shape

Summary: Creates a new shape whose path is a polyline along a given set of points.

Version: VISIO 4.1

Syntax: objRet = object.**DrawPolyline**(xyArray, flags)

Element	Description
objRet	A Shape object that represents the new polyline
object	The page, master or group in which to draw the shape
xyArray	An array of alternating x and y values that define points in the new shape's path
flags	Flags that influence how the shape is drawn

Remarks: DrawPolyline creates a new shape whose path consists of a sequence of line segments and whose end points match the points specified in xyArray. Calling DrawPolyline is equivalent to calling DrawSpline with a tolerance of 0 and a flag of visSplineAbrupt.

The points should be in internal drawing units (inches). The passed array should be a type SAFEARRAY of 8-byte floating point values passed by reference (VT_R8| VT_ARRAY|VT_BYREF). This is how Visual Basic 4.0 passes arrays to Automation objects.

The flags parameter is a bit mask that specifies options for drawing the new shape. Its value should be either 0 or visSpline1D (8). If flags is visSpline1D and if the first and last points in xyArray don't coincide, DrawPolyline produces a shape with 1D behavior; otherwise it produces a shape with 2D behavior.

If the first and last points in xyArray do coincide, DrawPolyline produces a filled shape.

See also: DrawBezier method, DrawLine method, DrawOval method, DrawRectangle method, DrawSpline method, FitCurve method

Example for DrawPolyline

```
'This VBA macro demonstrates drawing a polyline.

Public Sub TestDrawMethods_Example ()

  Dim shapeObj As Visio.Shape
  Dim arrXY(1 To 4 * 2) As Double

  'Initialize array with coordinates.
  arrXY(1) = 1
  arrXY(2) = 1
  arrXY(3) = 3
  arrXY(4) = 3
  arrXY(5) = 5
  arrXY(6) = 1
  arrXY(7) = 1
  arrXY(8) = 2

  'Use the DrawPolyLine method to draw a 2D shape.
  Set shapeObj = ActivePage.DrawPolyline(arrXY, 0)

  'Increase the Y coordinates of the array by 4.
  For i = 2 To UBound(arrXY) Step 2
    arrXY(i) = arrXY(i) + 4
  Next i

  'Use the DrawPolyLine method to draw a 1D shape.
  Set shapeObj = ActivePage.DrawPolyline(arrXY, visSpline1D)

End Sub
```

DrawRectangle method

Applies to: Master, Page, Shape

Summary: Adds a rectangle to the shapes collection of a page, master or group.

Version: VISIO 2.0

Syntax: objRet = object.**DrawRectangle** (x1, y1, x2, y2)

Element	Description
objRet	A Shape object that represents the new rectangle
object	The page, master or group on which to draw the rectangle
x1	The left side of the rectangle's width-height box
y1	The top of the rectangle's width-height box
x2	The right side of the rectangle's width-height box
y2	The bottom of the rectangle's width-height box

Remarks: Using this method is equivalent to using the rectangle tool in Visio. The arguments are in internal drawing units.

See also: DrawBezier method, DrawLine method, DrawOval method, DrawPolyline method, DrawSpline method

Example for DrawLine, DrawOval, DrawRectangle, FillStyle, LineStyle

```
'This VBA macro demonstrates drawing lines, ovals, and rectangles. It also
sets
'styles for shapes.

Public Sub DrawLineOvalRectangle_Example ()

  Dim shpObj As Visio.Shape

  Set shpObj = ActivePage.DrawRectangle(1, 4, 4, 1)
  shpObj.FillStyle = "Blue fill"

  Set shpObj = ActivePage.DrawOval(1.5, 10.5, 7.5, 6.5)
  shpObj.FillStyle = "Red fill"

  Set shpObj = ActivePage.DrawLine(5, 4, 7.5, 1)
  shpObj.LineStyle = "3pxl line"

End Sub
```

DrawSpline method

Applies to: Master, Page, Shape

Summary: Creates a new shape whose path follows a given sequence of points.

Version: VISIO 4.1

Syntax: objRet = object.**DrawSpline**(xyArray, tolerance, flags)

Element	Description
objRet	A Shape object that represents the new spline
object	The page, master or group in which to draw the new shape
xyArray	An array of alternating x and y values that define points in the new shape's path
tolerance	How closely the path of new shape must approximate the given points
flags	Flags that influence how the shape is drawn

Remarks: DrawSpline creates a new shape whose path falls within the given tolerance of the given array of points. To fit the given points exactly, specify a tolerance of 0. Typically, DrawSpline fits spline segments through the points, but it may sometimes produce line or circular arc segments in the new shape.

The points and tolerance should be in internal drawing units (inches). The passed array should be a type SAFEARRAY of 8-byte floating point values passed by reference (VT_R8|VT_ARRAY|VT_BYREF). This is how Visual Basic 4.0 passes arrays to Automation objects.

The error from the points to the path of the resulting shape will be roughly within tolerance. When the number of points is large, the actual error may sometimes exceed the prescribed tolerance.

The flags parameter is a bit mask that specifies options for drawing the new shape. Its value should be a combination of 0 or more of the following values:

visSplinePeriodic = 1
visSplineDoCircles = 2
visSplineAbrupt = 4
visSpline1D = 8

If Flags includes visSplinePeriodic and the following conditions are met, Visio attempts to draw a periodic spline. Otherwise, Visio draws a non-periodic spline:

* The last point must be a repetition of the first one.

* If the flag visSplineAbrupt is included as well, the entire closed path outlined by the points must be free of abrupt changes of direction and curvature.

If Flags includes visSplineDoCircles, Visio recognizes circular segments in the given array of points and generates circular arcs instead of spline rows for those segments.

If flags includes visSplineAbrupt, Visio breaks the spline whenever it detects an abrupt

change of direction or curvature in the points' trail. An abrupt change of direction is defined by three consecutive points A, B, C in the list, for which the distance between B and the line segment AC is more than twice the tolerance. Visio also considers point B to be an abrupt change if one of the segments AB or BC is more than twice as long as the other. At a point where an abrupt change is detected, Visio ends the current piece (line, arc or spline) and starts a fresh one.

If flags includes visSpline1D and if the first and last points in xyArray don't coincide, DrawSpline produces a shape with 1D behavior, otherwise it produces a shape with 2D behavior.

If the first and last points in xyArray do coincide, DrawSpline produces a filled shape.

See also: DrawBezier method, DrawLine method, DrawOval method, DrawRectangle method, DrawPolyline method, FitCurve method

Example for DrawBezier, DrawSpline

```
'This VBA macro demonstrates drawing a periodic spline through five arbitrary
'points, requiring that the spline approach within 0.25 (drawing) inches of
each
'point. It allows Visio to start new segments in the path of the new shape at
'points considered abrupt.

'Although the arguments for DrawBezier and DrawPolyline are different than
those
'of DrawSpline, they also accept an array of xy points which is set up just as
'shown here.

'A C++ program demonstrating the same thing follows the VBA macro.

Public Sub Spline_Example ()

  Dim shpObj as Visio.Shape
  Dim I as Integer
  Dim xypts(1 To 5 * 2) As Double

  For I = 1 To 5
    'Set x components to 1,2,3,4,5
    xypts(i * 2 - 1) = i
    'Set y components to f(i)
    xypts(i * 2) = i * i - 7 * i + 11
  Next I

  Set shpobj = ActivePage.DrawSpline(xypts, 0.25, visSplineAbrupt)

End Sub

-----------------------------------------------------
Here is how the same example might look in in C++.

    #define NPOINTS 5
    SAFEARRAYBOUND rgsabound[1];
    rgsabound[0].lLbound = 1;
    rgsabound[0].cElements = 2 * NPOINTS;

    SAFEARRAY FAR* psa;
    if ( (psa = SafeArrayCreate(VT_R8,1,rgsabound)) )
        {
        long ix[1];
        for ( int i = 1; i <= NPOINTS; i++ )
            {
            double x = 1.0 * i;
            double y = 1.0 * (i*i - 7*i + 11);
            ix[0] = i*2-1;
            SafeArrayPutElement(psa, ix, &x);
            ix[0] = i*2;
            SafeArrayPutElement(psa, ix, &y);
            }

        LPVISIOSHAPE ipIVShape = NULL;
        HRESULT hResult = ipIVPage->DrawSpline(
```

```
            &psa,0.25,
            visSplineAbrupt,
            &ipIVShape);

if ( NOERROR == hResult )
    ipIVShape->Release();

SafeArrayDestroy(psa);
}
```

Drop method

Applies to: Document, Page, Shape

Summary: Drops a Shape object into a stencil, drawing page, or group.

Version: VISIO 2.0

Syntax: objRet = object.**Drop**(dropObject, x, y)

Element	Description
objRet	The Master or Shape object created by dropping dropObject
object	The object to receive dropObject
dropObject	The Master or Shape object to drop
x	The x-coordinate at which to place the dropped shape's center of rotation or pinx
y	The y-coordinate at which to place the dropped shape's center of rotation or piny

Remarks: Using this method is similar to dragging and dropping a shape with the mouse. The object dropped may be a master or a shape on the drawing page.

To place a shape into a group or on a drawing page, apply the Drop method to a Shape or Page object, respectively. The shape's center of rotation is positioned at the specified coordinates, and a Shape object that represents the shape that is created is returned. When applying this method to a Shape object, make sure that the Shape object represents a group.

Note: If dropObject is a Master, the pin of the master is dropped at the specified coordinates. A master's pin will often, but not necessarily, lie at its center of rotation.

Note: DropObject is typically a Visio object such as a Master or a Shape. But these isn't required. Drop will accept any OLE object that provides an IDataObject interface.

To create a new master in a stencil, apply the Drop method to a Document object that represents a stencil (the stencil must be opened as an original or a copy rather than read-only). In this case, the x and y arguments are ignored, and the new master that is created is returned.

See also: Duplicate method, DropMany method

Example for Drop, Group

```
'This VBA macro demonstrates dropping shapes using the Page, Document, and
Shape
'objects.

Public Sub Drop_Example ()

  Dim shp1Obj As Visio.Shape
  Dim shp2Obj As Visio.Shape
  Dim shp3Obj As Visio.Shape
  Dim mstObj As Visio.Master

  Set shp1Obj = ActivePage.DrawRectangle(1, 2, 2, 1)
  Set shp2Obj = ActivePage.DrawRectangle(1, 4, 2, 3)

  'Example of Page.Drop

  ActivePage.Drop shp1Obj, 3.5, 3.5
  Set shp3Obj = ActivePage.Shapes(3)

  'Example of Document.Drop - Creates a master

  Set mstObj = ActiveDocument.Drop(shp3Obj, 0, 0)

  'Example of Shape.Drop

  ActiveWindow.Select shp1Obj, visSelect
  ActiveWindow.Select shp2Obj, visSelect
  ActiveWindow.Group

  Set shp1Obj = ActivePage.Shapes(2)

  shp1Obj.Drop shp3Obj, 3, 3

End Sub
```

DropMany method

Applies to: Master, Page, Shape

Summary: Creates several new shapes on a page, in a master or in a group.

Version: VISIO 4.5

Syntax: intRet = object.**DropMany**(ObjectsToInstance, xyArray, IDArray)

Element	Description
intRet	Number of entries in xyArray that processed successfully
object	The page, master or group in which to create new shapes
ObjectsToInstance	Identifies masters or other objects to make shapes from
xyArray	An array of alternating x and y values specifying the positions for the new shapes
IDArray	Returns IDs of created shapes

Remarks: The DropMany method is like using the Page, Master or Shape.Drop method, except that it can be used to create many new Shape objects at once, rather than one per call.

DropMany creates new Shape objects on the Page, Master or group shape to which it is applied (called the "target object" in the discussion below).

DropMany returns an array of the ID's of the Shape objects it produces.

ObjectsToInstance should be a one-dimensional array of n>=1 variants. Its entries identify what to make the new Shape objects from. Visio doesn't care what the lower and upper array bounds of ObjectsToInstance's entries are. Call these vlb and vub, respectively.

If ObjectsToInstance(i) is an unknown or dispatch (or a reference to one), then the referenced object will be instanced. This is essentially equivalent to calling Drop(ObjectsToInstance(i),x,y). ObjectsToInstance(i) will often, but not necessarily, refer to a Visio Master object. But this need not be the case. It might also refer to a Visio Shape object, Selection object, or even an object from another application.

If ObjectsToInstance(i) is the integer j, then an instance of the Master in the local stencil of the target object's document whose 1-based index is j will be made.

If ObjectsToInstance(i) is the string s (or a reference to the string s), then an instance of the Master with name s in the local stencil of the target object's document will be made. S can equal either the Master's UniqueID property or its Name property.

Note: If ObjectsToInstance(i) is an integer or string, then the OnDrop cell in the events section of the produced Shape will NOT be triggered. Use Drop instead if you want the OnDrop cell to trigger.

For vlb<i<=vub, if ObjectsToInstance(i) is empty (nothing or uninitialized in VB), then entry i will cause ObjectsToInstance(j) to be instanced again, where j is the largest value <i such that ObjectsToInstance(j) isn't empty. If you want to make n instances of the same thing, only ObjectsToInstance(vlb) need be non-empty.

xyArray should be a one-dimensional array of 2*m Doubles with lowerbound xylb and

upper bound xyub, where m>=n. The values in the array tell DropMany where to position the Shape objects it produces. ObjectsToInstance(i) will be dropped at (xy[(i-1)*2+xylb],xy[(i-1)*2+xylb+1]) for 1<=i<=m.

If the entity being instanced is a master, the pin of the new Shape will be positioned at the given xy. Otherwise, the center of the dropped Shape(s) will be positioned at the given xy.

Note that m>n is allowed. For n<i<=m, the i'th thing instanced will be the same thing as the n'th thing instanced. Thus to make m>=1 instances of the same thing, you can pass a one entry ObjectsToInstance array and an m entry xyArray array.

The value r returned by DropMany is the number of xy entries in xyArray that DropMany successfully processed. If all entries processed successfully then m will be returned. Note that if some entries are successfully processed prior to an error occuring, then the produced Shapes will not be deleted and this will raise an exception yet still return a positive r.

Presuming all m xy entries process correctly, the number of new Shape objects produced by DropMany will usually equal m. In rare cases, e.g. if a Selection object gets instanced, more than m Shapes may be produced. In the event the caller cares, the number of produced Shapes can be determined by comparing the number of shapes in the target object before and after DropMany executes. The caller can assert the new Shapes will be those with the highest indices in the target object's Shapes collection.

If DropMany returns 0, IDArray returns null (nothing). Otherwise it returns a one-dimensional array of m integers indexed from 0 to m-1. IDArray is an out arg that is allocated by DropMany and ownership of which is passed to DropMany's caller. The caller should eventually perform SafeArrayDestroy on the returned array. DropMany fails if called with !IDArray or *IDArray. (VB and VBA take care of all this for you. See the example for treatment in VBA.)

If IDArray returns non-null (not nothing), then IDArray(i), 1<=i<=r, returns the ID of the Shape produced by the i'th xyArray entry, provided the i'th xyArray entry produced exactly one Shape. If the i'th xyArray entry produced multiple Shapes, then -1 is returned in the entry. All entries i, r<i<=m, return -1.

See also: Drop method, ID property

Example for DropMany

```
'This VBA macro demonstrates using the DropMany method. It drops one instance
of
'every master in the local stencil of the macro's document onto Page1 of the
'macro's document.

'A C++ program demonstrating the same thing follows the VBA macro.

Public Sub DropEachMasterOntoPage1_Example ()

  On Error GoTo handleError

  Dim masters As Visio.Masters
  Dim nMas As Integer

  Set masters = ThisDocument.Masters
  nMas = masters.Count

  ReDim ObjectsToInstance(1 To nMas) As Variant
  ReDim xyArray(1 To nMas * 2) As Double

' Iterate through each master in document. You can identify which master to
' drop by passing the DropMany Master object or the master's index or the
' master's name. By passing an object, DropMany isn't constrained to just
' dropping a master from the local stencil of the document onto which it is
being
' dropped. The object can be master from another document or another type of
' object.
' Passing integers (master indices) or strings (master names) to DropMany
' is faster than passing objects, but integers or strings can identify only
' Masters in the local stencil of the document onto which it is being dropped.
' Hence your program has to somehow get the masters in question into the local
' stencil in the first place, provided they weren't there already.
'
' The following loop shows code for all three variations. You can
' use the variation that works best in your circumstances.

Dim i as Integer
For i = 1 To nMas
    ' Pass object to DropMany.
    'Set ObjectsToInstance(i) = masters(i)

    ' Pass index of master to drop to DropMany.
    ObjectsToInstance(i) = i

    ' Pass name of master to drop to DropMany. This code assumes master names
    ' are of form "master.i" which will rarely be the real case.
    'ObjectsToInstance(i) = "master." & Format(i - 1)

    ' Pass uniqueID of master to drop to DropMany.
    'ObjectsToInstance(i) = masters(i).UniqueID

    ' Set x components of where to drop to 2,4,6,2,4,6,2,4,6,...
    xyArray(i * 2 - 1) = (((i - 1) Mod 3) + 1) * 2
```

```
    ' Set y components to 2,2,2,4,4,4,6,6,6,...
    xyArray(i * 2) = Int((i + 2) / 3) * 2
Next i

Dim IDArray() As Integer
Dim nProcessed as Integer
nProcessed = ThisDocument.Pages(1).dropmany(ObjectsToInstance, xyArray,
IDArray)

Debug.Print nProcessed
For i = LBound(IDArray) To UBound(IDArray)
    Debug.Print i; IDArray(i)
Next i

Exit Sub

handleError:
MsgBox "Error"
Exit Sub

End Sub

-----------------------------------------------------
'A C++ example that demonstrates the DropMany method.

extern "C" int RunDemo(void)
{
    // This example drops each master in the local stencil of the
    // presently active document onto the first page of that
    // document using DropMany.
    //
    HRESULT hr;
    CVisioApplication app;
    CVisioDocument doc;
    CVisioPages pages;
    CVisioPage page;
    CVisioMasters masters;
    short nmasters;

    // This sequence gets first page of active document and
    // number of masters in local stencil of active document.

    if ( VAO_SUCCESS != vaoGetObjectWrap(app) )
        goto CU;

    hr= app.ActiveDocument(doc);
    check_valid(hr, doc);

    hr= doc.Pages(pages);
    check_valid(hr, pages);

    hr= pages.Item(VVariant(1L), page);
    check_valid(hr, page);

    hr= doc.Masters(masters);
    check_valid(hr, masters);
```

```
hr= masters.Count(&nmasters);
if ( NOERROR != hr )
    goto CU;

// Set up arguments to drop many. We're going to make an
// array with nmasters entries which will hold the index of
// each master we want to drop, i,e. 1,2,3,4,5... This is an
// array of variants. We could also set entries to be references
// to objects, or to names of masters to drop. See DropMany
// remarks. For this example, we set up the array of xy
// positions indicating where on the page to drop the masters
// to arbitrary values.

SAFEARRAYBOUND rgsabound[1];
rgsabound[0].lLbound = 1;
rgsabound[0].cElements = nmasters;

SAFEARRAY FAR* psaObjs;
if ( (psaObjs = SafeArrayCreate(VT_VARIANT,1,rgsabound)) )
    {
    VARIANT* pObj;
    if ( (NOERROR == SafeArrayAccessData(psaObjs,(void**)&pObj)) )
        {
        SAFEARRAY FAR* psaXY;
        rgsabound[0].cElements = nmasters*2;
        if ( (psaXY = SafeArrayCreate(VT_R8,1,rgsabound)) )
            {
            long ix[1], iy[1];
            *ix = 1; *iy = 2;
            double x, y;
            for ( int i = 0; i < nmasters; i++, *ix +=2, *iy += 2 )
                {
                V_VT(pObj+i) = VT_I2;
                V_I2(pObj+i) = i+1;

                x = y = 1.0 * i;
                SafeArrayPutElement(psaXY, ix, &x);
                SafeArrayPutElement(psaXY, iy, &y);
                }

            // DropMany will return an array of the ids of the
            // shapes it produces and the number of entries in
            // psaXY that it successfully processed. psaIDs is
            // an out arg created by DropMany and which this
            // owes Destroy of.

            short nXYsProcessed;
            SAFEARRAY FAR* psaIDs = NULL;

            hr= page.DropMany(&psaObjs,&psaXY,&psaIDs,&nXYsProcessed);

            if ( psaIDs )
                SafeArrayDestroy(psaIDs);

            SafeArrayDestroy(psaXY);
            }
        SafeArrayUnaccessData(psaObjs);
```

```
            }
        SafeArrayDestroy(psaObjs);
        }
CU:
    return 0;
}
```

Duplicate method

Applies to: Selection, Shape, Window

Summary: Duplicates the specified object or selection.

Version: VISIO 2.0

Syntax: object.**Duplicate**

Element	Description
object	The object to duplicate

Remarks: The Duplicate method duplicates the specified object or selection and adds a copy to the same page as the original. Using the Duplicate method is equivalent to choosing the Duplicate command located on the Edit menu in Visio.
When used with a Shape object, Duplicate duplicates the shape.
When used with a Window object, Duplicate duplicates the selection.

See also: Copy method, Cut method, Delete method, Paste method

Example for Duplicate

```
*Delete Method
```

Enabled property

Applies to: Addon, Event

Summary: Determines whether or not an Addon or Event object is currently enabled.

Version: VISIO 4.0

Syntax: intVal = object.**Enabled**
object.**Enabled** = intExpression

Element	Description
intVal	0 if the object is disabled, -1 if it is enabled
object	The object to get or set the enabled property of
intExpression	0 to disable the object; non-zero to enable the object

Remarks: You can query whether an add-on is enabled at the moment. You cannot tell an add-on to enable or disable itself. Visio will not send a run message to a disabled add-on. When shown in a Visio menu, the name of a disabled add-on will be grayed. An add-on implemented by an .EXE file always reports itself as enabled. An add-on implemented by a .VSL file reports itself as enabled or disabled according to the enabling policy that the .VSL has registered for that add-on.

You can get and set the Enabled property of an Event. An Event that is disabled won't perform its action when its event occurs.

Example for Enabled

End property

Applies to: Characters, Curve

Summary: Returns or sets the ending index of the indicated Characters object representing a range of text in a shape, or returns the end point of a Curve object.

Version: VISIO 3.0

Syntax: intRet = object.**End**
object.**End** = intExpression
retVal = object.**End**

Element	Description
intRet	The current ending index of the Characters object
retVal	Ending value of Curve's parameter domain
object	The Characters or Curve object that has or gets the setting
intExpression	The new ending index of the Characters object

Remarks: The End property of a Curve object returns the ending point of a curve. A Curve describes itself in terms of its parameter domain, which is the range [Start(),End()] where End() produces the curve's ending point.

The End property of a Characters object determines the end of the text range it presently represents. The value of the End property is an index that represents the boundary between two characters, similar to an insertion point in text. Like selected text in a drawing window, a Characters object represents the sequence of characters that are affected by subsequent actions, such as the Cut or Copy method. When you first retrieve a Characters object, its current text range includes all of the shape's text. You can change the text range by setting the object's Begin and End properties. Changing the text range of a Characters object has no effect on the text of the corresponding shape.

The End property can have a value from 0 to the value of CharCount for the corresponding shape. An index of 0 places End before the first character in the shape's text. An index of CharCount places End after the last character in the shape's text. If you specify a value less than 0, Visio sets End to 0. If you specify a value that would place End inside the expanded characters of a text field, Visio sets End to the end of the field.

The value of End must always be greater than or equal to the value of Begin. If you attempt to set End to a value less than Begin, Visio sets both Begin and End to the value specified for End.

See also: Begin property, Point method, PointAndDerivatives method, Start property

Example for End

```
*Point method
```

EndTransaction method

Applies to: ShapeData

Summary: Ends a transaction for the ShapeData object.

Version: VISIO 3.0 TECH

Syntax: object.**EndTransaction**

Element	Description
object	The ShapeData object that owns the transaction

Remarks: If you need to perform multiple operations on the ShapeData object and the object it provides (Attribute and Entity objects), you can speed operations by calling BeginTransaction and EndTransaction. Call BeginTransaction before you start working and EndTransaction when all operations are complete. This forces all database operations to be performed only when the EndTransaction call is received.

See also: BeginTransaction method

Example for EndTransaction

Entities property

Applies to: EntityApp

Summary: Retrieves the Entities collection for an EntityApp object.

Version: VISIO 3.0 TECH

Syntax: objRet = object.**Entities**

Element	Description
objRet	The Entities collection of an EntityApp object
object	The EntityApp object that owns the collection

Remarks: Use the Entities property to gain access to an EntityApp object's extended entity data.

See also: Entities object

Example for Entities

EntityApps property

Applies to: ShapeData

Summary: Returns the EntityApps collection for a given shape.

Version: VISIO 3.0 TECH

Syntax: objRet = object.**EntityApps**

Element	Description
objRet	The EntityApps collection of a ShapeData object
object	The ShapeData object that owns the collection

Remarks: Before retrieving the EntityApps collection, use the PutShape method to associate a Visio shape with the ShapeData object.

See also: EntityApp object, PutShape method

Example for EntityApps

```
*PutShape Method
```

EnumDirectories method

Applies to: Application

Summary: Returns an array naming the directories Visio would search given a list of paths.

Version: VISIO 4.5

Syntax: object.**EnumDirectories** PathList,NameArray

Element	Description
object	The Application object that is to perform the enumeration
PathList	A string of full or partial path names separated by semi-colons
NameArray	Array that receives the enumerated directory names

Remarks: Several Visio properties such as AddonPaths and TemplatePaths accept and receive a string interpreted to be a list of path (directory) names separated by semi-colons. Non-fully qualified names in the list are considered to be with respect to the folder that contains the Visio program files (appObj.Path). When Visio looks for items in the named paths, it looks in the paths plus all sub-directories of the paths. And there is no guarantee that paths named in the list actually exist.

Suppose d:\Add-ons is a path that exists and e:\Add-ons is a path that doesn't exist. If Visio's executable file is installed in c:\Visio, and AddonPaths is "Add-ons;d:\Add-ons;e", Visio looks for add-ons in both c:\Visio\Add-ons and d:\Add-ons, plus in any of their sub-directories.

The pupose of EnumDirectories is to accept a string such as one that AddonPaths might produce and return a list of the directories that Visio will actually enumerate when processing such a string.

If EnumDirectories succeeds, NameArray returns a one-dimensional array of n strings indexed from 0 to n-1. Each string will be the fully qualified name of a directory that does exist. The list will name those directories designated in the path list that actually do exist plus all sub-directories of those directories.

NameArray is an out argument that is allocated by EnumDirectories and ownership of which is passed back to the caller. The caller should eventually perform SafeArrayDestroy on the returned array. Note that SafeArrayDestroy has the side effect of freeing the strings referenced by the array's entries. EnumDirectories fails if called with !NameArray or *NameArray. (VB and VBA take care of all this for you. See the example for treatment in VBA.)

See also: Path property, AddonPaths property, DrawingPaths property, HelpPaths property, StartupPaths property, StencilPaths property, TemplatePaths property

Example for EnumDirectories

```
'This VBA macro demonstrates using the EnumDirectories property.

Public Sub EnumAddonDirs()

    ' Get the names of all the directories in which Visio looks for add-ons.
    Dim dirnames() As String
    Application.EnumDirectories Application.AddonPaths, dirnames
    Dim lb As Integer, ub As Integer
    lb = LBound(dirnames)
    ub = UBound(dirnames)
    While lb <= ub
        Debug.Print dirnames(lb)
        lb = lb + 1
    Wend

End Sub
```

Error property

Applies to: Cell

Summary: Returns the error code generated by the last evaluation of the indicated cell's formula.

Version: VISIO 2.0

Syntax: intRet = object.**Error**

Element	Description
intRet	The error code from the last evaluation of the cell's formula
object	The Cell object to examine

Remarks: When a cell's formula is evaluated, an error code is generated along with the result. The Error property provides access to this error code. Constants for valid error codes are declared by the Visio type library (and visconst.bas):

visErrorSuccess = 0
visErrorDividebyZero = 39
visErrorValue = 47
visErrorReference = 55
visErrorName = 61
visErrorNumber = 68
visErrorNotAvailable = 74

Note: visErrorSuccess means that no error occurred during the last evaluation of the indicated cell's formula.

Example for Error

Event property

Applies to: Event

Summary: Gets or sets the event code of an Event object.

Version: VISIO 4.0

Syntax: intRet = object.**Event**
object.**Event** = eventCode

Element	Description
intRet	The current event code
object	The Event object that has or gets the event code
eventCode	The new event code

Remarks: An Event object represents an event-action pair. When the event occurs, the action is performed.

If the action code of the Event object is visActCodeRunAddon, the event also specifies the target of the action and the arguments to send to the target. This information is stored in the Target and TargetArgs properties, respectively.

If the action code of the Event object is visActCodeAdvise, the event also specifies the object to receive event notifications (sometimes called the sink object) and arguments to send to the sink object along with the notification.

Event codes are declared by the Visio type library (and visconst.bas). They are prefixed with "visEvt" and are listed in event topics in this help file. For a list, display the Contents topic and choose List of Events.

A program can use the Trigger method to cause an Event object's action to be performed without waiting for the event to occur.

See also: Action property, Add method, AddAdvise method, EventInfo property, EventList object, Target property, TargetArgs property, Trigger method

Example for Event

EventInfo property

Applies to: Application

Summary: Gets additional information, if any, associated with an event.

Version: VISIO 4.0

Syntax: strRet = object.**EventInfo**(eventSeqNum)

Element	Description
strRet	Additional information about the event
object	The Application object to examine
eventSeqNum	The sequence number of the event to examine, or visEvtIDMostRecent (0)

Remarks: When Visio fires an event, there are a small number of cases when more information about the event is available than is actually sent with the event notification. To obtain this information, an event target can get the EventInfo property. If an event does not record extra information, EventInfo returns an empty string. If it does, EventInfo returns a string whose contents are specific to the event in question. If Visio no longer has information for the specified event, EventInfo generates an error.

To get information about the most recently fired event, pass visEvtIDMostRecent (0) as an argument to EventInfo.

You can also pass the firing sequence number of a fired event to EventInfo. The first event an instance of Visio fires will have sequence number 1, the second 2, and so on.

If the Action of an Event object is visActCodeRunAddon, then the command line string passed to the Addon will contain in it a substring of the form "/eventid=<sequence number>".

Note: Even though labeled with "/eventid," the <sequence number> passed in the command line string shouldn't be confused with the ID property of the firing Event object, which identifies the Event in its EventList collection. The number being passed is actually the firing sequence number.

If the Action is visActCodeAdvise, the sequence number is passed as an argument to the VisEventProc procedure implemented by the target object.

If an event target queries EventInfo immediately after being triggered, the most recent event and the event whose sequence number was passed to the target are usually the same. But if the target is an add-on implemented by an .EXE file, this may occasionally not be the case, because the .EXE and Visio are separate tasks that aren't modal with respect to each other. To ensure that the information returned by EventInfo is associated with the same event that triggered the add-on, it can pass <sequence number> as an argument to EventInfo.

The only events that presently pass extra data using EventInfo are ShapesDeleted (visEvtCodeShapeDelete) and ShapeChanged (visEvtMod+visEvtShape). The individual entries for these events specify the information passed in EventInfo.

See also: Action property, Event object, Event property, Target property, TargetArgs property,

ShapesDeleted event, ShapeChanged event

Example for EventInfo

EventList property

Applies to: Application, Cell, Characters, Document, Documents, Event, Layer, Layers, Master, Masters, Page, Pages, Selection, Shape, Shapes, Style, Styles, Window, Windows

Summary: Returns the EventList collection of an object or the EventList collection that contains an Event object.

Version: VISIO 4.0

Syntax: objRet = object.**EventList**

Element	Description
objRet	The EventList collection
object	The object that owns the collection or the Event object that belongs to the collection

See also: Event object, EventList object

Example for EventList

```
*Add Method
```

EventsEnabled property

Applies to: Application

Summary: Determines whether Visio fires events.

Version: VISIO 4.5

Syntax: intRet = object.**EventsEnabled**
object.**EventsEnabled** = intExpression

Element	Description
intRet	False (0) if event firing is inhibited; True (-1) if event firing is enabled
object	The object that has or gets the setting
intExpression	False (0) to inhibit event firing; True (non-zero) to enable event firing

Remarks: If EventsEnabled is false, Visio will not fire events, run Add-ons, or execute VBA code when evaluating RUNADDON operands in cell formulas.

By default, EventsEnabled will be true when an instance of Visio launches.

You may want to inhibit event firing if you have code behind events such as DocumentOpened or DocumentCreated that does not work properly, or if you suspect someone of attempting to incorporate a virus into a document. To set EventsEnabled to false:

1. From the Tools menu, choose Options.
2. In Options, click Advanced.
3. Uncheck Enable Automation Events.

See also: DocumentOpened event, DocumentCreated event

Example for EventsEnabled

ExecuteLine method

Applies to: Document

Summary: Executes a line of Visual Basic code

Version: VISIO 4.5

Syntax: object.**ExecuteLine** stringExpression

Element	Description
object	The Document object whose VBA project is to execute code
stringExpression	A string that will be interpreted as VBA code

Remarks: The VBA project of the Document object is told to execute the supplied string. VBA will treat the string much like it would treat the same string typed into its immediate window.

Some possibilities are:

```
ThisDocument.ExecuteLine("SomeMacro")
    ' Executes the macro (argumentless procedure) named SomeMacro
    ' that is in some module of the VBA project of ThisDocument.

ThisDocument.ExecuteLine("SomeProc 1, 2, 3")
    ' Executes the procedure named SomeProc and passes it 3 arguments.

ThisDocument.ExecuteLine("Module1.SomeProc 1, 2, 3")
    ' Same as prior example, but procedure name qualified with module name.

ThisDocument.ExecuteLine("UserForm1.Show")
    ' Shows the form UserForm1.

ThisDocument.ExecuteLine("debug.print ""some string""")
    ' Prints "some string" to VBA's immediate window.

ThisDocument.ExecuteLine("debug.print Documents.Count")
    ' Prints number of open documents to immediate window.

ThisDocument.ExecuteLine("ThisDocument.Save")
    ' Tells ThisDocument to save itself.
```

See also: ParseLine method, VBE property, VBProject property, AddonName property

Example for ExecuteLine

Export method

Applies to: Master, Page, Selection, Shape

Summary: Exports the indicated object from Visio.

Version: VISIO 3.0

Syntax: object.**Export** stringExpression

Element	Description
object	The object to export
stringExpression	The name of the file to receive the exported object

Remarks: The Export method exports a Page object, Master object, Selection object, or Shape object to the file specified by stringExpression. StringExpression must be a fully qualified pathname. Names specifying only a relative or partial path will generate an error.

The filename extension indicates which export filter to use. If the filter is not installed, Export returns an error. Export uses the default preference settings for the specified filter and does not prompt the user for non-default arguments.

The Export method of a Page object supports saving to HTML files using the extension .HTM or .HTML. Pages are exported using the settings that were last selected in the Save As HTML dialog box.

If specified file already exists, it is replaced without prompting the user.

See also: Import method

Example for Export

ExportIcon method

Applies to: Master

Summary: Export the icon for a Master to a named file or to the clipboard.

Version: VISIO 4.5

Syntax: object.**ExportIcon** stringExpression, flags

Element	Description
object	The master whose icon is to be exported
stringExpression	The file to export the icon to
format	The format to write to write the exported file in

Remarks: If the string is empty, the master's icon is copied to the clipboard.

If the value of flags is visIconFormatVisio (0), the icon is exported in Visio's internal icon format. ImportIcon accepts files written in this format.

If the value of flags is visIconFormatBMP (2), the icon is exported in bitmap (BMP) format.

See also: ImportIcon method

Example for ExportIcon

ExtraInfo property

Applies to: Hyperlink

Summary: Returns or sets extra URL request information used to resolve the hyperlink's URL.

Version: VISIO 5.0

Syntax: strRet = object.**ExtraInfo**
object.**ExtraInfo** = stringExpression

Element	Description
strRet	The current value of the field
object	The object that has or gets the value
stringExpression	The new value for the field

Remarks: Setting the ExtraInfo property of a shape's Hyperlink object is optional.

Setting the ExtraInfo property of a Hyperlink object is equivalent to setting the value of the ExtraInfo cell in the shape's hyperlink row.

You might, for example, set the Hyperlink object's ExtraInfo to be the coordinates of an image map, the contents of a form, or a file name.

If the ExtraInfo you're providing contains reserved characters other than spaces, they must be escaped. For example:

For "NAME=John Smith", set ExtraInfo to "Name=John Smith"
For "PATH=C:\TEMP", set ExtraInfo to "PATH=C%3A%5CTEMP"

See also: Address property, SubAddress property, Description property, Frame property, CreateURL method

Example for ExtraInfo

FieldCategory property

Applies to: Characters

Summary: Returns the field category for the field represented by the indicated object.

Version: VISIO 3.0

Syntax: intRet = object.**FieldCategory**

Element	Description
intRet	The field category
object	The Characters object to examine

Remarks: If the Characters object does not contain a field or if it contains non-field characters, FieldCategory returns an exception. Check the IsField property of the Characters object before getting its FieldCategory property.

Field categories correspond to those in the Category list in Visio's Field dialog box. Field category constants are declared by the Visio type library (and visconst.bas).

See also: AddField method, FieldCode property, FieldFormat property, FieldFormula property

Example for FieldCategory

FieldCode property

Applies to: Characters

Summary: Returns the field code for the field represented by the indicated object.

Version: VISIO 3.0

Syntax: intRet = object.**FieldCode**

Element	Description
intRet	The field code
object	The Characters object to examine

Remarks: If the Characters object does not contain a field or contains non-field characters, FieldCode returns an exception. Check the IsField property of the Characters object before getting its FieldCode property.

Field codes correspond to the fields in the Field list in Visio's Field dialog box. Field code declared by the Visio type library (and visconst.bas).

See also: AddField method, FieldCategory property, FieldFormat property, FieldFormula property

Example for FieldCode

FieldFormat property

Applies to: Characters

Summary: Returns the field format for the field represented by the indicated object.

Version: VISIO 3.0

Syntax: intRet = object.**FieldFormat**

Element	Description
intRet	The field format
object	The Characters object to examine

Remarks: If the Characters object does not contain a field or contains non-field characters, FieldFormat returns an exception. Check the IsField property of the Characters object before getting its FieldFormat property.

Field formats correspond to the formats in the Format list in Visio's Field dialog box. Field format constants are declared by the Visio type library (and visconst.bas).

See also: AddCustomField method, AddField method, FieldCategory property, FieldCode property, FieldFormula property

Example for FieldFormat

FieldFormula property

Applies to: Characters

Summary: Returns the formula of the custom field represented by the indicated object.

Version: VISIO 3.0

Syntax: strRet = object.**FieldFormula**

Element	Description
strRet	The formula of the custom field
object	The Characters object to examine

Remarks: If the Characters object does not contain a field, if it contains non-field characters, or if the field is not a custom field, FieldFormula returns an exception. Check the IsField and FieldCategory properties of the Characters object before getting its FieldFormula property.

The formula returned by FieldFormula corresponds to the formula that appears in the Custom Formula box in Visio's Field dialog box.

See also: AddCustomField method, FieldCategory property, FieldCode property, FieldFormat property

Example for FieldFormula

FillBasedOn property

Applies to: Style

Summary: Gets or sets the fill style that the Style object is based on.

Version: VISIO 4.0

Syntax: strVal = object.**FillBasedOn**
object.**FillBasedOn** = styleName

Element	Description
strVal	The name of the current fill style
object	The Style object that is based on the fill style
styleName	The name of the new fill style

Remarks: To base a style on no style, set FillBasedOn to a null string ("").

See also: BasedOn property, LineBasedOn property, TextBasedOn property

Example for FillBasedOn

FillStyle property

Applies to: Selection, Shape

Summary: Returns or sets the fill style for an object

Version: VISIO 2.0

Syntax: strRet = object.**FillStyle**
object.**FillStyle** = stringExpression

Element	Description
strRet	The current fill style
object	The Shape or Selection object that has or gets the fill style
stringExpression	The name of the fill style to apply

Remarks: Setting the FillStyle property is equivalent to selecting a style from the Fill style list on the toolbar.

Setting a style to a non-existent style generates an error. Setting one kind of style to an existing style of another kind (for example, setting FillStyle to a line style) does nothing. Setting one kind of style to an existing style that has more than one set of attributes changes only the attributes for that component. For example, setting FillStyle to a style with line, text, and fill attributes changes only the fill attributes.

To preserve a shape's local formatting, use the FillStyleKeepFmt property.

See also: FillStyleKeepFmt property

Example for FillStyle

*DrawRectangle Method

FillStyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a fill style to an object while preserving local formatting.

Version: VISIO 2.0

Syntax: object.**FillStyleKeepFmt** = stringExpression

Element	Description
object	The Shape or Selection object to which the fill style is applied
stringExpression	The name of the fill style to apply

Remarks: Setting the FillStyleKeepFmt property is equivalent to checking the Preserve Local Formatting option in Visio's Style dialog box.

Setting a style to a non-existent style generates an error. Setting one kind of style to an existing style of another kind (for example, setting FillStyleKeepFmt to a line style) does nothing. Setting one kind of style to an existing style that has more than one set of attributes changes only the attributes for that component (for example, setting FillStyleKeepFmt to a style with line, text, and fill attributes changes only the fill attributes).

See also: FillStyle property

Example for FillStyleKeepFmt

FilterPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for import and export filters.

