Visio OLE Automation Reference

Visio online help

Objects

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

Methods
Properties
Shapesheet Cells
Syntax Conventions

Close

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 in VISCONST.BAS prefixed with "visCmd".

Methods
Delete

Properties
Alt
CmdNum
Control
Key
Parent
Shift

Close

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.

Methods
Add

Properties
Count
Item
Parent

Close

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).

Methods
Delete

Properties
AccelItems
Parent
SetID
TableName

Close

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. Valid SetIDs for AccelTable objects are:

visUIObjSetNoDocument
visUIObjSetDrawing
visUIObjSetStencil
visUIObjSetShapeSheet
visUIObjSetIcon
visUIObjSetInPlace
visUIObjSetPrintPreview
visUIObjSetBinderInPlace

Methods
Add
AddAtID

Properties
Count
Item
ItemAtID
Parent

Close

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 OLE 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.

Methods
Run

Properties
Application
Enabled
Index
Name

Close

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.

Methods
Add

Properties
Application
Count
Item

Close

Application object

An Application object represents an instance of Visio. A program must 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. 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.

Methods
ClearCustomMenus
ClearCustomToolbars
Quit
Redo
SaveWorkspaceAs
SetCustomMenus
SetCustomToolbars
Undo

Properties
ActiveDocument
ActivePage
ActiveWindow
AddonPaths
Addons
Application
BuiltInMenus
BuiltInToolbars
CustomMenus
CustomMenusFile
CustomToolbars
CustomToolbarsFile
Documents
DrawingPaths
EventInfo
FilterPaths
HelpPaths
InstanceHandle
InstanceHandle32
IsVisio16
IsVisio32
Language
OnDataChangeDelay
ProcessID
ProfileName
PromptForSummary
ScreenUpdating
StartupPaths
StencilPaths

TemplatePaths
UserName
Version
WindowHandle
WindowHandle32
Windows

Close

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.

Visio 4.0 stores 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.

Methods
Delete

Properties
DefaultValue
Name
Prompt
Value

Close

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.

Visio 4.0 stores 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.

Methods
Add

Properties
Count
Item

Close

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.

Methods
GlueTo
GlueToPos
Trigger

Properties
Application
Column
Document
Error
Formula
FormulaForce
IsConstant
IsInherited
LocalName
Name
Result
ResultForce
ResultInt
ResultIntForce
ResultIU
ResultIUForce
ResultStr
Row
RowName
Section
Shape
Style
Units

Close

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.

Methods
AddCustomField
AddField
CharProps
Copy
Cut
ParaProps
Paste

Properties
Application
Begin
CharCount
CharPropsRow
Document
End
FieldCategory
FieldCode
FieldFormat
FieldFormula
IsField
ParaPropsRow
RunBegin
RunEnd
Shape
TabPropsRow
Text

Close

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.

Properties

Application
Blue
Document
Flags
Green
Index
PaletteEntry
Red

Close

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.

Properties

Application
Count
Document
Item

Close

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 collection of a Shape 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.

Properties

Application
Document
FromCell
FromPart
FromSheet
Index
ToCell
ToPart
ToSheet

Close

Connects object

A Connects collection includes a Connect object for each shape, group, or guide to which a shape is connected. Use the Connects property of a Shape object to retrieve its Connects collection. The default property of a Connects collection is Item.

Properties

Application
Count
Document
FromSheet
Item

Close

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.

Methods
ClearCustomMenus
ClearCustomToolbars
Close
Drop
Print
Save
SaveAs
SaveAsEx
SetCustomMenus
SetCustomToolbars

Properties
Application
Colors
Creator
CustomMenus
CustomMenusFile
CustomToolbars
CustomToolbarsFile
Description
EventList
Fonts
FullName
Index
InPlace
Keywords
Masters
Name
Pages
Path
ReadOnly
Saved
SavePreviewMode
Styles
Subject
Template
Title
Version

Close

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.

Methods
Add
Open
OpenEx

Properties
Application
Count
Item

Close

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.

Methods
Add

Properties
Count
Item
Name

Close

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.

Methods
Delete

Properties
Entities
Name

Close

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.

Methods
Add

Properties
Count
Item

Close

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.

Methods
Delete

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

Close

EventList object

The EventList collection includes an Event object for each event that is defined for a Document object. Use the EventList property of a Document object to retrieve its EventList collection. The default property of EventList is Item.

Methods
Add

Properties
Application
Count
Item

Close

Event object

An Event object is a member of the EventList collection of an object such as a Document. An Event encapsulates an event code, action code pair. When the event happens to the object, the action is performed.

The Target property of the Event object defines the target of the action, and the TargetArgs property defines the arguments, if any, that are sent to the target when the action is performed. For example, if the action is Run Add-on, the Target property contains the name of the add-on to run, and the TargetArgs property contains the argument string to be passed to the add-on.

The default property of an Event object is Event.

Methods
Delete
Trigger

Properties
Action
Application
Event
EventList
Index
Target
TargetArgs

Close

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.

Properties

Application
Attributes
CharSet
Document
ID
Index
Name
PitchAndFamily

Close

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.