Version: VISIO 4.0

Syntax: strRet = object.**FilterPaths**
object.**FilterPaths** = pathsStr

Element	Description
strRet	A text string containing a list of folders
object	An Application object
pathsStr	A text string containing a list of folders

Remarks: The string passed to and received from FilterPaths is the same string shown in Visio's File Paths dialog (accessible from the Tools/Options dialog). This same string is stored in Visio's profile (.ini) file (appObj.ProfileName) in the entry whose key is "FiltersPath."

Unlike similar methods such as AddonPaths and TemplatePaths, you can name only one path in FilterPaths and Visio will not look for filters in sub-folders of the path you name.

If the named path is not fully qualified, Visio looks for the folder in the folder that contains the Visio program files (appObj.Path).

See also: AddonPaths property, DrawingPaths property, HelpPaths property, StartupPaths property, StencilPaths property, TemplatePaths property, ProfileName property, Path property

Example for FilterPaths

FitCurve method

Applies to: Selection, Shape

Summary: Reduces the number of geometry segments in a shape or shapes by replacing them with similar spline, arc and line segments.

Version: VISIO 4.1

Syntax: object.**FitCurve** tolerance, flags

Element	Description
object	The shape or selection of shapes whose path is to be replaced
tolerance	How closely the resulting paths must match the shape's original paths
flags	Flags that influence how the shape is drawn

Remarks: FitCurve replaces the existing geometry segments of a shape with spline, arc and line segments that approximate the paths of the initial segments. Typically this reduces the number of segments in the shape.

When applied to a Selection object, FitCurve optimizes each of the shapes in the selection. It does not combine the selected shapes into a single shape.

The paths resulting from FitCurve fall within the given tolerance of the initial paths. Tolerance should be in internal drawing units (inches). To match the initial paths exactly, specify a tolerance of 0.

The flags parameter is a bit mask that specifies options for optimizing the paths. Its value should be a combination of 0 or more of the following values:

```
visSplinePeriodic    =    1
visSplineDoCircles   =    2
visSplineAbrupt      =    4
```

If flags includes visSplinePeriodic, Visio produces periodic splines if appropriate.

If flags includes visSplineDoCircles, Visio recognizes circular segments in the shape(s) and generates circular arcs instead of spline rows for those segments.

If flags includes visSplineAbrupt, Visio breaks the resulting splines whenever it detects an abrupt change of direction or curvature in a path.

See also: Combine method, DrawSpline method

Example for FitCurve

Flags property

Applies to: Color

Summary: Gets or sets the flags that specify how a Color object is used.

Version: VISIO 4.0

Syntax: intRet = object.**Flags**
object.**Flags** = intVal

Element	Description
intRet	The current value of the color's flags component
object	The Color object that has or gets the component
intVal	The new value of the color's flags component

Remarks: The Flags property of a Color object corresponds to the peFlags member of a Windows PALETTEENTRY data structure. For details, search the Windows SDK online help for PALETTEENTRY.

See also: Blue property, Green property, Red property, PaletteEntry property

Example for Flags

Flavor property

Applies to: UI Object

Summary: Visio 5.0 no longer supports the notion of flavor. The property is now ignored.

Version: VISIO 4.0

Syntax: intRet = object.**Flavor**
object.**Flavor** = newFlavor

Element	Description
intRet	The current product suite on which the toolbars are based
object	The UI object to examine
newFlavor	The product suite on which to base the toolbars

Remarks: The Visio user interface uses the Microsoft Office toolbar set.

See also: ShowToolbar property

Example for Flavor

FlipHorizontal method

Applies to: Selection, Shape

Summary: Flips an object horizontally.

Version: VISIO 2.0

Syntax: object.**FlipHorizontal**

Element	Description
object	The Shape or Selection object to flip

See also: FlipVertical method, ReverseEnds method, Rotate90 method

Example for FlipHorizontal

FlipVertical method

Applies to: Selection, Shape

Summary: Flips an object vertically.

Version: VISIO 2.0

Syntax: object.**FlipVertical**

Element	Description
object	The Shape or Selection object to flip

See also: FlipHorizontal method, ReverseEnds method, Rotate90 method

Example for FlipVertical

Follow method

Applies to: Hyperlink

Summary: Causes Visio to navigate the hyperlink.

Version: VISIO 5.0

Syntax: object.**Follow**

Element	Description
object	The Hyperlink object whose target is to be navigated to

See also: Hyperlink property, AddHyperlink method, FollowHyperlink method

Example for Follow, NewWindow

```
'The following VBA macro adds a Hyperlink object to a shape, sets its Address
'and NewWindow properties, and uses the Follow method to navigate the
hyperlink.

Sub NavigateHyperlink()

  dim myHlinkObj as Visio.Hyperlink

  Set myHlinkObj = ActivePage.DrawRectangle(0,0,5,5).AddHyperlink

  myHlinkObj.Address = "http://www.visio.com
  myHlinkObj.NewWindow = False
  myHlinkObj.Follow

End Sub
```

FollowHyperlink method

Applies to: Document

Summary: Navigates to an arbitrary document-based hyperlink.

Version: VISIO 5.0

Syntax: object.**FollowHyperlink** (Address, SubAddress, [ExtraInfo], [Frame], [NewWindow], [res1], [res2], [res3])

Element	Description
object	The Document object that is to navigate to the designated hyperlink
Address	The address that is to be navigated to
SubAddress	The subaddress that is to be navigated to. Optional.
ExtraInfo	Extra URL request information to use in resolving the URL.
Frame	The HTML frame that is to be navigated to. Optional.
NewWindow	Specifies if a new window is to be opened.
res1	Unused. Optional.
res2	Unused. Optional.
res3	Unused. Optional.

Remarks: The arguments to Document.Hyperlink are equivalent to the cell names of the Hyperlink row in the ShapeSheet section represented by visSectionObject.

If SubAddress information is not needed, pass an empty string. Handle optional arguments in the following manner. From VB or VBA, do not pass a value. From C/C++, pass an empty variant.

NOTE: Visio 4.5 provided an undocument Document.Hyperlink method with the following signature:

HRESULT FollowHyperlink[in] BSTR Target, [in] BSTR Location);

This method remains in Visio 5.0 but has been renamed to FollowHyperlink45 as follows.

HRESULT FollowHyperlink45[in] BSTR Target, [in] BSTR Location);

See also: Address property, SubAddress property, Frame property, NewWindow property, ExtraInfo property

Example for FollowHyperlink

Fonts property

Applies to: Document

Summary: Returns the Fonts collection of a Document object.

Version: VISIO 4.0

Syntax: objRet = object.**Fonts**

Element	Description
objRet	The Fonts collection of the Document object
object	The Document object that owns the collection

See also: Document object, Fonts object

Example for Fonts

ForeignType property

Applies to: OLEObject, Shape

Summary: Returns the subtype of a Shape object that represents an object foreign to Visio.

Version: VISIO 4.1

Syntax: retVal = object.**ForeignType**

Element	Description
retVal	The foreign type of the Shape object
object	The Shape object to examine

Remarks: If the Type property of a Shape object returns any value other than visTypeForeignObject, ForeignType returns the same value as the Shape's type property. If the Type property of a Shape object returns visTypeForeignObject, ForeignType returns a combination of the following values:

visTypeMetafile = &H0010
visTypeBitmap = &H0020
visTypeIsLinked = &H0100
visTypeIsEmbedded = &H0200
visTypeIsControl = &H0400
visTypeIsOLE2 = &H8000

If the shape represents an OLE 2.0 embedded object, for example, its foreign type will be &H8200.

See also: ClassID property, Object property, ObjectIsInherited property, ProgID property, Type property

Example for ForeignType

FormatResult method

Applies to: Application

Summary: Formats a string or number into a string according to a format picture, using specified units for scaling and formatting.

Version: VISIO 4.5

Syntax: stringRet = object.**FormatResult**(stringOrNumber, unitsIn, unitsOut, format)

Element	Description
stringRet	The evaluated result formatted according to format and unitsOut
object	The Application object that is to perform the formatting operation
stringOrNumber	String or number to be formatted
unitsIn	Measurement units to attribute to stringOrNumber
unitsOut	Measurement units to express result in
format	Picture of what result string should look like

Remarks: StringOrNumber is the expression to be formatted. StringOrNumber can be passed as a string, floating point number or integer.

If passed as a string, stringOrNumber might be the formula or prospective formula of a cell (what you'd see in a ShapeSheet window with View Formulas selected), or the result or prospective result of a cell (what you'd see with View Values selected) expressed as a string. FormatResult will evaluate the string and format the result. Because the string is being evaluated outside the context of being the formula of a particular cell, FormatResult will return an error if the string contains any cell references.

Possible values for StringOrNumber include:

1.7
3
"2.5"
"4.1 cm"
"12 ft - 17 in + (12 cm / SQRT(7))"

UnitsIn and UnitsOut can be strings such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property. If stringOrNumber is a string, UnitsIn specifies how to interpret the evaluated result and is only used if the result is a scalar. For example, the expression "4 * 5 cm" evaluates to 20 cm, which is not a scalar so unitsIn is ignored. The expression "4 * 5" evaluates to 20 which is a scalar and is interpreted using the specified unitsIn.

UnitsOut specifies what units the returned string should be expressed in. If you want the results expressed in the same units as the evaluated expression, pass "NOCAST" or visNoCast.

Format is a string that specifies a template or picture for what the string produced by FormatResult should look like. See the help for Visio's FORMAT() ShapeSheet function for details. A few of the possibilities are:

: Ouput a single digit, but not if it's a leading or trailing 0.

0 : Output a single digit, event if it is a leading or trailing 0.
. : Decimal placeholder.
, : Thousands separator.
"text" or 'text' : Output enclosed text as is.
\c : Output the character c.

Examples where string is specified:
Debug.Print application.FormatResult("0.5 * 2", "ft", "ft", "#.00 u") >>> 1.00 ft.
Debug.Print application.FormatResult("0.5 * 2", "ft", "in", "#.00 u") >>> 12.00 in.
Debug.Print application.FormatResult("1 cm", "ft", "in", "#.00 u") >>> .39 in.
Debug.Print application.FormatResult("1 cm", "ft", "NOCAST", "#.00 u") >>> 1.00 cm.
Debug.Print application.FormatResult("1 cm", "ft", "", "0.00 u") >>> 0.39
Debug.Print application.FormatResult("1 cm", "ft", "bozo", "#.00 u") >>> exception: Bad measurement unit.

Examples where number is specified:
Debug.Print application.FormatResult(1, "ft", "ft", "#.00 u") >>> 1.00 ft.
Debug.Print application.FormatResult(1, "ft", "in", "#.00 u") >>> 12.00 in.
Debug.Print application.FormatResult(1.0, "in", "ft", "#.00 u") >>> .08 ft.
Debug.Print application.FormatResult(1.0, visFeet, "", "#.00 u") >>> 12.00
Debug.Print application.FormatResult(1, "bozo", "in", "#.00 u") >>> exception: Bad measurement unit.

See also: ConvertResult method, Result property

Example for FormatResult

Formula property

Applies to: Cell

Summary: Returns or sets the formula for a Cell object.

Version: VISIO 2.0

Syntax: strRet = object.**Formula**
object.**Formula** = stringExpression

Element	Description
strRet	The cell's formula
object	The Cell object that contains the formula
stringExpression	The new formula for the cell

Remarks: If a cell's formula is protected with the GUARD function, you must use the FormulaForce property to change the cell's formula.

See also: FormulaForce property

Example for Formula

*AddSection Method

FormulaChanged event

Applies to: Application, Cell, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs after the formula changes in a cell in a Visio document.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtFormula (&H3000)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Cell whose formula just changed
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source FormulaChanged using the AddAdvise method. In addition, FormulaChanged is included in the event set of all the objects in the Applies to list except the Document object. For those objects you can use VBA Dim WithEvents variables to sink FormulaChanged. For performance considerations, the Document object's event set does not include FormulaChanged. To sink FormulaChanged from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, CellChanged event

Example for FormulaChanged

FormulaForce property

Applies to: Cell

Summary: Sets the formula in a Cell object, even if the formula is protected with a GUARD function.

Version: VISIO 2.0

Syntax: object.**FormulaForce** = stringExpression

Element	Description
object	The Cell object that contains the formula
stringExpression	The new formula for the cell

Remarks: Many of the SmartShapes provided with Visio have guarded cells to maintain their smart behavior. Changing the formula in a guarded cell might change the shape's behavior in unexpected ways.

See also: Formula property

Example for FormulaForce

Fragment method

Applies to: Selection, Window

Summary: Breaks selected shapes into smaller shapes.

Version: VISIO 2.0

Syntax: object.**Fragment**

Element	Description
object	The Window or Selection object that contains the shapes to fragment

Remarks: Using the Fragment method is equivalent to choosing the Fragment command located on the Operations submenu on the Shape menu in Visio. The produced shapes are the topmost shapes in the ContainingShape of the selected shapes. They inherit the formatting of the first selected shape and have no text. The original shapes are deleted.

If the object being operated on is a Selection object, it has no shapes selected in it when the operation is complete.

See also: Combine method, Intersect method, Join method, Subtract method, Trim method, Union method, ContainingShape property

Example for Fragment

Frame property

Applies to: Hyperlink

Summary: Returns or sets the name of an HTML frame in the shape's Hyperlink object to which a shape's hyperlink will navigate.

Version: VISIO 5.0

Syntax: strRet = object.**Frame**
object.**Frame** = stringExpression

Element	Description
strRet	The current value of the field
object	The object that has or gets the value
stringExpression	The new value for the field

Remarks: Setting the Frame property of a shape's Hyperlink object is optional and only applies when Visio is open as an ActiveX Document in an ActiveX browser, for example, Microsoft Internet Explorer 3.0 or later.

This is equivalent to setting the result of the Frame cell in the shape's hyperlink row.

See also: Address property, SubAddress property, NewWindow property, ExtraInfo property, Description property

Example for Frame

FromCell property

Applies to: Connect

Summary: Returns the cell from which a connection originates.

Version: VISIO 2.0

Syntax: objRet = object.**FromCell**

Element	Description
objRet	The cell from which the connection originates
object	The Connect object to examine

Remarks: A connection is defined by a reference in a cell in the shape from which the connection originates to a cell in the shape to which the connection is made. For a 2-D shape, a connection may be defined in any of the six cells in its Alignment section. In this case, the FromCell property returns the Alignment cell that is involved in the connection.

For a 1-D shape, a connection may be defined in its 1-D Endpoints section. In this case, the FromCell property returns the appropriate cell object if the 1-D shape is glued to a guide.

If the 1-D shape is glued to a 1-D or 2-D shape, FromCell returns either the BeginX or EndX cell object, depending on which endpoint is glued.

For both 2-D and 1-D shapes, a connection may be defined in the X and Y cells of one row in their Controls section. In this case, the FromCell property returns the X cell object.

See also: FromPart property, FromSheet property, GlueTo method, ToCell property

Example for FromCell, FromPart, FromSheet, ToCell, ToPart, ToSheet

```
'This Visual Basic program demonstrates extracting connection information.
'This program displays the connection information in the debug window.

Sub ListConnections ()
    Dim appVisio As object
    Dim docObj As object
    Dim pagsObj As object
    Dim pagObj As object
    Dim shpsObj As object
    Dim shpObj As object
    Dim fromObj As object
    Dim fromData As Integer
    Dim fromStr As String
    Dim toObj As object
    Dim toData As Integer
    Dim toStr As String
    Dim consObj As object
    Dim conObj As object
    Dim curShapeIX As Integer
    Dim i As Integer

    Set appVisio = GetObject(, "visio.application")
    Set docObj = appVisio.ActiveDocument
    Set pagsObj = docObj.Pages
    Set pagObj = pagsObj(1)
    Set shpsObj = pagObj.Shapes

    For curShapeIX = 1 To shpsObj.Count
        Set shpObj = shpsObj(curShapeIX)
        Set consObj = shpObj.Connects

        For i = 1 To consObj.Count
            Set conObj = consObj(i)
            Set fromObj = conObj.FromSheet
            fromData = conObj.FromPart
            Set toObj = conObj.ToSheet
            toData = conObj.ToPart

        'FromPart property values
            If fromData = visConnectError Then
                fromStr = "error"
            ElseIf fromData = visNone Then
                fromStr = "none"
            ElseIf fromData = visLeftEdge Then
                fromStr = "left"
            ElseIf fromData = visCenterEdge Then
                fromStr = "center"
            ElseIf fromData = visRightEdge Then
                fromStr = "right"
            ElseIf fromData = visBottomEdge Then
                fromStr = "bottom"
            ElseIf fromData = visMiddleEdge Then
                fromStr = "middle"
```

```
            ElseIf fromData = visTopEdge Then
                fromStr = "top"
            ElseIf fromData = visBeginX Then
                fromStr = "beginX"
            ElseIf fromData = visBeginY Then
                fromStr = "beginY"
            ElseIf fromData = visBegin Then
                fromStr = "begin"
            ElseIf fromData = visEndX Then
                fromStr = "endX"
            ElseIf fromData = visEndY Then
                fromStr = "endY"
            ElseIf fromData = visEnd Then
                fromStr = "end"
            ElseIf fromData >= visControlPoint Then
                fromStr = "controlPt_" & CStr(fromData - visControlPoint + 1)
            Else
                fromStr = "???"
            End If

            If toData = visConnectError Then
                toStr = "error"
            ElseIf toData = visNone Then
                toStr = "none"
            ElseIf toData = visGuideX Then
                toStr = "guideX"
            ElseIf toData = visGuideY Then
                toStr = "guideY"
            ElseIf toData >= visConnectionPoint Then
                toStr = "connectPt_" & CStr(toData - visConnectionPoint + 1)
            Else
                toStr = "???"
            End If

            Debug.Print "from " & fromObj.Name & " " & fromStr;
            Debug.Print " to "; toObj.Name & " " & toStr & "."
        Next i
    Next curShapeIX
End Sub
```

FromConnects property

Applies to: Shape

Summary: Returns a Connects collection of the shapes connected to a shape.

Version: VISIO 4.5

Syntax: objRet = object.**FromConnects**

Element	Description
objRet	Connects collection of shapes connected to this shape
object	The Shape object that owns the collection

Remarks: The FromConnects property of a shape returns a Connects collection that contains every Connect for which the shape is the ToSheet. This tells you all the shapes connected to the indicated shape.

To obtain a Connects collection that contains every Connect for which the shape is the FromSheet, use the shape's Connects property. This tells you all the shapes to which the shape is connected.

See also: Connects property, Connect object, Connects object, FromSheet property, ToSheet property

Example for FromConnects

FromPart property

Applies to: Connect

Summary: Returns the part of a shape from which a connection originates.

Version: VISIO 2.0

Syntax: retVal = object.**FromPart**

Element	Description
retVal	The part of the shape where the connection originates
object	The Connect object to examine

Remarks: The following constants declared by the Visio type library (and visconst.bas) show return values for the FromPart property:

visConnectFromError = -1
visFromNone = 0
visLeftEdge = 1
visCenterEdge = 2
visRightEdge = 3
visBottomEdge = 4
visMiddleEdge = 5
visTopEdge = 6
visBeginX = 7
visBeginY = 8
visBegin = 9
visEndX = 10
visEndY = 11
visEnd = 12
visControlPoint = 100

See also: FromSheet property, ToPart property

Example for FromPart

```
*FromCell Property
```

FromSheet property

Applies to: Connect, Connects

Summary: Returns the shape from which a connection or connections originate.

Version: VISIO 2.0

Syntax: objRet = object.**FromSheet**

Element	Description
objRet	The shape from which the connections originate
object	The Connect object or Connects collection to examine

Remarks: Connect.FromSheet is always unambiguous. It will always return the shape from which Connect originates.

Connects represents several connections. If every connection represented by the collection originates from the same shape, Connects.FromSheet will return that shape. Otherwise it returns nothing and does not raise an exception.

See also: GlueTo method, ToSheet property

Example for FromSheet

```
*FromCell Property
```

FullName property

Applies to: Document

Summary: Returns the name of a document, including the drive and path.

Version: VISIO 2.0

Syntax: strRet = object.**FullName**

Element	Description
strRet	The filename of the document
object	The Document object to examine

Remarks: Use the FullName property to obtain a document's drive, folder path, and filename as one string. The returned value can include UNC drive names (for example, \\bob\leo.)

See also: Name property, Path property

Example for FullName

GeometryCount property

Applies to: Shape

Summary: Returns the number of Geometry sections for a shape.

Version: VISIO 2.0

Syntax: intRet = object.**GeometryCount**

Element	Description
intRet	The number of Geometry sections for the shape
object	The Shape object to examine

Remarks: GeometryCount equals 0 for groups and guides.

See also: AddSection method, CellsSRC property

Example for GeometryCount

```
'This VBA macro demonstrates using the GeometryCount property.

'To run this macro, first insert a user form with a list box into your
project.

Public Sub IterateGeometry_Example ()

Dim shpObj As Visio.Shape       ' shape object to work on
Dim curGeomSect As Integer      ' Section number for accessing geometry
section
Dim curGeomSectIndx As Integer  ' Loop variable for geometry sections
Dim nRows As Integer            ' number of rows in section
Dim nCells As Integer           ' number of cells in row
Dim curRow As Integer           ' current row number (0 based)
Dim curCell As Integer          ' current cell index (0 based)
Dim nSects As Integer           ' number of geometry sections in shape

    ' Retrieve the first shape from the ActivePage
    Set shpObj = ActivePage.Shapes(1)

    ' Make sure the listbox is cleared
    UserForm1.ListBox1.Clear

    ' Get the count of gemetry section in the shape
    ' Note: If the shape is a group this will be 0
    nSects = shpObj.GeometryCount

    ' Iterate through all geometry sections for the shape
    ' Because we are adding the current geometry section index to
    ' the constant visSectionFirstComponent, we must start with 0
    For curGeomSectIndx = 0 To nSects - 1
        ' Set a variable to use when accessing the current
        ' geometry section
        curGeomSect = visSectionFirstComponent + curGeomSectIndx

        ' Get the count of rows in the current geometry section
        nRows = shpObj.RowCount(curGeomSect)

        ' Loop through the rows. Remember the count is zero based
        For curRow = 0 To (nRows - 1)
            ' Get the count of cells in the current row
            nCells = shpObj.RowsCellCount(curGeomSect, curRow)

            ' Loop through the cells. Again this is zero based
            For curCell = 0 To (nCells - 1)
                ' Retrieve the cell's formula and add to the listbox
                UserForm1.ListBox1.AddItem _
                    shpObj.CellsSRC(curGeomSect, curRow, curCell).LocalName _
                    & ": " & shpObj.CellsSRC(curGeomSect, curRow,
curCell).Formula
            Next curCell
        Next curRow
    Next curGeomSectIndx
```

```
    ' Display the user form
    UserForm1.Show

End Sub
```

GetFormulas method

Applies to: Master, Page, Shape, Style

Summary: Returns the formulas of many cells.

Version: VISIO 4.5

Syntax: PageOrMasterObj.**GetFormulas** SID_SRCStream, formulas
ShapeOrStyleObj.**GetFormulas** SRCStream, formulas

Element	Description
PageOrMasterObj	The page or master object whose cells are to be queried
ShapeOrStyleObj	The shape or style object whose cells are to be queried
SRCStream	Stream identifying cells to be queried
SID_SRCStream	Stream identifying cells to be queried
formulas	Array that receives formulas of queried cells

Remarks: GetFormulas is like Cell.Formula, except that it can be used to obtain the formulas of many cells at once, rather than one cell at a time.

GetFormulas is a specialization of GetResults, which can be used to obtain cell formulas or results. Setting up a call to GetFormulas involves slightly less work than setting up GetResults.

Shape.GetFormulas can be used to obtain the formulas of any set of cells of Shape. Style.GetFormulas can be used to obtain the formulas of any set of cells of Style.

In both of these cases, you tell GetFormulas which cells you want the formulas of by passing an array of integers in SRCStream. SRCStream should be a one-dimensional array of 3*n two byte integers for some n>=1. GetFormulas interprets the stream as:

{ sectionIdx, rowIdx, cellIdx }n

where sectionIdx is the section index of the desired cell, rowIdx is its row index and cellIdx is its cell index.

Page and Master.GetFormulas are more general in that they can be used to get formulas of any set of cells in any set of shapes of the page or master. SID_SRCStream should be a one-dimensional array of 4*n two byte integers for some n>=1. GetFormulas interprets the stream as:

{ sheetID, sectionIdx, rowIdx, cellIdx }n

where sheetID is the ID property of the Shape on the page or master whose cell formula is desired.

Note: If the sheetID in an entry is visInvalShapeID (-1) or if the bottom byte of sectionIdx is visSectionInval (255), then the entry will be ignored and an empty variant will be returned in the corresponding formula array entry. The motivation for this is so that the same [SID_]SRCStream array can be used on several calls to GetFormulas, SetFormulas, and the like, with the caller only needing to make minor changes to the stream between calls.

If GetFormulas succeeds, formulas returns a one-dimensional array of n variants indexed from 0 to n-1. Each variant returns a formula as a string. Formulas is an out argument that is allocated by GetFormulas and ownership of which is passed back to the caller. The

caller should eventually perform SafeArrayDestroy on the returned array. Note that SafeArrayDestroy has the side effect of clearing the variants referenced by the array's entries, hence deallocating any strings GetFormulas returns. GetFormulas fails if called with !formulas or *formulas. (VB and VBA take care of all this for you. See the example for treatment in VBA.)

See also: Formula property, ID property, Section property, Row property, Cells property, GetResults method, SetFormulas method

Example for GetFormulas, SetFormulas

```
'This VBA macro demonstrates using the GetFormulas and SetFormulas methods.
'A C++ program demonstrating the same thing follows the VBA macro.

Public Sub GetSetFormulas()
' This example assumes there is an active page that has at least 3 shapes on
it.
' It uses GetFormulas to get the width of shape 1, the height of shape 2 and
the
' angle of shape 3. It then uses SetFormulas to set the width of shape 1 to
the
' height of shape 2 and the height of shape 2 to the width of shape 1. The
angle
' of shape 3 is left unaltered.

On Error GoTo handleError

' You're going to get 3 cell formulas using page.GetFormulas. The input array
has
' 4 slots for each cell you're going to get, as it also would if you were
using
' master.GetFormulas. If you were using Shape or Style.GetFormulas, you'd only
' need to supply 3 slots for each cell (section, row and cell).
'
Dim ssrcarray(1 To 3 * 4) As Integer

ssrcarray(1) = ActivePage.Shapes(1).ID
ssrcarray(2) = visSectionObject
ssrcarray(3) = visRowXFormOut
ssrcarray(4) = visXFormWidth

ssrcarray(5) = ActivePage.Shapes(2).ID
ssrcarray(6) = visSectionObject
ssrcarray(7) = visRowXFormOut
ssrcarray(8) = visXFormHeight

ssrcarray(9) = ActivePage.Shapes(3).ID
ssrcarray(10) = visSectionObject
ssrcarray(11) = visRowXFormOut
ssrcarray(12) = visXFormAngle

' Tell Visio to return the formulas of the cells.
Dim formulaArray() As Variant
ActivePage.GetFormulas ssrcarray, formulaArray

' Use SetFormulas to:
'   Set width of shape 1 to height of shape 2.
'   Set height of shape 2 to width of shape 1.
'   Leave angle of shape 3 alone.
' Note: formulaArray is indexed from 0 to 2.

Dim temp as variant
temp = formulaArray(0)
formulaArray(0) = formulaArray(1)
formulaArray(1) = temp
```

```
' You're going to pass the same ssrcarray back to SetFormulas that you
' just passed to GetFormulas. But you want to leave angle alone.
' By setting sheet ID entry in third slot in ssrcarray to
' visInvalShapeID, you tell SetFormulas to ignore that slot.
ssrcarray(9) = visInvalShapeID

' Tell Visio to set the formulas of the cells.
ActivePage.SetFormulas ssrcarray, formulaArray, 0

Exit Sub

handleError:
MsgBox "Error"
Exit Sub

End Sub


------------------------------------------------------------------------------
'This C++ program demonstrates using the GetFormulas and SetFormulas methods.

extern "C" int RunDemo(void)
{
    // This example assumes there is an active page that has at least 3
    // shapes on it. It uses GetFormulas to get the width of shape 1, the
    // height of shape 2 and the angle of shape 3.
    //
    HRESULT hr;
    CVisioApplication app;
    CVisioPage page;
    CVisioShapes shapes;

    // Get shapes collection of active page.

    if ( VAO_SUCCESS != vaoGetObjectWrap(app) )
        goto CU;

    hr= app.ActivePage(page);
    check_valid(hr, page);

    hr= page.Shapes(shapes);
    check_valid(hr, shapes);

    // Set up arguments to GetFormulas. You're going to get 3 cell values
    // using page.GetFormulas. The input array has 4 slots for each cell
    // you're going to get, as it also would if you were using
    // master.GetFormulas.
    // If you were using Shape or Style.GetFormulas, you'd only need to supply
3
    // slots for each cell (section, row and cell).

    #define N_CELLS_TO_GET 3
    #define SLOTS_PER_CELL 4

    SAFEARRAYBOUND rgsabound[1];
    rgsabound[0].lLbound = 1;
```

```
rgsabound[0].cElements = (N_CELLS_TO_GET * SLOTS_PER_CELL);

SAFEARRAY FAR* psaSSRC;
if ( (psaSSRC = SafeArrayCreate(VT_I2,1,rgsabound)) )
    {
    // Set up ssrc array to get width of shape 1, height of shape 2
    // and angle of shape 3. Need shape id's to do this. If you were
    // getting formulas of several cells in same shape, then you'd
    // use shape.getFormulas, rather than page.getFormulas, as you're
    // showing here, and you wouldn't need to gather the shape ids.
    // Also, if you'd just made shapes using DropMany, it would have
    // returned the id's of the shapes it made to you. Note that
    // methods similar to GetFormulas, such as SetFormulas, can use
    // the same ssrc array as set up here. And you can mark
    // individual entries in the array such that they'll be ignored
    // on one call and not on the next. See remarks for GetFormulas.

    #define N_SHAPES_TO_GET N_CELLS_TO_GET
    CVisioShape shape;
    short shapeID;
    static short secIdx = visSectionObject;
    static short rowIdx = visRowXFormOut;
    static short cellIndices[] =
        {visXFormWidth, visXFormHeight, visXFormAngle};

    long rgIndex[1];
    *rgIndex = 1;

    for ( long i = 1; i <= N_SHAPES_TO_GET && NOERROR == hr; i++ )
        {
        if ( NOERROR == (hr = shapes.Item(VVariant(i),shape)) &&
             NOERROR == (hr = shape.ID(&shapeID)) )
            {
            SafeArrayPutElement(psaSSRC, rgIndex, &shapeID);
            *rgIndex += 1;
            SafeArrayPutElement(psaSSRC, rgIndex, &secIdx);
            *rgIndex += 1;
            SafeArrayPutElement(psaSSRC, rgIndex, &rowIdx);
            *rgIndex += 1;
            SafeArrayPutElement(psaSSRC, rgIndex, &(cellIndices[i-1]));
            *rgIndex += 1;
            }
        }

    if ( NOERROR == hr )
        {
        // Now ask Visio to return formulas of cells.

        SAFEARRAY FAR* psaFormulas = NULL;
        if ( NOERROR == page.GetFormulas(&psaSSRC,&psaFormulas) )
            {
            // Got the formulas, which will be an array of variants of
            // type BSTR. You're going to use SetFormulas to set the
            // width of shape 1 to the height of shape 2 and the height of
            // shape 2 to the width of shape 1. Leave angle
            // of shape 3 as is, so you mark 3rd slot in ssrc array in a
            // fashion that will cause SetFormulas to ignore it.
```

```
                if ( psaFormulas &&
                     SafeArrayGetDim(psaFormulas) == 1 &&
                     SafeArrayGetElemsize(psaFormulas) == sizeof(VARIANT) )
                    {
                    // Note: psaFormulas is indexed from 0 to 2.
                    VARIANT temp0,temp1;
                    VariantInit(&temp0);
                    VariantInit(&temp1);
                    *rgIndex = 0;
                    SafeArrayGetElement(psaFormulas,rgIndex,&temp0);
                    *rgIndex = 1;
                    SafeArrayGetElement(psaFormulas,rgIndex,&temp1);
                    *rgIndex = 0;
                    SafeArrayPutElement(psaFormulas,rgIndex,&temp1);
                    *rgIndex = 1;
                    SafeArrayPutElement(psaFormulas,rgIndex,&temp0);
                    VariantClear(&temp0);
                    VariantClear(&temp1);

                    short nProcessed;
                    page.SetFormulas(&psaSSRC,&psaFormulas,0,&nProcessed);
                    }
                SafeArrayDestroy(psaFormulas);
                }
            }
        SafeArrayDestroy(psaSSRC);
        }
CU:
    return 0;
}
```

GetNames method

Applies to: Addons, Documents, Masters, Pages, Styles

Summary: Returns the names of all items in a documents, pages, masters, styles or addons collection.

Version: VISIO 4.5

Syntax: object.**GetNames** NameArray

Element	Description
object	The collection whose member names are to be gotten
NameArray	Array that receives names of members of indicated object

Remarks: If GetNames succeeds, NameArray returns a one-dimensional array of n strings indexed from 0 to n-1, where n will equal the Count property of the indicated object. NameArray is an out argument that is allocated by GetNames and ownership of which is passed back to the caller. The caller should eventually perform SafeArrayDestroy on the returned array. Note that SafeArrayDestroy has the side effect of freeing the strings referenced by the array's entries. GetNames fails if called with !NameArray or *NameArray. (VB and VBA take care of all this for you. See the example for treatment in VBA.)

See also: Documents object, Pages object, Masters object, Styles object, Addons object

Example for GetNames

```
'This VBA macro demonstrates using the GetNames property.
'A C++ program demonstrating the same thing follows the VBA macro.

Public Sub GetMasterNames()

    ' Get the names of all masters in the active
    ' document and displays the names in the Immediate window.
    Dim masnames() As String
    ActiveDocument.Masters.GetNames masnames
    Dim lb As Integer, ub As Integer
    lb = LBound(masnames)
    ub = UBound(masnames)
    Debug.Print ActiveDocument; " lb:"; lb; "ub:"; ub
    While lb <= ub
        Debug.Print masnames(lb)
        lb = lb + 1
    Wend

End Sub


------------------------------------------------------------
'This C++ program demonstrates using the GetNames property.

extern "C" int RunDemo(void)
{
    // This gets the names of the masters in the active document.
    //
    HRESULT hr;
    CVisioApplication app;
    CVisioDocument doc;
    CVisioMasters masters;
    SAFEARRAY FAR* psaNames = NULL;

    // Get masters collection of active document.

    if ( VAO_SUCCESS != vaoGetObjectWrap(app) )
        goto CU;

    hr= app.ActiveDocument(doc);
    check_valid(hr, doc);

    hr= doc.Masters(masters);
    check_valid(hr, masters);

    // Now ask Visio to return names of all masters.
    if ( NOERROR == masters.GetNames(&psaNames) )
        {
        // Got the names, which will be an array of BSTRs.
        // Iterate through them and you can verify
        // they're the right names using the debugger.

        long rgIndex[1];
        LONG llb, lub;
```

```
        if ( psaNames &&
             SafeArrayGetDim(psaNames) == 1 &&
             SafeArrayGetElemsize(psaNames) == sizeof(BSTR) &&
             (NOERROR == SafeArrayGetLBound(psaNames, 1, &llb)) &&
             (NOERROR == SafeArrayGetUBound(psaNames, 1, &lub)) )
            {
            BSTR bstr;
            for ( *rgIndex = llb; llb <= lub; llb++, (*rgIndex)++ )
                {
                SafeArrayGetElement(psaNames,rgIndex,&bstr);
                // SafeArrayElement makes copy of string.
                SysFreeString(bstr);
                }
            }
        SafeArrayDestroy(psaNames);
        }
CU:
    return 0;
}
```

GetResults method

Applies to: Master, Page, Shape, Style

Summary: Gets the results or formulas of many cells.

Version: VISIO 4.5

Syntax: PageOrMasterObj.**GetResults** SID_SRCStream, flags, units, results
ShapeOrStyleObj.**GetResults** SRCStream, flags, units, results

Element	Description
ShapeOrStyleObj	The shape or style object whose cells are to be queried
PageOrMasterObj	The page or master object whose cells are to be queried
SRCStream	Stream identifying cells to be queried
SID_SRCStream	Stream identifying cells to be queried
flags	Flags that influence the type of entries returned in results
units	Measurement units results are to be returned in
results	Array that receives results or formulas of queried cells

Remarks: GetResults is like Cell.Result, except that it can be used to get the results (values) of many cells at once, rather than one cell at a time.

Shape.GetResults can be used to get results of any set of cells of Shape.
Style.GetResults can be used to get results of any set of cells of Style.

In both of these cases, you tell GetResults which cells you want to get by passing an array of integers in SRCStream. SRCStream should be a one-dimensional array of 3*n two-byte integers for n>=1. GetResults interprets the stream as:

{ sectionIdx, rowIdx, cellIdx }n

where sectionIdx is the section index of the desired cell, rowIdx is its row index and cellIdx is its cell index.

Page and Master.GetResults are more general in that they can be used to get results of any set of cells in any set of shapes of the page or master. SID_SRCStream should be a one-dimensional array of 4*n two-byte integers for n>=1. GetResults interprets the stream as:

{ sheetID, sectionIdx, rowIdx, cellIdx }n

where sheetID is the ID property of the Shape on the page or master whose cell result is desired.

Note: If the sheetID in an entry is visInvalShapeID (-1) or if the bottom byte of sectionIdx is visSectionInval (255), then the entry will be ignored and an empty variant will be returned in the corresponding results array entry. The motivation for this is that the same [SID_]SRCStream array can be used on several calls to GetResults, SetResults and the like with the caller only needing to make minor changes to the stream between calls.

flags indicates what data type the returned results should be expressed in. Its value should be one of the following:

```
visGetFloats (0)            ' Results returned as doubles (VT_R8's)
visGetTruncatedInts (1)     ' Results returned as truncated long integers (VT_I4's)
visGetRoundedInts (2)       ' Results returned as rounded long integers (VT_I4's)
visGetStrings (3)           ' Results returned as strings (VT_BSTR's)
```

visGetFormulas (4) ' Formulas returned as strings (VT_BSTR's)

units is an array that controls what measurement units individual results are returned in. Each entry in the array can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also indicate desired units with integer constants (visCentimeters, visInches, etc.) declared by the Visio type library (and visconst.bas). See the Cell.Result property also. Note that the values specified in the units array have no effect if flags is visGetFormulas.

If not null, we expect units to be a one-dimensional array of 1<=u variants. Each entry can be a string or integer code, or empty (nothing). If the I'th entry is empty, then the i'th returned result will be returned in the units designated by units(j), where j is the index most recent prior non-empty entry. Thus if you want all returned values to be in the same units, you need only pass a units array with one entry. If there is no prior non-empty entry, or if no units array is supplied, then visNumber (0x20) will be used. This causes internal units (like the Cell.ResultIU property) to be returned.

If GetResults succeeds, results returns a one-dimensional array of n variants indexed from 0 to n-1. The type of the returned variants is a function of flags. Results is an out argument that is allocated by GetResults and ownership of which is passed back to the caller. The caller should eventually perform SafeArrayDestroy on the returned array. Note that SafeArrayDestroy has the side effect of clearing the variants referenced by the array's entries, hence deallocating any strings GetResults returns. GetResults fails if called with !results or *results. (VB and VBA take care of all this for you. See the example for treatment in VBA.)

See also: Result property, ResultIU property, ID property, Section property, Row property, Cells property, GetFormulas method, SetResults method

Example for GetResults, SetResults

```
'VBA macro that demonstrates using the GetResults and SetResults method.
'A C++ program demonstrating the same thing follows the VBA macro.

Public Sub GetSetResults()
' This example assumes there is an active page that has at least 3 shapes on
it.
' It uses GetResults to get the width of shape 1, the height of shape 2 and
the
' angle of shape 3. It then uses SetResults to set the width of shape 1 to the
' height of shape 2 and the height of shape 2 to the width of shape 1. The
angle
' of shape 3 is left unaltered.

On Error GoTo handleError

' This code gets 3 cell values using page.GetResults. The input array has 4
' slots for each cell you're going to get, as it also would if you were using
' master.GetResults. If you were using Shape or Style.GetResults, you'd only
need
' to supply 3 slots for each cell (section, row and cell).
'
Dim ssrcarray(1 To 3 * 4) As Integer

ssrcarray(1) = ActivePage.Shapes(1).ID
ssrcarray(2) = visSectionObject
ssrcarray(3) = visRowXFormOut
ssrcarray(4) = visXFormWidth

ssrcarray(5) = ActivePage.Shapes(2).ID
ssrcarray(6) = visSectionObject
ssrcarray(7) = visRowXFormOut
ssrcarray(8) = visXFormHeight

ssrcarray(9) = ActivePage.Shapes(3).ID
ssrcarray(10) = visSectionObject
ssrcarray(11) = visRowXFormOut
ssrcarray(12) = visXFormAngle

' Get first two values in inches. You can leave the second entry in units
' uninitialized (empty) since you want the second result in the same units as
the
' first result. Get the third result in degrees. Note that you can express
' units as either a string or an integer constant.
Dim unitsarray(1 To 3) As Variant
unitsarray(1) = "in."
unitsarray(3) = visDegrees

' Tell Visio to return results of cells as an array of floating point numbers.
Dim resultArray() As Variant
ActivePage.GetResults ssrcarray, visGetFloats, unitsarray, resultArray

' Use SetResults to:
'   Set width of shape 1 to height of shape 2.
'   Set height of shape 2 to width of shape 1.
```

```
'   Leave angle of shape 3 alone.
' Note: resultArray is indexed from 0 to 2.

Dim temp as variant
temp = resultArray(0)
resultArray(0) = resultArray(1)
resultArray(1) = temp

' You're going to pass the same ssrcarray back to SetResults that you
' just passed to GetResults. But you want to leave the angle alone.
' By setting the sheet ID entry in the third slot in the ssrcarray to
' visInvalShapeID, you tell SetResults to ignore that slot.
ssrcarray(9) = visInvalShapeID

' Tell Visio to set the results of the cells.
ActivePage.SetResults ssrcarray, unitsarray, resultArray, 0

Exit Sub

handleError:
MsgBox "Error"
Exit Sub

End Sub


------------------------------------------------------------------------------
'This C++ program demonstrates using the GetResults and SetResults methods.

extern "C" int RunDemo(void)
{
    // This example assumes there is an active page that has at least 3
    // shapes on it. It uses GetResults to get the width of shape 1, the
    // height of shape 2 and the angle of shape 3.
    //
    HRESULT hr;
    CVisioApplication app;
    CVisioPage page;
    CVisioShapes shapes;

    // Get shapes collection of active page.

    if ( VAO_SUCCESS != vaoGetObjectWrap(app) )
        goto CU;

    hr= app.ActivePage(page);
    check_valid(hr, page);

    hr= page.Shapes(shapes);
    check_valid(hr, shapes);

    // Set up arguments to GetResults. We're going to get 3 cell values
    // using page.GetResults. The input array has 4 slots for each cell
    // we're going to get, as it also would were we using master.GetResults.
    // Were we using Shape or Style.GetResults, we'd only need to supply 3
    // slots for each cell (section, row and cell).
```

```
#define N_CELLS_TO_GET 3
#define SLOTS_PER_CELL 4

SAFEARRAYBOUND rgsabound[1];
rgsabound[0].lLbound = 1;
rgsabound[0].cElements = (N_CELLS_TO_GET * SLOTS_PER_CELL);

SAFEARRAY FAR* psaSSRC;
if ( (psaSSRC = SafeArrayCreate(VT_I2,1,rgsabound)) )
    {
    // Set up ssrc array to get width of shape 1, height of shape 2
    // and angle of shape 3. Need shape id's to do this. If you were
    // getting results of several cells in same shape, then you'd
    // use shape.getresults, rather than page.getresults, as we're
    // showing here, and you wouldn't need to gather the shape ids.
    // Also, if you'd just made shapes using DropMany, it would have
    // returned to you the id's of the shapes it made. Note that
    // methods similar to GetResults, such as SetResults, can use
    // the same ssrc array as is being set up here. And you can mark
    // individual entries in the array such that they'll be ignored
    // on one call and not on the next. See remarks for GetResults.

    #define N_SHAPES_TO_GET N_CELLS_TO_GET
    CVisioShape shape;
    short shapeID;
    static short secIdx = visSectionObject;
    static short rowIdx = visRowXFormOut;
    static short cellIndices[] =
        {visXFormWidth, visXFormHeight, visXFormAngle};

    long rgIndex[1];
    *rgIndex = 1;

    for ( long i = 1; i <= N_SHAPES_TO_GET && NOERROR == hr; i++ )
        {
        if ( NOERROR == (hr = shapes.Item(VVariant(i),shape)) &&
             NOERROR == (hr = shape.ID(&shapeID)) )
            {
            SafeArrayPutElement(psaSSRC, rgIndex, &shapeID);
            *rgIndex += 1;
            SafeArrayPutElement(psaSSRC, rgIndex, &secIdx);
            *rgIndex += 1;
            SafeArrayPutElement(psaSSRC, rgIndex, &rowIdx);
            *rgIndex += 1;
            SafeArrayPutElement(psaSSRC, rgIndex, &(cellIndices[i-1]));
            *rgIndex += 1;
            }
        }

    // Set up array that tells GetResults what measurement units to
    // return results in. Get first two values in inches. We can leave
    // the second entry in units array in VT_EMPTY state since we want
    // the second result in the same units as the first result. Get the
    // third result in degrees. Note that we can express desired units as
    // either a string or an integer constant.

    SAFEARRAY FAR* psaUnits;
```

```
        rgsabound[0].cElements = N_CELLS_TO_GET;

        if ( NOERROR == hr &&
             (psaUnits = SafeArrayCreate(VT_VARIANT,1,rgsabound)) )
            {
            VARIANT* pUnit;
            if ( (NOERROR == SafeArrayAccessData(psaUnits,(void**)&pUnit)) )
                {
                V_VT(pUnit+0) = VT_I2;
                V_I2(pUnit+0) = visInches;
                // Leave pUnit+1 empty. Get 2nd cell in same units as 1st
cell.
                V_VT(pUnit+2) = VT_I2;
                V_I2(pUnit+2) = visDegrees;

                // Now ask Visio to return results of cells as an array
                // of floating point numbers.

                SAFEARRAY FAR* psaResults = NULL;
                if ( NOERROR == page.GetResults(&psaSSRC,visGetFloats,
                                                &psaUnits,&psaResults) )
                    {
                    // Got the results, which will be an array of variants
                    // of type double because we specified visGetFloats. If
                    // we'd called GetResults with visGetTruncatedInts or
                    // visGetRoundedInts we'd now have integer variants.
                    // If we'd specified visGetStrings or visGetFormulas
                    // we'd now have BSTR variants.

                    // We're now going to use SetResults to set width of
                    // shape 1 to height of shape 2 and height of shape 2
                    // to width of shape 1. We want to leave angle of
                    // shape 3 as is, so we mark 3rd slot in ssrc array in
                    // fashion that will cause SetResults to ignore it.

                    if ( psaResults &&
                         SafeArrayGetDim(psaResults) == 1 &&
                         SafeArrayGetElemsize(psaResults) == sizeof(VARIANT) )
                        {
                        // Note: psaResults is indexed from 0 to 2.
                        VARIANT temp0,temp1;
                        VariantInit(&temp0);
                        VariantInit(&temp1);
                        *rgIndex = 0;
                        SafeArrayGetElement(psaResults,rgIndex,&temp0);
                        *rgIndex = 1;
                        SafeArrayGetElement(psaResults,rgIndex,&temp1);
                        *rgIndex = 0;
                        SafeArrayPutElement(psaResults,rgIndex,&temp1);
                        *rgIndex = 1;
                        SafeArrayPutElement(psaResults,rgIndex,&temp0);
                        VariantClear(&temp0);
                        VariantClear(&temp1);

                        short nProcessed;
                        page.SetResults(&psaSSRC,&psaUnits,
                                        &psaResults,0,&nProcessed);
```

```
                    }
                SafeArrayDestroy(psaResults);
                }
            SafeArrayUnaccessData(psaUnits);
            }
        SafeArrayDestroy(psaUnits);
        }
    SafeArrayDestroy(psaSSRC);
    }
CU:
    return 0;
}
```

GlueTo method

Applies to: Cell

Summary: Glues one shape to another from a cell in the first shape to a cell in the second shape.

Version: VISIO 2.0

Syntax: object.**GlueTo** gluetocell

Element	Description
object	A Cell object that represents the part of the shape to glue
gluetocell	A Cell object that represents the part of the shape to glue to

Remarks: You can glue to any X or Y cell in a Connection Point section row; any X or Y cell in a Geometry section vertex row; and cells in the Alignment and Guide Info sections. Gluing to an Alignment cell or a Geometry vertex cell creates a connection point if one doesn't exist.

Gluing the X cell of a Controls section row or a BeginX or EndX cell automatically glues the Y cell of the Controls section row or the BeginY or EndY cell, respectively. (The reverse is also true.)

You can glue to cells PinX or PinY when using dynamic glue. PinX indicates dynamic glue with a horizontal walking preference. PinY indicates dynamic glue with a vertical walking preference.

See also: FromCell property, GlueToPos method, ToCell property

Example for GlueTo, GlueToPos

```
'This VBA macro demonstrates gluing shapes together.


Public Sub GlueTo_Example()

  Dim shp1DObj As Visio.Shape
  Dim shp2DObj As Visio.Shape
  Dim shp2DObj2 As Visio.Shape
  Dim cellGlueFromBegin As Visio.Cell
  Dim cellGlueFromEnd As Visio.Cell
  Dim cellGlueToObj As Visio.Cell

  'Draw a line and 2 rectangles.
  Set shp1DObj = ActivePage.DrawLine(3, 5, 5, 3)
  Set shp2DObj = ActivePage.DrawRectangle(1, 1, 4, 2)
  Set shp2DObj2 = ActivePage.DrawRectangle(5, 5, 8, 6)

  'Get the cell objects needed to make the connections.
  Set cellGlueFromBegin = shp1DObj.Cells("BeginX")
  Set cellGlueFromEnd = shp1DObj.Cells("EndX")
  Set cellGlueToObj = shp2DObj.Cells("Geometry1.X3")

  'Use the GlueTo method to glue the begin point of the 1D shape
  'to the top right vertex (Geometry1.X3) of the lower 2D shape.
  cellGlueFromBegin.GlueTo cellGlueToObj

  'Use the GlueToPos method to glue the end point of the 1D shape
  'to the bottom center of the upper 2D shape.
  cellGlueFromEnd.GlueToPos shp2DObj2, 0.5, 0

  'You can also use the GlueTo method to glue referencing a connection point
cell.
  Set shp1DObj = ActivePage.DrawLine(3, 5, 5, 3)
  Set cellGlueFromEnd = shp1DObj.Cells("EndX")
  Set cellGlueToObj = shp2DObj.Cells("Connections.X1")
  cellGlueFromEnd.GlueTo cellGlueToObj

End Sub
```

GlueToPos method

Applies to: Cell

Summary: Glues one shape to another from a cell in the first shape to an x,y position in the second shape.

Version: VISIO 2.0

Syntax: object.**GlueToPos** shpObject, x, y

Element	Description
object	A Cell object that represents the part of the shape to glue
shpObject	The Shape object to be glued to
x	The x-coordinate of the position to glue to
y	The y-coordinate of the position to glue to

Remarks: The GlueToPos method creates a new connection point at the location determined by x and y, which represent decimal fractions of the specified shape's width and height, respectively, rather than coordinates. For example, celObj.GlueToPos shpObject, 0.5, 0.5 creates a connection point at the center of shpObject and glues the part of the shape that celObj represents to that point.

Gluing the X cell of a Controls section row or a Begin X or EndX cell automatically glues the Y cell of the Controls section row or the BeginY or EndY cell, respectively. (The reverse is also true.)

See also: FromCell property, GlueTo method, ToSheet property

Example for GlueToPos

```
*GlueTo Method
```

Green property

Applies to: Color

Summary: Gets or sets the intensity of the green component of a Color object.

Version: VISIO 4.0

Syntax: intRet = object.**Green**
object.**Green** = intVal

Element	Description
intRet	The current value of the color's green component
object	The Color object that has or gets the component
intVal	The new value of the color's green component

Remarks: The Green property can be a value from 0 to 255.

A color is represented by red, green and blue components. It also has flags that indicate how the color is to be used. These correspond to members of the Windows PALETTEENTRY data structure. For details, search the Windows SDK online help for PALETTEENTRY.

See also: Blue property, Flags property, PaletteEntry property, Red property

Example for Green

Group method

Applies to: Selection, Shape, Window

Summary: Groups the objects that are selected in the indicated window or selection, or turns the indicated shape into a group.

Version: VISIO 2.0

Syntax: object.**Group**

Element	Description
object	The object to group

See also: AddToGroup method, ConvertToGroup method, RemoveFromGroup method, Ungroup method

Example for Group

```
*Drop Method
```

Group property

Applies to: Entity

Summary: Specifies the Group type of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**Group**
object.**Group** = Expression

Element	Description
RetVal	The current group value as a long integer
object	The Entity object that has or gets the value
Expression	The new group value as a long integer

Remarks: The Group property of an Entity object can be used in one of two ways. You can get the Group property to determine what type of data is stored in an Entity. When accessing an Entity object's data, you can use only certain properties depending upon the Group returned. These properties are listed below:

1000 - String
1002 - Control
1003 - LayerName
1004 - BinaryData & BinaryLength
1005 - Handle

1010, 1020, 1030 - VectorX, VectorY, & VectorZ
1011, 1021, 1031
1012, 1022, 1032
1013, 1023, 1033

1040, 1041, 1042 - RealValue
1070 - ShortValue
1071 - LongValue

When setting any of the above properties, the Group property is automatically set. When setting the RealValue property, the Group property is set by default to 1040; when setting any of the VectorX/Y/Z properties for the first time, the Group property is set by default to 1010. You can change the Group property only between similar groups (for example, you can change the Group property from 1010 to 1021, but not to 1000). This also applies to the RealValue property groups 1040, 1041, and 1042.

Example for Group

Handle property

Applies to: Entity

Summary: Specifies the current handle of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**Handle**
object.**Handle** = Expression

Element	Description
RetVal	The current handle
object	The Entity object that has or gets the handle
Expression	The new handle

Remarks: If an Entity object's Group property is set to 1005, it contains a database handle. A database handle is stored as a string of up to 8 characters.

See also: Group property

Example for Handle

Help property

Applies to: Shape

Summary: Sets or returns the help string for a shape.

Version: VISIO 3.0

Syntax: strRet = object.**Help**
object.**Help** = strExpression

Element	Description
strRet	The current help string
object	The Shape object that has or gets the help string
strExpression	The new help string

Remarks: Use this property to set or get the help string for a Shape object. This is equivalent to setting the help field for a shape in the Special dialog box. The limit for a help string is 127 characters.

Example for Help

HelpContextID property

Applies to: MenuItem, StatusBarItem, ToolbarItem

Summary: Gets or sets the help context ID to be used by a menu item, status bar item, or toolbar item.

Version: VISIO 4.0

Syntax: object.**HelpContextID** = intVal
intVal = object.**HelpContextID**

Element	Description
object	The object that has or gets the context ID
intVal	The context ID of a topic in a help file

Remarks: For Visio commands, the HelpContextID property is usually the same value as the CmdNum property, which contains the command ID. Command IDs are declared by the Visio type library (and visconst.bas). Command ID constants have the prefix "visCmd."

By default, HelpContextID is 0, which displays the Contents topic of the file indicated by the HelpFile property.

If HelpContextID is null and the object's CmdNum property is set to one of Visio's command IDs, it uses the default help context ID from the built-in Visio user interface.

See also: HelpFile property, MiniHelp property

Example for HelpContextID

HelpFile property

Applies to: MenuItem, StatusBarItem, ToolbarItem

Summary: Gets or sets the help file to be used by a MenuItem, StatusbarItem, or ToolbarItem.

Version: VISIO 4.0

Syntax: object.**HelpFile** = fileStr
fileStr = object.**HelpFile**

Element	Description
object	The object that has or gets the help file
fileStr	The name of the help file

Remarks: Set the HelpContextID property of the object to display a particular topic within the help file.

If fileStr is not a fully qualified path, Visio searches the directories specified in the HelpPaths property of the Application object.

If HelpFile is null and the object's CmdNum property is set to one of Visio's command IDs, it uses the default help file from the built-in Visio user interface.

See also: CmdNum property, HelpContextID property, HelpPaths property

Example for HelpFile

HelpPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for help files.

Version: VISIO 4.0

Syntax: strRet = object.**HelpPaths**
object.**HelpPaths** = pathsStr

Element	Description
strRet	A text string containing a list of folders
object	An Application object
pathsStr	A text string containing a list of folders

Remarks: The string passed to and received from HelpPaths is the same string shown in Visio's File Paths dialog (accessible from the Tools/Options dialog). This same string is stored in Visio's profile (.ini) file (appObj.ProfileName) in the entry whose key is "HelpPath."

To indicate more than one folder, separate individual items in the path string with semicolons. If a path is not fully qualified, Visio looks for the folder in the folder that contains the Visio program files (appObj.Path).

For example, if Visio's executable file is installed in c:\Visio, and HelpPaths is "Help;d:\Help", Visio looks for help files in both c:\Visio\Help and d:\Help.

When Visio looks for help files, it will look in all paths named in HelpPaths plus in all sub-folders of those paths. Also, the fact that a path is named in HelpPaths does not imply the path actually exists. If you pass HelpPaths to the EnumDirectories method, it will return a complete list of fully qualified paths that Visio will actually look in.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, StartupPaths property, StencilPaths property, TemplatePaths property, ProfileName property, Path property, EnumDirectories method

Example for HelpPaths

HitTest property

Applies to: Shape

Summary: Determines if a given x,y position hits outside, inside or on the boundary of a shape.

Version: VISIO 4.5

Syntax: intRet = object.**HitTest**(x, y, tolerance)

Element	Description
intRet	visHitOutside (0), visHitOnBoundary (1) or visHitInside (2)
object	The shape to be hit tested
x	The x-coordinate to be hit tested
y	The y-coordinate to be hit tested
tolerance	How close x,y must be to shape for hit to occur

Remarks: x, y and tolerance should be in internal drawing units (inches in the drawing) and with respect to the coordinate space of the page, master or group shape that contains the shape being hit tested.

See also: BoundingBox method

Example for HitTest

Hyperlink property

Applies to: Shape

Summary: Returns a Hyperlink object that represents a shape's hyperlink.

Version: VISIO 5.0

Syntax: objRet = object.**Hyperlink**

Element	Description
objRet	A Hyperlink object that represents the shape's hyperlink
object	The Shape object to examine

Remarks: Use the Hyperlink property of a Shape object to obtain a Hyperlink object that represents the behavior of navigating to a referenced location.

Use the AddHyperlink method or its equivalent to add a Hyperlink object to the shape before retrieving this property. Otherwise, an exception will be raised if no Hyperlink object exists.

See also: Hyperlink object, AddHyperlink method

Example for Hyperlink

*AddHyperlink method

HyperlinkBase property

Applies to: Document

Summary: Returns or sets the value of the HyperlinkBase field in a document's properties.

Version: VISIO 5.0

Syntax: strRet = object.**HyperlinkBase**
object.**HyperlinkBase** = stringExpression

Element	Description
strRet	The current value of the field
object	The Document object that has or gets the value
stringExpression	The new value of the field

Remarks: Setting the HyperlinkBase property is equivalent to entering information in the Hyperlink Base field in the Properties dialog box, accessed from the File menu.

See also: Description property, Keywords property, Subject property, Title property, Company property, Category property, Manager property

Example for HyperlinkBase

*Document Property

IconFileName method

Applies to: StatusBarItem, ToolbarItem

Summary: Sets a custom icon file to be used for an item in a toolbar or status bar.

Version: VISIO 4.0

Syntax: object.**IconFileName** fileString

Element	Description
object	The object that loads the icon file
fileString	The name of the icon file to load

Remarks: The icon file is loaded and the bits are saved. The file name is discarded.

Visio uses the 32 x 32 icon in the file. You should create a 16 x 16 icon in the center of the 32 x 32 icon.

Unless fileStr is a fully qualified path, Visio searches for the .ICO file in the directories indicated by the Application object's AddonPaths property (assuming that the UI object is in Visio's process.)

See also: AddonPaths property

Example for IconFileName

```
*BuiltInToolbars Property
```

IconSize property

Applies to: Master

Summary: Returns or sets the size of a master icon.

Version: VISIO 2.0

Syntax: intRet = object.**IconSize**
object.**IconSize** = newSize

Element	Description
intRet	The current size of the master icon
object	The Master object that owns the icon
newSize	The new size for the master icon

Remarks: The following constants declared by the Visio type library (and visconst.bas) show the possible values for IconSize:

visNormal = 1
visTall = 2
visWide = 3
visDouble = 4

See also: IconUpdate property

Example for IconSize

IconUpdate property

Applies to: Master

Summary: Determines whether a master icon is updated manually or automatically.

Version: VISIO 2.0

Syntax: intRet = object.**IconUpdate**
object.**IconUpdate** = updateMode

Element	Description
intRet	The current update mode for the icon
object	The Master object that owns the icon
updateMode	The new update mode for the icon

Remarks: The following constants declared by the Visio type library (and visconst.bas) show the possible values for IconUpdate:

visAutomatic = 1
visManual = 0

See also: IconSize property

Example for IconUpdate

ID property

Applies to: Event, Font, Master, Page, Shape, Style

Summary: Returns the ID of a Page, Master, Shape, Style, Font, or Event object.

Version: VISIO 4.0

Syntax: intVal = object.**ID**

Element	Description
intVal	The ID of the object
object	The object to examine

Remarks: The ID of a shape is unique only within the scope of the page or master. The ID of a page, master, or style is unique within the scope of the document.

If a shape, page, master, or style is deleted, future objects in the same scope may be assigned the same ID. Therefore persisting shape or style IDs in separate data stores are generally not as sound as persisting UniqueIDs.

Shape IDs are useful when using methods such as GetResults and PutResults, which can be used to get or set many cell values at once, possibly cells in many different shapes. To do this, you need to pass shape IDs to such methods. If you create shapes using DropMany, it will return the IDs of the shapes it creates to your program.

The ID of a font object corresponds to the number stored in the Font cell of a row in a shape's Character Properties section. For example, to apply the font named "Arial" to the text of a shape, create a Font object representing "Arial" and get the ID of that font, then set the CharProps property of the Shape object to that ID, as is shown in the following example.

Note: The ID associated with a particular font varies from system to system or as fonts are installed and removed on a given system.

The ID of an Event uniquely identifies an Event in its EventList. As long as a reference is held on an EventList, or on the source object of an EventList, the ID of any Event in the list can be cached. Even if other Events are added to or removed from the list, the cached ID can be used later to identify the original event. If an Event is Persistent, its ID can be cached indefinitely. While the Event with that ID might be removed, no new event in the same EventList will be given the same ID.

See also: NameID property, UniqueID property, DropMany method, GetResults method, SetResults method, GetFormulas method, SetFormulas method, CharProps property, ItemFromID property, Persistent property

Example for ID

Import method

Applies to: Master, Page, Shape

Summary: Imports a file into Visio.

Version: VISIO 3.0

Syntax: objRet = object.**Import**(stringExpression)

Element	Description
objRet	A Shape object that represents the new shape imported from the file
object	The page, master or group to receive the new shape
stringExpression	The name of the file to import

Remarks: The Import method imports the file specified by stringExpression onto a page, master or group. StringExpression must be a fully qualified pathname. Names specifying only a relative or partial path will generate an error.

The filename extension indicates which import filter to use. If the filter is not installed, Import returns an error. Import uses the default preference settings for the specified filter and does not prompt the user for non-default arguments.

See also: Export method

Example for Import

ImportIcon method

Applies to: Master

Summary: Imports the icon for a Master from a named file.

Version: VISIO 4.5

Syntax: object.**ImportIcon** stringExpression

Element	Description
object	The master to receive the new icon
stringExpression	The name of the file to import

Remarks: ImportIcon can only import files that were produced by exporting a master icon in Visio's internal icon format (visIconFormatVisio). ImportIcon will not accept icons represented in other formats.

See also: ExportIcon method

Example for ImportIcon

IncludesFill property

Applies to: Style

Summary: Indicates whether the style includes fill attributes.

Version: VISIO 4.0

Syntax: intRet = object.**IncludesFill**
object.**IncludesFill** = intExpression

Element	Description
intRet	0 if the object doesn't define fill attributes, -1 if it does
object	The Style object that has or gets the fill attributes
intExpression	0 to disable fill attributes, or non-zero to enable them

Remarks: This property corresponds to the Fill check box in the Includes section of Visio's Define Styles dialog box.

See also: IncludesLine property, IncludesText property

Example for IncludesFill

IncludesLine property

Applies to: Style

Summary: Indicates whether the style includes line attributes.

Version: VISIO 4.0

Syntax: intRet = object.**IncludesLine**
object.**IncludesLine** = intExpression

Element	Description
intRet	0 if the object doesn't define line attributes, -1 if it does
object	The Style object that has or gets the line attributes
intExpression	0 to disable line attributes, or non-zero to enable them

Remarks: This property corresponds to the Line check box in the Include section of Visio's Define Styles dialog box.

See also: IncludesFill property, IncludesText property

Example for IncludesLine

IncludesText property

Applies to: Style

Summary: Indicates whether the style includes text attributes.

Version: VISIO 4.0

Syntax: intRet = object.**IncludesText**
object.**IncludesText** = intExpression

Element	Description
intRet	0 if the object doesn't define text attributes, -1 if it does
object	The Style object that has or gets the text attributes
intExpression	0 to disable text attributes, or non-zero to enable them

Remarks: This property corresponds to the Text check box in the Include section of Visio's Define Styles dialog box.

See also: IncludesFill property, IncludesLine property

Example for IncludesText

Index property

Applies to: Addon, Color, Connect, Document, Entity, Event, Font, Layer, Master, Menu, MenuItem, Page, Shape, StatusBarItem, Style, Toolbar, ToolbarItem, Window

Summary: Returns the ordinal position of an object in a collection.

Version: VISIO 2.0

Syntax: intRet = object.**Index**

Element	Description
intRet	The index of the object within its collection
object	The object to examine

Remarks: Most collections are indexed starting with 1 rather than 0, so the index of the first element is 1, the index of the second element is 2, and so forth. The index of the last element in a collection is the same as the value of that collection's Count property. You can iterate through a collection by using these index values. Adding objects to or deleting objects from a collection can change the index values of other objects in the collection.

The Color collection is indexed starting with 0. This is to be consistent with the numbering displayed alongside the colors displayed in Visio's color palette dialog box.

The following collections are also indexed starting with 0:

AccelItems
AccelTables
MenuSets
MenuItems
Menus
StatusBarItems
StatusBars
ToolbarItems
Toolbars
ToolbarSets

See also: Item property

Example for Index

InPlace property

Applies to: Document

Summary: Determines whether or not a Document object is open in place.

Version: VISIO 3.0

Syntax: intRet = object.**InPlace**

Element	Description
intRet	TRUE (-1) if the Document object is open in place; otherwise FALSE (0)
object	The Document object to examine

Remarks: For an instance of Visio open in place, a Document object that represents a stencil will report itself as in place (return True) even though the stencil does not appear to be within the container.

Example for InPlace

InsertFromFile method

Applies to: Master, Page, Shape

Summary: Adds a linked or embedded object to a page, master, or group.

Version: VISIO 4.1

Syntax: objRet = object.**InsertFromFile**(filename, flags)

Element	Description
objRet	A Shape object that represents the newly linked or embedded object
object	The page, master, or group in which to embed or link the object
filename	The name of the file that contains the object to link or embed
flags	Flags that influence how the object is inserted. See Remarks.

Remarks: InsertFromFile creates a new shape that represents a linked or embedded OLE object.

Flags is a bit mask whose value should be a combination of the following values:

visInsertLink = 8
visInsertIcon = 16

If flags includes visInsertLink, the new shape represents an OLE link to the named file. Otherwise, InsertFromFile produces an OLE object from the contents of the named file and embeds it in the document that contains the page, master, or group.

If flags includes visInsertIcon, Visio displays the new shape as an icon.

See also: InsertObject method

Example for InsertFromFile

InsertObject method

Applies to: Master, Page, Shape

Summary: Adds a new embedded object or ActiveX control to a page, master, or group.

Version: VISIO 4.1

Syntax: objRet = object.**InsertObject**(ClassOrProgID, flags)

Element	Description
objRet	A Shape object that represents the newly created object or control
object	The page, master or group in which to create the object or control
ClassOrProgID	Identifies the type of object or control to create
flags	Flags that influence the operation. See Remarks.

Remarks: ClassOrProgID is a string that identifies the kind of object or control to create. It can be either the object or control's class ID (guid) in string form or the object or control's program ID of the handler for the class.

If ClassOrProgID is a string representing a class ID, it will look like "{D3E34B21-9D75-101A-8C3D-00AA001A1652}", which is the ClassID registered by the version of Microsoft Paint that is distributed with Windows 95.

If ClassOrProgID is a string representing a program ID, it will look like "paint.picture" or "forms.combobox.1".

See vendor-specific documentation or browse the Windows registry to determine which class IDs and program IDs are associated with objects and controls provided by other applications.

Flags is a bit mask that can include one of the following values:

visInsertIcon = &H10
visInsertDontShow = &H1000

Flags can also include one of the following values:

visInsertAsControl = &H2000
visInsertAsEmbed = &H4000

If flags includes visInsertIcon, Visio displays the new shape as an icon.

If flags includes visInsertDontShow, Visio refrains from executing the newly created object's show verb after creating the new object. Whether you want to execute the show verb depends in part on why your program is creating the new object, and may differ from one type of object to another.

If both visInsertIcon and visInsertDontShow are specified, InsertObject will fail. If you want to insert an object that displays as an icon, you must allow Visio to execute the object's show verb.

Values in visInsertAsControl and visInsertAsEmbed will only have an effect if the class identified by ClassOrProgID is identified in the registry as being a control and as insertable. If neither visInsertAsControl or visInsertAsEmbed is specified and the object can be either a control or an embed, Visio will insert it as a control. In rare cases this means Visio 5.0 may insert a control whereas prior versions of Visio would have responded to the same call by inserting an embedded object.

If a control is inserted, this method will place the document in design mode, causing any code executing in the document to halt until the document is returned to run mode.

See also: InsertFromFile method, ClassID property, ProgID property, Mode property

Example for InsertObject

InstanceHandle property

Applies to: Application

Summary: Returns the instance handle of the Application object.

Version: VISIO 4.0

Syntax: intRet = object.**InstanceHandle**

Element	Description
intRet	The instance handle of the object (a 2-byte value)
object	The Application object to examine

Remarks: InstanceHandle returns a 2-byte value, which is appropriate to use with an instance of 16-bit Visio.

If you're working with an instance of 32-bit Visio, use InstanceHandle32 instead.

See also: InstanceHandle32 property, IsVisio16 property, IsVisio32 property, WindowHandle property, WindowHandle32 property

Example for InstanceHandle

InstanceHandle32 property

Applies to: Application

Summary: Returns the instance handle of the Application object.

Version: VISIO 4.0

Syntax: intRet = object.**InstanceHandle32**

Element	Description
intRet	The instance handle of the object (a 4-byte value)
object	The Application object to examine

Remarks: InstanceHandle32 returns a 4-byte value, which is appropriate to use with an instance of 32-bit Visio.

If the Application object represents an instance of 16-bit Visio, InstanceHandle32 returns 0.

See also: InstanceHandle property, IsVisio16 property, IsVisio32 property, WindowHandle property, WindowHandle32 property

Example for InstanceHandle32

Intersect method

Applies to: Selection, Window

Summary: Creates one closed shape from the area in which selected shapes overlap or intersect.

Version: VISIO 4.0

Syntax: object.**Intersect**

Element	Description
object	The Window or Selection object that contains the shapes to intersect

Remarks: The Intersect method is equivalent to choosing the Intersect command from the Operations submenu on the Shape menu in Visio. The produced shape will be the topmost shape in its ContainingShape and will inherit the text and formatting of the first selected shape. The original shapes are deleted.

If the object being operated on is a Selection object, it will have no shapes selected in it when the operation is complete.

See also: Combine method, Fragment method, Join method, Subtract method, Trim method, Union method, ContainingShape property

Example for Intersect

IsConstant property

Applies to: Cell

Summary: Returns TRUE if the formula of the cell is a constant expression.

Version: VISIO 4.0

Syntax: intRet = object.**IsConstant**

Element	Description
intRet	TRUE (-1) if the object's formula is a constant; otherwise False (0)
object	The Cell object to examine

See also: IsInherited property

Example for IsConstant

IsField property

Applies to: Characters

Summary: Returns TRUE if the object represents the expanded text of a single field with no additional non-field characters.

Version: VISIO 3.0

Syntax: intRet = object.**IsField**

Element	Description
intRet	TRUE (-1) if the object represents only the expanded text of a field; otherwise False (0)
object	The Characters object to examine

Remarks: If the Characters object contains characters in addition to the expanded text of a field, IsField returns FALSE. To change the range of text represented by a Character object, set its Begin and End properties.

See also: Begin property, End property

Example for IsField

IsHierarchical property

Applies to: MenuItem

Summary: Indicates whether a MenuItem represents a hierarchical submenu.

Version: VISIO 4.0

Syntax: intRet = object.**IsHierarchical**

Element	Description
intRet	TRUE (-1) if the object represents a submenu; otherwise False (0)
object	The MenuItem object to examine

Remarks: The CmdNum property of a MenuItem object that represents a submenu should be visCmdHierarchical.

See also: CmdNum property

Example for IsHierarchical

```
'This VBA macro demonstrates deleting a hierarchical menu.

Public Sub DeleteHierarchicalMenuItem_Example()

  Dim uiObj As Visio.UIObject
  Dim menuSetObj As Visio.MenuSet
  Dim menuObj As Visio.Menu
  Dim menuItemsObj As Visio.MenuItems
  Dim menuItemObj As Visio.MenuItem
  Dim hiermenuItemsObj As Visio.MenuItems
  Dim hiermenuItemObj As Visio.MenuItem

  'True if variable represents a hierarchical menu item
  Dim hierState As Boolean
  Dim i, j As Integer                          'Loop variables

  'Retrieve the UIObject for the copy of the BuiltInMenus
  Set uiObj = Visio.Application.BuiltInMenus

  'Set menuSetObj to the Drawing menu set
  Set menuSetObj = uiObj.MenuSets.ItemAtID(visUIObjSetDrawing)

  'Retrieve the Tools menu.
  'Because you retrieved the built-in menus, you know that you can find the
Tools
  'menu by its position. If you had retrieved a custom UI, you would have to
loop
  'through the menus checking the caption to find the Tools menu.
  'When using a custom menu there is no guarantee that you will find a
  'Tools menu because it could be deleted.
  Set menuObj = menuSetObj.Menus(5)

  'Retrieve the MenuItems collection for the Tools menu
  Set menuItemsObj = menuObj.MenuItems

  'Locate the Macro menu item
  'Because you retrieved the built-in menus you know you will find it. If you
had
  'started from a custom menu you would need to handle the case of not finding
  'the menu item.
  For i = 0 To menuItemsObj.Count - 1
    'Retrieve the current menu item from the collection
    Set menuItemObj = menuItemsObj(i)

    'Check the CmdNum to see if it is Macro
    If menuItemObj.CmdNum = visCmdHierarchical And _
       menuItemObj.Caption = "&Macro" Then

       'The value of hierState is true
       hierState = menuItemObj.IsHierarchical

       'Retrieve the Menuitems collection for the hierarchical menu
       Set hiermenuItemsObj = menuItemObj.MenuItems
```

```
        'Locate the Visual Basic Editor menu item
        'As with the Macro menu item, you know you will find the VBE menu item
        'because you started with a copy of the built-in menus.
        For j = 0 To hiermenuItemsObj.Count - 1
          'Retrieve menu item from collection
          Set hiermenuItemObj = hiermenuItemsObj(j)

          'Check the CmdNum
            If hiermenuItemObj.CmdNum = visCmdToolsRunVBE Then

              'Delete the Visual Basic Editor menu item
              hiermenuItemObj.Delete

              'Exit the inside for loop
              Exit For
            End If
         Next j

       'Exit the outer for loop
       Exit For
     End If
  Next i

  'Tell Visio to use the custom user interface while the document is active.
  ThisDocument.SetCustomMenus uiObj

End Sub
```

IsInherited property

Applies to: Cell

Summary: Returns TRUE if the formula of the cell is inherited from a master or a style.

Version: VISIO 4.0

Syntax: intRet = object.**IsInherited**

Element	Description
intRet	TRUE (-1) if the object's formula is inherited; otherwise False (0)
object	The Cell object to examine

Remarks: In Visio's ShapeSheet window, the values and formulas of cells with local values are shown in blue. Values and formulas of cells that inherit from a master or style are shown in black.

See also: CellExists property, IsConstant property, RowCount property

Example for IsInherited

IsSeparator property

Applies to: MenuItem

Summary: Indicates whether a MenuItem object represents a separator on a menu.

Version: VISIO 4.0

Syntax: intRet = object.**IsSeparator**

Element	Description
intRet	TRUE (-1) if the menu item is a separator; otherwise False (0)
object	The MenuItem object to examine

Remarks: The CmdNum property of a MenuItem object that represents a separator is 0.

See also: CmdNum property

Example for IsSeparator

IsVisio16 property

Applies to: Application

Summary: Returns TRUE if the instance of Visio represented by the object is an instance of 16-bit Visio.

Version: VISIO 4.0

Syntax: intRet = object.**IsVisio16**

Element	Description
intRet	TRUE (-1) if object is a Win16 instance; otherwise False (0)
object	The Application object to examine

See also: IsVisio32 property

Example for IsVisio16

IsVisio32 property

Applies to: Application

Summary: Returns TRUE if the instance of Visio represented by the object is an instance of 32-bit Visio.

Version: VISIO 4.0

Syntax: intRet = object.**IsVisio32**

Element	Description
intRet	TRUE (-1) if the object is a Win32 instance; otherwise False (0)
object	The Application object to examine

See also: IsVisio16 property

Example for IsVisio32

Item property

Applies to: AccelItems, AccelTables, Addons, Attributes, Colors, Connects, Documents, Entities, EntityApps, EventList, Fonts, Layers, Masters, MenuItems, Menus, MenuSets, OLEObjects, Pages, Path, Paths, Selection, Shapes, StatusBarItems, StatusBars, Styles, ToolbarItems, Toolbars, ToolbarSets, Windows

Summary: Returns an object from a collection.

Version: VISIO 2.0

Syntax: objRet = object.**Item**(index)
objRet = object.**Item**(stringExpression)

Element	Description
objRet	The object retrieved from the collection
object	The collection that contains the object
index	The index of the object to retrieve
stringExpression	The name or unique ID of the object to retrieve

Remarks: You can retrieve an object from its collection by passing its index within that collection as the argument for the Item property. Item is the default property for all collections. When retrieving objects from a collection, the following statements are equivalent to the syntax examples given above (notice that Item is omitted from the expression):

```
objRet = object(index)
objRet = object(stringExpression)
```

You can retrieve an object in a Pages, Documents, Fonts, Layers, Masters, Styles, Shapes, or OLEObjects collection by passing the object's name as a string expression.

You can also pass the unique ID string of a Master or Shape to Item. For example:

```
objRet = shpObj.Item("{2287DC42-B167-11CE-88E9-0020AFDDD917}")
```

If such a string is passed to Shapes.Item, all shapes immediately contained in Shapes will be searched, but shapes within group shapes contained by Shapes will not be searched. To search all shapes in Shapes, including those not immediately contained by it, prefix the unique ID string with "*". For example:

```
objRet = shpObj.Item("*{2287DC42-B167-11CE-88E9-0020AFDDD917}")
```

See also: Index property, UniqueID property

Example for Item

```
*Shapes Property
```

ItemAtID property

Applies to: AccelTables, MenuSets, StatusBars, ToolbarSets

Summary: Returns the AccelTable, MenuSet, StatusBar, or ToolbarSet object for the indicated ID within the collection.

Version: VISIO 4.0

Syntax: objRet = object.**ItemAtID**(id)

Element	Description
objRet	The object retrieved from the collection
object	The collection that contains the object
id	The Visio context ID of the object to retrieve

Remarks: The ID corresponds to a window or context menu. Constants for IDs are prefixed with visUIObjSet and are declared by the Visio type library (and visconst.bas).

See also: AddAtID method, Item property

Example for ItemAtID

*BuiltInToolbars Property

ItemFromID property

Applies to: EventList, Fonts, Shapes, Styles

Summary: Returns an item of a collection given the ID of that item.

Version: VISIO 4.0

Syntax: objRet = object.**ItemFromID**(id)

Element	Description
objRet	The object retrieved from the collection
object	The collection that contains the object
id	The ID of the object to retrieve

Remarks: The ID of a shape uniquely identifies the shape within its page or master.

The ID of a style uniquely identifies the style within its document.

The ID of a font corresponds to the number stored in the Font cell of a row in a shape's Character properties section. Note that the ID associated with a particular font varies between systems or as fonts are installed and removed on a given system.

The ID of an Event uniquely identifies the Event in its EventList for the life of the EventList.

See also: CharProps property, Event object, ID property

Example for ItemFromID

Join method

Applies to: Selection, Window

Summary: Creates a new shape by joining selected shapes.

Version: VISIO 4.1

Syntax: object.**Join**

Element	Description
object	The Window or Selection object that contains the shapes to join

Remarks: The Join method is equivalent to choosing the Join command from the Operation submenu on the Shape menu in Visio. The new shape produced by Join inherits the text and formatting of the first selected shape and is the topmost shape in its container--the nth shape in the Shapes collection of its ContainingShape, where n = Count. The original shapes are deleted.

If the object being operated on is a Selection object, it has no shapes selected when the operation is complete.

Join and Combine are similar. Combine produces a shape that has one geometry section for each original shape. The resulting shape has holes in regions where the original shapes overlapped. Join differs from Combine in that it coalesces abutting line and curve segments in the original shapes into a single geometry section in the resulting shape.

You might want to Join shapes after importing a non-Visio drawing in which apparent polylines are represented by many independent shapes, each possessing a single line or curve segment. By joining the shapes that constitute a polyline in such a drawing, you can replace many single-segment shapes with one multiple-segment shape.

See also: Combine method, Fragment method, Intersect method, Subtract method, Trim method, Union method

Example for Join

Key property

Applies to: AccelItem

Summary: Gets or sets the ASCII key code value for an accelerator.