Properties
Application
Count
Document
Item
ItemFromID

Close

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.

Methods
Add
Delete
Remove

Properties
Application
CellsC
Document
Index
Master
Name
Page
Row

Close

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.

Methods
Add

Properties
Application
Count
Document
Item
Master
Page

Close

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.

Methods
Delete

Properties
AlignName
Application
Document
IconSize
IconUpdate
Index
Layers
Name
OneD
PageSheet
Prompt
Shapes
UniqueID

Close

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.

Properties

Application
Count
Document
Item

Close

MenuItem object

A MenuItem object represents a single menu item on a Visio menu, such as the New menu item on the File 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.

Methods
Delete

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

Close

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.

Methods
Add
AddAt

Properties
Count
Item
Parent

Close

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.

Methods
Delete

Properties
Caption
Index
MDIWindowMenu
MenuItems
Parent

Close

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.

Methods
Delete

Properties
Caption
Menus
Parent
SetID

Close

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. Valid SetIDs for MenuSet objects are:

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

Methods
Add
AddAtID

Properties
Count
Item
ItemAtID
Parent

Close

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.

Methods
Add
AddAt

Properties
Count
Item
Parent

Close

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.

Methods
AddGuide
CenterDrawing
Delete
DrawLine
DrawOval
DrawRectangle
Drop
Export
Import
Paste
Print

Properties
Application
Background
BackPage
Document
Index
Layers
Name
PageSheet
Shapes

Close

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.

Methods
Add

Properties
Application
Count
Document
Item

Close

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.

Methods
BringForward
BringToFront
ConvertToGroup
Copy
Cut
Delete
DeselectAll
Duplicate
Export
FlipHorizontal
FlipVertical
Group
ReverseEnds
Rotate90
Select
SelectAll
SendBackward
SendToBack
Ungroup

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

Close

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.

Methods
BeginTransaction
EndTransaction
PutShape

Properties
Attributes
EntityApps
Shape

Close

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.

Methods

AddNamedRow
AddRow
AddRows
AddSection
BringForward
BringToFront
ConvertToGroup
Copy
Cut
Delete
DeleteRow
DeleteSection
Drop
Duplicate
Export
FlipHorizontal
FlipVertical
Group
ReverseEnds
Rotate90
SendBackward
SendToBack
SetBegin
SetCenter
SetEnd
Ungroup

Properties

Application
AreaIU
CellExists
Cells
CellsSRC
CellsSRCExists
Characters
CharCount
Connects
ContainingMaster
ContainingPage
ContainingShape
Data1
Data2
Data3
Document

FillStyle
FillStyleKeepFmt
GeometryCount
Help
Index
Layer
LayerCount
LengthIU
LineStyle
LineStyleKeepFmt
Master
Name
NameID
OneD
Parent
RowCount
RowExists
RowsCellCount
RowType
SectionExists
Shapes
Style
StyleKeepFmt
Text
TextStyle
TextStyleKeepFmt
Type
UniqueID

Close

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.

Properties
Application
ContainingMaster
ContainingPage
ContainingShape
Count
Document
Item

Close

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.

Methods
Delete

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

Close

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.

Methods
Add
AddAt

Properties
Count
Item
Parent

Close

StatusBar object

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

Methods
Delete

Properties
Caption
Parent
SetID
StatusBarItems

Close

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. Valid SetIDs for StatusBar objects are:

visUIObjSetNoDocument
visUIObjSetDrawing
visUIObjSetStencil
visUIObjSetShapeSheet
visUIObjSetIcon
visUIObjSetInPlace
visUIObjSetPrintPreview
visUIObjSetText
visUIObjSetBinderInPlace

Methods
Add
AddAtID

Properties
Count
Item
ItemAtID
Parent

Close

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.

Methods
Delete

Properties
Application
BasedOn
Cells
Document
FillBasedOn
IncludesFill
IncludesLine
IncludesText
Index
LineBasedOn
Name
TextBasedOn

Close

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.

Methods
Add

Properties
Application
Count
Document
Item

Close

ToolbarItem object

A ToolbarItem object represents one item in a toolbar row. 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.

Methods
Delete

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

Close

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.

Methods
Add
AddAt

Properties
Count
Item
Parent

Close

Toolbar object

A Toolbar object represents one toolbar row 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 four toolbars can be displayed in a Visio window at one time.

The default property of Toolbar is Caption.

Methods
Delete

Properties
Caption
Index
Parent
ToolbarItems

Close

ToolbarSet object

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

Methods
Delete

Properties
Caption
Parent
SetID
Toolbars

Close

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. Valid SetIDs for ToolbarSet objects are:

visUIObjSetNoDocument
visUIObjSetDrawing
visUIObjSetStencil
visUIObjSetShapeSheet
visUIObjSetIcon
visUIObjSetInPlace
visUIObjSetPrintPreview
visUIObjSetText
visUIObjSetBinderInPlace

Methods
Add
AddAtID

Properties
Count
Item
ItemAtID
Parent

Close

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 4 Toolbar objects.