Version: VISIO 4.0

Syntax: object.**Key** = keyVal
keyVal = object.**Key**

Element	Description
object	An AccelItem object
keyVal	The ASCII value of the key used by the accelerator

Remarks: For a list of ASCII key code values, search the Windows SDK online help for Virtual Key Codes.

See also: Alt property, Control property, Shift property

Example for Key

Keywords property

Applies to: Document

Summary: Returns or sets the value of the Keywords field in a document's properties.

Version: VISIO 2.0

Syntax: strRet = object.**Keywords**
object.**Keywords** = stringExpression

Element	Description
strRet	The current value of the field
object	The Document object that has or gets the value
stringExpression	The new value of the field

Remarks: Setting the Keywords property is equivalent to entering information in the Keywords field in the Properties dialog box located on the File menu.

See also: Creator property, Description property, Subject property, Title property, Manager property, Company property, Category property, HyperlinkBase property

Example for Keywords

*Document Property

Language property

Applies to: Application

Summary: The language ID of the version of the Visio instance represented by the Application object.

Version: VISIO 3.0

Syntax: intRet = object.**Language**

Element	Description
intRet	The language ID
object	The Application object to examine

Remarks: This returns the language ID recorded in the object's VERSIONINFO resource. The IDs returned are the standard IDs used by Windows to encode different language versions. For example, the Language property returns &H0409 for the U.S. English version of Visio. For details, search the Windows SDK online help for VERSIONINFO.

Example for Language

Layer property

Applies to: Shape

Summary: Returns the i'th layer to which a shape is assigned.

Version: VISIO 4.0

Syntax: objRet = object.**Layer**(index)

Element	Description
objRet	A Layer object that represents the requested layer
object	The Shape object to examine
index	The ordinal of the layer to get

Remarks: A shape is assigned to 0 or more layers. The number of layers to which a shape is assigned equals the LayerCount property of that shape. If a shape is assigned to n layers, then the valid indexes that can be passed to its Layer property are 1 through n.

See also: Layer object, LayerCount property

Example for Layer

```
*Layers Property
```

LayerCount property

Applies to: Shape

Summary: Returns the number of layers to which a shape is assigned.

Version: VISIO 4.0

Syntax: intRet = object.**LayerCount**

Element	Description
intRet	The number of layers the shape is assigned to
object	The Shape object to examine

Remarks: A shape is assigned to 0 or more layers.

See also: Layer property

Example for LayerCount

*Layers Property

LayerName property

Applies to: Entity

Summary: Specifies the layer name represented by an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**LayerName**
object.**LayerName** = Expression

Element	Description
RetVal	The current layer name
object	The Entity object that has or gets the layer name
Expression	The new layer name

Remarks: If the group type of an Entity is 1003, then it contains the name of a layer. The layer name is stored as a string of up to 31 characters.

Example for LayerName

Layers property

Applies to: Master, Page

Summary: Returns the Layers collection of the indicated object.

Version: VISIO 4.0

Syntax: objRet = object.**Layers**

Element	Description
objRet	The Layers collection of the Master or Page object
object	The Master or Page object that owns the collection

See also: Layer property, Layers object, Master object, Page object

Example for Layer, LayerCount, Layers, Name

```
'This VBA macro demonstrates creating and adding shapes to layers.
'It also uses the LayerName and LayerCount properties.


Public Sub LayerProps_Example()

  Dim pageObj As Visio.Page
  Dim shapeObj As Visio.Shape
  Dim layerObj As Visio.Layer
  Dim layersObj As Visio.Layers

  If ActiveDocument Is Nothing Then
    Documents.Add ("")
  End If

  Set pageObj = ActivePage

  If pageObj Is Nothing Then
    Set pageObj = ActiveDocument.Pages(1)
  End If

  'Draw a rectangle
  Set shapeObj = pageObj.DrawRectangle(1, 5, 5, 1)

  'Get the layers collection
  Set layersObj = pageObj.Layers

  'Create a layer named ExampleLayer1 and add the shape to that layer
  Set layerObj = layersObj.Add("ExampleLayer1")
  layerObj.Add shapeObj, 1

  'Create a layer named ExampleLayer2 and add the shape to that layer
  Set layerObj = layersObj.Add("ExampleLayer2")
  layerObj.Add shapeObj, 1

  'Verify that the shape has been assigned to 2 layers
  Debug.Print "The page has " & shapeObj.LayerCount & " layers."

  'Set 1st layer as the layer object
  Set layerObj = shapeObj.Layer(1)

  'Verify by using the Name property
  Debug.Print "Current layerObj name is """ & layerObj.Name & """."

End Sub
```

Layout method

Applies to: Master, Page, Selection, Shape

Summary: Lays out the shapes and/or re-routes the connectors of the page, master, group or selection.

Version: VISIO 4.5

Syntax: object.**Layout**

Element	Description
object	The page, master, group or selection whose shapes are to be repositioned

Remarks: The Layout method performs the same operation as the Lay Out Shapes item in Visio's Tools menu.

Behavior of the Layout method can be influenced by setting the formulas or results of various user section cells of the page, master or group to be laid out. You can infer how these cells influence the behavior of the Layout method by examining the effect of choosing various options in the Lay Out Shapes dialog on the values of cells in the user section.

To lay out a subset of the shapes of a page, master or group, establish a Selection object in which the shapes to be laid out are selected. Then invoke Selection.Layout.

If Selection.Layout is performed and Selection has no shapes selected, all shapes in the page, master or group of the selection are laid out.

Example for Layout

LeftMargin property

Applies to: Document

Summary: Specifies the left margin for printing a document's pages.

Version: VISIO 4.0

Syntax: retVal = object.**LeftMargin**(units)
object.**LeftMargin**(units) = newValue

Element	Description
retVal	The margin value expressed in the given units
object	The Document object that has or gets the margin value
units	The units to use when retrieving or setting the margin value
newValue	The new margin value

Remarks: This property corresponds to the Left Margin control in Visio's Page Setup dialog box.

Units can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property.

See also: RightMargin property, TopMargin property, BottomMargin property, Result property

Example for LeftMargin

LengthIU property

Applies to: Shape

Summary: Returns the length (perimeter) of the object in internal units.

Version: VISIO 4.0

Syntax: retVal = object.**LengthIU**

Element	Description
retVal	The length (perimeter) of the object in internal units
object	The Shape object to examine

Remarks: The value returned is in inches.

See also: AreaIU property, BoundingBox method

Example for LengthIU

LineBasedOn property

Applies to: Style

Summary: Gets or sets the line style that the indicated Style object is based on.

Version: VISIO 4.0

Syntax: strVal = object.**LineBasedOn**
object.**LineBasedOn** = styleName

Element	Description
strVal	The name of the current based-on line style
object	The Style object that is based on the style
styleName	The name of the new based-on line style

Remarks: To base a style on no style, set LineBasedOn to a null string ("").

See also: BasedOn property, FillBasedOn property, TextBasedOn property

Example for LineBasedOn

LineStyle property

Applies to: Selection, Shape

Summary: Specifies the line style for an object.

Version: VISIO 2.0

Syntax: strRet = object.**LineStyle**
object.**LineStyle** = stringExpression

Element	Description
strRet	The name of the current line style
object	The Shape or Selection object that has or gets the line style
stringExpression	The name of the line style to apply

Remarks: Setting the LineStyle property is equivalent selecting a line style from the Line style list in Visio.

Setting a style to a non-existent style generates an error. Setting one kind of style to an existing style of another kind (for example, setting LineStyle to a fill style) does nothing. Setting one kind of style to an existing style that has more than one set of attributes changes only the attributes for that component. For example, setting LineStyle to a style with line, text, and fill attributes changes only the line attributes.

To preserve a shape's local formatting, use the LineStyleKeepFmt property.

See also: LineStyleKeepFmt property

Example for LineStyle

```
*DrawRectangle Method
```

LineStyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a line style to an object while preserving local formatting.

Version: VISIO 2.0

Syntax: object.**LineStyleKeepFmt** = stringExpression

Element	Description
object	The Shape or Selection object that has or gets the line style
stringExpression	The name of the style to apply

Remarks: Setting the LineStyleKeepFmt property is equivalent to checking the Preserve Local Formatting option in the Style dialog box in Visio.

Setting a style to a non-existent style generates an error. Setting one kind of style to an existing style of another kind (for example, setting LineStyleKeepFmt to a fill style) does nothing. Setting one kind of style to an existing style that has more than one set of attributes changes only the attributes for that component (for example, setting LineStyleKeepFmt to a style with line, text, and fill attributes changes only the line attributes).

See also: LineStyle property

Example for LineStyleKeepFmt

LoadFromFile method

Applies to: UI Object

Summary: Loads a Visio UI object from a file.

Version: VISIO 4.0

Syntax: object.**LoadFromFile** stringExpression

Element	Description
object	The UI object to receive data from the file
stringExpression	The name of the file to load

Remarks: You must use the SaveToFile method to save a UI object in a file that can be loaded with LoadToFile.

See also: SaveToFile method

Example for LoadFromFile

```
*SaveToFile Method
```

LocalName property

Applies to: Cell

Summary: Returns the local name of a cell.

Version: VISIO 4.0

Syntax: strRet = object.**LocalName**

Element	Description
strRet	The local name of the cell
object	The Cell object to examine

Remarks: A cell has both a local name and a universal name. The local name differs according to the locale for which Windows is installed on the user's system. The universal name is the same regardless of locale.

To get the universal name of a cell, use the Name property.

See also: Name property

Example for LocalName

LongValue property

Applies to: Entity

Summary: Gets or sets the long integer value of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**LongValue**
object.**LongValue** = Expression

Element	Description
RetVal	The current long integer value
object	The Entity object that has or gets the value
Expression	The new long integer value

Remarks: If an Entity object has a Group of 1071, then it contains a 32 bit long integer.

See also: Group property, RealValue property, ShortValue property

Example for LongValue

Manager property

Applies to: Document

Summary: Returns or sets the value of the Manager field in a document's properties.

Version: VISIO 5.0

Syntax: strRet = object.**Manager**
object.**Manager** = stringExpression

Element	Description
strRet	The current value of the field
object	The Document object that has or gets the value
stringExpression	The new value of the field

Remarks: Setting the Manager property is equivalent to entering information in the Manager field in the Properties dialog box, accessed from the File menu.

See also: Description property, Keywords property, Subject property, Title property, Company property, Category property, HyperlinkBase property

Example for Manager

*Document Property

MarkerEvent event

Applies to: Application

Summary: The event that occurs when the QueueMarkerEvent method is invoked.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtMarker (&H1100)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Application object emitting this event
moreInfo	Nothing for this event

Remarks:

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: QueueMarkerEvent method, Action property, Add method, AddAdvise method, Event object, EventList object

Example for MarkerEvent

```
*QueueMarkerEvent method
```

Master property

Applies to: Layer, Layers, Shape, Window

Summary: Gets the master that is displayed in the indicated window, or returns the master from which the Shape object was created or the master that contains the Layer or Layers object.

Version: VISIO 2.0

Syntax: objRet = object.**Master**

Element	Description
objRet	A Master object that represents the object's master
object	The object to examine

Remarks: If the indicated window is not showing a master, Master returns Nothing. If the indicated window is a master that is opened for editing, the master returned is the actual master being edited, not the temporary master that exists while the actual master is being edited.

If the Shape object is not an instance of a master, its Master property returns Nothing. If the Shape object is in a group, its Master property is the same as the group's.

If the Layer or Layers object is from a page rather than a master, its Master property returns Nothing.

See also: Master object, Page property

Example for Master

MasterAdded event

Applies to: Application, Document, Documents, Masters

Summary: The event that occurs after a new master is added to a document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtAdd+visEvtMaster (&H8008)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Master that was just created
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, Master object

Example for MasterAdded

MasterChanged event

Applies to: Application, Document, Documents, Master, Masters

Summary: The event that occurs after certain properties of a master are changed.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtMaster (&H2008)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Master that just changed
moreInfo	Nothing for this event

Remarks: The MasterChanged event indicates that the following has occurred:

A change to the master has caused its properties to be propagated to its instances.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for MasterChanged

Masters property

Applies to: Document

Summary: Returns the Masters collection for the indicated document's stencil.

Version: VISIO 2.0

Syntax: objsRet = object.**Masters**

Element	Description
objsRet	The Masters collection for the indicated document
object	The Document object that owns the collection

See also: Masters object

Example for Masters

```
'This Visual Basic program demonstrates printing all the master shape names in
'the current document to the debug window.

'Make sure you have a document open before running this program.

Sub DumpMasterNames ()

  Dim I As Integer, iMastCount As Integer
  Dim appVisio As Object, CurDoc As Object, DocMstrs As Object

  Set appVisio = GetObject(, "visio.application")

  If appVisio Is Nothing Then
    MsgBox "Visio not loaded"
    Exit Sub
  End If

  Set CurDoc = appVisio.ActiveDocument

  If CurDoc Is Nothing Then
    MsgBox "No Stencil Loaded"
    Exit Sub
  End If

  Set DocMstrs = CurDoc.Masters

  Debug.Print "Master Name Dump For Document : "; CurDoc.Name

  iMastCount = DocMstrs.Count

  If iMastCount > 0 Then
    For I = 1 To iMastCount
      Debug.Print "  "; DocMstrs.Item(I).Name
    Next I
  Else
    Debug.Print "  No Masters"
  End If

End Sub
```

MatchByName property

Applies to: Master

Summary: Determines how Visio decides if a local master is already present when an instance of a master is dropped on the drawing page.

Version: VISIO 5.0

Syntax: intRet = object.**MatchByName**
object.**MatchByName** = intExpression

Element	Description
intRet	True (-1) if match by name is enabled; otherwise False (0)
object	The Master object that has or gets the setting
intExpression	True (non-zero) if match by name is enabled; otherwise False (0)

Remarks: The MatchByName property allows changes made to a local master to apply to new instances of the master, even if the instances are dragged from a standalone stencil file.

Setting MatchByName is equivalent to checking or unchecking Match Master Name On Drop in the Properties dialog box, accessed from the Master shortcut menu.

Suppose you create an instance of a master from a stencil in a document (producing a local copy of the master in that document), and then make modifications to the local master (such as changing its fill color).

If the MatchByName property of the local master is false, then dragging the original master from the standalone stencil into the drawing will make an instance with the standalone master's attributes and will produce a second local master.

If the MatchByName property of the local master is true, then dragging the original master from the standalone stencil into the drawing will make an instance with the local master's attributes and will not produce a second local master.

Example for MatchByName

MDIWindowMenu property

Applies to: Menu

Summary: Determines whether this menu can be used by the MDI window manager to list the currently open MDI windows.

Version: VISIO 4.0

Syntax: object.**MDIWindowMenu** = intVal
intVal = object.**MDIWindowMenu**

Element	Description
object	The Menu object that has or gets the setting
intVal	Non-zero if the Menu object should be the MDI window menu; otherwise 0

Remarks: The MDIWindowMenu property usually refers to the Window menu.

Example for MDIWindowMenu

MenuItems property

Applies to: Menu, MenuItem

Summary: Returns the MenuItems collection of a Menu or MenuItem object.

Version: VISIO 4.0

Syntax: objRet = object.**MenuItems**

Element	Description
objRet	The MenuItems collection of the object
object	The Menu or MenuItem object that owns the collection

Remarks: If a Menu object represents a hierarchical submenu, its MenuItems collection contains submenu items. Otherwise, its MenuItems collection is empty.

See also: MenuItems object

Example for MenuItems

*ActionText Property

Menus property

Applies to: MenuSet

Summary: Returns the Menus collection of a MenuSet object.

Version: VISIO 4.0

Syntax: objRet = object.**Menus**

Element	Description
objRet	The Menus collection of the MenuSet object
object	The MenuSet object that owns the collection

Remarks: A Menu object's index within the Menus collection determines its left-to-right position on the menu bar.

See also: Menus object

Example for Menus

*ActionText Property

MenuSets property

Applies to: UI Object

Summary: Returns the MenuSets collection of a UI object.

Version: VISIO 4.0

Syntax: objRet = object.**MenuSets**

Element	Description
objRet	The MenuSets collection of a UI object
object	The UI object that owns the collection

Remarks: If a UI object represents menus and accelerators (for example, if the object was retrieved using the BuiltInMenus property of an Application or Document object), its MenuSets collection represents all of the menus for that UI object.

Use the ItemAtID property of a MenuSets object to retrieve menus for a particular window context such as the drawing window. If a context does not include menus (as only a few do not), it has no MenuSets collection. For a list, see the MenuSets object.

See also: ItemAtID property, MenuSets object

Example for MenuSets

```
*ActionText Property
```

MiniHelp property

Applies to: MenuItem

Summary: Gets or sets the string that appears in the status bar when a menu item is selected.

Version: VISIO 4.0

Syntax: object.**MiniHelp** = miniHelpStr
miniHelpStr = object.**MiniHelp**

Element	Description
object	The MenuItem object that has or gets the minihelp string
miniHelpStr	The minihelp string

Remarks: If MiniHelp is null and the MenuItem object's CmdNum property is set to one of Visio's command IDs, Visio uses the default minihelp text from the built-in Visio user interface.

See also: CmdNum property

Example for MiniHelp

```
*ActionText Property
```

Mode property

Applies to: Document

Summary: Determines whether a document is in run mode or design mode.

Version: VISIO 5.0

Syntax: intRet = object.**Mode**
object.**Mode** = intExpression

Element	Description
intRet	Current mode of the document
object	The Document object that has or gets the setting
intExpression	visDocModeRun (0) or visDocModeDesign (1)

Remarks: Use the Mode property of a Document object to obtain the document's mode. A Visio 5.0 document is either in run mode or in design mode, just as a Visual Basic form is either running or being designed.

The fundamental distinctions between run mode and design mode are:

(1) ActiveX controls hosted in a document are told not to fire events when the document is in design mode and to fire events when in run mode.
2) Visio will not source events from any object whose document is in design mode.

Document.Mode will report one of the following values:
visDocModeRun (0)
visDocModeDesign (1)

The run/design mode of a Visio document is reported in the Visio UI by the Design Mode button on the Developer toolbar. The appearance of this button is the same as the Design Mode button in the Visual Basic Editor window. If pressed, the document (project) is in design mode. If not pressed, the document (project) is in run mode.

The run/design mode of a Visio document is synchronized with the run/design state of the document's VBA project, provided the document has a project. If the document transitions to/from run mode, then the project's mode will switch, and vice versa. This means that if code in a document's project sets the document's mode to design mode (ThisDocument.Mode = visDocModeDesign), the project in which the code executes will transition to design mode and any statements following the mode assignment statement will not execute. However, code in a document can put another document (project) into design mode and keep running.

A document's mode is not a persistent property. A document opens in run mode unless the user chooses the Disable Macros option from the Document Macro Warning dialog box that appears if the Visio Macro Virus Protection option is enabled. In this latter case, the document remains in design mode for as long as it remains open, and attempts by the user or programs to put the document in run mode will fail.

See also: DesignModeEntered event, RunModeEntered event

Example for Mode

Name property

Applies to: Addon, Attribute, Cell, Document, Entities, Entity, EntityApp, Font, Layer, Master, Page, Shape, Style, UI Object

Summary: Specifies the name of an object.

Version: VISIO 2.0

Syntax: strRet = object.**Name**
object.**Name** = stringExpression

Element	Description
strRet	The current name of the object
object	The object that has or gets the name
stringExpression	The new name of the object

Remarks: You cannot set the Name property of a Document object. If a document is not yet named, this property returns the document's temporary name, such as Drawing1 or Stencil1.

You cannot set the Name property of an Addon object or a Font object.

You can set the Name property of a Style object that represents a style that is not a Visio default style (e.g., "Text Only", "None", "Normal", or "No Style"). If you attempt to set the Name property of a default style, an error is generated.

You can get, but not set, the name of a cell. Some cells are in named rows. You can both get and set the name of a named row using the RowName property.

A cell has both a local name and a universal name. The local name will differ depending on which locale the running version of Windows is installed for. The universal name will be the same regardless of what locale is installed.

To get the universal name of a cell, use the Name property. To get the local name, use LocalName.

See also: LocalName property, RowName property, GetNames method

Example for Name

*Layers Property

NameID property

Applies to: Shape

Summary: Returns unique name for a shape.

Version: VISIO 2.0

Syntax: strRet = object.**NameID**

Element	Description
strRet	The unique name of the shape
object	The Shape object to examine

Remarks: The NameID property returns a unique identifier for each shape on a page or master. The identifier has the following form:

sheet.X

where X is the shape's ID property. This will be a number from 1 to 4095.

NameID is unique within a page or master, but not across pages or masters. At any moment, no other shape on the indicated shape's page or master will have the same NameID. But shapes on other pages or masters may have the same NameID. A shape's UniqueID is unique across pages and masters.

Also, NameID's are reused. If a shape whose NameID is sheet.X is deleted, then a shape subsequently added to the same context may be assigned sheet.X as its NameID. Therefore, persisting NameIDs in separate data stores is generally not as sound as persisting UniqueIDs.

See also: EventInfo property, Name property, UniqueID property, ID property

Example for NameID

NewWindow property

Applies to: Hyperlink

Summary: Returns or sets whether Visio will open a new window when Visio navigates to the hyperlink target.

Version: VISIO 5.0

Syntax: intRet = object.**NewWindow**
object.**NewWindow** = intExpression

Element	Description
intRet	0 if navigate won't open new window; -1 if navigate will open new window
object	The object that has or gets the setting
intExpression	0 to cause navigate not to open new window; non-zero to cause navigate to open window

Remarks: Setting the NewWindow property of a Hyperlink object is equivalent to setting the NewWindow cell in the shape's hyperlink row.

See also: Address property, SubAddress property, Frame property, Description property, ExtraInfo property

Example for NewWindow

```
*Follow method
```

Object property

Applies to: OLEObject, Shape

Summary: Returns an IDispatch interface on the ActiveX control or embedded or linked OLE 2.0 object represented by a Shape object or an OLEObject object.

Version: VISIO 4.1

Syntax: dispRet = shpobj.**Object**

Element	Description
dispRet	IDispatch interface on ActiveX control or OLE object represented by shape
shpobj	The Shape object or OLEObject object to examine

Remarks: ShpObj.Object will raise an exception if ShpObj doesn't represent an ActiveX control or an OLE 2.0 embedded or linked object. A shape represents an ActiveX control or an OLE 2.0 embedded or linked object if the visTypeIsOLE2 bit (&H8000) is set in the value returned by ShpObj.ForeignType.

If ShpObj.Object succeeds, it will return an IDispatch interface on the control or object. You owe an eventual release on the returned value. (Set it to nothing or let it go out of scope if you're using Visual Basic.) You can determine the kind of object you've obtained an interface on by using ShpObj.ClassID or ShpObj.ProgID.

Beginning with Visio 5.0, if the object returned by ShpObj.Object is embedded and if the shape inherits the object from its master, then ShpObj.Object will sever the instance, that is, copy the inherited data into the instance. Otherwise if the client receiving the IDispatch from ShpObj.Object makes changes to the object, all instances of the master, not just the instance being queried will change. If the object returned by ShpObj.Object is linked, a sever is not performed since, by definition, there may be other entities referencing the link. ShpObj.ObjectIsInherited was added to Visio 5.0 so that client programs can know if a shape inherits its object and access the master's object(s) if that is what it really wants to do.

See also: ForeignType property, ClassID property, ProgID property, ObjectIsInherited property

Example for Object

ObjectIsInherited property

Applies to: Shape

Summary: Indicates if a shape represents an ActiveX or OLE object that is inherited from the shape's master.

Version: VISIO 5.0

Syntax: intRet = object.**ObjectIsInherited**

Element	Description
intRet	TRUE (-1) if object is inherited; otherwise FALSE (0)
object	The Shape object that has the setting

See also: ForeignType property, Object property

Example for ObjectIsInherited

ObjectType property

Applies to: Addon, Addons, Application, Cell, Characters, Color, Colors, Connect, Connects, Curve, Document, Documents, Event, EventList, Font, Fonts, Hyperlink, Layer, Layers, Master, Masters, OLEObject, OLEObjects, Page, Pages, Path, Paths, Selection, Shape, Shapes, Style, Styles, Window, Windows

Summary: Returns the object's type.

Version: VISIO 4.1

Syntax: intRet = object.**ObjectType**

Element	Description
object	The object to examine
intRet	The type of the object

Remarks: Constants representing object types are prefixed with visObjType and are declared by the Visio type library (and visconst.bas).

Versions of Visio prior to 4.1 had an undocumented method called Dump, which took an integer argument (typically ignored) and returned an integer that indicated the type of its object. ObjectType is essentially identical to Dump, except it takes no arguments.

Although Dump was undocumented and unsupported, it may have been used in some programs. Dump should be replaced with ObjectType for compatibility with Visio 4.1.

Example for ObjectType

OLEObjects property

Applies to: Document, Master, Page

Summary: Returns the OLEObjects collection of a document, master, or page.

Version: VISIO 5.0

Syntax: objRet = object.**OLEObjects**

Element	Description
objRet	An OLEObjects collection
object	The object to examine

Remarks: The OLEObjects property returns an OLEObjects collection that includes any OLE 2.0 linked or embedded objects or ActiveX controls contained in a document, master, or page.

See also: OLEObject object

Example for OLEObjects

```
*Progid property
```

OnDataChangeDelay property

Applies to: Application

Summary: Controls whether a container application updates a Visio object that is in place in the container.

Version: VISIO 3.0

Syntax: intRet = object.**OnDataChangeDelay**
object.**OnDataChangeDelay** = intExpression

Element	Description
intRet	The current OnDataChangeDelay setting of the object
object	The Application object that has or gets the setting
intExpression	The new OnDataChangeDelay setting of the object

Remarks: The OnDataChangeDelay property is used to control how frequently Visio will send OnDataChange advises to the container of a Visio document.This only affects instances of Visio that were run from within an OLE container document.

Setting OnDataChangeDelay to 0 will cause Visio to send immediate advises to the container as changes occur to the documents Visio has open.

Setting OnDataChangeDelay to -1 causes Visio to use the interval specified in the [OLEUpdateDelay] entry in VISIO.INI. If VISIO.INI has no such entry, Visio defaults to using a value of 10000 (milliseconds). When an instance of Visio runs, it initializes its OnDataChangeDelay value to 0. If both OnDataChangeDelay and OLEUpdateDelay are 0, Visio will never send advises to the container.

Setting OnDataChangeDelay to any value other than -1 or 0 will set the delay between advises to that number of milliseconds.

Example for OnDataChangeDelay

OneD property

Applies to: Master, Shape

Summary: Determines whether an object behaves as a 1-D object.

Version: VISIO 2.0

Syntax: retVal = object.**OneD**
object.**OneD** = intExpression

Element	Description
retVal	TRUE if the shape is 1-D; FALSE if the shape is 2-D
object	The Master or Shape object that has or gets the setting
intExpression	0 to declare object as 2-D; non-zero to declare it 1-D

Remarks: Setting the OneD property is equivalent to changing a shape's interaction style in the Behavior dialog box. Setting the OneD property for a 1-D shape to FALSE deletes the 1-D Endpoints section from its ShapeSheet, even if the cells in that section were protected with the GUARD function.

You cannot set the OneD property of a Master object. A guide has no OneD property. The OneD property of an object from another application is always FALSE.

Example for OneD

Open method

Applies to: Documents, Master

Summary: Opens an existing Visio file, or opens a Visio master so it can be editted.

Version: VISIO 2.0

Syntax: docObjRet = docsObj.**Open** (stringExpression)
masterObjCopy = masterObj.**Open**

Element	Description
masterObjCopy	A temporary copy of masterObj
docObjRet	A Document object that represents the file that was opened
masterObj	A Master object that is to be editted
docsObj	The Documents collection to receive the opened file
stringExpression	The name of a file to open

Remarks: Documents.Open opens a Visio file as an original. Depending on the filename extension, the Open method opens a drawing (.VSD), a stencil (.VSS), a template (.VST), or a workspace (.VSW).

If the file does not exist or the filename is invalid, no Document object is returned and an error is generated.

If a valid stencil (.VSS) filename is passed, the original stencil file is opened, which means you can edit its masters. Unless you want to create or edit the masters, it is recommended that you open a stencil read-only through an associated template or by using the OpenEx method.

Beginning with Visio 4.1, Master.Open can be used in conjunction with Master.Close to reliably edit the shapes and cells of a Visio Master. In previous versions of Visio you could edit a Master's shapes and cells, but the changes would not be pushed to instances of the master, and alignment box information displayed when instancing the edited master would not be correct.

To edit the shapes and cells of a Master from a program, you should:
1. Open the Master for editing using masterObjCopy = masterObj.Open. This will fail if there is a drawing window open into masterObj or if other programs already have masterObj open. If Open succeeds, masterObjCopy will be a copy of masterObj.
2. Change the shapes and cells you want to alter. Change them in masterObjCopy, not masterObj.
3. Close the Master using masterObjCopy.Close. Close will fail if masterObjCopy isn't a Master that resulted from a prior masterObj.Open call. Otherwise, Close will merge the changes made in step 2 from masterObjCopy back into masterObj. It will also update all instances of masterObj to reflect the changes and update information cached in masterObj that is used for purposes such as drawing its bounding box when it is dragged. If masterObj.IconUpdate isn't visManual (0), Close will update the icon shown in the stencil window for masterObj to depict an image of masterObjCopy.

If you change the shapes and cells of a master directly, as opposed to opening and closing it as described above, the effects listed in step 3 won't occur.

Note: A program that makes an open for edit copy of a Master should both Close and release the copy. Visual Basic will typically do the release automatically. From C/C++ you must explicitly release the copy, just as you would do for any other object.

See also: Add method, Drop method, OpenEx method, Close method

Example for Open

```
'This VBA macro demonstrates opening files and opening, editing, and
'closing a Visio master.

Public Sub OpenDoc ()

  Dim docObj As Visio.Document
  Dim mastObj As Visio.Master

  'Open a blank document (not based on a template).
  Set docObj = Documents.Add("")

  'Open a new document based on a template.
  Set docObj = Documents.Add("c:\visio\solutions\flowchart\flowchrt.vst")

  'Open a stencil docked and in the read-only mode.
  Documents.OpenEx "flowchart.vss", visOpenDocked

  'Open a document in original mode.
  Set docObj = Documents.Open("c:\visio\solutions\flowchart\flowchrt.vst")

  'Open a master shape, edit it, then close it.
  Set mastObj = ThisDocument.Masters(1).Open
  mastObj.DrawRectangle 1, 2, 3, 4
  mastObj.Close

End Sub
```

OpenDrawWindow method

Applies to: Master, Page, Shape

Summary: Opens a new drawing window that displays a page, master or group.

Version: VISIO 4.1

Syntax: objRet = object.**OpenDrawWindow**

Element	Description
objRet	A Window object that represents the opened window
object	The page, master or group to display in the drawing window

Remarks: OpenDrawWindow opens a new drawing window, even if the page, master, or group is already displayed in a drawing window.

See also: OpenIconWindow method, OpenSheetWindow method, OpenStencilWindow method

Example for OpenDrawWindow

OpenEx method

Applies to: Documents

Summary: Opens an existing Visio file using extra information passed in an argument.

Version: VISIO 4.0

Syntax: objRet = object.**OpenEx** (fileName, openFlags)

Element	Description
objRet	A Document object that represents the file that was opened
object	The Documents collection to receive the opened file
fileName	The name of the file
openFlags	Flags that indicate how to open the file

Remarks: OpenEx is identical to Open, except that it provides an extra argument in which the caller can specify how the document opens. OpenFlags should be a combination of zero or more of the following:

visOpenCopy = 1
visOpenRO = 2
visOpenDocked = 4
visOpenDontList = 8

If visOpenCopy is specified, a copy of the file is opened.

If visOpenRO is specified, the file is opened read-only.

If visOpenDocked is specified, the file is shown in a docked rather than an MDI window, provided that the file is a stencil file and there is an active drawing window in which to put the docked stencil window.

If visOpenDontList is specified, the name of the opened file won't appear in the list of recently opened documents on the File menu.

See also: Open method, SaveAsEx method

Example for OpenEx

```
*SaveAs Method
```

OpenIconWindow method

Applies to: Master

Summary: Opens an icon window that shows a master's icon.

Version: VISIO 4.1

Syntax: objRet = object.**OpenIconWindow**

Element	Description
objRet	A Window object that represents the opened window
object	The Master object whose icon is to be displayed in the icon window

Remarks: If the master's icon is already displayed in an icon window, OpenIconWindow activates that window rather than opening another window.

See also: OpenDrawWindow method, OpenSheetWindow method, OpenStencilWindow method

Example for OpenIconWindow

```
*Type Property
```

OpenSheetWindow method

Applies to: Shape

Summary: Opens a ShapeSheet window for a Shape object.

Version: VISIO 4.1

Syntax: objRet = object.**OpenSheetWindow**

Element	Description
objRet	A Window object that represents the opened window
object	The Shape object whose ShapeSheet is to be displayed

Remarks: OpenSheetWindow opens a new ShapeSheet window for the shape even if its ShapeSheet is already displayed in another window.

See also: OpenDrawWindow method, OpenIconWindow method, OpenStencilWindow method

Example for OpenSheetWindow

```
*Type Property
```

OpenStencilWindow method

Applies to: Document

Summary: Opens a stencil window that shows the masters in the stencil of a document.

Version: VISIO 4.1

Syntax: objRet = object.**OpenStencilWindow**

Element	Description
objRet	A Window object that represents the opened window
object	The Document object whose stencil is to be displayed

Remarks: If the document's stencil is already displayed in a stencil window, OpenStencilWindow activates that window rather than opening another window.

See also: OpenDrawWindow method, OpenIconWindow method, OpenSheetWindow method

Example for OpenStencilWindow

```
*Type Property
```

Page property

Applies to: Layer, Layers, Window

Summary: Gets or sets the page that is displayed in the indicated window, or gets the page that contains the indicated layer or layers.

Version: VISIO 2.0

Syntax: objVariantRet = windowObj.**Page**
windowObj.**Page** = stringVariant

objRet = LayerOrLayersObj.**Page**

Element	Description
objVariantRet	A Page object that represents the page being shown returned in a variant
windowObj	The Window object that has or gets the setting
stringVariant	A variant to which is assigned a string that names the page to be shown
objRet	The Page object that contains the layer or layers
LayerOrLayersObj	The Layer object or Layers collection that has the setting

Remarks: If the indicated window is not showing a page (maybe it is showing a master), Page returns an empty variant. Otherwise the returned variant refers to the Page object that the window is showing.

[Note: In earlier versions of Visio (through version 4.1), Window.Page returned an object (as opposed to a variant of type object) and Window.Page accepted a string (as opposed to a variant of type string). Due to changes in Automation support tools, it became necessary to change the property to accept and return variants. For backward compatibility, PageAsObj and PageFromName were added. PageAsObj and PageFromName have the same signatures and occupy the same vtble slots as did the prior version of Page.]

If the Layer object or Layers collection is in a master rather than in a page, the Page property returns Nothing. You cannot set the Page property of a Layer object or Layers collection.

See also: Master property, Page object, PageAsObj property, PageFromName property

Example for ActiveWindow, Page

```
'This VBA macro demonstrates adding and naming pages and using the
'Page property of various objects. It also demonstrates setting the
'active window's page.

Public Sub PageProp_Example()

  Dim pageObj1 As Visio.Page, pageObj2 As Visio.Page
  Dim tmpPageObj As Visio.Page
  Dim layerObj1 As Visio.Layer, layerObj2 As Visio.Layer
  Dim layersObj1 As Visio.Layers, layersObj2 As Visio.Layers

  'Set the current page name to MyPage1
  ActivePage.Name = "MyPage1"

  'Use the Page property to return the page object from the window object.
  Set pageObj1 = ActiveWindow.Page

  'Verify that the expected page was received.
  Debug.Print "The active window contains: " & pageObj1.Name

  'Add a second page named MyPage2.
  Set pageObj2 = ActiveDocument.Pages.Add
  pageObj2.Name = "MyPage2"

  'Get the layers collection from each page.
  Set layersObj1 = pageObj1.Layers
  Set layersObj2 = pageObj2.Layers

  'Create a layer for each of the layers collections.
  Set layerObj1 = layersObj1.Add("ExampleLayer1")
  Set layerObj2 = layersObj2.Add("ExampleLayer2")

  'Use the Page property to return the page object from a layers object.
  Set tmpPageObj = layersObj1.Page

  'Verify that the expected page was received.
  Debug.Print "layersObj1 is from: " & tmpPageObj.Name

  'Use the Page property to return the page object from a layer object.
  Set tmpPageObj = layerObj2.Page

  'Verify that the expected page was received.
  Debug.Print "layerObj2 is from: " & tmpPageObj.Name

  'Set the active window's page to "MyPage1."
  ActiveWindow.Page = "MyPage1"

End Sub
```

PageAdded event

Applies to: Application, Document, Documents, Pages

Summary: The event that occurs after a new page is added to a Visio document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtAdd+visEvtPage (&H8010)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Page that was just created
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, Page object

Example for PageAdded

*DocumentSaved Event

PageAsObj property

Applies to: Window

Summary: Gets the page that is displayed in the indicated window.

Version: VISIO 4.5

Syntax: objRet = object.**PageAsObj**

Element	Description
objRet	A Page object that represents the page being shown or nothing
object	The Window object that is showing the page

Remarks: If the indicated window is not showing a page, PageAsObj returns nothing. Otherwise the Page object that is being shown in the window is returned

[Note: Up through and including version 4.1 of Visio, WindowObj.Page returned an object. Due to changes in Automation support tools, it became necessary to change the Page property to return a variant of type object. For backwards compatibility, PageAsObj was added. It behaves like the Page property used to, and occupies the same slot in the vtble as the old property. If you're developing new code, you'll likely find very few occasions when you must use PageAsObj.]

See also: Page property, PageFromName property

Example for PageAsObj

PageChanged event

Applies to: Application, Document, Documents, Page, Pages

Summary: The event that occurs after certain properties of a page are changed.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtPage (&H2010)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Page that just changed
moreInfo	Nothing for this event

Remarks: The PageChanged event indicates that one of the following has changed:

* The name of the page
* The background page assigned to this page
* Whether this page itself is a background page or a foreground page

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Background property, BackPage property, Event object, EventList object

Example for PageChanged

PageFromName property

Applies to: Window

Summary: Sets the page that is displayed in the indicated window.

Version: VISIO 4.5

Syntax: object.**PageFromName** = stringExpression

Element	Description
object	The Window object whose name is returned
stringExpression	The name of the page to be shown

Remarks: [Note: Up through and including version 4.1 of Visio, WindowObj.Page accepted a string. Due to changes in Automation support tools, it became necessary to change the Page property to return a variant of type string. For backwards compatibility, PageFromName was added. It behaves like the Page property used to, and occupies the same slot in the vtble as the old property. If you're developing new code, you'll likely find very few occasions when you must use PageFromName.]

See also: Page property, PageAsObj property

Example for PageFromName

Pages property

Applies to: Document

Summary: Returns the Pages collection of a document.

Version: VISIO 2.0

Syntax: objsRet = object.**Pages**

Element	Description
objsRet	The Pages collection for the indicated document
object	The Document object that owns the collection

See also: Document object, Pages object

Example for Pages

```
'This VBA macro demonstrates printing the names of a document's pages.

Public Sub PrintPageNames ()

  Dim I As Integer
  Dim docObj As Visio.Document
  Dim PageList As Visio.Pages

  'Get the Pages collection for the active document
  Set PageList = ActiveDocument.Pages

  Debug.Print "Page names for document : "; ActiveDocument.Name

  'Iterate through the pages and print the Page name in the VBA debug window.
  For I = 1 To PageList.Count
    Debug.Print "  "; PageList.Item(I).Name
  Next I

End Sub
```

PageSheet property

Applies to: Master, Page

Summary: Returns the page sheet of a page or master.

Version: VISIO 4.0

Syntax: objRet = object.**PageSheet**

Element	Description
objRet	A Shape object that represents a page sheet
object	The Master or Page object that owns the page sheet

Remarks: Every page and master contains a tree of shape objects. Shapes can be of the following types:

visTypePage = 1
visTypeGroup = 2
visTypeShape = 3
visTypeForeignObject = 4
visTypeGuide = 5

In the tree of shapes of a master or page, there is exactly one shape of type visTypePage. This shape is always the root shape in the tree, and it is this shape that this PageSheet property returns.

The page sheet contains important settings for the page or master such as its size and scale. It also contains the Layers section that defines the layers for that page or master.

An alternative way to obtain a page's or master's page shape is to use the following:

```
shpObj = pageOrMasterObj.Shapes("ThePage")
```

See also: Shape object

Example for PageSheet

PaletteEntry property

Applies to: Color

Summary: Gets or sets the red, green, blue, and flags components of the color.

Version: VISIO 4.0

Syntax: intRet = object.**PaletteEntry**
object.**PaletteEntry** = intVal

Element	Description
intRet	The current value of the color's components
object	The Color object that has or gets the components
intVal	The new value of the color's components

Remarks: A color is represented by 1-byte red, green, and blue components. It also has a 1-byte flags field indicating how the color is to be used. These correspond to members of the Windows PALETTEENTRY data structure. For details, search the Windows SDK online help for PALETTEENTRY.

The value passed is 4 tightly packed BYTE fields. The correspondence between PaletteEntry and red, green, blue, and flags values is:

```
palentry == r+256*(b+256*(g+256*f))
```

See also: Blue property, Flags property, Green property, Red property

Example for PaletteEntry

PaperHeight property

Applies to: Document

Summary: Returns the height of a document's pages.

Version: VISIO 4.5

Syntax: retVal = object.**PaperHeight**(units)

Element	Description
retVal	The document's paper height expressed in the given units
object	The Document object that has the setting
units	The units to use when retrieving the paper height

Remarks: Units can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property.

See also: PaperWidth property, PaperSize property, Result property

Example for PaperHeight

PaperSize property

Applies to: Document

Summary: Gets or sets a code that determines the size of a document's pages.

Version: VISIO 4.5

Syntax: retVal = object.**PaperSize**
object.**PaperSize** = intVal

Element	Description
retVal	Integer code designating present page size
object	The Document object that has or gets the setting
intVal	Integer code designating new page size

Remarks: The value gotten or returned from PaperSize is the integer code stored in the dmPaperSize field of the DEVMODE structure maintained in a Visio document. For a complete listing of page size codes, search the Windows SDK online help for "DEVMODE."

Some common settings are:
- Letter size pages: 1
- Legal size pages: 5
- A4 pages: 9

See also: PaperHeight property, PaperWidth property

Example for PaperSize

PaperWidth property

Applies to: Document

Summary: Returns the width of a document's pages.

Version: VISIO 4.5

Syntax: retVal = object.**PaperWidth**(units)

Element	Description
retVal	The document's paper width expressed in the given units
object	The Document object that has the setting
units	The units to use when retrieving the paper width

Remarks: Units can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property.

See also: PaperHeight property, PaperSize property, Result property

Example for PaperWidth

ParaProps property

Applies to: Characters

Summary: Sets the indicated paragraph property of a Characters object to a new value.

Version: VISIO 3.0

Syntax: object.**ParaProps**(intWhichProp) = intExpression

Element	Description
object	The Characters object that gets the new value
intWhichProp	The property to set
intExpression	The new value of the property

Remarks: The values of the intWhichProp argument correspond to named cells in the Paragraph section of the ShapeSheet. Constants for intWhichProp are declared by the Visio type library (and visconst.bas):

visIndentFirst = 0
visIndentLeft = 1
visIndentRight = 2
visSpaceLine = 3
visSpaceBefore = 4
visSpaceAfter = 5
visHorzAlign = 6

For information about types of formatting, see information about the applicable cell in the Visio online Help.

To retrieve information about existing formats, use the ParaPropsRow property.

Depending on the extent of the text range and the format, setting the ParaProps property may cause rows to be added or removed from the Paragraph section of the ShapeSheet.

See also: CharProps property, ParaPropsRow property, ShapeSheet Cells

Example for ParaProps

ParaPropsRow property

Applies to: Characters

Summary: Returns the index of the row in the Paragraph section of a ShapeSheet that contains paragraph formatting information for a Characters object.

Version: VISIO 3.0

Syntax: intRet = object.**ParaPropsRow**(bias)

Element	Description
intRet	The index of the row that defines the Character object's paragraph format
object	The Characters object to examine
bias	The direction of the search

Remarks: If the formatting for the Characters object is represented by more than one row in the Paragraph section in the ShapeSheet, ParaPropsRow returns -1. If the Characters object represents an insertion point rather than a sequence of characters (that is, if its Begin and End properties return the same value), use the bias argument to determine which row index to return:

visBiasLeft = 1
visBiasRight = 2
visBiasLetVisioChoose = 0

Specify visBiasLeft for the row that covers paragraph formatting for the character to the left of the insertion point, or visBiasRight for the row that covers paragraph formatting for the character to the right of the insertion point.

See also: CharPropsRow property, ParaProps property, TabPropsRow property

Example for ParaPropsRow

Parent property

Applies to: AccelItem, AccelItems, AccelTable, AccelTables, Menu, MenuItem, MenuItems, Menus, MenuSet, MenuSets, Shape, StatusBar, StatusBarItem, StatusBarItems, StatusBars, Toolbar, ToolbarItem, ToolbarItems, Toolbars, ToolbarSet, ToolbarSets

Summary: Determines the parent of an object.

Version: VISIO 3.0

Syntax: objRet = object.**Parent**

Element	Description
objRet	The parent of the indicated object
object	The object to examine

Remarks: In general, an object's parent is the object that contains it. For example, the parent of a Menu object is the Menus collection that contains the Menu object.

If a Shape object is a member of a group, the parent is that group. Otherwise, its parent is a Page or a Master object.

Example for Parent

ParseLine method

Applies to: Document

Summary: Parses a line of Visual Basic code.

Version: VISIO 4.5

Syntax: object.**ParseLine** stringExpression

Element	Description
object	The Document object whose VBA project is to parse code
stringExpression	A string that will be interpreted as VBA code

Remarks: The VBA project of the Document object is told to parse the supplied string. VBA will treat the string much like it would treat the same string typed into its immediate window.

See also: ExecuteLine method, AddonName property

Example for ParseLine

Paste method

Applies to: Characters, Page, Window

Summary: Pastes the contents of the Windows Clipboard into the indicated object.

Version: VISIO 2.0

Syntax: object.**Paste**

Element	Description
object	The object to paste

Remarks: If the contents of the Clipboard are valid for pasting into the indicated object, the Paste method pastes them. For example, if the Clipboard contains a shape, it can be pasted onto a page.

See also: Copy method, Cut method, Delete method, Duplicate method

Example for Paste

```
*Copy Method
```

Path property

Applies to: Application, Document

Summary: Returns the drive and folder path of Visio or a document.

Version: VISIO 2.0

Syntax: strRet = object.**Path**

Element	Description
strRet	The path of Visio or the indicated document
object	The Application or Document object to examine

Remarks: AppObj.Path returns the name of the folder that contains the Visio program files. If Visio's program files are located in C:\VISIO, AppObj.Path will return C:\VISIO\.

The Path property of a document with a name of C:\VISIO\DRAWINGS\MYDRAW.VSD returns C:\VISIO\DRAWINGS\. If the document has not been saved, the Path property returns a null string.

The returned value can include UNC drive names (for example, \\bob\leo.)

See also: FullName property, Name property

Example for Path

*Points property

Paths property

Applies to: Shape

Summary: Returns a Paths collection that reports the coordinates of a shape's paths in the coordinate system of the shape's parent.

Version: VISIO 5.0

Syntax: objRet = object.**Paths**

Element	Description
objRet	A Paths object that represents the shape's strokes
object	The Shape object to get the Paths of

See also: Paths object, PathsLocal property

Example for Paths

*Points property

PathsLocal property

Applies to: Shape

Summary: Returns a Paths collection that reports the coordinates of a shape's paths in the the shape's local coordinate system.

Version: VISIO 5.0

Syntax: objRet = object.**PathsLocal**

Element	Description
objRet	A Paths object that represents the shape's strokes
object	The Shape object to get the Paths of

See also: Paths object, Paths property

Example for PathsLocal

PatternFlags property

Applies to: Master

Summary: Determines whether a master will behave as a custom pattern.

Version: VISIO 5.0

Syntax: intRet = object.**PatternFlags**
object.**PatternFlags** = intExpression

Element	Description
intRet	The current PatternFlags value of the master
object	The Master object that has or gets the setting
intExpression	The new PatternFlags value of the master

Remarks: Visio allows a master to be used as a custom line pattern, line end, or fill pattern.

Master.PatternFlags determines whether a master is meant to be used as a pattern (non-zero); whether it is a line, fill, or line end pattern; and which pattern mode to use when applying it to shapes.

If Master.PatternFlags is meant to be a pattern (non-zero), it can include a combination of the following bits:

visMasIsLinePat (&H1) -- line pattern
visMasIsLineEnd (&H2) -- line end pattern
visMasIsFillPat (&H4) -- fill pattern

If visMasIsLinePat is selected, the pattern mode should be one of the following values, which coincide with the options shown in the Properties dialog box, accessed from the Master's shortcut menu, when Line Pattern is selected:

VisMasLPTileDeform (&H0)
visMasLPTile (&H10)
visMasLPStretch (&H20)
visMasLPAnnotate (&H30)

In addition, visMasLPScale (&H40) can optionally be included in PatternFlag's value.

If visMasIsLineEnd is selected, the pattern mode should be one of the following values, which coincide with the options shown in the Properties dialog box, accessed from the Master's shortcut menu, when Line End is selected:

visMasLEDefault (&H0)
visMasLEUpright (&H100)

In addition, visMasLEScale (&H400) can optionally be included in PatternFlag's value.

If visMasIsFillPat is selected, the pattern mode should be one of the following values, which coincide with the options shown in the Properties dialog box, accessed from the Master's shortcut menu, when Fill Pattern is selected:

visMasFPTile (&H0)
visMasFPCenter (&H1000)
visMasFPStretch (&H2000)

In addition, visMasFPScale (&H4000) can optionally be included in PatternFlag's value.

Example for PatternFlags

Persistable property

Applies to: Event

Summary: Determines whether an Event can potentially persist within its document.

Version: VISIO 4.1

Syntax: intRet = object.**Persistable**

Element	Description
intRet	False (0) if the Event cannot be made persistent; True (-1) if it can
object	The Event object to examine

Remarks: The Persistable property of an Event object indicates whether the event can persist--whether the event can be stored with a Visio document between executions of a program. Whether an event can persist depends on two conditions:

* The action code of the Event object must be visActCodeRunAddon. If the action code is visActCodeAdvise, the event won't persist and must be re-created by a program at run time.

* The source object must be capable of containing persistent events in its EventList. The source object's PersistsEvents property indicates whether it can contain persistent events.

Although an Event object's Persistable property indicates whether an event can persist, its Persistent property indicates whether that event actually will persist. When an Event object is first created, its Persistent property is set to the same value as its Persistable property. That is, a persistable event's Persistent property is set to True, and a non-persistable event's Persistent property is set to False.

A non-persistent event exists as long as a reference is held on the Event object, the EventList object that contains the Event object, or the source object that has the EventList object. When the last reference to any of these objects is released, the non-persistent event ceases to exist.

You can change the initial setting for a persistable event by setting its Persistent property to False. In this case, the event will not persist with its document, even though it could. However, you cannot change the Persistent property of a non-persistent event--attempting to do so will cause an exception.

See also: EventList object, Persistent property, PersistsEvents property

Example for Persistable

Persistent property

Applies to: Event

Summary: Determines whether or not an event will persist with its document.

Version: VISIO 4.1

Syntax: intRet = object.**Persistent**
object.**Persistent** = intExpression

Element	Description
intRet	False (0) if the event won't be saved with the document; True (-1) if it will
object	The Event object that has or gets the setting
intExpression	False (0) to make the event non-persistent; True (non-zero) to make it persistent

Remarks: The Persistent property determines whether a persistable event will actually persist with its document. An event is persistable if its action code is visActCodeRunAddon and if the event's source object is capable of containing persistent events.

When an event is first created, its Persistent property is set to the same value as its Persistable property--if an event can persist, Visio assumes it should persist. You can change the initial setting for a persistable event by setting its Persistent property to False. However, you cannot change the Persistent property of a non-persistable event--attempting to do so will cause an exception.

A non-persistent event exists as long as a reference is held on the Event object, the EventList object that contains the Event object, or the source object that has the EventList object. When the last reference to any of these objects is released, the non-persistent event ceases to exist.

A persistent event exists until its Event object is deleted from the source object's EventList.

See also: Action property, EventList object, Persistable property, PersistsEvents property

Example for Persistent

PersistsEvents property

Applies to: Application, Cell, Characters, Document, Documents, Layer, Layers, Master, Masters, Page, Pages, Selection, Shape, Shapes, Style, Styles, Window, Windows

Summary: Indicates whether this object is capable of containing persistent events in its EventList.

Version: VISIO 4.1

Syntax: intRet = object.**PersistsEvents**

Element	Description
intRet	False (0) if this object cannot contain persistent events; True (-1) if it can
object	The object to examine

Remarks: PersistsEvents is a property of every object that has an EventList property. To be persistable, an event's action code must be visActCodeRunAddon, but it must also be in the EventList of an object whose PersistsEvents property is True.

Whether a persistable event actually does persist depends on the setting of its Persistent property.

See also: EventList object, Persistable property, Persistent property

Example for PersistsEvents

PitchAndFamily property

Applies to: Font

Summary: Returns the pitch and family code for a Font object.

Version: VISIO 4.0

Syntax: intRet = object.**PitchAndFamily**

Element	Description
intRet	The pitch and family code of the Font object
object	The Font object to examine

Remarks: Use the PitchAndFamily property to specify a font's pitch and assign it to a font family. You can specify pitch, family, or both. To specify both, use an Or expression. Font families are used to specify a font when an exact typeface is unavailable.

The possible values of the PitchAndFamily property correspond to those of the lfPitchAndFamily member of the Windows LOGFONT data structure. For details, search the Windows SDK online help for LOGFONT.

See also: CharSet property

Example for PitchAndFamily

Point method

Applies to: Curve

Summary: Returns a point at some position along a curve.

Version: VISIO 5.0

Syntax: object.**Point**(t, x, y)

Element	Description
object	The Curve object to get a point of
t	The value in the Curve's parameter domain to get the point at
x	Returns x value of Curve at t
y	Returns y value of Curve at t

Remarks: A Curve object describes itself in terms of its parameter domain, which is the range [Start(),End()]. The Point method of a Curve object returns the x,y coordinates at position t, which is any position along the curve's path. The Point method can be used to extrapolate the curve's path outside of [Start(),End()].

See also: End property, PointAndDerivatives method, Start property, Points property

Example for End, Point, PointAndDerivatives, Start

```
' This VBA macro places a shape on the document's active page,
' then retrieves it and iterates through its paths collection
' and each path object to display the coordinates of various
' points along the curve.

Sub CurveAndPointExample()

Dim shapeObj    As Visio.Shape      'a shape object
Dim pathsObj    As Visio.Paths      'the paths collection
Dim pathObj     As Visio.Path       'a path object
Dim curveObj    As Visio.curve      'a curve object
Dim startpoint As Double, endpoint As Double
Dim x As Double, y As Double
Dim dx As Double, dy As Double, ddx As Double, ddy As Double
Dim i As Integer, j As Integer

'get the Paths collection for this shape
Set pathsObj = ActivePage.DrawOval(1, 1, 4, 4).Paths

'step through for each Path object in the Paths collection
For i = 1 To pathsObj.Count
Set pathObj = pathsObj.Item(i)
Debug.Print "Path object " & i

    'For each curve in a path object
    For j = 1 To pathObj.Count

        Set curveObj = pathObj(j)
        Debug.Print "Curve number " & j

        'display the start point of the curve'
        startpoint = curveObj.start
        Debug.Print "Startpoint= " & startpoint

        endpoint = curveObj.End
        Debug.Print "Endpoint= " & endpoint

        'display the midpoint of the curve
        curveObj.Point endpoint / 2, x, y
        Debug.Print "Midpoint= " & x, y

        'demonstrate the PointAndDerivatives method asking for 1 derivative
        curveObj.PointAndDerivatives startpoint - 1, 1, x, y, dx, dy, ddx, ddy
        Debug.Print "PointAndDerivative= " & x, y, dx, dy

    Next j
    Debug.Print "This path has " & j - 1 & " curve object(s)."

Next i
Debug.Print "This shape has " & i - 1 & " path object(s)."

End Sub
```

PointAndDerivatives method

Applies to: Curve

Summary: Returns a point and, optionally, derivatives at some position along a curve's path.

Version: VISIO 5.0

Syntax: object.**PointAndDerivatives**(t, n, x, y, dx, dy, ddx, ddy)

Element	Description
object	The Curve object to get point and derivatives of
t	The value in the Curve's parameter domain to evaluate
n	0: get point; 1: point and 1st derivative; 2: point plus first and second derivative
x	Returns x value of Curve at t
y	Returns y value of Curve at t
dx	Returns dxdt at t if n > 0
dy	Returns dydt at t if n > 0
ddx	Returns ddxdt at t if n > 1
ddy	Returns ddydt at t if n > 1

Remarks: Use the PointAndDerivatives method of the Curve object to obtain the coordinates of a point within the curve's parameter domain and, optionally, its first and second derivatives.

A Curve object describes itself in terms of its parameter domain which is the range [Start(),End()]. The PointAndDerivatives method can be used to extrapolate the curve's path outside [Start(),End()].

See also: End property, Point method, Start property, Points property

Example for PointAndDerivatives

```
*Point method
```

Points property

Applies to: Curve, Path

Summary: Returns an array of points that defines a polyline that approximates a Path or Curve object to within a given tolerance.

Version: VISIO 5.0

Syntax: object.**Points**(Tolerance, xyArray)

Element	Description
object	The Path or Curve object to get the point stream of
Tolerance	Specifies how close returned array of points must approximate true path
xyArray	Returns array of alternating x and y values specifying points along Path or Curve's stroke

Remarks: Use the Points property of the Path or Curve object to obtain an array of x,y coordinates specifying points along the Path or Curve within a given tolerance. Tolerance and the returned x,y values are expressed in internal drawing units (inches).

If Shape.Paths was used to obtain the Path or Curve being queried, the coordinates will be expressed in the parent's coordinate system. If Shape.PathsLocal was used to obtain the Path or Curve, the coordinates will be expressed in the local coordinate system.

If Visio is unable to achieve the requested tolerance, Visio will approximate the points as close to the requested tolerance as possible. Generally speaking, the lower the tolerance, the more points Visio will return. Visio will not accept a tolerance of 0.

The array returned includes both the starting and ending points of the Path or Curve even if the Path or Curve is closed.

See also: Point method, PointAndDerivatives method, Start property, End property

Example for Path, Paths, Points

```
'This VBA macro places a shape on the page, retrieves its paths collection,
'then uses the Points property of the Path object to return an array of points
'that defines a polyline approximating the Path object.

Public Sub DisplayPath()

    Dim shpobj As Visio.Shape
    Dim xyArray() As Double
    Dim str As String
    Dim i As Integer
    Dim j As Integer

    Set shpobj = ActivePage.DrawOval(1, 1, 4, 4)

    For i = 1 To shpobj.Paths.Count

        shpobj.Paths(i).Points 1#, xyArray
        For j = LBound(xyArray) To UBound(xyArray)
            str = str & xyArray(j) & Chr(10)
        Next j

    Next i
    Debug.Print str

End Sub
```

Print method

Applies to: Document, Page

Summary: Prints the contents of an object to the default printer.

Version: VISIO 2.0

Syntax: object.**Print**

Element	Description
object	The Page or Document object to print

Remarks: For a Document object, this method prints all of the indicated document's pages. Background pages are printed on the same sheet of paper as the foreground pages they are assigned to.

For a Page object, this method prints the indicated page and its background page (if any) on the same sheet of paper.

If you're using VBA or Visual Basic 4.0, you must assign the method result to a dummy variable and you must apply the method to a variable of type Object, not of type Visio.Document or Visio.Page. For example, if docObj is of type Visio.Document:

```
dim docObjTemp as Object
Set docObjTemp = docObj
dim dummy as String
dummy = docObjTemp.Print
```

Example for Print

PrintCenteredH property

Applies to: Document

Summary: Indicates whether drawings will be centered between the left and right edges of the paper when printed.

Version: VISIO 4.0

Syntax: intRet = object.**PrintCenteredH**
object.**PrintCenteredH** = newValue

Element	Description
intRet	-1 if the document will center drawings horizontally when printing, otherwise 0
object	The Document object that has or gets the setting
newValue	Non-zero to center drawings horizontally when printing, otherwise zero

Remarks: This property corresponds to the Center Left/Right control in Visio's Page Setup dialog box.

See also: PrintCenteredV property

Example for PrintCenteredH

PrintCenteredV property

Applies to: Document

Summary: Indicates whether drawings will be centered between the top and bottom edges of the paper when printed.

Version: VISIO 4.0

Syntax: intRet = object.**PrintCenteredV**
object.**PrintCenteredV** = newValue

Element	Description
intRet	-1 if the document will center drawings vertically when printing, otherwise 0
object	The Document object that has or gets the setting
newValue	Non-zero to center drawings vertically, otherwise zero

Remarks: This property corresponds to the Center Up/Down control in Visio's Page Setup dialog box.

See also: PrintCenteredH property

Example for PrintCenteredV

PrintFitOnPages property

Applies to: Document

Summary: Indicates that drawings in a document will printed on a specified number of sheets across and down.

Version: VISIO 4.0

Syntax: intRet = object.**PrintFitOnPages**
object.**PrintFitOnPages** = newValue

Element	Description
intRet	-1 if the document will fit drawings on a specified number of sheets, otherwise 0
object	The Document object that has or gets the setting
newValue	Non-zero to fit drawings on a specified number of sheets, otherwise zero

Remarks: This property corresponds to the Fit On control in Visio's Page Setup dialog box. If this property is set, Visio prints the document's drawings on the number of sheets across and down specified by PrintPagesAcross and PrintPagesDown.

See also: PrintPagesAcross property, PrintPagesDown property

Example for PrintFitOnPages

PrintLandscape property

Applies to: Document

Summary: Indicates whether a document's drawings will print in landscape or portrait orientation.

Version: VISIO 4.0

Syntax: intRet = object.**PrintLandscape**
object.**PrintLandscape** = newValue

Element	Description
intRet	-1 if the document will print drawings in landscape orientation, otherwise 0
object	The Document object that has or gets the setting
newValue	Non-zero to print drawings in landscape orientation, otherwise zero

Remarks: This property corresponds to the Portrait/Landscape controls in Visio's Page Setup dialog box.

Example for PrintLandscape

PrintPagesAcross property

Applies to: Document

Summary: Indicates the number of sheets of paper across on which a drawing will be printed if PrintFitOnPages is set.

Version: VISIO 4.0

Syntax: intRet = object.**PrintPagesAcross**
object.**PrintPagesAcross** = newValue

Element	Description
intRet	The number of sheets across on which drawings will be printed
object	The Document object that has or gets the setting
newValue	The number of sheets across on which to print drawings

Remarks: This property corresponds to the Pages Across control in Visio's Page Setup dialog box. If PrintFitOnPages is false, this setting is ignored.

See also: PrintFitOnPages property, PrintPagesDown property

Example for PrintPagesAcross

PrintPagesDown property

Applies to: Document

Summary: Indicates how many sheets of paper down on which a drawing will be printed if PrintFitOnPages is set.

Version: VISIO 4.0

Syntax: intRet = object.**PrintPagesDown**
object.**PrintPagesDown** = newValue

Element	Description
intRet	The number of sheets down on which drawings will be printed
object	The Document object that has or gets the setting
newValue	The number of sheets down on which to print drawings

Remarks: This property corresponds to the Pages Down control in Visio's Page Setup dialog box. If PrintFitOnPages is false, this setting is ignored.

See also: PrintFitOnPages property, PrintPagesAcross property

Example for PrintPagesDown

PrintScale property

Applies to: Document

Summary: Indicates how much drawings will be reduced or enlarged when printed.

Version: VISIO 4.0

Syntax: retVal = object.**PrintScale**
object.**PrintScale** = newValue

Element	Description
retVal	The scale at which drawings will be printed; 1.0 equals 100%
object	The Document object that has or gets the setting
newValue	The new scale value

Remarks: This property corresponds to the Scale control in Visio's Page Setup dialog box. To print a drawing at half its size, specify 0.5. To print a drawing at twice its size, specify 2.0.

See also: PrintFitOnPages property

Example for PrintScale

Priority property

Applies to: StatusBarItem, ToolbarItem

Summary: Determines when a toolbar or status bar item is dropped from view when the Visio window is too narrow to show all items.

Version: VISIO 4.0

Syntax: object.**Priority** = intVal
intVal = object.**Priority**

Element	Description
object	The object that has or gets the priority
intVal	The priority of the status bar item or toolbar item

Remarks: The higher the value of the Priority property, the more likely the item is to be dropped from the toolbar or status bar on low-resolution monitors or in narrow windows. For example, an object with a priority of 10 is more likely to be dropped than an object with a priority of 2.

Example for Priority

*BuiltInToolbars Property

ProcessID property

Applies to: Application

Summary: Returns a unique process ID for the indicated instance of Visio.

Version: VISIO 2.0

Syntax: retVal = object.**ProcessID**

Element	Description
retVal	The process ID for the instance of Visio
object	The Application object that represents the instance

Remarks: The ProcessID property returns a value unique to the indicated instance. The value is not reused until 4294967296 (2^32) further processes have been created on the current workstation.

Example for ProcessID

```
'This VB program demonstrates using the ProcessID property.

Sub ShowProcessID ()

  Dim appVisio As Visio.Application

  'Creates an instance of Visio
  Set appVisio = CreateObject("visio.application")

  'Prints the Process ID in the VB debug window.
  Debug.Print "Visio Process ID : "; appVisio.ProcessID

End Sub
```

ProfileName property

Applies to: Application

Summary: Returns the name of the Visio application object's profile (.ini) file.

Version: VISIO 4.0

Syntax: strRet = object.**ProfileName**

Element	Description
strRet	The name of Visio's profile (.ini) file
object	An Application object

Example for ProfileName

ProgID property

Applies to: OLEObject, Shape

Summary: Returns the programmatic identifier of a shape representing an ActiveX control or an embedded or linked OLE 2.0 object.

Version: VISIO 5.0

Syntax: strRet = object.**ProgID**

Element	Description
strRet	The program identifier of the OLE object represented by the shape
object	The Shape object to examine

Remarks: The ProgID property will raise an exception if the shape doesn't represent an ActiveX control or OLE 2.0 embedded or linked object. A shape represents an ActiveX control or an OLE 2.0 embedded or linked object if the visTypeIsOLE2 bit (&H8000) is set in the value returned by shpObj.ForeignType.

Use the ProgID property of a Shape object or OLEObject to obtain the programmatic identifier of the object. Every OLE object class stores a programmatic identifier for itself in the registry. Typically this occurs when the program that services the object installs itself. Client programs use this identifier to identify the object. You are using the Visio identifier when you execute a statement such as GetObject(,"Visio.Application") from a Visual Basic program.

The following are examples of strings ProgID might return:

```
Visio.Drawing.5
MSGraph.Chart.5
Forms.CommandButton.1
```

After using a shape's Object property to obtain an IDispatch interface on the object the shape represents, you can obtain the shape's ClassID or ProgID to determine the methods and properties provided by that interface.

See also: ClassID property, ForeignType property, Object property

Example for ClassID, OLEObjects, ProgID

```
' This VBA macro demonstrates getting the OLEObjects collection on an active
page
' and printing the ProgID and ClassID for each OLEObject object in the
Immediate
' window. To run this macro, make sure that the active page has at least one
OLE
' 2.0 embedded or linked object, or an ActiveX Control.
'
Public Sub DisplayOLEObjects()

    Dim i As Integer
    Dim theObjects As Visio.OLEObjects


    ' Retrieve the OLEObjects collection for the active page
    Set theObjects = ActivePage.OLEObjects

    ' Step through the collection of OLEObjects on the page
    For i = 1 To theObjects.Count
        Debug.Print theObjects(i).ProgID
        Debug.Print theObjects(i).ClassID
    Next i

End Sub
```

Prompt property

Applies to: Attribute, Master

Summary: Returns or sets the prompt string for the indicated object.

Version: VISIO 2.0

Syntax: strRet = object.**Prompt**
object.**Prompt** = stringExpression

Element	Description
strRet	The current prompt string
object	The Master or Attribute object that has or gets the prompt string
stringExpression	The new prompt string

Remarks: For a Master object this property returns or sets the status bar prompt for the corresponding master.

Example for Prompt

PromptForSummary property

Applies to: Application

Summary: Determines whether Visio prompts for document properties when it saves a document.

Version: VISIO 4.0

Syntax: intRet = object.**PromptForSummary**
object.**PromptForSummary** = intExpression

Element	Description
intRet	0 if prompting is off, -1 if it is on
object	The Application object that has or gets the setting
intExpression	0 to turn prompting off, non-zero to turn it on

Remarks: This property corresponds to the "Prompt for document properties on save" checkbox in Visio's Options dialog box.

See also: Creator property, Description property, Keywords property, Subject property, Title property

Example for PromptForSummary

PurgeUndo method

Applies to: Application

Summary: Empties the Visio queue of undo actions.

Version: VISIO 5.0

Syntax: object.**PurgeUndo**

Element	Description
object	The Application object whose undo queue is to be purged

Remarks: After calling Application.PurgeUndo, the queue of undo actions will be empty and no operation performed before the call can be reversed.

See also: Undo method, Redo method

Example for PurgeUndo

PutShape method

Applies to: ShapeData

Summary: Sets the shape to be used by the ShapeData object.

Version: VISIO 3.0 TECH

Syntax: object.**PutShape** objShape

Element	Description
object	The ShapeData object to use the Shape object
objShape	The Visio Shape object to associate with the ShapeData object

Remarks: The ShapeData object is external to Visio, so you must indicate a Shape object to work with before you can retrieve any of its properties. The object parameter for the PutShape method must be a Visio Shape object. After setting this property, you can retrieve the shape through the Shape property. There is no way to remove this shape from the ShapeData object after you have set it. Therefore, when a program is finished with a ShapeData object or any of its collections, it is important to release every outstanding object either through an explicit Set object = Nothing assignment or by making sure all objects go out of scope.

Releasing the object is important because the ShapeData object keeps a local copy of the Shape object, and this pointer may become invalid over time. For example, if you create a global ShapeData object and give it a Shape object that a user later deletes, the ShapeData object doesn't recognize this, and any operations performed on it will fail.

You cannot reuse a ShapeData object by using the PutShape method more than once. After the first valid PutShape call, the ShapeData object will no longer accept new Shape objects. You must create a new ShapeData object for every shape to which you want to attach the database.

Example for Attributes, DefaultValue, EntityApps, PutShape, Value

```
'This VBA macro demonstrates using the Put method and Put property.

Public Sub DatabaseTest ()

  Dim shapeDataObj As Object
  Dim shpObj As Visio.Shape
  Dim attribsObj As Object
  Dim attribObj As Object
  Dim entityAppsObj As Object
  Dim entityAppObj As Object
  Dim entitiesObj As Object
  Dim entityObj As Object

  'Get the Shape database
  Set ShapeDataObj = CreateObject("Visio.ShapeDatabase")

  Set shpObj = ThisDocument.Pages(1).DrawRectangle(1, 2, 2, 1)

  'Before using the Shape database you must initalize it using
  'the PutShape property.  To speed things up, use the
  'BeginTransaction property to force the database to
  'delay all reads and writes until a matching EndTransaction.
  shapeDataObj.PutShape shpObj
  shapeDataObj.BeginTransaction

  'Manipulate the attribute data
  Set attribsObj = shapeDataObj.Attributes
  Set attribObj = attribsObj.Add("Part #")
  attribObj.Value = "XYZ-123"
  attribObj.DefaultValue = "xxx-xxx"
  attribObj.Prompt = "Please enter the part #"

  'To add Extended Entity Data to a shape, you must first create an EntityApp
  'object to represent your application.
  Set entityAppsObj = shapeDataObj.EntityApps
  Set entityAppObj = entityAppsObj.Add("TestApp")

  'After you get an EntityApp, you can retrieve its Entities
  'collection and add an Entity.
  Set entitiesObj = entityAppObj.Entities

  'Add a vector Entity.
  Set entityObj = entitiesObj.Add

  entityObj.VectorX = 1.23
  entityObj.VectorY = -423.1002
  entityObj.VectorZ = 12.932

  'To ensure that the transactions above are written back to the
  'shape, call EndTransaction.  If you don't call EndTransaction,
  'changes are ignored.
  shapeDataObj.EndTransaction
```

```
End Sub
```

QueueMarkerEvent method

Applies to: Application

Summary: Queues a "marker event" that will fire after all presently queued events.

Version: VISIO 5.0

Syntax: intRet = object.**QueueMarkerEvent**(stringExpression)

Element	Description
intRet	The sequence number of the event that will eventually fire
object	The Application object to queue the event
stringExpression	An arbitrary string that will be passed with the event that eventually fires

Remarks: The QueueMarkerEvent method enables a client to identify events it caused as opposed to events it did not cause.

The QueueMarkerEvent method returns the sequence number of the MarkerEvent it queues. This number will be unique within an instance of Visio for the course of that instance. A client can compare event sequence numbers to determine which events it caused and respond accordingly.

ContextString (legally null or empty) is a string that the client can pass to the QueueMarkerEvent method that will be recorded in the EventInfo of the MarkerEvent event when it fires. The QueueMarkerEvent method of the Visio Application object's event set is passed the ContextString directly.

The following is an example, expressed in pseudocode, of one possible way that a client program can use the QueueMarkerEvent method to ignore events that it caused:

```
‘ Toggles to true if received events were caused by this app
ICausedThisEvent = false

‘ bracket a group of operations
app.queuemarkerevent("ContextStrIdentifyingMe")
     code to perform operations
app.queuemarkerevent("ContextStrIdentifyingMe")

'Depending on the method chosen to handle the event, you could use one of the two
following examples
'of how to interrogate the event to determine if it is your MarkerEvent
          ‘ If using addadvise to connect viseventproc to markerevent
             viseventproc ()
             if event is markerevent and app.eventinfo is "ContextStrIdentifyingMe"
                       IcausedThisEvent =True
             endif

            ‘ if using Dim WithEvents app as Visio.Application
             app.markerevent(app,seqn,bstrctx)
             if bstrctx is "ContextStrIdentifyingMe"
                        ICausedThisEvent = True
```

```
            endif

' In other event procs:
if ICausedThisEvent   = False
        respond to this event
else
        this is something I caused - don't do anything
endif
```

See also: MarkerEvent event

Example for MarkerEvent, QueueMarkerEvent

```
'Paste this code into ThisDocument and run UseMarker
'Output will be displayed in the VBA Immediate Window

Dim WithEvents appVisio As Visio.Application

Private Sub appVisio_MarkerEvent(ByVal app As Visio.IVApplication, ByVal
SequenceNum As Long, ByVal ContextString As String)
    Debug.Print "Marker: " & app.EventInfo(0)
End Sub

Private Sub appVisio_ShapeAdded(ByVal Shape As Visio.IVShape)
    Debug.Print "  ShapeAdded: " & Shape.Name
End Sub

Public Sub UseMarker()
    Set appVisio = ThisDocument.Application

    'Marker Events can be used to comment a segment of events in the queue
    appVisio.QueueMarkerEvent "I am starting..."
    ActivePage.DrawRectangle 0, 0, 3, 3
    appVisio.QueueMarkerEvent "I am finished..."

    'Sample output from the VBA Immediate Window:
        'Marker: I am starting...
        '  ShapeAdded: Sheet.1
        'Marker: I am finished...
End Sub
```

Quit method

Applies to: Application

Summary: Closes the indicated instance of Visio.

Version: VISIO 2.0

Syntax: object.**Quit**

Element	Description
object	The Application object that represents the instance to close

Remarks: If the Quit method is invoked when a document with unsaved changes is open in the indicated instance of Visio, a dialog box appears asking if you want to save the document. If you want to quit Visio without saving and without seeing the dialog box, set the Saved property of the Document object representing the document to True immediately before quitting. Set the Saved property to True only if you are sure you want to close the document without saving changes.

See also: AlertResponse property

Example for Quit

ReadOnly property

Applies to: Document

Summary: Indicates whether the file was opened as read-only.

Version: VISIO 2.0

Syntax: retVal = object.**ReadOnly**

Element	Description
retVal	TRUE if the document was opened read-only; otherwise FALSE
object	The Document object to examine

Example for ReadOnly

RealValue property

Applies to: Entity

Summary: Specifies the real value of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**RealValue**
object.**RealValue** = Expression

Element	Description
RetVal	The current real value as a double
object	The Entity object that has or gets the value
Expression	The new real value as a double

Remarks: If an Entity object has a Group of 1040, 1041, or 1042, then it contains a real value as a double.

See also: Group property

Example for RealValue

Red property

Applies to: Color

Summary: Gets or sets the intensity of the red component of a Color object.

Version: VISIO 4.0

Syntax: intRet = object.**Red**
object.**Red** = intVal

Element	Description
intRet	The current value of the color's red component
object	The Color object that has or gets the component
intVal	The new value of the color's red component

Remarks: The Red property can be a value from 0 to 255.

A color is represented by red, green and blue components. It also has flags that indicate how the color is to be used. These correspond to members of the Windows PALETTEENTRY data structure. For details, search the Windows SDK online help for PALETTEENTRY.

See also: Blue property, Flags property, Green property, PaletteEntry property

Example for Red

Redo method

Applies to: Application

Summary: Reverses the most recent Undo command.

Version: VISIO 2.0

Syntax: object.**Redo**

Element	Description
object	The Application object in which to reverse Undo

Remarks: To reverse the effect of the Undo method, use the Redo method. For example, if you clear an item and restore it with the Undo method, use the Redo method to clear the item again.

See also: Undo method, PurgeUndo method

Example for Redo, Undo

```
'This VBA macro demonstrates undoing and redoing actions.

Public Sub UnDoReDo_Example ()

  Dim shpObj As Visio.Shape

  'Draw a rectangle, delete it using Undo, and then redraw it using Redo.
  Set shpObj = ActivePage.DrawRectangle(1, 5, 5, 1)

  Visio.Application.Undo                'Delete the shape.
  Visio.Application.Redo                'Bring it back.

End Sub
```

Remove method

Applies to: Layer

Summary: Removes a shape from a layer.

Version: VISIO 4.0

Syntax: object.**Remove** shapeObj, fPreserveMembers

Element	Description
object	The Layer object from which to remove the shape
shapeObj	The Shape object to remove
fPreserveMembers	Whether to remove members of a group

Remarks: If the shape is a group and fPreserveMembers is non-zero, member shapes of the group are unaffected. If fPreserveMembers is zero, the group's member shapes are also removed from the layer.

Removing a shape from a layer does not delete the shape.

Example for Remove

RemoveFromGroup method

Applies to: Window

Summary: Removes selected objects from groups.

Version: VISIO 2.0

Syntax: object.**RemoveFromGroup**

Element	Description
object	The Window object that contains the group and the selected objects

See also: AddToGroup method, Group method, Ungroup method

Example for RemoveFromGroup

Result property

Applies to: Cell

Summary: Returns or sets a cell's value.

Version: VISIO 2.0

Syntax: retVal = object.**Result** (units)
object.**Result** (units) = newValue

Element	Description
retVal	The value in the cell
object	The Cell object that contains the value to get or set
units	The units to use when retrieving or setting the cell's value
newValue	The new value for the cell

Remarks: Use the Result property to set the value of an unguarded cell. If the cell's formula is protected with the GUARD function, the formula is not changed and an error is generated. If the cell contains only a text string, then 0 is returned.

Units can be a string such as "inches", "inch", "in.", or "i". If the string is invalid, an error is generated. Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the following constants declared by the Visio type library (and visconst.bas):

```
visNumber = 32      'non-dimensional number
visDate = 40      ' date
visTypeUnits = 48      'number with no explicit units
visPoints = 50      'points
visPicas = 51      'picas
visDidots = 53      'didots
visCiceros = 54      'ciceros
visPageUnits = 63      'use default page units
visDrawingUnits = 64      'use default drawing units
visInches = 65      'inches (decimal)
visFeet = 66      'feet
visFeetAndInches = 67      'feet and inches
visMiles = 68      'miles (decimal)
visCentimeters = 69      'centimeters
visMillimeters = 70      'millimeters
visMeters = 71      'meters
visKilometers = 72      'kilometers
visInchFrac = 73      'inches (fractional)
visMileFrac = 74      'miles (fractional)
visYards = 75      'yards
visAngleUnits = 80      'angle with no explicit units
visDegrees = 81      'angle in decimal degrees
visDegreeMinSec = 82      ' degrees, minutes, seconds
visRadians = 83      'angle in radians
visMin = 84      'angle in minutes-seconds
visSec = 85      'angle in seconds
visCurrency = 111      ' currency
```

visNoCast = 252 ' leave in present units

To specify internal units, pass a null string (""). Internal units are inches for distance and radians for angles. To specify implicit units, you must use the Formula property.

See also: Formula property, FormulaForce property, ResultForce property, ResultIU property, ResultIUForce property, ResultInt property, ResultFromInt property, ResultStr property

Example for Result

```
'This VB code excerpt demonstrates using the Result property.

Dim shpObj as Visio.Shape
Dim celObj as Visio.Cell

...

Set celObj = shpObj.Cells("LocPinX")
localCenterX = celObj.Result("inches")

'You can also use the constants defined by the Visio type library.
localCenterX = celObj.Result(visInches)
```

ResultForce property

Applies to: Cell

Summary: Sets a cell's value, even if the cell's formula is protected with the GUARD function.

Version: VISIO 2.0

Syntax: object.**ResultForce** (units) = newValue

Element	Description
object	The Cell object that contains the value to set
units	The units to use when setting the cell's value
newValue	The new value for the cell

Remarks: Use the ResultForce method to set a cell's value even if the cell's formula is protected with a GUARD function.

You can specify units as a valid string such as "inches". If the string is invalid, an error is generated. For a list of units, see the Result method.

To specify internal units, pass a null string (""). Internal units are inches for distance and radians for angles. To specify implicit units, you must use the Formula property.

See also: Formula property, Result property, ResultInt property, ResultFromInt property, ResultIU property, ResultIUForce property, ResultStr property

Example for ResultForce

ResultFromInt property

Applies to: Cell

Summary: Sets the value of a cell to an integer value.

Version: VISIO 4.5

Syntax: object.**ResultFromInt**(units) = newValue

Element	Description
object	The Cell object that receives the value
units	The units to use when setting the cell's value
newValue	The new value for the cell

Remarks: ResultFromInt is analogous to setting a cell's Result property. The difference is that ResultFromInt accepts an integer for the value of the cell, whereas Result accepts a floating point number.

Units can be a string such as "inches", "inch", "in.", or "i". If the string is invalid, an error is generated. Units can also be one of the integer constants declared by the Visio type library (and visconst.bas). For a list of valid unit strings or constants, see the Result property.

If the cell's formula is protected with a GUARD function, use ResultFromIntForce.

Note: Up through and including version 4.1 of Visio, ResultInt was both a read and write property and this ResultFromInt property did not exist. Due to changes in Automation support tools, it became no longer possible to support properties whose read and write variations took differing arguments. Therefore, ResultInt is now only a read property and this write only ResultFromInt property was added.

See also: Result property, ResultFromIntForce property, ResultInt property, ResultIU property, ResultStr property

Example for ResultFromInt

ResultFromIntForce property

Applies to: Cell

Summary: Sets the value of a cell to an integer value, even if the cell's formula is protected with the GUARD function.

Version: VISIO 4.0

Syntax: object.**ResultFromIntForce**(units) = newValue

Element	Description
object	The Cell object that gets the new value
units	The units to use when setting the cell's value
newValue	The new value for the cell

Remarks: Use the ResultFromIntForce property to set a cell's value even if the cell's formula is protected with a GUARD function. Otherwise it is identical in behavior to ResultFromInt.

Note: Up through and including version 4.1 of Visio, ResultInt was both a read and write property and this ResultFromIntForce property was called ResultIntForce. Due to changes in Automation support tools, it became no longer possible to support properties whose read and write variations took differing arguments. Therefore, ResultInt is now only a read property and a write only ResultFromInt property was added. This ResultIntForce property was renamed to ResultFromIntForce to be consistent.

See also: Result property, ResultInt property, ResultFromInt property, ResultStr property

Example for ResultFromIntForce

ResultInt property

Applies to: Cell

Summary: Gets the value of a cell expressed as an integer.

Version: VISIO 4.0

Syntax: intRet = object.**ResultInt**(units,roundFlag)

Element	Description
intRet	The cell's value returned as an integer
object	The Cell object that contains the value
units	The units to use when retrieving the cell's value
roundFlag	0 to truncate the value, non-zero to round it

Remarks: ResultInt is analogous to a cell's Result property. The difference is that ResultInt returns an integer for the value of the cell, whereas Result returns a floating point number.

Units can be a string such as "inches", "inch", "in.", or "i". If the string is invalid, an error is generated. Units can also be one of the integer constants declared by the Visio type library (and visconst.bas). For a list of valid unit strings or constants, see the Result property.

When getting a cell's result as an integer, you can indicate whether you want the returned value to be rounded or truncated. Use 0 to truncate the result or a non-zero value to round it.

Note: Up through and including version 4.1 of Visio, ResultInt was both a read and write property. Due to changes in Automation support tools, it became no longer possible to support properties whose read and write variations took differing arguments. Therefore, ResultInt is now only a read property and a new write only ResultFromInt property was added.

See also: Result property, ResultFromInt property, ResultFromIntForce property, ResultIU property, ResultStr property

Example for ResultInt

ResultIU property

Applies to: Cell

Summary: Returns or sets a cell's value in internal units.

Version: VISIO 2.0

Syntax: retVal = object.**ResultIU**
object.**ResultIU** = newValue

Element	Description
retVal	The cell's value in internal units
object	The Cell object that contains the value
newValue	The new value for the cell

Remarks: Use the ResultIU property to set the value of an unguarded cell. If the cell's formula is protected with a GUARD function, the formula is not changed and an error is generated.

The units default to Visio's internal units, which are inches for distance and radians for angles.

See also: Formula property, Result property, ResultForce property, ResultInt property, ResultIUForce property, ResultStr property

Example for ResultIU

ResultIUForce property

Applies to: Cell

Summary: Sets a cell's value in internal units, even if the cell's formula is protected with the GUARD function.

Version: VISIO 2.0

Syntax: object.**ResultIUForce** = newValue

Element	Description
object	The Cell object that gets the new value
newValue	The new value for the cell

Remarks: Use ResultIUForce to set a cell's value in internal units if the cell's formula is protected with the GUARD function. The cell's units default to Visio's internal units, which are inches for distance and radians for angles.

See also: Formula property, Result property, ResultInt property, ResultFromInt property, ResultForce property, ResultIU property, ResultStr property

Example for ResultIUForce

ResultStr property

Applies to: Cell

Summary: Gets the value of a cell expressed as a string.

Version: VISIO 4.0

Syntax: stringRet = object.**ResultStr**(units)

Element	Description
stringRet	The cell's value returned as a string
object	The Cell object that contains the value
units	The units to use when retrieving the value

Remarks: The ResultStr property is analogous to a cell's Result property. The difference is that ResultStr returns a string for the value of the cell, whereas Result returns a floating point number.

Units can be a string such as "inches", "inch", "in.", or "i". If the string is invalid, an error is generated. For a list of valid unit strings, see the Result property.

ResultStr is often useful for filling controls such as edit boxes with the value of a cell.

ResultStr is also a useful mechanism for converting between units. For example, you can get the value in inches, then get the same value in centimeters.

See also: Result property, ResultInt property

Example for ResultStr, RowCount

```
'This VBA macro demonstrates extracting a custom property and setting the
shape's
'text.

'To run this macro, first open a blank drawing and the VBA Samples Stencil,
'then insert a user form with a label, text box, and list box.

Public Sub ExtractCustomProperties()

  Dim stnObj As Visio.Document    ' Stencil that contains master
  Dim mastObj As Visio.Master     ' Master to drop
  Dim pagsObj As Visio.Pages      ' Pages collection of document
  Dim pagObj As Visio.Page        ' Page to work with
  Dim shpObj As Visio.Shape       ' Instance of master on page
  Dim celObj As Visio.Cell        ' Cell object for custom property
  Dim nRows As Integer            ' Number of rows in custom prop section
  Dim i As Integer                ' Loop variable

    ' Get the pages collection for the document
    ' Note the use of ThisDocument to refer to the current document
    Set pagsObj = ThisDocument.Pages

    ' Get a reference to the first page of the collection
    Set pagObj = pagsObj(1)

    ' Get the document object for the stencil
    Set stnObj = Documents("VBA Samples.vss")

    ' Retrieve the master object for the Desktop PC shape
    Set mastObj = stnObj.Masters("Desktop PC")

    ' Drop the shape in the approximate middle of the page
    ' Coordinates passed with Drop are always in inches
    ' The Drop method returns a reference to the new shape object
    Set shpObj = pagObj.Drop(mastObj, 4.25, 5.5)

    ' This example demonstrates 2 methods of extracting custom properties
    ' The first method retrieves the value of a custom property by name
    ' Note that Prop.Computer implies Prop.Computer.Value
    Set celObj = shpObj.Cells("Prop.Computer")

    ' Now that you have the cell, get the value as a string and put it
    ' into the textbox on the form
    UserForm1.TextBox1.Text = celObj.ResultStr(Visio.visNone)

    ' Set the caption of the label
    UserForm1.Label1.Caption = "Prop.Computer"

    ' The second method of accessing custom properties uses
    ' section, row, cell. This method is best when you want to iterate
    ' through all the properties
    nRows = shpObj.RowCount(Visio.visSectionProp)

    ' Make sure the list box is cleared
```

```
    UserForm1.ListBox1.Clear

    ' Loop through all the rows and add the value of Prop.Computer
    ' to the list box. Remember, Rows are numbered starting with 0.
    For i = 0 To nRows - 1
        Set celObj = shpObj.CellsSRC(Visio.visSectionProp, i,
visCustPropsValue)
        UserForm1.ListBox1.AddItem celObj.LocalName & vbTab & _
        celObj.ResultStr(Visio.visNone)
    Next i

    ' Display the user form
    UserForm1.Show

End Sub
```

ReverseEnds method

Applies to: Selection, Shape

Summary: Reverses an object by flipping it both horizontally and vertically.

Version: VISIO 2.0

Syntax: object.**ReverseEnds**

Element	Description
object	The Shape or Selection object to reverse

See also: FlipHorizontal method, FlipVertical method, Rotate90 method

Example for ReverseEnds

RightMargin property

Applies to: Document

Summary: Specifies the right margin for printing a document's pages.

Version: VISIO 4.0

Syntax: retVal = object.**RightMargin**(units)
object.**RightMargin**(units) = newValue

Element	Description
retVal	The margin value expressed in the given units
object	The Document object that has or gets the margin value
units	The units to use when retrieving or setting the margin value
newValue	The new margin value

Remarks: This property corresponds to the Right Margin control in Visio's Page Setup dialog box.

Units can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property.

See also: LeftMargin property, TopMargin property, BottomMargin property, Result property

Example for RightMargin

Rotate90 method

Applies to: Selection, Shape

Summary: Rotates an object 90 degrees counterclockwise.

Version: VISIO 2.0

Syntax: object.**Rotate90**

Element	Description
object	The Shape or Selection object to rotate

See also: FlipHorizontal method, FlipVertical method, ReverseEnds method

Example for Rotate90

Row property

Applies to: Cell, Layer

Summary: Returns the row index of a cell or layer.

Version: VISIO 4.0

Syntax: intRet = object.**Row**

Element	Description
intRet	The index of the row that defines the cell or layer
object	The Cell or Layer object to examine

See also: Cell object, CellsSRC property, Column property, Layer object, Section property

Example for Row

RowCount property

Applies to: Shape

Summary: Returns the number of rows in a ShapeSheet section.

Version: VISIO 2.0

Syntax: retVal = object.**RowCount** (section)

Element	Description
retVal	The number of rows in the section
object	The Shape object to examine
section	The section to count

Remarks: The section argument must be a section constant. For a list of section constants, see the AddSection method.

RowCount is intended primarily to be used with sections that contain a variable number of rows, such as geometry and control point sections. The value returned by RowCount for sections that have a fixed number of rows, such as the object properties section, is the number of rows in the section that possess at least one cell whose value is local to the shape. This is opposed to rows whose cells are all inherited from a master or style. In a Visio ShapeSheet window, cells with local values are shown in blue. Cells with inherited values are shown in black. Black is typically better than blue in that for most cases less information is stored. Using Automation, you can determine if a cell is inherited using Cell.IsInherited.

See also: AddRow method, AddSection method, GeometryCount property, RowsCellCount property, IsInherited property

Example for RowCount

*ResultStr Property

RowExists property

Applies to: Shape

Summary: Returns TRUE if the indicated ShapeSheet row exists.

Version: VISIO 4.0

Syntax: intRet = object.**RowExists**(section,row,fExistsLocally)

Element	Description
intRet	FALSE (0) if row doesn't exist; otherwise TRUE (-1)
object	The Shape object to examine
section	The row's section index
row	The row's row index
fExistsLocally	The scope of the search

Remarks: If fExistsLocally is FALSE (0), RowExists returns TRUE if the object either contains or inherits the specified row. If fExistsLocally is TRUE (non-zero), RowExists returns TRUE only if the object contains the specified row locally; if the row is inherited, RowExists returns FALSE.

See also: Cells property, CellExists property, CellsSRC property, CellsSRCExists property, SectionExists property

Example for RowExists

RowName property

Applies to: Cell

Summary: Gets or sets the name of the row that contains the cell.

Version: VISIO 4.0

Syntax: strRet = object.**RowName**
object.**RowName** = stringExpression

Element	Description
strRet	The current name of the row
object	The Cell object that has or gets the row name
stringExpression	The new name to assign to the row

Remarks: If the cell is from a row in a shape's User-defined Cells, Custom Properties, or Connections sections, the RowName property returns the name of the row and can be used to set the row name.

If the cell is not in one of these two sections, RowName returns a null string (""), and attempting to set it generates an error.

See also: AddNamedRow method, Cells property

Example for RowName

RowsCellCount property

Applies to: Shape

Summary: Returns the number of cells in a row of a ShapeSheet section.

Version: VISIO 2.0

Syntax: intRet = object.**RowsCellCount** (section, row)

Element	Description
intRet	The number of cells in the row
object	The Shape object to examine
section	The index of the section that contains the row
row	The index of the row to count

Remarks: Use section and row index constants declared by the Visio type library (and visconst.bas). For a list of constants, see the AddRow and AddSection methods.

See also: AddRow method, AddSection method, RowCount property

Example for RowsCellCount

RowType property

Applies to: Shape

Summary: Returns or sets the type of a row in a Geometry section of a ShapeSheet.

Version: VISIO 2.0

Syntax: retVal = object.**RowType** (section,row)
object.**RowType** (section, row) = rowTag

Element	Description
retVal	The current type of the row
object	The Shape object that owns the row
section	The index of the section that contains the row
row	The index of the row
rowTag	The new type for the row

Remarks: To change the type of a row in a Geometry section, use the RowType property. For example, you can change a LineTo row to an ArcTo row. If an inappropriate row tag is passed or the row does not exist, no changes occur.

After its row type has been changed, a row may have a different number of cells or the cells may have different meanings. A program must provide meaningful formulas for the new or changed cells.

The RowType property is read-only for tab rows.

See the AddRow and AddSection methods for lists of valid row and section constants.

See also: AddRow method, Formula property

Example for RowType

```
*AddSection Method
```

Run method

Applies to: Addon

Summary: Runs the add-on represented by an Addon object.

Version: VISIO 4.0

Syntax: object.**Run** argString

Element	Description
object	The Addon object that represents the add-on to be run
argString	The argument string to pass to the add-on

Remarks: If the add-on is implemented by an .EXE file, the arguments are passed in the command line string. If the add-on is implemented by a .VSL file, the arguments are passed in a field of the argument structure that accompanies the run message sent to the .VSL's VisioLibMain procedure.

Example for Run

RunBegin property

Applies to: Characters

Summary: Returns the beginning index of a run of the specified type that is at or before the beginning index of a Characters object.

Version: VISIO 3.0

Syntax: intRet = object.**RunBegin**(runType)

Element	Description
intRet	The beginning index of the run
object	The Characters object to examine
runType	The type of run to get

Remarks: A "run" is a sequence of characters that share a particular attribute: character format, paragraph format, tab format, a word, a paragraph, or a field. For example, certain words may be bold or italic, or one paragraph may be centered and another left-aligned. Each change of format represents a run of that format. Similarly, delimiters such as spaces and paragraph marks represent the beginnings and endings of words, paragraphs, and fields.

In a ShapeSheet, each row in the Character and Paragraph sections represents a run of the corresponding format in the text of a shape. In addition, rows that represent runs of character, paragraph, and tab formats can be retrieved by specifying a row index as an argument to the CellsSRC property of a shape.

Use the RunBegin property to determine the beginning of a sequence of identically formatted characters or the beginning of a word, paragraph, or field. If the Begin property of the Characters object is already at the start of a run, the value of RunBegin is identical to the value of Begin.

Use the runType argument to specify the type of run you want:

visCharPropRow = 1
visParaPropRow = 2
visTabPropRow = 3
visWordRun = 10
visParaRun = 11
visFieldRun = 20

Use visWordRun and visParaRun to mimic double-clicking and triple-clicking to select text.

Use visFieldRun to specify a run of field or non-field text. Check the IsField property to determine whether the run is a field.

See also: RunEnd property

Example for RunBegin

RunEnd property

Applies to: Characters

Summary: Returns the ending index of a run of the specified type that is at or after the ending index of a Characters object.

Version: VISIO 3.0

Syntax: intRet = object.**RunEnd**(runType)

Element	Description
intRet	The ending index of the run
object	The Characters object to examine
runType	The type of run to get

Remarks: A "run" is a sequence of characters that share a particular attribute. For example, certain words may be bold or italic, or one paragraph may be centered and another left-aligned. Each change of format represents a run of that format. Similarly, delimiters such as spaces and paragraph marks represent the beginnings and endings of words, paragraphs, and fields.

In a ShapeSheet, each row in the Character and Paragraph sections represents a run of the corresponding format in the text of a shape. In addition, rows that represent runs of character, paragraph, and tab formats can be retrieved by specifying a row index as an argument to the CellsSRC property of a shape.

Use the RunEnd property to determine the end of a sequence of identically formatted characters or the end of a word, paragraph, or field. If the End property of the Characters object is already at the end of a run, the value of RunEnd is identical to the value of End.

Use the runType argument to specify the type of run you want:

visCharPropRow = 1
visParaPropRow = 2
visTabPropRow = 3
visWordRun = 10
visParaRun = 11
visFieldRun = 20

Use visWordRun and visParaRun to mimic double-clicking and triple-clicking to select text.

Use visFieldRun to specify a run of field or non-field text. Check the IsField property to determine whether the run is a field.

See also: RunBegin property

Example for RunEnd

RunModeEntered event

Applies to: Application, Document, Documents

Summary: The event that occurs after a Visio document enters run mode.

Version: VISIO 5.0

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeDocRunning (5)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Document that just entered run mode
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, EventsEnabled property, DesignModeEntered event, Mode property

Example for RunModeEntered

Save method

Applies to: Document

Summary: Saves a document.

Version: VISIO 2.0

Syntax: object.**Save**

Element	Description
object	The Document object to save

Remarks: Until a document has been saved, the Save method generates an error. Use the SaveAs method to save and name a new document.

See also: SaveAs method, SaveAsEx method

Example for Save

SaveAs method

Applies to: Document

Summary: Saves a document under the specified filename.

Version: VISIO 2.0

Syntax: object.**SaveAs** stringExpression

Element	Description
object	The Document object to save
stringExpression	The filename under which to save the document

Remarks: The specified file can accept UNC drive names (for example, \\bob\leo).

See also: Save method, SaveAsEx method

Example for OpenEx, SaveAs, SaveAsEx, Saved, SaveWorkspaceAs

```
'This VBA macro demonstrates various ways of opening and saving files in
Visio.

Public Sub Save_Example()

  Dim docObj1 As Visio.Document
  Dim docObj2 As Visio.Document
  Dim pageObj As Visio.Page
  Dim shpObj As Visio.Shape

  Set pageObj = ThisDocument.Pages(1)
  Set shpObj = pageObj.DrawOval(2.5, 7, 3.5, 9)

  'Use the SaveAs method to save the document for the first time.
  ThisDocument.SaveAs "c:\My Drawing1.vsd"

  'Use the Saved property to verify the save. Result = True (-1)
  Debug.Print ThisDocument.Saved

  'Force a change to the document by adding a shape.
  Set shpObj = pageObj.DrawOval(4, 7, 5, 9)

  'Use the Saved property to verify that the document changed since the last
  'time is was saved.  Result = False (0)
  Debug.Print ThisDocument.Saved

  'Use the Save method to save any new changes.
  ThisDocument.Save
  Debug.Print ThisDocument.Saved          'Result = True (-1)

  'The Saved property can also be set.
  'For example, change the document so the Saved property becomes false.
  Set shpObj = pageObj.DrawRectangle(1, 1, 7, 7)

  'Set the Saved property to True.
  'Setting the Saved property to True *does not* save the document.
  ThisDocument.Saved = True

  'Close the document and reopen it.  Notice, the rectangle was not saved.
  Set docObj1 = ThisDocument
  docObj1.Close
  Set docObj1 = Documents.Open("c:\My Drawing1.VSD")

  'Add a new shape to the drawing.
  Set shpObj = pageObj.DrawOval(3.5, 6.5, 4, 7)

  'Open a stencil read-only using the OpenEx method.
  Documents.OpenEx "basic.vss", visOpenDocked

  'Use the SaveAsEx method to save the drawing as a new read-only drawing.
  'In this example, SaveAsEx also uses the flag to save the workspace.
  docObj1.SaveAsEx "c:\My Drawing2.VSD", visSaveAsRO + visSaveAsWS
```

```
  'Re-open the 1st drawing
  'The previous save caused it to automatically close.
  Set docObj2 = Documents.Open("c:\My Drawing1.vsd")
  Windows.Arrange
  Visio.Application.SaveWorkspaceAs ("c:\My Workspace.vsw")

End Sub
```

SaveAsEx method

Applies to: Document

Summary: Saves a document under a specified filename using extra information passed in an argument.

Version: VISIO 4.0

Syntax: object.**SaveAsEx** fileName, saveFlags

Element	Description
object	The Document object to save
fileName	The filename under which to save the document
saveFlags	How to save the file

Remarks: SaveAsEx is identical to SaveAs, except that it provides an extra argument in which the caller can specify how the document is to be saved. SaveFlags should be a combination of zero or more of the following:

visSaveAsRO = 1
visSaveAsWS = 2

If visSaveAsRO is specified, the document is saved as read-only.

If visSaveAsWS is specified, the current workspace is saved with the file so that the next time the document is opened, that workspace is restored.

See also: Save method, SaveAs method

Example for SaveAsEx

```
*SaveAs Method
```

Saved property

Applies to: Document

Summary: Determines whether a document has any unsaved changes.

Version: VISIO 2.0

Syntax: intRet = object.**Saved**
object.**Saved** = intExpression

Element	Description
intRet	TRUE (-1) if the document has no unsaved changes; otherwise False (0)
object	The Document object that has or gets the setting
intExpression	True (non-zero) to indicate the document is saved; FALSE (0) to indicate unsaved changes

Remarks: Setting the Saved property for a document to TRUE should be done with caution. Data loss could occur when Saved is set to TRUE and a user or another program makes changes to the document before it is closed. If the document has unsaved changes and a user or a program closes the document before more changes occur, there is no prompt to save the unsaved changes before the document closes. A document that contains embedded or linked OLE objects may report itself as unsaved even if the document's Saved property is set to TRUE.

Example for Saved

```
*SaveAs Method
```

SavePreviewMode property

Applies to: Document

Summary: Determines whether Visio saves a preview of a document when the document is saved.

Version: VISIO 4.0

Syntax: intRet = object.**SavePreviewMode**
object.**SavePreviewMode** = intExpression

Element	Description
intRet	FALSE (0) if the property is not set; otherwise TRUE (-1)
object	The Document object that has or gets the setting
intExpression	0 to turn the property off, or non-zero to turn it on

Example for SavePreviewMode

SaveToFile method

Applies to: UI Object

Summary: Saves the user interface represented by a UI object in a file.

Version: VISIO 4.0

Syntax: object.**SaveToFile** stringExpression

Element	Description
object	The UI object to save to the file
stringExpression	The name of the file in which to save the UI object

Remarks: The file can be loaded back into Visio by using the LoadFromFile method of a UI object.

See also: LoadFromFile method

Example for Application, LoadFromFile, SaveToFile

```
'This VBA macro demonstrates saving and loading a custom user interface
'file (.VSU). It does not manipulate any menus or menu items.


Public Sub SaveMenusToFile_Example ()

    Dim UIObj As Visio.UIObject
    Dim strPath As String

    'Get Menus object from Visio
    Set UIObj = Visio.Application.BuiltInMenus

    'Save Menus object to a file
    strPath = "C:\Temp\Menus.vsu"
    UIObj.SaveToFile (strPath)
    MsgBox ("Menus successfully saved to " & strPath)

    'Load Menus from file
    UIObj.LoadFromFile (strPath)
    Visio.Application.SetCustomMenus UIObj
    MsgBox ("Menus successfully loaded from " & strPath)

End Sub
```

SaveWorkspaceAs method

Applies to: Application

Summary: Saves the current workspace.

Version: VISIO 4.0

Syntax: object.**SaveWorkspaceAs** fileName

Element	Description
object	The Application object that owns the workspace to save
fileName	The name of the file in which to save the workspace

Remarks: The SaveAsWorkspace method performs the same action as the Save Workspace command located on Visio's File menu. The specified filename should have a .VSW extension.

Example for SaveWorkspaceAs

```
*SaveAs Method
```

ScreenUpdating property

Applies to: Application

Summary: Determines whether or not the Visio screen is updated (redrawn) during a series of actions.

Version: VISIO 3.0

Syntax: intRet = object.**ScreenUpdating**
object.**ScreenUpdating** = intExpression

Element	Description
intRet	0 if screen updating is off; -1 if screen updating is on
object	The Application object that has or gets the setting
intExpression	0 to turn screen updating off; non-zero to turn screen updating on

Remarks: The ScreenUpdating property is a read-write property. Use this property to increase performance during a series of actions. For example, you can turn off screen updating while a series of shapes are created so the screen is not redrawn after each shape appears, and then turn screen updating on to update the screen.

If you send a large number of commands to Visio while screen updating is turned off, Visio may redisplay the screen occasionally in order to flush its buffers.

If a program neglects to turn screen updating on after turning it off, Visio turns screen updating back on when a user performs an operation.

See also: DeferRecalc property

Example for ScreenUpdating

```
'This VB code excerpt demonstrates using the ScreenUpdating property.

  Visio.Application.ScreenUpdating = False
  ...
  'Drop the shapes
  'Set the shapes' text
  'Connect the shapes
  'Format the connectors
  ...
  Visio.Application.ScreenUpdating = True
```

Section property

Applies to: Cell

Summary: Returns the section index of a cell.

Version: VISIO 4.0

Syntax: intRet = object.**Section**

Element	Description
intRet	Returns the section index of a cell
object	The Cell object to examine

See also: Cell object, CellsSRC property, Column property, Row property

Example for Section

SectionExists property

Applies to: Shape

Summary: Returns TRUE if the indicated ShapeSheet section exists.

Version: VISIO 4.0

Syntax: intRet = object.**SectionExists**(section,fExistsLocally)

Element	Description
intRet	FALSE (0) if section doesn't exist; otherwise TRUE (-1)
object	The Shape object to examine
section	The section index
fExistsLocally	The scope of the search

Remarks: If fExistsLocally is FALSE (0), SectionExists returns TRUE if the object either contains or inherits the section. If fExistsLocally is TRUE (non-zero), SectionExists returns TRUE only if the object contains the section locally; if the section is inherited, SectionExists returns FALSE.

See also: Cells property, CellExists property, CellsSRC property, CellsSRCExists property, RowExists property

Example for SectionExists

Select method

Applies to: Selection, Window

Summary: Selects or deselects an object.

Version: VISIO 2.0

Syntax: object.**Select** addObj, selectType

Element	Description
object	The Window or Selection object that contains the shapes
addObj	A Shape object to select or deselect
selectType	The type of selection to make

Remarks: The following constants declared by the Visio type library (and visconst.bas) show valid values for selection types:

visDeselect = 1
visSelect = 2
visSubSelect = 3
visSelectAll = 4
visDeselectAll = 256

You can combine visDeselectAll with visSelect and visSubSelect to deselect all shapes prior to selecting or subselecting other shapes.

If the object being operated on is a Selection, and if Select is told to select a Shape whose ContainingShape is different than the ContainingShape of the selection, then Select will deselect everything presently selected, even if selectType doesn't say to deselect.

See also: DeselectAll method, SelectAll method, Selection object, Selection property, ContainingShape property

Example for Select

```
'This VBA macro demonstrates selecting, deselecting, and subselecting shapes.

Public Sub Select_Example ()

  Const MAX_SHAPES = 6
  ReDim shpObjs(1 To MAX_SHAPES) As Visio.Shape

  Dim I As Integer
  Dim shpObj As Visio.Shape

  'Draw some shapes.
  For I = 1 To MAX_SHAPES
    Set shpObjs(I) = ActivePage.DrawRectangle(I, I + 1, I + 1, I)
  Next I

  'Set up a group for testing subselections by selecting the first
  'three shapes on the page, grouping them, and storing the group as shpObj.
  'Although the first three shapes are grouped, the array shpsObj() still
  'contains them.

  ActiveWindow.DeselectAll

  For I = 1 To 3
    ActiveWindow.Select shpObjs(I), visSelect
  Next I

  ActiveWindow.Group

  Set shpObj = ActivePage.Shapes(ActivePage.Shapes.Count)

  'You have (MAX_SHAPES - 3) shapes on the page with three shapes in the
group.
  'Subselection is accomplished by selecting the parent shape first or one of
  'the groups shapes already subselected.

  ActiveWindow.Select shpObj, visDeselectAll + visSelect   'Select parent
  ActiveWindow.Select shpObjs(1), visSubSelect
  ActiveWindow.Select shpObjs(3), visSubSelect

  'Next, select just one shape.  At this point two shapes are
  'subselected but you want to start a new selection with the last two
  'shapes on the page and the group.  Note that the subselections that were
  'made in the group are cancelled by selecting another shape that is
  'at the same level as their parents.

  ActiveWindow.Select shpObjs(MAX_SHAPES), visDeselectAll + visSelect
  ActiveWindow.Select shpObjs(MAX_SHAPES - 1), visSelect
  ActiveWindow.Select shpObj, visSelect

  'Remove only one shape from the window selection.
  ActiveWindow.SelectAll
  ActiveWindow.Select shpObjs(MAX_SHAPES - 1), visDeselect

  'Close the document.
```

```
  ActivePage.Document.Close

End Sub
```

SelectAll method

Applies to: Selection, Window

Summary: Selects all possible shapes in a window or selection.

Version: VISIO 2.0

Syntax: object.**SelectAll**

Element	Description
object	The Window or Selection object that contains the shapes

Remarks: In the case of a selection, all shapes that can be selected are all immediate children of the selection's ContainingShape.

See also: ContainingShape property, DeselectAll method, Select method, Selection object, Selection property

Example for SelectAll

```
*Add Method
```

Selection property

Applies to: Window

Summary: Returns a Selection object.

Version: VISIO 2.0

Syntax: objRet = object.**Selection**

Element	Description
objRet	A Selection object
object	The object that owns the selection

Remarks: A Selection object is a set of shapes in a common context that can be operated on. A Selection object is analogous to shapes that display selection handles when you shift+click them with the pointer tool in a drawing window. Once a Selection object retrieved, the set of shapes it represents can be changed by using the Select method.

Window.Selection returns a Selection object that represents what is presently selected in the window. The Selection object is independent of the selection in the window, which can subsequently change as a result of user actions.

See also: DeselectAll method, Select method, SelectAll method, Selection object

Example for Selection

```
*Add Method
```

SelectionAdded event

Applies to: Application, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs after shape(s) are added to a Visio document.

Version: VISIO 4.5

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeSelAdded (902)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Selection whose entries were just added
moreInfo	Nothing for this event

Remarks: A Shape object can serve as the source object for SelectionAdded iff the shape's Type property is visTypeGroup (2) or visTypePage(1).

The SelectionAdded and ShapeAdded events are similar in that they both fire after shape(s) are created. They differ in how they behave when a single operation creates several shapes. Suppose a paste operation produces 3 new shapes. ShapeAdded will fire 3 times and the respective subject objects will be the 3 added shapes. SelectionAdded will fire once and its subject object will be a Selection object in which the 3 new shapes are selected.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source SelectionAdded using the AddAdvise method. In addition, SelectionAdded is included in the event set of all the objects in the Applies to list except the Document object. For those objects you can use VBA Dim WithEvents variables to sink SelectionAdded. For performance considerations, the Document object's event set does not include SelectionAdded. To sink SelectionAdded from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, Selection object, ShapeAdded event

Example for SelectionAdded

SelectionChanged event

Applies to: Application, Window, Windows

Summary: The event that occurs after the set of shapes selected in a window changes.

Version: VISIO 4.5

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeWinSelChange (701)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Window whose selection has changed
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for SelectionChanged

SendBackward method

Applies to: Selection, Shape

Summary: Moves a shape or selected shapes back one position in the z-order.

Version: VISIO 2.0

Syntax: object.**SendBackward**

Element	Description
object	The Shape or Selection object to send backward

See also: BringForward method, BringToFront method, SendToBack method

Example for BringForward, BringToFront, SendBackward, SendToBack

```
'This VBA macro demonstrates positioning shapes in the z-order on a page.

Public Sub ZOrder_Example ()

  Dim shapeObj1 As Visio.Shape, shapeObj2 As Visio.Shape, shapeObj3 As
Visio.Shape


  'Draw three rectangles.
  Set shapeObj1 = ActivePage.DrawRectangle(1, 1, 5, 5)
  shapeObj1.Text = "1"
  Set shapeObj2 = ActivePage.DrawRectangle(2, 2, 6, 6)
  shapeObj2.Text = "2"
  Set shapeObj3 = ActivePage.DrawRectangle(3, 3, 7, 7)
  shapeObj3.Text = "3"

  'Move shapeObj3 to the back of the z-order.
  shapeObj3.SendToBack

  'Move shapeObj2 back one position in the z-order.
  shapeObj2.SendBackward

  'Bring shapeObj2 forward one position in the z-order.
  shapeObj2.BringForward

  'Bring shapeObj3 to the front of the z-order.
  shapeObj3.BringToFront

End Sub
```

SendToBack method

Applies to: Selection, Shape

Summary: Moves the shape or selected shapes to the back of the z-order.

Version: VISIO 2.0

Syntax: object.**SendToBack**

Element	Description
object	The Shape or Selection object to send to back

See also: BringForward method, BringToFront method, SendBackward method

Example for SendToBack

```
*SendBackward Method
```

SetBegin method

Applies to: Shape

Summary: Moves the begin point of a 1-D shape to the coordinates represented by x and y.

Version: VISIO 2.0

Syntax: object.**SetBegin** x, y

Element	Description
object	The Shape object to set
x	The new x-coordinate of the begin point
y	The new y-coordinate of the begin point

Remarks: The SetBegin method applies to only 1-D shapes. If the indicated shape is a 2-D shape, an error is generated.

The coordinates represented by the x and y arguments are parent coordinates, measured from the origin of the shape's parent (the page or group that contains the shape).

See also: SetCenter method, SetEnd method

Example for SetBegin

SetCenter method

Applies to: Shape

Summary: Moves a shape so that its pin is positioned at the coordinates represented by x and y.

Version: VISIO 2.0

Syntax: object.**SetCenter** x, y

Element	Description
object	The Shape object to set
x	The new x-coordinate of the center of rotation
y	The new y-coordinate of the center of rotation

Remarks: This sets the shape's pin position. The pin is often, but not necessarily, equal to the shape's center of rotation.

The coordinates represented by the x and y arguments are parent coordinates, measured from the origin of the shape's parent (the page or group that contains the shape).

See also: SetBegin method, SetEnd method

Example for SetCenter

SetCustomMenus method

Applies to: Application, Document

Summary: Replaces the current built-in or custom menus of an Application or Document object.

Version: VISIO 4.0

Syntax: object.**SetCustomMenus** UIObject

Element	Description
object	The Application or Document object to receive the custom menus
UIObject	A UI object that represents the new custom menus

Remarks: If the UI object was created in a separate process by using CreateObject instead of getting the appropriate property of an Application or Document object, SetCustomMenus returns an error.

See also: CustomMenus property, SetCustomToolbars method, UI Object object

Example for SetCustomMenus

```
*ActionText Property
```

SetCustomToolbars method

Applies to: Application, Document

Summary: Replaces the current built-in or custom toolbars of an Application or Document object.

Version: VISIO 4.0

Syntax: object.**SetCustomToolbars** UIObject

Element	Description
object	The Application or Document object to receive the custom toolbars
UIObject	A UI object that represents the new custom toolbars

Remarks: If the UI object was created in a separate process by using CreateObject instead of getting the appropriate property of an Application or Document object, SetCustomToolbars returns an error.

See also: CustomToolbars property, SetCustomMenus method, UI Object object

Example for SetCustomToolbars

```
*BuiltInToolbars Property
```

SetEnd method

Applies to: Shape

Summary: Moves the end point of a 1-D shape to the coordinates represented by x and y.

Version: VISIO 2.0

Syntax: object.**SetEnd** x, y

Element	Description
object	The Shape object to set
x	The new x-coordinate of the end point
y	The new y-coordinate of the end point

Remarks: The SetEnd method applies only to 1-D shapes. If the indicated shape is a 2-D shape, an error is generated.

The coordinates represented by the x and y arguments are parent coordinates, measured from the origin of the shape's parent (the page or group that contains the shape).

See also: SetBegin method, SetCenter method

Example for SetEnd

SetFormulas method

Applies to: Master, Page, Shape, Style

Summary: Sets the forumlas of many cells.

Version: VISIO 4.5

Syntax: intRet PageOrMasterObj.**SetFormulas** SID_SRCStream, formulas, flags
intRet ShapeOrStyleObj.**SetFormulas** SRCStream, formulas, flags

Element	Description
intRet	Number of SRCStream entries which processed successfully
PageOrMasterObj	The page or master object whose cells are to be modified
ShapeOrStyleObj	The shape or style object whose cells are to be modified
SID_SRCStream	Stream identifying cells to be modified
SRCStream	Stream identifying cells to be modified
formulas	Formulas to be assigned to identified cells
flags	Flags that influence the behavior of SetFormulas

Remarks: SetFormulas is like Cell.Formula, except that it can be used to set the formulas of many cells at once, rather than one cell at a time.

Shape.SetFormulas can be used to set formulas of any set of cells of Shape.
Style.SetFormulas can be used to set formulas of any set of cells of Style.

In both of these cases, you tell SetFormulas which cells you want to set by passing an array of integers in SRCStream. SRCStream should be a one-dimensional array of 3*n two-byte integers for n>=1. SetFormulas interprets the stream as:

{ sectionIdx, rowIdx, cellIdx }n

where sectionIdx is the section index of the desired cell, rowIdx is its row index and cellIdx is its cell index.

Page and Master.SetFormulas are more general in that they can be used to set formulas of any set of cells in any set of shapes of the page or master. SID_SRCStream should be a one-dimensional array of 4*n two-byte integers for n>=1. SetFormulas interprets the stream as:

{ sheetID, sectionIdx, rowIdx, cellIdx }n

where sheetID is the ID property of the Shape on the page or master whose cell formula is to be modified.

Note: If the sheetID in an entry is visInvalShapeID (-1) or if the bottom byte of sectionIdx is visSectionInval (255), then the entry will be ignored by SetFormulas. The motivation for this is that the same [SID_]SRCStream array can be used on several calls to SetFormulas, GetFormulas and the like with the caller only needing to make minor changes to the stream between calls.

Formulas should be a one-dimensional array of 1<=m variants. Each variant should be a string, a reference to a string or empty. If formulas(i) is empty, then the i'th cell will be set to the formula in formulas(j), where j is the index of the most recent prior entry which is not empty. If there is no prior non-empty entry, the corresponding cell is not altered. If fewer formulas than cells are specified (m<n), then the i'th cell, i>m, will be set to the same formula as was chosen to set the m'th cell to. Thus to set many cells to the same

formula, you need only pass one copy of the formula.

flags should be a mask of the following:

```
visSetBlastGuards (2) ' Override present cell values even if they're guarded
visSetTestCircular (4) ' Test for establishment of circular cell references
```

The value returned by SetFormulas is the number of entries in SRCStream that were successfully processed. If i<n entries process correctly, but an error occurs on the i+1st entry, then SetFormulas will raise an exception and return i. Otherwise n will be returned.

See also: Formula property, ID property, Section property, Row property, Cells property, SetResults method, GetFormulas method

Example for SetFormulas

```
*GetFormulas Method
```

SetID property

Applies to: AccelTable, MenuSet, StatusBar, ToolbarSet

Summary: Returns the set ID of the object in its collection.

Version: VISIO 4.0

Syntax: intRet = object.**SetID**

Element	Description
intRet	The set ID of the object
object	The object to examine

Remarks: The set ID of an object can be set by using the AddAtID method. These IDs correspond to Visio window and context (shortcut) menu sets. Not all IDs need to be present in a given collection. Valid ID values declared by the Visio type library (and visconst.bas) are as follows.

visUIObjSetNoDocument = 1
visUIObjSetDrawing = 2
visUIObjSetStencil = 3
visUIObjSetShapeSheet = 4
visUIObjSetIcon = 5
visUIObjSetInPlace = 6
visUIObjSetPrintPreview = 7
visUIObjSetText = 8
visUIObjSetCntx_DrawObjSel = 9
visUIObjSetCntx_DrawOleObjSel = 10
visUIObjSetCntx_DrawNoObjSel = 11
visUIObjSetCntx_InPlaceNoObj = 12
visUIObjSetCntx_TextEdit = 13
visUIObjSetCntx_StencilRO = 14
visUIObjSetCntx_ShapeSheet = 15
visUIObjSetCntx_Toolbar = 16
visUIObjSetCntx_Icon = 17
visUIObjSetBinderInPlace = 18
visUIObjSetCntx_Debug = 19
visUIObjSetCntx_StencilRW = 20
visUIObjSetCntx_StencilDocked = 21

See also: AddAtID method

Example for SetID

SetResults method

Applies to: Master, Page, Shape, Style

Summary: Sets the results or formulas of many cells.

Version: VISIO 4.5

Syntax: intRet PageOrMasterObj.**SetResults** SID_SRCStream, units, results, flags
intRet ShapeOrStyleObj.**SetResults** SRCStream, units, results, flags

Element	Description
intRet	Number of SRCStream entries which processed successfully
ShapeOrStyleObj	The shape or style object whose cells are to be modified
PageOrMasterObj	The page or master object whose cells are to be modified
SID_SRCStream	Stream identifying cells to be modified
SRCStream	Stream identifying cells to be modified
units	Measurement units to be attributed to entries in results array
results	Results or formulas to be assigned to identified cells
flags	Flags that influence the behavior of SetResults

Remarks: SetResults is like Cell.Result, except that it can be used to set the results (values) of many cells at once, rather than one cell at a time.

Shape.SetResults can be used to set results of any set of cells of Shape.
Style.SetResults can be used to set results of any set of cells of Style.

In both of these cases, you tell SetResults which cells you want to set by passing an array of integers in SRCStream. SRCStream should be a one-dimensional array of 3*n two-byte integers for n>=1. SetResults interprets the stream as:

{ sectionIdx, rowIdx, cellIdx }n

where sectionIdx is the section index of the desired cell, rowIdx is its row index and cellIdx is its cell index.

Page and Master.SetResults are more general in that they can be used to set results of any set of cells in any set of shapes of the page or master. SID_SRCStream should be a one-dimensional array of 4*n two-byte integers for n>=1. SetResults interprets the stream as:

{ sheetID, sectionIdx, rowIdx, cellIdx }n

where sheetID is the ID property of the Shape on the page or master whose cell result is to be modified.

Note: If the sheetID in an entry is visInvalShapeID (-1) or if the bottom byte of sectionIdx is visSectionInval (255), then the entry will be ignored by SetResults. The motivation for this is that the same [SID_]SRCStream array can be used on several calls to SetResults, GetResults and the like with the caller only needing to make minor changes to the stream between calls.

Units is an array that controls what measurement units individual entries in results are deemed to be in. Each entry in the array can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also indicate desired units with integer constants (visCentimeters, visInches, etc.) declared by the Visio type library (and visconst.bas). See also remarks for

cell.result. Note that the values specified in the units array have no effect if visSetFormulas is set in flags.

If not null, we expect units to be a one-dimensional array of 1<=u variants. Each entry can be a string or integer code, or empty (nothing). If the i'th entry is empty, then the i'th entry in results will be deemed to be in the units designated by units(j), where j is the most recent prior non-empty entry. Thus if you want all entries in result to be interpreted to be in the same units, then you need only pass a units array with one entry. If there is no prior non-empty entry, or if no units array is supplied, then visNumber (0x20) will be used. This causes Visio to default to internal units (like the Cell.ResultIU property).

Results should be a one-dimensional array of 1<=m variants. A result can be passed as double, integer, string, or reference to a string. If visSetFormulas is set in flags, strings are interpreted as formulas, otherwise strings are interpreted as string constants. If results(i) is empty, then the i'th cell will be set to the value in results(j), where j is the index of the most recent prior entry which is not empty. If there is no prior non-empty entry, the corresponding cell is not altered. If fewer results than cells are specified (m<n), then the i'th cell, i>m, will be set to the same value as was chosen to set the m'th cell to. Thus to set many cells to the same value, you need only pass one copy of the value.

Flags should be a mask of the following:

```
visSetFormulas (1)        ' Treat strings in results as formulas
visSetBlastGuards (2) ' Override present cell values even if they're guarded
visSetTestCircular (4) ' Test for establishment of circular cell references
```

The value returned by SetResults is the number of entries in SRCStream that were successfully processed. If i<n entries process correctly, but an error occurs on the i+1st entry, then SetResults will raise an exception and return i. Otherwise n will be returned.

See also: Result property, ResultIU property, ID property, Section property, Row property, Cells property, SetFormulas method, GetResults method

Example for SetResults

```
*GetResults Method
```

Shape property

Applies to: Cell, Characters, Hyperlink, OLEObject, ShapeData

Summary: Returns the Shape object that owns a Cell or Characters object or that is associated with an OLEObject, Hyperlink, or ShapeData object.

Version: VISIO 3.0

Syntax: objRet = object.**Shape**

Element	Description
objRet	The Shape object that contains or is associated with the object
object	The object to examine

Example for Shape

```
*Text Property
```

ShapeAdded event

Applies to: Application, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs after shape(s) are added to a Visio document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtAdd+visEvtShape (&H8040)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Shape that was added
moreInfo	Nothing for this event

Remarks: A Shape object can serve as the source object for ShapeAdded iff the shape's Type property is visTypeGroup (2) ot visTypePage(1).

The SelectionAdded and ShapeAdded events are similar in that they both fire after shape(s) are created. They differ in how they behave when a single operation creates several shapes. Suppose a paste operation produces 3 new shapes. ShapeAdded will fire 3 times and the respective subject objects will be the 3 added shapes. SelectionAdded will fire once and its subject object will be a Selection object in which the 3 new shapes are selected.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, Shape object, SelectionAdded event

Example for ShapeAdded

```
*DocumentCreated Event
```

ShapeChanged event

Applies to: Application, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs after non-cell properties of a shape are changed in a Visio document.

Version: VISIO 4.5

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtShape (&H2040)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Shape that just changed
moreInfo	Nothing for this event

Remarks: The ShapeChanged event indicates that a property of a shape that isn't stored in a cell has changed.

Note that most shape properties are stored in cells. Sign up for CellChanged rather than ShapeChanged to determine when properties stored in cells have changed.

To determine which properties have changed when ShapeChanged fires, use EventInfo. The string returned by EventInfo will contain a list of substrings identifying the changed properties.

Shape properties that will cause ShapeChanged to fire are:
- shape name (EventInfo will contain "/name")
- data1 (EventInfo will contain "/data1")
- data2 (EventInfo will contain "/data2")
- data3 (EventInfo will contain "/data3")
- uniqueid (EventInfo will contain "/uniqueid")

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source ShapeChanged using the AddAdvise method. In addition, ShapeChanged is included in the event set of all the objects in the Applies to list except the Document object. For those objects you can use VBA Dim WithEvents variables to sink ShapeChanged. For performance considerations,

the Document object's event set does not include ShapeChanged. To sink ShapeChanged from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, CellChanged event, EventInfo property

Example for ShapeChanged

Shapes property

Applies to: Master, Page, Shape

Summary: Returns the Shapes collection for an object.

Version: VISIO 2.0

Syntax: objsRet = object.**Shapes**

Element	Description
objsRet	The Shapes collection of the object
object	The object that owns the collection

Remarks: Use the Shapes property to retrieve the Shapes collection for page, master, or group.

See also: Shapes object

Example for Item, Shapes

```
'This VBA macro demonstrates getting the Shapes collection. It also uses the
Item
'and Count properties. It prints all shapes on Page1 in the Immediate window.
'To run this macro, make sure the active document is a Visio drawing with
shapes
'on Page1.

Public Sub ShapeNames_Example ()

  Dim I As Integer, iShapeCount As Integer
  Dim shpsObj As Visio.Shapes

  Set shpsObj = ActiveDocument.Pages.Item(1).Shapes

  Debug.Print "Shape Name List For Document : "; ActiveDocument.Name
  Debug.Print "                          Page : ";
ActiveDocument.Pages.Item(1).Name

  iShapeCount = shpsObj.Count

  If iShapeCount > 0 Then
      For I = 1 To iShapeCount
          Debug.Print "  "; shpsObj.Item(I).Name
      Next I
  Else
      Debug.Print "  No Shapes On Page"
  End If

End Sub
```

ShapesDeleted event

Applies to: Document

Summary: The event that occurs after shape(s) are deleted from a Visio document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeShapeDelete (801). See remarks.
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Selection which just lost member(s)
moreInfo	The name(s) of the deleted shape(s)

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above. An Event object created with this event code triggers the action after shapes are deleted and passes visEvtCodeShapeDelete with the notification.

Because ShapesDeleted is an after event, the deleted shapes are gone when the notification is received. To receive notification just before shapes are deleted, use the BeforeShapeDelete, BeforeSelectionDelete or BeforeWindowSelDelete event instead of ShapesDeleted.

How you determine which page or master contained the deleted shapes depends on the Action property of the Event object whose target has been triggered. If the event's Action is visActCodeRunAddon, then the index of the document and page or master containing the shapes is passed in the command string. If the Action is visActCodeAdvise, then the subject object passed to VisEventProc is a Selection object whose ContainingShape is the parent shape of the shapes that got deleted.

To determine the names of which shapes were deleted, use EventInfo. If one shape is deleted, the EventInfo string is in the following form:

    /shapes=shapename

where shapename is the shape's unique id if it has one; otherwise it is the shape's nameID (sheet.n).

If more than one shape is deleted, the EventInfo string will be of the following form, unless the total number of characters in the EventInfo string would exceed 8096:

    /shapes=shapename1;shapename2;shapename3;...

The shape list does not include subshapes of deleted shapes. If a group is deleted, only the group is included in the EventInfo string. The group's members are not included.

If the total number of characters in the EventInfo string would exceed 8096 characters, the EventInfo string is as follows:

/shapes=many

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, BeforeShapeDelete event, BeforeSelectionDelete event, BeforeWindowSelDelete event, Event object, EventList object, EventInfo property

Example for ShapesDeleted

```
*DocumentSaved Event
```

Shift property

Applies to: AccelItem

Summary: Gets or sets whether the Shift key is a modifier for the AccelItem object.

Version: VISIO 4.0

Syntax: object.**Shift** = intExpression
intRet = object.**Shift**

Element	Description
object	The AccelItem object that has or gets the setting
intRet	True (-1) if Shift modified Key; otherwise False (0)
IntExpression	True (non-zero) if Shift modified Key; otherwise False (0)

See also: Alt property, Control property, Key property

Example for Shift

ShortValue property

Applies to: Entity

Summary: Specifies the short integer value of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**ShortValue**
object.**ShortValue** = Expression

Element	Description
RetVal	The current short integer value
object	The Entity object that has or gets the value
Expression	The new short integer value

Remarks: If an Entity object has a Group of 1070, then it contains a 16-bit integer.

See also: Group property, LongValue property, RealValue property

Example for ShortValue

ShowConnectPoints property

Applies to: Window

Summary: Determines whether Visio shows or doesn't show connection points in the window.

Version: VISIO 4.5

Syntax: intRet = object.**ShowConnectPoints**
object.**ShowConnectPoints** = intExpression

Element	Description
intRet	0 if connection point display is off; -1 if connection point display is on
object	The Window object that has or gets the setting
intExpression	0 to turn connection point display off; non-zero to turn connection point display on

Remarks: Setting ShowConnectPoints is analogous to toggling connection point display using the Connection Points item in Visio's View menu.

See also: ShowMenus property, ShowStatusBar property, ToolbarStyle property, ShowRulers property, ShowGrid property, ShowGuides property, ShowPageBreaks property

Example for ShowConnectPoints

ShowGrid property

Applies to: Window

Summary: Determines whether Visio shows or doesn't show a grid in the window.

Version: VISIO 4.5

Syntax: intRet = object.**ShowGrid**
object.**ShowGrid** = intExpression

Element	Description
intRet	0 if grid display is off; -1 if grid display is on
object	The Window object that has or gets the setting
intExpression	0 to turn grid display off; non-zero to turn grid display on

Remarks: Setting ShowGrid is analogous to toggling the grid display using the Grid command located on Visio's View menu.

See also: ShowMenus property, ShowStatusBar property, ToolbarStyle property, ShowRulers property, ShowGuides property, ShowConnectPoints property, ShowPageBreaks property

Example for ShowGrid

```
'This VBA macro demonstrates hiding the grid.

Public Sub HideGrid_Example()

    ' Check whether current window is a drawing window
    If ActiveWindow.Type = visDrawing Then
        ' Hide the grid
        ActiveWindow.ShowGrid = False
    Else
        ' Tell the user why you're not hiding the grid
        MsgBox "Current window is not a drawing window.", vbOKOnly, DVS_TITLE
    End If

End Sub
```

ShowGuides property

Applies to: Window

Summary: Determines whether Visio shows or doesn't show guides in the window.

Version: VISIO 4.5

Syntax: intRet = object.**ShowGuides**
object.**ShowGuides** = intExpression

Element	Description
intRet	0 if guide display is off; -1 if guide display is on
object	The Window object that has or gets the setting
intExpression	0 to turn guide display off; non-zero to turn guide display on

Remarks: Setting ShowGuides is analogous to toggling guide display using the Guides item in Visio's View menu.

See also: ShowMenus property, ShowStatusBar property, ToolbarStyle property, ShowRulers property, ShowGrid property, ShowConnectPoints property, ShowPageBreaks property

Example for ShowGuides

ShowMenus property

Applies to: Application

Summary: Determines whether Visio shows or doesn't show menus.

Version: VISIO 4.5

Syntax: intRet = object.**ShowMenus**
object.**ShowMenus** = intExpression

Element	Description
intRet	0 if menu display is off; -1 if menu display is on
object	The Application object that has or gets the setting
intExpression	0 to turn menu display off; non-zero to turn menu display on

Remarks: The ShowMenus setting does not persist each time you run Visio--it is only valid for an instance of Visio. However, the ShowStatusBar and ToolbarStyle settings do persist.

See also: ShowStatusBar property, ShowToolbar property, ToolbarStyle property

Example for ShowMenus, ShowStatusBar

```
'This VBA macro demonstrates hiding the menus, toolbars, and status bars.

Public Sub NoVisioUI_Example()

    'Hide the status bars
    Visio.Application.ShowStatusBar = False

    'Hide the toolbars
    Visio.Application.ToolbarStyle = Visio.visToolBarNone

    'Hide the menus
    Visio.Application.ShowMenus = False

    MsgBox "Restore the status bars, toolbars, and menus now."
    Visio.Application.ShowStatusBar = True
    Visio.Application.ToolbarStyle = Visio.visToolbarMSOffice
    Visio.Application.ShowMenus = True

End Sub
```

ShowPageBreaks property

Applies to: Window

Summary: Determines whether Visio shows or doesn't show page break positions in the window.

Version: VISIO 4.5

Syntax: intRet = object.**ShowPageBreaks**
object.**ShowPageBreaks** = intExpression

Element	Description
intRet	0 if page break display is off; -1 if page break display is on
object	The Window object that has or gets the setting
intExpression	0 to turn page break display off; non-zero to turn page break display on

Remarks: Setting ShowPageBreaks is analogous to toggling page break display using the Page Breaks item in Visio's View menu.

See also: ShowMenus property, ShowStatusBar property, ToolbarStyle property, ShowRulers property, ShowGrid property, ShowGuides property, ShowConnectPoints property

Example for ShowPageBreaks

ShowProgress property

Applies to: Application

Summary: Determines whether Visio shows a progress indicator while performing certain operations.

Version: VISIO 4.1

Syntax: intRet = object.**ShowProgress**
object.**ShowProgress** = intExpression

Element	Description
intRet	False (0) if progress indicators are suppressed; True (-1) if they are shown
object	The object that has or gets the setting
intExpression	False (0) to suppress progress indicators; True (non-zero) to show them

Remarks: If you want to perform some operation, such as printing, that typically displays a progress indicator but don't want the progress indicator to appear, set ShowProgress to False (0). By default, ShowProgress is True (non-zero).

In most cases you should restore the setting to its prior value when you've completed the operation.

See also: AlertResponse property

Example for ShowProgress

ShowRulers property

Applies to: Window

Summary: Determines whether Visio shows or doesn't show rulers in the window.

Version: VISIO 4.5

Syntax: intRet = object.**ShowRulers**
object.**ShowRulers** = intExpression

Element	Description
intRet	0 if ruler display is off; -1 if ruler display is on
object	The Window object that has or gets the setting
intExpression	0 to turn ruler display off; non-zero to turn ruler display on

Remarks: Setting ShowRulers is analogous to toggling ruler display using the Rulers commnad located on Visio's View menu.

See also: ShowMenus property, ShowStatusBar property, ToolbarStyle property, ShowGrid property, ShowGuides property, ShowConnectPoints property, ShowPageBreaks property

Example for ShowRulers

```
'This VBA macro demonstrates hiding the rulers.

Public Sub HideRulers_Example()

    ' Check whether the current window is a drawing window
    If ActiveWindow.Type = visDrawing Then
        ' Hide the rulers
        ActiveWindow.ShowRulers = False
    Else
        ' Tell the user why you're not hiding the rulers
        MsgBox "Current window is not a drawing window.", vbOKOnly, DVS_TITLE
    End If

End Sub
```

ShowStatusBar property

Applies to: Application

Summary: Determines whether Visio shows or doesn't show a status bar.

Version: VISIO 4.5

Syntax: intRet = object.**ShowStatusBar**
object.**ShowStatusBar** = intExpression

Element	Description
intRet	0 if status bar display is off; -1 if status bar display is on
object	The Application object that has or gets the setting
intExpression	0 to turn status bar display off; non-zero to turn status bar display on

Remarks: The ShowStatusBar and ToolbarStyle settings persist each time you run Visio. ShowMenus, however, does not persist each time you run Visio--it is only valid for an instance of Visio.

See also: ShowMenus property, ShowToolbar property, ToolbarStyle property

Example for ShowStatusBar

```
*ShowMenus Property
```

ShowToolbar property

Applies to: Application

Summary: Determines whether the Visio toolbar is visible or hidden.

Version: VISIO 5.0

Syntax: intRet = object.**ShowToolbar**
object.**ShowToolbar** = intExpression

Element	Description
intRet	0 if toolbar display is off; -1 if toolbar display is on
object	The Application object that has or gets the setting
intExpression	0 to turn toolbar display off; non-zero to turn toolbar display on

Remarks: The ShowStatusBar and ShowToolbar settings persist each time you run Visio. ShowMenus, however, does not persist each time you run Visio--it is only valid for an instance of Visio.

Prior to Visio 5.0 Application.BuiltInToolbars and Application.ToolbarStyle supported toolbar "flavors." Supported flavors were:

visToolBarNone
visToolBarMSOffice
visToolBarLotusSS

Toolbar flavors were removed from Visio 5.0 and toolbars are simply present or absent. The ShowToolbar property was added for this reason.

See also: ShowMenus property, ShowStatusBar property, ToolbarStyle property, BuiltInToolbars property

Example for ShowToolbar

Spacing property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the spacing before and after a toolbar or status bar item.

Version: VISIO 4.0

Syntax: object.**Spacing** = intVal
intVal = object.**Spacing**

Element	Description
object	The object that has or gets the spacing
intVal	The spacing before and after the item

Remarks: Valid spacings are declared by the Visio type library (and visconst.bas):

visCtrlSpacingNONE = &H0
visCtrlSpacingVARIABLE_BEFORE = &H1
visCtrlSpacingVARIABLE_AFTER = &H2
visCtrlSpacingFIXED_BEFORE = &H4
visCtrlSpacingFIXED_AFTER = &H8

Example for Spacing

Start property

Applies to: Curve

Summary: Returns the start of a Curve object's parameter domain.

Version: VISIO 5.0

Syntax: retVal = object.**Start**

Element	Description
retVal	Starting value of Curve's parameter domain
object	The Curve object that has the value

Remarks: The Start property of Curve returns the coordinates of the curve's starting point. A Curve object describes itself in terms of its parameter domain, which is the range [Start(),End()] where Start() produces the curve's starting point.

See also: End property, Point method, PointAndDerivatives method

Example for Start

```
*Point method
```

StartupPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio will look for add-ons to run automatically when Visio is started.

Version: VISIO 4.0

Syntax: strRet = object.**StartupPaths**
object.**StartupPaths** = pathsStr

Element	Description
strRet	A list of directories
object	An Application object
pathsStr	A list of directories

Remarks: The string passed to and received from StartupPaths is the same string shown in Visio's File Paths dialog (accessible from the Tools/Options dialog). This same string is stored in Visio's profile (.ini) file (appObj.ProfileName) in the entry whose key is "StartUpPath."

To indicate more than one folder, separate individual items in the path string with semicolons. If a path is not fully qualified, Visio looks for the folder in the folder that contains the Visio program files (appObj.Path).

For example, if Visio's executable file is installed in c:\Visio, and StartupPaths is "Startup;d:\Startup", when Visio is started it looks for add-ons in both c:\Visio\Startup and d:\Startup.

When Visio looks for startup add-ons, it will look in all paths named in StartupPaths plus in all sub-folders of those paths. Also, the fact that a path is named in StartupPaths does not imply the path actually exists. If you pass StartupPaths to the EnumDirectories method, it will return a complete list of fully qualified paths that Visio will actually look in.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, HelpPaths property, StencilPaths property, TemplatePaths property, ProfileName property, Path property, EnumDirectories method

Example for StartupPaths

Stat property

Applies to: Application, Cell, Characters, Color, Colors, Connect, Connects, Curve, Document, Font, Fonts, Hyperlink, Layer, Layers, Master, Masters, OLEObject, OLEObjects, Page, Pages, Path, Paths, Selection, Shape, Shapes, Style, Styles, Window

Summary: Returns status information for an object.

Version: VISIO 3.0

Syntax: intRet = object.**Stat**

Element	Description
intRet	A bit mask of status bits. See remarks.
object	The object whose status is being examined

Remarks: If an object is a reference to some entity in a document, and if that document closes, object.Stat returns a value in which the visStatClosed bit is set.

If an object is a reference to an entity that has been deleted, object.Stat will returns a value in which the visStatDeleted bit is set.

See also: BeforeDocumentClose event

Example for Stat

StatusBarItems property

Applies to: StatusBar

Summary: Returns the StatusBarItems collection of a StatusBar object.

Version: VISIO 4.0

Syntax: objRet = object.**StatusBarItems**

Element	Description
objRet	The StatusBarItems collection of the StatusBar object
object	The StatusBar object that owns the collection

See also: StatusBarItems object

Example for StatusBarItems

StatusBars property

Applies to: UI Object

Summary: Returns the StatusBars collection of a UI object.

Version: VISIO 4.0

Syntax: objRet = object.**StatusBars**

Element	Description
objRet	The StatusBars collection of the UI object
object	The UI object that owns the collection

Remarks: If a UI object represents toolbars and status bars (for example, if the object was retrieved using the BuiltInToolbars property of an Application object), its StatusBars collection represents all of the status bars for that UI object.

Use the ItemAtID property of a StatusBars object to retrieve status bars for a particular window context, for example, the drawing window. If a context does not include status bars, it has no StatusBars collection. For a list, see the StatusBars object.

See also: StatusBars object

Example for StatusBars

StencilPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for stencils.

Version: VISIO 4.0

Syntax: strRet = object.**StencilPaths**
object.**StencilPaths** = pathsStr

Element	Description
strRet	A text string containing a list of folders
object	An Application object
pathsStr	A text string containing a list of folders

Remarks: The string passed to and received from StencilPaths is the same string shown in Visio's File Paths dialog (accessible from the Tools/Options dialog). This same string is stored in Visio's profile (.ini) file (appObj.ProfileName) in the entry whose key is "StencilPath."

To indicate more than one folder, separate individual items in the path string with semicolons. If a path is not fully qualified, Visio looks for the folder in the folder that contains the Visio program files (appObj.Path).

For example, if Visio's executable file is installed in c:\Visio, and StencilPaths is "Stencils;d:\Stencils", Visio looks for stencils in both c:\Visio\Stencils and d:\Stencils.

When Visio looks for stencils, it will look in all paths named in StencilPaths plus in all sub-folders of those paths. Also, the fact that a path is named in StencilPaths does not imply the path actually exists. If you pass StencilPaths to the EnumDirectories method, it will return a complete list of fully qualified paths that Visio will actually look in.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, HelpPaths property, StartupPaths property, TemplatePaths property, ProfileName property, Path property, EnumDirectories method

Example for StencilPaths

String property

Applies to: Entity

Summary: Specifies the String value contained in an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**String**
object.**String** = Expression

Element	Description
RetVal	The current string value
object	The Entity object that has or gets the value
Expression	The new string value

Remarks: If an Entity object has a Group of 1000, then it contains a string of up to 255 characters.

See also: Group property

Example for String

Style property

Applies to: Cell, Selection, Shape

Summary: Gets or sets the style for a Shape object, or gets the style that contains a Cell object.

Version: VISIO 2.0

Syntax: strRet = object.**Style**
object.**Style** = stringExpression
objRet = cellObject.**Style**

Element	Description
strRet	The fill style component of the style
object	The Shape or Selection object that has or gets the style
stringExpression	The name of the style to apply
objRet	A Style object that represents the style containing the cell
cellObject	The Cell object to examine

Remarks: If a style has diverse text, line, and fill styles applied to it, the Style property returns the fill style. Setting the Style property to a non-existent style generates an error.

To preserve local formatting, use the StyleKeepFmt property.

If a Cell object is in a style, its Style property returns the style that contains the cell, and its Shape property returns Nothing. If a Cell object is in a shape, its Shape property returns the shape that contains the cell, and its Style property returns Nothing.

See also: FillStyle property, LineStyle property, Shape property, StyleKeepFmt property, TextStyle property

Example for Style

StyleAdded event

Applies to: Application, Document, Documents, Styles

Summary: The event that occurs after a new style is added to a Visio document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtAdd+visEvtStyle (&H8004)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Style that was just created
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, Style object

Example for StyleAdded

StyleChanged event

Applies to: Application, Document, Documents, Style, Styles

Summary: The event that occurs after certain properties of a style are changed in a Visio document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtStyle (&H2004)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Style that just changed
moreInfo	Nothing for this event

Remarks: The StyleChanged event indicates that the name of a style has changed or that a change to the style has caused its properties to be propagated to objects to which the style is applied.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for StyleChanged

StyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a style to an object while preserving local formatting.

Version: VISIO 2.0

Syntax: object.**StyleKeepFmt** = stringExpression

Element	Description
object	The Shape or Selection object that gets the style
stringExpression	The name of the style to apply

Remarks: Setting the StyleKeepFmt property is equivalent to checking the Preserve Local Formatting option in the Style dialog box in Visio. Setting a style to a non-existent style generates an error.

Example for StyleKeepFmt

Styles property

Applies to: Document

Summary: Returns the Styles collection for a document.

Version: VISIO 2.0

Syntax: objRet = object.**Styles**

Element	Description
objRet	The Styles collection of the Document object
object	The Document object that owns the collection

See also: Style object, Styles object

Example for Styles

```
*Add Method
```

SubAddress property

Applies to: Hyperlink

Summary: Returns or sets the Subaddress in a shape's Hyperlink object that represents the subaddress to which a shape's hyperlink will navigate.

Version: VISIO 5.0

Syntax: strRet = object.**SubAddress**
object.**SubAddress** = stringExpression

Element	Description
strRet	The current value of the field
object	The object that has or gets the value
stringExpression	The new value for the field

Remarks: Setting the SubAddress property of a shape's hyperlink is optional unless the Address property is blank. In this case the SubAddress must contain the name of the drawing page.

Setting a Hyperlink's Subaddress is equivalent to entering information in the Named Location In File field of the Hyperlink dialog box, accessed from the Insert menu. This is also equivalent to setting the result of the Subaddress cell in the shape's hyperlink row.

A Hyperlink's SubAddress property specifies a sublocation within the hyperlink's address. For Visio files, this can be a page name. For Excel, this can be a worksheet or a range within a worksheet. For HTML pages, this can be a sub-anchor.

The hyperlink address for which a subaddress is being supplied must support SubAddress linking. As of this writing, only Visio 4.5 and later, and Microsoft Office 97 and later, provide this support.

See also: Address property, ExtraInfo property

Example for SubAddress

*AddHyperlink method

Subject property

Applies to: Document

Summary: Returns or sets the value of the Subject field in a document's properties.

Version: VISIO 2.0

Syntax: strRet = object.**Subject**
object.**Subject** = stringExpression

Element	Description
strRet	The current value of the field
object	The Document object that has or gets the value
stringExpression	The new value for the field

Remarks: Setting the Subject property is equivalent to entering information in the Subject field in the Properties dialog box located on the File menu.

See also: Creator property, Description property, Keywords property, Title property, Manager property, Company property, Category property, HyperlinkBase property

Example for Subject

*Document Property

Subtract method

Applies to: Selection, Window

Summary: Subtracts from one selected shape those areas that overlap other selected shapes.

Version: VISIO 4.0

Syntax: object.**Subtract**

Element	Description
object	The Window or Selection object that contains the shapes to subtract

Remarks: The Subtract method is equivalent to choosing the Subtract command from the Operations submenu on the Shape menu in Visio. The first selected shape is the one that will have the other selected shapes subtracted from it. The other shapes will be deleted.

If the object being operated on is a Selection object, it will have no shapes selected in it when the operation is complete.

See also: Combine method, Fragment method, Intersect method, Join method, Trim method, Union method, ContainingShape property

Example for Subtract

SubType property

Applies to: Window

Summary: Returns the subtype of a Window object that represents a drawing window.

Version: VISIO 4.0

Syntax: intRet = object.**SubType**

Element	Description
intRet	The subtype of the Window object
object	The Window object to examine

Remarks: If the Type property of a Window object returns any value other than visDrawing, SubType returns the same value as the Type property. If the Type property of a Window object returns visDrawing, SubType returns one of the following values:

visPageWin = 128
visPageGroupWin = 160
visMasterWin = 64
visMasterGroupWin = 96

visPageWin indicates a drawing window showing a page.

visPageGroupWin indicates a group editing window of a group on a page.

visMasterWin indicates a master drawing page window.

visMasterGroupWin indicates a group editing window of a group in a master.

See also: Type property

Example for SubType

TableName property

Applies to: AccelTable

Summary: Gets or sets the name of an AccelTable object.

Version: VISIO 4.0

Syntax: object.**TableName** = nameStr
strRet = object.**TableName**

Element	Description
object	The AccelTable object that has or gets the name
nameStr	The new name for the object
strRet	The current name of the object

Remarks: This property is not currently used by Visio in its user interface.

Example for TableName

TabPropsRow property

Applies to: Characters

Summary: Returns the index of the tab properties row that contains tab formatting information for a Characters object.

Version: VISIO 3.0

Syntax: intRet = object.**TabPropsRow**(bias)

Element	Description
intRet	The index of the row that defines the Character object's formatting
object	The Characters object to examine
bias	The direction of the search

Remarks: Rows that represent runs of tab formatting can be retrieved by specifying a row index as an argument to the CellsSRC property of a shape. Tab formats may be viewed or changed in Visio's Tabs dialog box.

If the tab format for the Characters object is represented by more than one tab properties row, TabPropsRow returns -1. If the Characters object represents an insertion point rather than a sequence of characters (that is, if its Begin and End properties return the same value), use the bias argument to determine which row index to return:

visBiasLeft = 1
visBiasRight = 2
visBiasLetVisioChoose = 0

Specify visBiasLeft for the row that covers tab formatting for the character to the left of the insertion point, or visBiasRight for the row that covers tab formatting for the character to the right of the insertion point.

See also: CharPropsRow property, ParaPropsRow property

Example for TabPropsRow

Target property

Applies to: Event

Summary: Gets or sets the target of an event.

Version: VISIO 4.0

Syntax: strRet = object.**Target**
object.**Target** = stringExpression

Element	Description
strRet	The current target
object	The Event object that has or gets the target
stringExpression	The target to set

Remarks: An event consists of an event-action pair. When the event occurs, the action is performed. An event also specifies the target of the action and arguments to send to the target.

If the action code of the event is visActCodeRunAddon, the Target property contains the name of the add-on to run.

If the action code of the event is visActCodeAdvise, the Target property is not available. Attempting to get or set the Target property for such an event will cause an exception.

See also: Action property, Event property, EventInfo property, TargetArgs property

Example for Target

TargetArgs property

Applies to: Event

Summary: Gets or sets the arguments to be sent to the target of an event.

Version: VISIO 4.0

Syntax: strRet = object.**TargetArgs**
object.**TargetArgs** = stringExpression

Element	Description
strRet	The current arguments
object	The Event object that has or gets the arguments
stringExpression	The new arguments to set

Remarks: An event consists of an event-action pair. When the event occurs, the action is performed. An event also specifies the target of the action and arguments to send to the target.

Visio supports the following actions, which determine the target arguments of the event:

* visActCodeRunAddon. In this case, the TargetArgs property contains the arguments to send to the add-on when it is run.

* visActCodeAdvise. In this case, the TargetArgs property contains the string specified with the AddAdvise method when the Event object was created. When the program receives notification of the event, it can get the Event object and get its TargetArgs property to obtain the string.

See also: Action property, Event property, EventInfo property, Target property

Example for TargetArgs

Template property

Applies to: Document

Summary: Returns the name of the template from which the document was created.

Version: VISIO 4.0

Syntax: strRet = object.**Template**

Element	Description
strRet	The name of the template from which the Document object was created
object	The Document object to examine

Example for Template

```
'This VBA macro demonstrates the using the Template property to get the name
of
'the template from which the document was created.

Public Sub TemplateProp_Example()

  Dim documentObj As Visio.Document
  Dim strTemplateName As String

  strTemplateName = ThisDocument.Template
  Debug.Print strTemplateName             'Verify the proper string was
returned.

End Sub
```

TemplatePaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for templates.

Version: VISIO 4.0

Syntax: strRet = object.**TemplatePaths**
object.**TemplatePaths** = pathsStr

Element	Description
strRet	A text string containing a list of folders
object	An Application object
pathsStr	A text string containing a list of folders

Remarks: The string passed to and received from TemplatePaths is the same string shown in Visio's File Paths dialog (accessible from the Tools/Options dialog). This same string is stored in Visio's profile (.ini) file (appObj.ProfileName) in the entry whose key is "TemplatePath."

To indicate more than one folder, separate individual items in the path string with semicolons. If a path is not fully qualified, Visio looks for the folder in the folder that contains the Visio program files (appObj.Path).

For example, if Visio's executable file is installed in c:\Visio and TemplatePaths is "Template;d:\Template", Visio looks for add-ons in both c:\Visio\Template and d:\Template.

When Visio looks for templates, it will look in all paths named in TemplatePaths plus in all sub-folders of those paths. Also, the fact that a path is named in TemplatePaths does not imply the path actually exists. If you pass TemplatePaths to the EnumDirectories method, it will return a complete list of fully qualified paths that Visio will actually look in.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, HelpPaths property, StartupPaths property, StencilPaths property, ProfileName property, Path property, EnumDirectories method

Example for TemplatePaths

Text property

Applies to: Characters, Shape

Summary: Returns or sets the text of the object.

Version: VISIO 2.0

Syntax:

strRet = shapeObj.**Text**
shapeObj.**Text** = stringExpression

stringVariantRet = charsObj.**Text**
charsObj.**Text** = stringOrObjVariant

Element	Description
strRet	The text of the Shape object returned as a string
shapeObj	The Shape object that owns the text
stringExpression	New text for the Shape object
stringVariantRet	The text of the Characters object returned in a variant of type string
charsObj	The Characters object that owns the text
stringOrObjVariant	A variant to which is assigned a string, Shape object or Characters object

Remarks: The Text property of a Shape object returns the entire text of the shape.

The Text property of a Characters object returns the range of text represented by that object, which may be a subset of the shape's text depending on the values of the Characters object's Begin and End properties. The text is returned in a variant of type string, as opposed to in a string. This is typically transparent if you're using VB or VBA.

If you are using C/C++ and want a string rather than a variant, use TextAsString.

[Note: In earlier versions of Visio (through version 4.1), CharsObj.Text did return a string. Due to changes in Automation support tools, it became necessary to change the property to return a variant of type string. For backward compatibility, TextAsString was added. TextAsString has the same signature and occupies the same vtble slot as did the prior version of CharsObj.Text.]

In the text returned by the Text property of a Shape object, fields are represented as 4-character escape sequences.

In the text returned by a Characters object, fields are expanded to the number of characters that are visible in the drawing window.

For example, if a Shape object's text contains a field that displays the filename of a drawing, the Shape object's Text property returns a 4-character escape sequence. For a Characters object, the Text property returns the filename (provided the Begin and End properties were not altered).

Objects from other applications have no Text property.
Guides have no Text property.

See also: Characters object, Shape object, TextAsString property

Example for Characters, Shape, Text

```
'This VBA macro demonstrates setting the Shape object properties.

Public Sub ShapeProp_Example ()

  Dim shpRectObj As Visio.Shape
  Dim shpOvalObj As Visio.Shape
  Dim shpObjFromCell As Visio.Shape
  Dim shpObjFromChars As Visio.Shape
  Dim cellObj As Visio.Cell
  Dim charsObj As Visio.Characters

  'Create 2 different shapes and add different text to each shape.
  Set shpRectObj = ActivePage.DrawRectangle(2, 3, 5, 4)
  Set shpOvalObj = ActivePage.DrawOval(2, 5, 5, 7)
  shpRectObj.Text = "Rectangle Shape"
  shpOvalObj.Text = "Oval Shape"

  'Get a Cell object from the first shape.
  Set cellObj = shpRectObj.Cells("Width")
  'Get a Characters object from the second shape.
  Set charsObj = shpOvalObj.Characters

  'Use the shape property to get the shape object.
  Set shpObjFromCell = cellObj.Shape
  Set shpObjFromChars = charsObj.Shape

  'Use each shape's text to verify the proper shape object was returned.
  Debug.Print shpObjFromCell.Text
  Debug.Print shpObjFromChars.Text

End Sub
```

TextAsString property

Applies to: Characters

Summary: Returns the text of the object.

Version: VISIO 4.5

Syntax: strRet = object.**TextAsString**

Element	Description
strRet	The text of the Characters object returned as a string
object	The Characters object that owns the text

Remarks: The TextAsString property of a Characters object returns the range of text represented by that object, which may be a subset of the shape's text depending on the values of the Characters object's Begin and End properties. The text is returned as a string. Object.Text would return the text in a variant of type string.

[Note: Up through and including version 4.1 of Visio, CharsObj.Text did return a string. Due to changes in Automation support tools, it became necessary to change the Text property to return a variant of type string. For backwards compatibility, TextAsString was added. It behaves like the Text property used to, and occupies the same slot in the vtble as the old property. If you're developing new code, you'll likely find very few occasions when you must use TextAsString.]

In the text returned by a Characters object, fields are expanded to the number of characters that are visible in the drawing window.

See also: Text property

Example for TextAsString

TextBasedOn property

Applies to: Style

Summary: Gets or sets the text style that a Style object is based on.

Version: VISIO 4.0

Syntax: strVal = object.**TextBasedOn**
object.**TextBasedOn** = styleName

Element	Description
strVal	The name of the current based-on text style
object	The Style object that has or gets the based-on style
styleName	The name of the new based-on style

Remarks: To base a style on no style, set TextBasedOn to a null string ("").

See also: BasedOn property, FillBasedOn property, LineBasedOn property

Example for TextBasedOn

TextChanged event

Applies to: Application, Characters, Document, Documents, Master, Masters, Page, Pages, Shape

Summary: The event that occurs after the text of a shape is changed in a Visio document.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtMod+visEvtText (&H2080)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Shape whose text just changed
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

The Applies to list shown above identifies objects that can source TextChanged using the AddAdvise method. In addition, TextChanged is included in the event set of all the objects in the Applies to list except the Document object. For those objects you can use VBA Dim WithEvents variables to sink TextChanged. For performance considerations, the Document object's event set does not include TextChanged. To sink TextChanged from a Document (including the ThisDocument object in a VBA project), you must use AddAdvise.

See also: Action property, Add method, AddAdvise method, Event object, EventList object

Example for TextChanged

TextStyle property

Applies to: Selection, Shape

Summary: Returns or sets the text style for an object.

Version: VISIO 2.0

Syntax: strRet = object.**TextStyle**
object.**TextStyle** = stringExpression

Element	Description
strRet	The current text style
object	The Shape or Selection object that has or gets the style
stringExpression	The name of the text style to apply

Remarks: Setting this property is equivalent to selecting a style from the Text style list in Visio.

Setting a style to a non-existent style generates an error. Setting one kind of style to an existing style of another kind (for example, setting TextStyle to a fill style) does nothing. Setting one kind of style to an existing style that has more than one set of attributes changes only the attributes for that component (for example, setting TextStyle to a style with line, text, and fill attributes changes only the text attributes).

To preserve a shape's local formatting, use the TextStyleKeepFmt property.

See also: TextStyleKeepFmt property

Example for TextStyle

TextStyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a text style to an object while preserving local formatting.

Version: VISIO 2.0

Syntax: object.**TextStyleKeepFmt** = stringExpression

Element	Description
object	The Shape or Selection object to which the style is applied
stringExpression	The name of the style to apply

Remarks: Setting the TextStyleKeepFmt property is equivalent to checking the Preserve Local Formatting option in the Style dialog box in Visio.

Setting a style to a non-existent style generates an error. Setting one kind of style to an existing style of another kind (for example, setting TextStyleKeepFmt to a fill style) does nothing. Setting one kind of style to an existing style that has more than one set of attributes changes only the attributes for that component (for example, setting TextStyleKeepFmt to a style with line, text, and fill attributes changes only the text attributes).

See also: TextStyle property

Example for TextStyleKeepFmt

Title property

Applies to: Document

Summary: Returns or sets the value of the Title field in a document's properties.

Version: VISIO 2.0

Syntax: strRet = object.**Title**
object.**Title** = stringExpression

Element	Description
strRet	The current value of the field
object	The Document object that has or gets the value
stringExpression	The new value for the field

Remarks: Setting the Title property is equivalent to entering information in the Title field in the Properties dialog box located on the File menu.

See also: Creator property, Description property, Keywords property, Subject property, Manager property, Company property, Category property, HyperlinkBase property

Example for Title

*Document Property

ToCell property

Applies to: Connect

Summary: Returns the cell to which a connection is made.

Version: VISIO 2.0

Syntax: objRet = object.**ToCell**

Element	Description
objRet	The cell to which the connection is made
object	The Connect object to examine

Remarks: A connection is defined by a reference in a cell in the shape from which the connection originates to a cell in the shape to which the connection is made. The ToCell property returns the Cell object for the cell referred to by the shape from which the connection originates.

For a 2-D shape, a connection may be defined in one of the six cells in its Alignment section. In this case, the ToCell property returns the cell object referred to in that Alignment cell, either a GuidePosX or a GuidePosY cell object.

For a 1-D shape, a connection may be defined in its 1-D Endpoints section. If a 1-D shape is glued to a guide, the ToCell property returns the cell object referred to in the appropriate cell, either a GuidePosX or a GuidePosY cell object. If the 1-D shape is glued to another 1-D shape or a 2-D shape, ToCell returns the Cell object referred to in either the BeginX or EndX cell, depending on which endpoint is involved in the connection. The Cell object returned is the X cell in a Connections section row of the shape to which the connection is made.

For both 2-D and 1-D shapes, a connection may be defined in the X and Y cells of one row in the Controls section. In this case, the ToCell property returns the cell object referred to in the X cell, either the X cell object in a Connections row of the shape to which the connection is made or a GuidePosX or GuidePosY cell object.

This property will return PinX or PinY for a Connect object that represents a connection with dyamic glue. PinX indicates dynamic glue with a horizontal walking preference; PinY indicates a vertical walking preference.

See also: FromCell property, GlueTo method

Example for ToCell

```
*FromCell Property
```

ToolbarItems property

Applies to: Toolbar

Summary: Returns the ToolbarItems collection of a Toolbar object.

Version: VISIO 4.0

Syntax: objRet = object.**ToolbarItems**

Element	Description
objRet	The ToolbarItems collection of the Toolbar object
object	The Toolbar object that owns the collection

See also: ToolbarItems object

Example for ToolbarItems

*BuiltInToolbars Property

Toolbars property

Applies to: ToolbarSet

Summary: Returns the Toolbars collection of a ToolbarSet object.

Version: VISIO 4.0

Syntax: objRet = object.**Toolbars**

Element	Description
objRet	The Toolbars collection of the ToolbarSet object
object	The ToolbarSet object that owns the collection

See also: Toolbars object

Example for Toolbars

*BuiltInToolbars Property

ToolbarSets property

Applies to: UI Object

Summary: Returns the ToolbarSets collection of a UI object.

Version: VISIO 4.0

Syntax: objRet = object.**ToolbarSets**

Element	Description
objRet	The ToolbarSets collection of the UI object
object	The UI object that owns the collection

Remarks: If a UI object represents toolbars and status bars (for example, if the object was retrieved using the BuiltInToolbars property of an Application object), its ToolbarSets collection represents all of the toolbars for that UI object.

Use the ItemAtID property of a ToolbarSets object to retrieve toolbars for a particular window context, for example, the drawing window. If a context does not include toolbars, it has no ToolbarSets collection. For a list, see the ToolbarSets object.

See also: ToolbarSets object

Example for ToolbarSets

*BuiltInToolbars Property

ToolbarStyle property

Applies to: Application

Summary: Determines whether Visio shows or does not show a toolbar.

Version: VISIO 4.5

Syntax: intRet = object.**ToolbarStyle**
object.**ToolbarStyle** = fWhichToolbars

Element	Description
intRet	The present toolbar style
object	The Application object that has or gets the setting
fWhichToolbars	The set of built-in toolbars to get

Remarks: The notion of toolbar style was removed from Visio 5.0 and the toolbar is now simply present or absent. To remain compatible with earlier releases, ToolbarStyle now accepts either of the following values:

(1) visToolBarNone, which will hide the toolbar
(2) any other value, which will show the Microsoft Office toolbar

Visio 5.0 now provides Application.ShowToolbar, which has the same effect as ToolbarStyle.

The ToolbarStyle setting will persist each time you run Visio. This is also true for ShowStatusBar.
ShowMenus, however, does not persist each time you run Visio.

See also: ShowMenus property, ShowToolbar property, ShowStatusBar property

Example for ToolbarStyle

```
'This VBA macro demonstrates getting the ToolbarStyle property to find out
'which built-in Visio toolbar is in use.

Sub WhichToolbar_Example()

  Dim toolbarState As Integer

  toolbarState = Visio.Application.ToolbarStyle
  Debug.Print toolbarState

End Sub
```

ToPart property

Applies to: Connect

Summary: Returns the part of a shape to which a connection is made.

Version: VISIO 2.0

Syntax: intRet = object.**ToPart**

Element	Description
intRet	The part of the shape to which the connection is made
object	The Connect object to examine

Remarks: The ToPart property identifies the part of a shape to which another shape is glued, such as its begin point or end point, one of its edges, or a connection point. The following constants declared by the Visio type library (and visconst.bas) show possible return values for the ToPart property:

visConnectToError = -1
visToNone = 0
visGuideX = 1
visGuideY = 2
visConnectionPoint = 100
visWholeShape = 3

See also: FromPart property, GlueTo method, ToSheet property

Example for ToPart

```
*FromCell Property
```

TopMargin property

Applies to: Document

Summary: Specifies the top margin for printing a document's pages.

Version: VISIO 4.0

Syntax: retVal = object.**TopMargin**(units)
object.**TopMargin**(units) = newValue

Element	Description
retVal	The margin value expressed in the given units
object	The Document object that has or gets the margin
units	The units to use when retrieving or setting the margin value
newValue	The new margin value

Remarks: This property corresponds to the Top Margin control in Visio's Page Setup dialog box.

Units can be a string such as "inches", "inch", "in.", or "i". Strings may be used for all supported Visio units such as centimeters, meters, miles, and so on. You can also use any of the units constants declared by the Visio type library (and visconst.bas). See the Cell.Result property.

See also: LeftMargin property, RightMargin property, BottomMargin property, Result property

Example for TopMargin

ToSheet property

Applies to: Connect, Connects

Summary: Returns the shape to which a connection or connections are made.

Version: VISIO 2.0

Syntax: objRet = object.**ToSheet**

Element	Description
objRet	The shape to which the connection is made
object	The Connect object or Connects collection to examine

Remarks: Connect.ToSheet always returns the shape to which the connection is made.

Connects represents several connections. If every connection represented by the collection is made to the same shape, Connects.ToSheet will return that shape. Otherwise it returns nothing and does not raise an exception.

See also: FromSheet property, GlueTo method

Example for ToSheet

*FromCell Property

TraceFlags property

Applies to: Application

Summary: Records designated events that happen during a Visio instance to the VBA Immediate window.

Version: VISIO 5.0

Syntax: intRet = object.**TraceFlags**
object.**TraceFlags** = intExpression

Element	Description
intRet	Current trace flags
object	The Application object that has or gets the setting
intExpression	New trace flags

Remarks: The TraceFlags property can get or set the events that will be logged during a Visio instance. The value can be any combination of the following:

visTraceEvents (&H1) -- event occurrences
visTraceAdvises (&H2) -- outgoing advise calls
visTraceAddonInvokes (&H4) -- addon invocations
visTraceCallsToVBA (&H8) -- VBA invocations

visTraceEvents will cause the Immediate window to log most Visio events as they happen. In most cases this will occur even if no external agent is listening or responding to the event. In a few cases Visio knows there is no listener for an event and will not log the fact that the event occurred. Also, some events are specializations of other events and won't be recorded. For example, SelectionAdded is manufactured from distinct ShapeAdded events, so the Immediate window will record the ShapeAdded events but not the SelectionAdded events.

Here is a string Visio might log when visTraceEvents is selected:
-event: 0x8040 /doc=1 /page=1 /shape=Sheet.1

The number after -event: is the code of the event that happened. In this case 0x8040 is the code for the ShapeAdded event. The text following the event code will differ from event to event.

visTraceAdvises writes a line to the Immediate window just before Visio calls an event handler procedure and another line just after the event handler returns. This includes event procedures in VBA projects such as a procedure of ThisDocument. Here is an example of what you might see:
>advise seq=4 event=0x8040 sink=0x40097598
<advise seq=4

This indicates the call to and return from an event handler. This was the 4th event fired by Visio. The code of the event was 0x8040 and the address of the interface Visio called was 0x40097598.

visTraceAddonInvokes records when Visio invokes an executable or VSL add-on and when Visio regains control from the invocation. visTraceAddonInvokes will also trace attempts to invoke non-present add-ons. For example, if a cell's formula is

=RunAddon("xxx") and there is no add-on named "xxx", then a message such as "-invokeAO: Failed to load 'ARRAY32.VSL'" will be logged. Here is an example:

```
>invokeAO: SHOWARGS.EXE
<invokeAO: completed
```

visTraceCallToVBA writes a line to the Immediate window just before it makes a call into VBA other than to an event procedure, and another line just after VBA returns control to Visio. This includes macro invocations, calls to VBA procedures resulting from evaluation of cells that make use of RunAddon or CallThis operands, and calls resulting from selection of custom menu or toolbar items. Use visTraceAdvises to log calls to VBA event procedures.

```
>invokeVBA: Module1.MyMacro
<invokeVBA: completed
```

No messages will appear in the Immediate window if no document with a VBA project is open. Visio will queue a small number of messages to log when such document does open. But messages will be lost if no document with a project is available for lengthy periods. Messages will also be lost if VBA resets or if there are undismissed breakpoints.

Code in VBA projects can intersperse their messages with those logged by Visio using standard Debug.Print statements. Code in non-VBA projects can log messages to VBA's Immediate window using Document.VBProject.ExecuteLine("Debug.Print ""somestring""").

The TraceFlags property is recorded in the TraceFlags entry of the [application] section of Visio's initialization file (visio.ini).

Example for TraceFlags

Trigger method

Applies to: Cell, Event

Summary: Evaluates the formula of a cell or causes an event's action to be performed.

Version: VISIO 4.0

Syntax: cellObject.**Trigger**
eventObject.**Trigger** contextString

Element	Description
object	The Cell or Event object to trigger
contextString	The string to send to the target of the event

Remarks: Triggering a cell simply evaluates the formula of that cell. If the formula has side effects such as running an add-on, those side effects occur.

Triggering an event causes the action associated with the event to be performed. The specified context string is passed to the target of the action:

* If the action is to run an add-on (visEvtCodeRunAddon), the string is passed in the command line string sent to the add-on.

* If the action is to send a notification to the calling program (visEvtCodeAdvise), the string is passed in the moreInfo parameter of the notification.

See also: Action property, Add method, AddAdvise method, Event property

Example for Trigger

Trim method

Applies to: Selection, Window

Summary: Trims selected shapes into smaller shapes.

Version: VISIO 4.1

Syntax: object.**Trim**

Element	Description
object	The Window or Selection object that contains the shapes to trim

Remarks: The Trim method is equivalent to choosing the Trim command from the Operation submenu on the Shape menu in Visio.

The new shapes produced by Trim inherit the formatting of the first selected shape, have no text, and are the topmost shapes in their container--the nth, nth-1, nth-2, and so forth shape in the Shapes collection of their ContainingShape, where n = Count. The original shapes are deleted.

If the object being operated on is a Selection object, it will have no shapes selected when the operation is complete.

Trim is similar to Fragment. The shapes produced by Fragment coincide with the distinct regions of the selected shapes, taking overlap into account. The shapes produced by Trim coincide with the distinct paths of the selected shapes, also taking overlap into account.

See also: Combine method, Fragment method, Intersect method, Join method, Subtract method, Union method, ContainingShape property

Example for Trim

Type property

Applies to: Shape, Window

Summary: Returns the type of the object.

Version: VISIO 2.0

Syntax: retVal = object.**Type**

Element	Description
retVal	The type of the Shape or Window object
object	The Shape or Window object to examine

Remarks: The following constants declared by the Visio type library (and visconst.bas) show possible values that Type returns.

The Type property of a Shape object returns one of the following:

visTypePage = 1
visTypeGroup = 2
visTypeShape = 3
visTypeForeignObject = 4
visTypeGuide = 5

The Type property of a Window object returns one of the following:

visDrawing = 1
visStencil = 2
visSheet = 3
visIcon = 4

If a Window object is type visDrawing, use the SubType property to determine the type of drawing window represented by the object.

See also: SubType property

Example for OpenIconWindow, OpenSheetWindow, OpenStencilWindow, Type

```
'This VBA macro demonstrates using the Type property.

Public Sub TypeProp_Example()

  Dim masterObj As Visio.Master
  Dim shpObj As Visio.Shape
  Dim shpGroupObj As Visio.Shape
  Dim shpGuideObj As Visio.Shape
  Dim winStenObj As Visio.Window
  Dim winSheetObj As Visio.Window
  Dim winIconObj As Visio.Window

  'Create a group.
  Set shpObj = ActivePage.DrawRectangle(1, 1, 2, 3)
  shpObj.Duplicate
  ActiveWindow.SelectAll
  ActiveWindow.Selection.Group

  'Get three different types of shape objects: a group, a shape, and a guide.
  Set shpGroupObj = ActivePage.Shapes(1)
  Set shpObj = ActivePage.DrawRectangle(1, 4, 2, 6)
  Set shpGuideObj = ActivePage.AddGuide(visVert, 4, 0)

  'Use the Type method to verify each shape's type.
  Debug.Print shpGroupObj.Type                      'visTypeGroup = 2
  Debug.Print shpObj.Type                           'visTypeShape = 3
  Debug.Print shpGuideObj.Type                      'visTypeGuide = 5

  'Get the document's stencil window.
  Set winStenObj = ThisDocument.OpenStencilWindow

  'Add a master and get it's Icon window.
  Set masterObj = ThisDocument.Masters.Add
  Set winIconObj = masterObj.OpenIconWindow

  'Get the ShapeSheet window of shpObj.
  Set winSheetObj = shpObj.OpenSheetWindow

  'Use the Type property to verify each window's type.
  Debug.Print winStenObj.Type                       'visStencil = 2
  Debug.Print winSheetObj.Type                      'visSheet = 3
  Debug.Print winIconObj.Type                       'visIcon = 4

End Sub
```

TypeSpecific1 property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the type of a toolbar or status bar item.

Version: VISIO 4.0

Syntax: object.**TypeSpecific1** = intVal
intVal = object.**TypeSpecific1**

Element	Description
object	The StatusBarItem or ToolbarItem object that has or gets the type
intVal	The type of the toolbar or status bar item

Remarks: The value of an object's TypeSpecific1 property depends on the value of its CntrlType property. If CntrlType is any of the following, TypeSpecific1 can be any constant prefixed with visIconIX that is declared by the Visio type library (and visconst.bas):

visCtrlTypeBUTTON
visCtrlTypeSTATE_BUTTON
visCtrlTypeHIERBUTTON
visCtrlTypeSTATE_HIERBUTTON
visCtrlTypeDROPBUTTON
visCtrlTypeSTATE_DROPBUTTON
visCtrlTypeSPINBUTTON

If CntrlType is any of the following, TypeSpecific1 is 0:

visCtrlTypeEDITBOX
visCtrlTypeCOMBOBOX
visCtrlTypeCOMBODRAW
visCtrlTypeLISTBOX
visCtrlTypeLISTBOXDRAW

If CntrlType is visCtrlTypeCOLORBOX, TypeSpecific1 is an integer index into Visio's color table.

If CntrlType is visCtrlTypeLABEL, TypeSpecific1 can be any constant prefixed with visStrID that is declared by the Visio type library (and visconst.bas).

If CntrlType is visCtrlTypeMESSAGE, TypeSpecific1 can be any constant prefixed with visCtrlAlignment that is declared by the Visio type library (and visconst.bas).

See also: CntrlID property, CntrlType property, TypeSpecific2 property

Example for TypeSpecific1

TypeSpecific2 property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the type of a toolbar or status bar item.

Version: VISIO 4.0

Syntax: object.**TypeSpecific2** = intVal
intVal = object.**TypeSpecific2**

Element	Description
object	The StatusBarItem or ToolbarItem object that has or gets the type
intVal	The type of the toolbar or status bar item

Remarks: The value of an object's TypeSpecific2 property depends on the value of its CntrlType property. If CntrlType is any of the following, TypeSpecific2 can be an integer to group buttons together, or 0.

visCtrlTypeBUTTON
visCtrlTypeSTATE_BUTTON
visCtrlTypeHIERBUTTON
visCtrlTypeSTATE_HIERBUTTON
visCtrlTypeDROPBUTTON
visCtrlTypeSTATE_DROPBUTTON
visCtrlTypeSPINBUTTON

If CntrlType is any of the following, TypeSpecific2 represents the minimum and maximum width of the control expressed in number of characters.

visCtrlTypeEDITBOX
visCtrlTypeCOMBOBOX
visCtrlTypeCOMBODRAW
visCtrlTypeLISTBOX
visCtrlTypeLISTBOXDRAW
visCtrlTypeMESSAGE

For example, if the minimum width is 10 characters and the maximum width is 20 characters, calculate the value of TypeSpecific2 as follows:

(20 = 0x14, 10 = 0x0A) = 0x140A = 5130 decimal

If CntrlType is any of the following, TypeSpecific2 is not used:

visCtrlTypePUSHBUTTON
visCtrlTypeCOLORBOX
visCtrlTypeLABEL

See also: CntrlID property, CntrlType property, TypeSpecific1 property

Example for TypeSpecific2

Undo method

Applies to: Application

Summary: Reverses the most recent action, if the action can be reversed.

Version: VISIO 2.0

Syntax: object.**Undo**

Element	Description
object	The Application object in which to reverse the action

Remarks: You can reverse actions, one action at a time. The number of undo actions depends on the number set in the VISIO.INI file (10 is the default). You can undo most actions, but not all. Use the Redo method to reverse the effect of the Undo method.

See also: Redo method, PurgeUndo method

Example for Undo

```
*Redo Method
```

Ungroup method

Applies to: Selection, Shape

Summary: Ungroups a group.

Version: VISIO 2.0

Syntax: object.**Ungroup**

Element	Description
object	The Shape or Selection object to ungroup

See also: AddToGroup method, Group method, RemoveFromGroup method

Example for Ungroup

```
'This VBA macro demonstrates using the Ungroup method.

Sub Ungroup_Example ()

  Dim shp1Obj As Visio.Shape
  Dim shp2Obj As Visio.Shape
  Dim shp3Obj As Visio.Shape
  Dim mstObj As Visio.Master

  Set shp1Obj = ActivePage.DrawRectangle(1, 2, 2, 1)
  Set shp2Obj = ActivePage.DrawRectangle(1, 4, 2, 3)

  ActivePage.Drop shp1Obj, 3.5, 3.5
  Set shp3Obj = ActivePage.Shapes(3)

  Set mstObj = ThisDocument.Drop(shp3Obj, 0, 0)

  ActiveWindow.Select shp1Obj, visSelect
  ActiveWindow.Select shp2Obj, visSelect
  ActiveWindow.Group

  Set shp1Obj = ActivePage.Shapes(2)
  shp1Obj.Drop shp3Obj, 3, 3
  shp1Obj.Ungroup

End Sub
```

Union method

Applies to: Selection, Window

Summary: Creates a new shape from the perimeter of selected shapes.

Version: VISIO 2.0

Syntax: object.**Union**

Element	Description
object	The Window or Selection object that contains the shapes to unite

Remarks: The Union method is equivalent to choosing the Union command from the Operations submenu on the Shape menu in Visio. The produced shape will be the topmost shape in its ContainingShape and will inherit the text and formatting of the first selected shape. The original shapes are deleted.

If the object being operated on is a Selection object, it will have no shapes selected in it when the operation is complete.

See also: Combine method, Fragment method, Intersect method, Join method, Subtract method, Trim method, ContainingShape property

Example for Union

UniqueID property

Applies to: Master, Shape

Summary: Returns or clears the unique ID of the indicated object.

Version: VISIO 4.0

Syntax: strRet = object.**UniqueID**
strRet = object.**UniqueID**(flag)

Element	Description
strRet	The unique id of the Master or Shape object
object	The Master or Shape object that has the unique ID
flag	Gets, assigns, or clears the unique ID of a Shape object

Remarks: In Visio, a Shape or Master object can have a unique ID. If a shape has a unique ID, you can reliably assume that no other shape in the same or any other document also has that ID. The same is true for a master.

Every master has a unique ID. You can determine this ID using:

```
idStr = mastObj.UniqueID
```

The value it returns is a string in the form:

```
{2287DC42-B167-11CE-88E9-0020AFDDD917}
```

By default, a shape does not have a unique ID. A shape gains a unique ID only if its UniqueID property is set.

The flag parameter controls the behavior of UniqueID. It should have one of the following values:

```
visGetGUID = 0
visGetOrMakeGUID = 1
visDeleteGUID = 2
```

shpObj.UniqueID(visGetGUID) returns the unique ID string like the one shown above only if the shape already has a unique ID. Otherwise it returns a null string ("").

shpObj.UniqueID(visGetOrMakeGUID) returns the unique ID string of the shape. If the shape does not yet have a unique ID, it assigns one to the shape and returns the new ID.

shpObj.UniqueID(visDeleteGUID) clears the unique ID of the shape and returns a null string ("").

Given the unique id of a shape you can access that shape using Shapes.Item(uniqueIDString).

Given the unique id of a master you can access that master using Masters.Item(uniqueIDString).

See also: EventInfo property, Item property, Masters object, Shapes object

Example for UniqueID

Units property

Applies to: Cell

Summary: Indicates the unit of measure associated with a Cell object.

Version: VISIO 3.0

Syntax: intRet = object.**Units**

Element	Description
object	The Cell object to examine
intRet	The units associated with a cell's current value

Remarks: This property can be used to determine the unit of measure currently associated with a cell's value. The various unit codes are declared by the Visio type library (and visconst.bas). For example, a cell's width might be expressed in inches (visInches) or in centimeters (visCentimeters). In some cases a program might want to behave differently depending on whether a cell's value is in metric or in English units.

Example for Units

UpdateUI method

Applies to: UI Object

Summary: Causes Visio to display changes to the user interface represented by a UI object.

Version: VISIO 4.0

Syntax: object.**UpdateUI**

Element	Description
object	The UI object that represents the user interface that was changed

Remarks: The UpdateUI method updates the Visio user interface with changes made to a UI object during a session. Use the CustomMenus or CustomToolbars property of an Application object or Document object to obtain the UI object initially.

See also: CustomMenus property, CustomToolbars property

Example for UpdateUI

UserName property

Applies to: Application

Summary: Gets or sets the user name of an Application object.

Version: VISIO 4.0

Syntax: strRet = object.**UserName**
object.**UserName** = strExpression

Element	Description
strRet	The current user name
object	The Application object that has or gets the user name
strExpression	The new user name

Remarks: The UserName property corresponds to the User Name option in Visio's Options dialog box.

Example for UserName

Value property

Applies to: Attribute

Summary: Returns or sets the value of an Attribute object.

Version: VISIO 3.0 TECH

Syntax: strRet = object.**Value**
object.**Value** = strValue

Element	Description
strRet	The current value of the attribute
object	The Attribute object that has or gets the value
strValue	The new value of the attribute

Example for Value

```
*PutShape Method
```

VBE property

Applies to: <Global>, Application

Summary: Returns root object of object model exposed by Visual Basic for Applications.

Version: VISIO 4.5

Syntax: objRet = object.**VBE**

Element	Description
objRet	Programmable object that exposes VBA methods and properties
object	The Application object that has the setting

Remarks: Through this property you can access and manipulate the VBA projects associated with currently open Visio documents.

You can get information about the object returned by the VBE property by following these steps:

1) Open the Visual Basic Editor from Visio.
2) Select "Microsoft Visual Basic for Applications Extensibility" in the References dialog box located on the Tools menu.
3) Open the VBA Object Browser and view the type library named VBIDE.

See also: VBProject property

Example for VBE

```
'This VBA macro displays a message box that shows how many VBA projects are
open
'in an instance of Visio.

Public Sub VBE_Example ()

  MsgBox (Visio.Application.VBE.VBProjects.Count)

End Sub
```

VBProject property

Applies to: Document

Summary: Returns programmable object through which the VBA project of the document can be controlled.

Version: VISIO 4.5

Syntax: objRet = object.**VBProject**

Element	Description
objRet	Programmable object that exposes methods and properties of the document's VBA project
object	The Document object that has the setting

Remarks: You can get information about the object returned by the VBProject property by following these steps:

1) Open the Visual Basic Editor from Visio.
2) Select "Microsoft Visual Basic for Applications Extensibility" in the References dialog box located on the Tools menu.
3) Open the VBA Object Browser and view the type library named VBIDE.
4) Examine the class named VBProject.

See also: VBE property

Example for VBProject

```
'This VBA macro demonstrates printing the names of libraries referenced by a
'VBA project in the Immediate window.

Public Sub ShowThisProjectsRefs_Example ()

  Dim thisProject As VBProject
  Dim nrefs As Integer

  Set thisProject = ThisDocument.VBProject

  nrefs = thisProject.References.Count
  While nrefs > 0
    Debug.Print thisProject.References(nrefs).Name
    nrefs = nrefs - 1
  Wend

End Sub
```

VectorX property

Applies to: Entity

Summary: Specifies the X component of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**VectorX**
object.**VectorX** = Expression

Element	Description
RetVal	The current X component of a vector as a double
object	The Entity object that has or gets the X component
Expression	The new X component of the vector as a double

Remarks: If an Entity object has a Group of 1010, 1020, 1030, 1011, 1021, 1031, 1012, 1022, or 1032, then it contains vector/point information. The vector/point information can be manipulated using the VectorX, VectorY, and VectorZ properties. Note that Visio does not support the automatic scaling of any vector or point data at this time.

Notice that all three components of a vector are stored using one Entity, not three. Therefore, you should use a Group of 1010 for points/vectors, 1011 for world space position vectors, 1012 for world displacement vectors, and 1013 for world direction vectors.

See also: VectorY property, VectorZ property

Example for VectorX

VectorY property

Applies to: Entity

Summary: Specifies the Y component of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**VectorY**
object.**VectorY** = Expression

Element	Description
RetVal	The current Y component of the vector as a double
object	The Entity object that has or gets the Y component
Expression	The new Y component of the vector as a double

Remarks: If an Entity object has a Group of 1010, 1020, 1030, 1011, 1021, 1031, 1012, 1022, or 1032, then it contains vector/point information. The vector/point information can be manipulated using the VectorX, VectorY, and VectorZ properties. Note that Visio does not support the automatic scaling of any vector or point data at this time.

Notice that all three components of a vector are stored using one Entity, not three. Therefore, you should use a Group of 1010 for points/vectors, 1011 for world space position vectors, 1012 for world displacement vectors, and 1013 for world direction vectors.

See also: VectorX property, VectorZ property

Example for VectorY

VectorZ property

Applies to: Entity

Summary: Specifies the Z component of an Entity object.

Version: VISIO 3.0 TECH

Syntax: RetVal = object.**VectorZ**
object.**VectorZ** = Expression

Element	Description
RetVal	The current Z component of the vector as a double
object	The Entity object that has or gets the Z component
Expression	The new Z component of the vector as a double

Remarks: If an Entity object has a Group of 1010, 1020, 1030, 1011, 1021, 1031, 1012, 1022, or 1032, it contains vector/point information. The vector/point information can be manipulated using the VectorX, VectorY, and VectorZ properties. Note that Visio does not support the automatic scaling of any vector or point data at this time.

Notice that all three components of a vector are stored using one Entity, not three. Therefore, you should use a Group of 1010 for points/vectors, 1011 for world space position vectors, 1012 for world displacement vectors, and 1013 for world direction vectors.

See also: VectorX property, VectorY property

Example for VectorZ

Version property

Applies to: Application, Document

Summary: Returns the version of a running instance of Visio or determines the version of a saved document.

Version: VISIO 2.0

Syntax:

strRet = appObject.**Version**
intRet = docObject.**Version**
docObject.**Version** = intExpression

Element	Description
strRet	Visio's major and minor version numbers
appObject	The Application object to examine
intRet	The file format version the document is saved in
docObject	The Document object that has or gets the setting
intExpression	The file format version in which to save the document

Remarks: Use this property to verify the version of a particular instance of Visio. This information is helpful if your program requires a particular version. Both the major and minor version numbers are returned. The string returned by Visio 4.5 is "4.5".

Setting the Version property of a document tells Visio which file format version to save the document in the next time the document is saved. To set the file version number, it's easiest to use hexadecimal notation. For example, docObj.Version = &H20000.

Constants for file format versions are declared by the Visio type library (and visconst.bas). Visio 4.5 can save the following versions:

visVersion20&	=&H20000	Save as a Visio 2.0 document.
visVersion30&	=&H30003	Save as a Visio 3.0 document.
visVersion40&	=&H40000	Save as a Visio 4.0 document.

When Visio is about to save a document in a prior version format, it always displays an alert that requires the user to confirm the operation.

Visio 4.x always reports the version of a document it opens as &H40000. This is true even if the opened document was last saved as a prior version format, because Visio 4.x converts the in-memory representation of every document it opens to 4.0 format.

The version of document that hasn't been saved yet will be reported as 0.

Example for Version

```
'This VB program demonstrates printing the version of a Visio instance in the
VB
'debug window.

Public Sub ShowVersion_Example ()

  Dim appVisio As Visio.Application
  Dim strVer As String
  Dim iDotPos As Integer

  Set appVisio = CreateObject("visio.application")

  strVer = appVisio.Version
  iDotPos = InStr(strVer, ".")

  Debug.Print "  Major Version : "; Left(strVer, iDotPos - 1)
  Debug.Print "  Minor Version : "; Right(strVer, Len(strVer) - iDotPos)

End Sub
```

WindowActivated event

Applies to: Application, Window, Windows

Summary: The event that occurs after the active window changes in a Visio instance.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventCode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtApp+visEvtWinActivate (&H1080)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events in this instance
subject	The Application object in which this event occurred
moreInfo	Nothing for this event

Remarks: The WindowActivated event indicates that the active window has changed in a Visio instance. This event implies that the ActiveDocument and ActivePage properties of the Application object may also have changed, but not necessarily; in contrast, any time the ActiveDocument or ActivePage property changes, WindowActivated event is always generated.

For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, ActiveDocument property, ActivePage property, Add method, AddAdvise method, Event object, EventList object

Example for WindowActivated

WindowHandle property

Applies to: Application, Window

Summary: Returns the (16-bit) handle of a Visio window.

Version: VISIO 2.0

Syntax: retVal = object.**WindowHandle**

Element	Description
retVal	The HWND of the object's window
object	The object to get the window handle of

Remarks: Use AppObj.WindowHandle to obtain the HWND for Visio's main (frame) window.

Use WinObj.WindowHandle to obtain the HWND for a window in the Windows collection of an instance of Visio.

You can use the obtained HWND in Windows API calls.

WindowHandle returns a 2-byte value. This is proper in Win16 where handles are 2-byte values. In Win32 handles are 4-byte values.

If you are dealing with the Win16 version of Visio, then WindowHandle returns the true handle of the instance.

If you are dealing with the Win32 version of Visio, then WindowHandle returns its 4-byte handle cast into the 2-byte value returned.

By observation, it appears that using the 2-byte value returned by WindowHandle is always valid, regardless of which of the 4 possible Visio Controller/Visio instance combinations is in effect.

You can determine which type of Visio instance you're dealing with by using IsVisio16 or IsVisio32. If you are dealing with Visio32 and would prefer to obtain a 4-byte handle, use WindowHandle32.

See also: InstanceHandle property, InstanceHandle32 property, IsVisio16 property, IsVisio32 property, WindowHandle32 property

Example for WindowHandle

WindowHandle32 property

Applies to: Application, Window

Summary: Returns the (32-bit) handle of a Visio window.

Version: VISIO 4.0

Syntax: retVal = object.**WindowHandle32**

Element	Description
retVal	The HWND of the object's window
object	The object to get the window handle of

Remarks: Use AppObj.WindowHandle32 to obtain the HWND for Visio's main (frame) window.

Use WinObj.WindowHandle32 to obtain the HWND for a window in the Windows collection of an instance of Visio.

You can use the obtained HWND in Windows API calls.

WindowHandle32 returns a 4-byte value. See also WindowHandle which returns a 2-byte value. (In Win16 handles are 2-byte values. In Win32 they're 4-byte values.)

If the Visio instance being dealt with is an instance of 16-bit Visio, WindowHandle32 returns 0. You can determine the type of Visio instance you're dealing with by using IsVisio16 or IsVisio32.

By observation, it appears that using the 2-byte value returned by WindowHandle is always valid, regardless of which of the 4 possible Visio Controller/Visio instance combinations is in effect.

See also: InstanceHandle property, InstanceHandle32 property, IsVisio16 property, IsVisio32 property, WindowHandle property

Example for WindowHandle32

WindowOpened event

Applies to: Application, Windows

Summary: The event that occurs after a Visio window is opened.

Version: VISIO 4.1

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtAdd+visEvtWindow (&H8001)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Window that just opened
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, Window object

Example for WindowOpened

Windows property

Applies to: <Global>, Application

Summary: Returns the Windows collection for an instance of Visio.

Version: VISIO 2.0

Syntax: objRet = object.**Windows**

Element	Description
objRet	The Windows collection of the Application object
object	The Application object that owns the collection

See also: Window object, Windows object

Example for Windows

WindowTurnedToPage event

Applies to: Application, Window, Windows

Summary: The event that occurs after a window shows a different page in itself.

Version: VISIO 4.5

Syntax: object.VisEventProc(eventcode,source,id,sequence,subject,moreInfo)

Element	Description
object	The event sink object passed with AddAdvise when this Event object was created
eventCode	visEvtCodeWinPageTurn (704)
source	The Visio object whose EventList contains the Event object
id	The ID of the Event object in the source object's EventList
sequence	The ordinal position of this event relative to past events
subject	The Window that is now showing a different page
moreInfo	Nothing for this event

Remarks: For an action to be triggered by this event, the EventList of the source object must contain an Event object with the eventCode shown above.

The syntax describes the signature of the method Visio will call if the Action property of the Event object is visActCodeAdvise. To create this kind of Event object, use the AddAdvise method of an EventList object.

If the Action property of the Event object is visActCodeRunAddon, Visio runs the add-on named by the Target property of the Event object and sends it a command string that identifies the subject of the event, among other things. To create this kind of Event object, use the Add method of an EventList object.

See also: Action property, Add method, AddAdvise method, Event object, EventList object, BeforeWindowPageTurn event

Example for WindowTurnedToPage

Zoom property

Applies to: Window

Summary: Returns or sets the current display size (magnification factor) for a page in a window.

Version: VISIO 2.0

Syntax: retVal = object.**Zoom**
object.**Zoom** = newZoom

Element	Description
retVal	The current display size for the window
object	The Window object that has or gets the display size
newZoom	The new display size for the window

Remarks: Valid values range from 0.05 to 9.99 (5% to 999%). The value -1 fits the page into the window.

Example for Zoom

Automation Reference Glossary

Argument
Collection
Connection
Constant
Document
Method
Object
Program
Property
Return Value
Shape
Sheet

Object
An item in Visio that you can control from a program. An object has attributes called properties whose values you can set or retrieve. An object also has methods that you can invoke to make the object perform an action.

Collection
An object that includes one or more other objects, almost always of the same type. For example, the Documents collection includes all open documents in the current instance of Visio. A collection differs from an array in that the position of a given object may change when an operation affecting the collection is carried out.

Method
An action that can be performed by an object. For example, Window objects have an **Activate** method, which can be used to activate a specific window. A method usually returns the value that results from its action. For example, a method that creates a new object returns the new object.

Property
An attribute of an object that defines its behavior or appearance. For example, every collection has a **Count** property whose value is the number of elements in the collection; a Shape object's **OneD** property determines whether the shape behaves as a 1-D or 2-D shape. Some properties are read-only, which means they can return a value but cannot be set. Other properties are write-only, which means they can be set but don't return a value.

Argument
An item that is passed with a method or property to supply additional information for the requested action. For example, the Open method's *stringExpression* argument specifies the filename of the document to open.

Constant
A declared item whose value cannot change during the course of a program's execution. Visio constants are included in the file VISCONST.BAS.

Return value
The value returned by a method, property, or function. A return value may indicate whether an operation was successful. For example, the return value for opening a file might be 0 if the file could not be opened. Return values may also include objects, strings, or integers.

Program
Executable code that controls Visio using OLE Automation.

Document
A Visio file. All Visio files have the same format and can contain the same data. The file extension determines how Visio opens the file and what it displays.

Shape
Anything that can be selected in Visio with the pointer tool, including a shape drawn in Visio, a group, a guide or guide point, a linked or embedded object, or an object imported from another application.

Sheet
A synonym for shape. Internally, a shape is defined in a spreadsheet similar to that displayed in a ShapeSheet window. Some programming language terms derive their names from this internal spreadsheet. For example shpObj.**Connects**(1).ToSheet returns a reference to a Shape object that is connected to shpObj. Visio also uses sheet as the default name for any shape that is not an instance of a named master.

Connection
The relationship between two shapes that are connected, represented by a Connect object in a program. The term connector is used to refer to certain masters used to draw lines in diagrams such as flowcharts and organization charts. A connector has no special role in a connection¾it behaves no differently from any other 1-D shape.

Syntax conventions

The description for each property and method shows the syntax for using that property or method. Syntax follows this general format:

return value = object.**keyword** (argument1,...)

Syntax element	Description
return value	The variable to receive the value returned by the property or method. In syntax examples, variable names indicate the type of value returned, for example: retVal Generic value (Variant) intRet Integer strRet String objRet Object objsRet Collection
object	An object variable that represents the object to be acted on.
keyword	The name of the property or method.
argument1,...	One or more arguments that can be passed with the property or method. The argument descriptions give the argument's type and possible values. Constants defined for Visio may be used as arguments for some methods or properties. Valid constants for a method or property are listed in the help topic for that method or property. All constants are included in the file VISCONST.BAS provided with Visio.
[keyword]	A keyword in brackets indicates a property or method that is also a Visual Basic keyword. Use brackets to distinguish the property or method from the Visual Basic keyword.

Compound statements

Objects, keywords, and arguments may be concatenated in compound statements. For example:

Documents(1).Pages(3).Shape(1)

returns the first shape on the third page of the first document in the current instance of Visio.

Visio Type Library

Visio version 5.0 products include a type library that contains Automation descriptions of the objects, properties, methods, events, and constants that Visio exposes to Automation controllers.

A type library is useful for several reasons:

- The information in a type library serves as input to object browsers supplied by Visual Basic for Applications (VBA) and other development environments. You can use object browsers to view the Automation descriptions for Automation servers (such as Visio) installed on your system. For example, you can view the syntax for a Visio property, method, or event.
- A type library allows development environments to bind your program's code to Automation server code at compile (design) time rather than dynamically at runtime. The result is that your program will often run faster. For example, you can use Visio object types instead of general variable types. You can declare a variable as *Visio.Page*, *Visio.Shape*, *Visio.Document*, and so on instead of *Object*.
- You can copy and paste code templates from object browsers into your program. For example, you can view the syntax for a particular property, copy the code template from the object browser, paste it into your program, and fill in the actual values.

Referencing the Visio type library

To use the Visio type library, a development environment must reference it. The VBA project of a Visio document automatically references the Visio type library. In other development environments you must take appropriate steps to reference the library.

To reference the Visio type library in Visual Basic:

1. From the Tools menu, choose References.
2. In the Available References list, select the Visio type library, then click OK.

   Note: Visio automatically registers its type library with Windows the first time it runs. If you do not see the Visio type library in the Available References list, run Visio, exit Visio, and then follow these steps again.

Viewing Visio Automation descriptions

To browse Visio objects, properties, methods, events, and constants from VBA, choose Object Browser from the View menu. The Object Browser initially displays items declared by all libraries referenced by your project. To display only Visio items, select Visio in the Project/Library list. You can also view additional information about an item by selecting the item and clicking Help.

Using Visio object types

Earlier versions of Visio didn't include a Visio type library; you defined an object variable as an Object and used it to hold a reference to a Visio object. For example:

```
Dim docObj as Object
Dim pagObj as Object
Dim shpObj as Object
```

By using Visio object types declared in the Visio type library, you can declare variables as specific types. For example:

```
Dim docObj As Visio.Document
Dim pagObj As Visio.Page
Dim shpObj As Visio.Shape
```

For a list of Visio object types, view the Class names in the Classes list in the Object Browser.

Note: Programs using Visio object types will fail if you run them with versions of Visio older than version

4.5 because earlier versions of Visio did not include a type library. With earlier versions of Visio you must use general object types.

Resolving object name ambiguities

Your VBA or Visual Basic program can reference many type libraries. Libraries will sometimes declare items with the same name. For example, both Visio and Excel expose an object called Application. When more than one library declares an item with the same name, VBA and Visual Basic bind the name to the library with the highest priority.

Note: The names of the libraries your project references are displayed in the Project/Library list in the Object Browser.

To resolve object name ambiguities, you can change the priority of libraries in the Object Browser or prefix object types with the corresponding library name. For details on changing the priority of libraries, see the online Microsoft Visual Basic Help.

The best way to resolve name ambiguities is to prefix object types with the corresponding library name. For example:

```
Dim visioObj As Visio.Application
Dim excelObj As Excel.Application
```

If your code runs exclusively in the context of a VBA project of a Visio document, you don't have to prefix names of Visio object types with *Visio*, although it is a good idea, because the Visio type library has a higher priority than other libraries that may declare conflicting names. VBA will not let you change the priority of the Visio type library when you are using VBA within Visio, but in other development environments you can change the priority of the Visio type library.

ThisDocument Object

The VBA project of every Visio document has a class module called ThisDocument. When referenced from code in the project, ThisDocument returns a reference to the project's Document object. For example, you can display the name of the VBA project's document in a message box with this statement:

```
MsgBox ThisDocument.Name
```

You can get the first page of the VBA project's document by using this code:

```
Dim pagObj as Visio.Page
Set pagObj = ThisDocument.Pages.Item(1)
```

If you want to manipulate the document associated with your VBA project, use the ThisDocument object. If you want to manipulate a document, but not necessarily the document associated with your VBA project, get the Document object from the Documents collection.

The ActiveDocument property will often, but not necessarily, return a reference to the same document as ThisDocument. The ActiveDocument and ThisDocument are the same if the document shown in the Visio active window is the document containing ThisDocument's project. Whether your code uses ActiveDocument or ThisDocument depends on the purpose of your program.

Note: ThisDocument is not available to code that isn't part of the VBA project of a Visio document.

Extending ThisDocument's properties and methods

You can extend the set of properties and methods of a project's Document object by adding public properties and methods to that project's ThisDocument class module. The new methods and properties are exposed just like the built-in methods and properties implemented by Visio. The new methods and properties aren't available when you reference other Document objects.