Methods
Add
AddAt

Properties
Count
Item
Parent

Close

UI Object object

The UI object represents Visio's menus, toolbars, accelerators, and status bars, from either Visio's built-in 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 disk file and loaded into Visio. Use the SaveToFile property 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.

Methods
Delete
LoadFromFile
SaveToFile
UpdateUI

Properties
AccelTables
Flavor
MenuSets
Name
StatusBars
ToolbarSets

Close

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.

Methods
Activate
AddToGroup
Close
Combine
Copy
Cut
Delete
DeselectAll
Duplicate
Fragment
Group
Paste
RemoveFromGroup
Select
SelectAll
Union

Properties
Application
Document
Index
Page
Selection
SubType
Type
Zoom

Close

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.

Methods
Arrange

Properties
Application
Count
Item

Contents Close

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.

Action
Active
AlignBottom
AlignCenter
AlignLeft
AlignMiddle
AlignRight
AlignTop
Angle
ArrowSize
BeginArrow
BeginX
BeginY
BottomMargin
CanGlue
Case
Color
Color
DrawingScale
EndArrow
EndX

EndY
EventDblClick
EventXFMod
Field
FillBkgnd
FillForegnd
FillPattern
FlipX
FlipY
Font
Format
Glue
GuideFlags
GuidePosX
GuidePosY
Height
Help
HorzAlign
ImgHeight
ImgOffsetX
ImgOffsetY
ImgWidth
IndFirst
IndLeft
IndRight
Invisible
Label
LayerMember
LeftMargin
LineColor
LinePattern
LineWeight
LockAspect
LockBegin
LockCalcWH
LockCrop
LockDelete
Locked
LockEnd
LockFormat
LockGroup
LockHeight
LockMoveX
LockMoveY
LockRotate
LockSelect
LockTextEdit
LockVtxEdit
LockWidth
LocPinX
LocPinY
Menu
Name
NoAlignBox
NoCtlHandles
NonPrinting

NoObjHandles
PageHeight
PageScale
PageWidth
PinX
PinY
Pos
Print
Prompt (Action section)
Prompt (Custom Properties section)
Prompt (User-defined Cells section)
ResizeMode
RightMargin
Rounding
ShdwBkgnd
ShdwForegnd
ShdwOffsetX
ShdwOffsetY
ShdwPattern
Size
Snap
SortKey
SpAfter
SpBefore
SpLine
Status
Style
TextBkgnd
TheData
TheText
Tip
TopMargin
TxtAngle
TxtHeight
TxtLocPinX
TxtLocPinY
TxtPinX
TxtPinY
TxtWidth
Type
UpdateAlignBox
Value (Custom Properties section)
Value (User-defined Cells section)
Verify
VerticalAlign
Visible
Width
XCon
XDyn
YCon
YDyn

Close

Properties

A
B
C
D
E
F
G
H
I
J
K
L
M

Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close

Z

Background
BackPage
BasedOn
Begin
BinaryData
BinaryLength
Blue
BuiltInMenus
BuiltInToolbars
Caption
CellExists
Cells
CellsC
CellsSRC
CellsSRCExists
Characters
CharCount
CharPropsRow
CharSet
CmdNum
CntrlID
CntrlType
Colors
Column
Connects
ContainingMaster
ContainingPage
ContainingShape
Control
Count
Creator
CustomMenus
CustomMenusFile
CustomToolbars
CustomToolbarsFile
Data1
Data2
Data3
DefaultValue
Description
Document
Documents
DrawingPaths
Enabled
End
Entities
EntityApps
Error
Event
EventInfo
EventList
FieldCategory
FieldCode
FieldFormat
FieldFormula
FillBasedOn

FillStyle
FillStyleKeepFmt
FilterPaths
Flags
Flavor
Fonts
Formula
FormulaForce
FromCell
FromPart
FromSheet
FullName
GeometryCount
Green
Group
Handle
Help
HelpContextID
HelpFile
HelpPaths
IconFileName
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
LengthIU
LineBasedOn
LineStyle
LineStyleKeepFmt
LocalName
LongValue
Master
Masters

MDIWindowMenu
MenuItems
Menus
MenuSets
MiniHelp
Name
NameID
OnDataChangeDelay
OneD
Page
Pages
PageSheet
PaletteEntry
ParaPropsRow
Parent
Path
PitchAndFamily
Priority
ProcessID
ProfileName
Prompt
PromptForSummary
ReadOnly
RealValue
Red
Result
ResultForce
ResultInt
ResultIntForce
ResultIU
ResultIUForce
ResultStr
Row
RowCount
RowExists
RowName
RowsCellCount
RowType
RunBegin
RunEnd
Saved
SavePreviewMode
ScreenUpdating
Section
SectionExists
Selection
SetID
Shape
Shapes
Shift
ShortValue
Spacing
StartupPaths
StatusBarItems
StatusBars
StencilPaths

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

Close

Methods

Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close
Close

Z

AddRow
AddRows
AddSection
AddToGroup
Arrange
BeginTransaction
BringForward
BringToFront
CenterDrawing
CharProps
ClearCustomMenus
ClearCustomToolbars
Close
Combine
ConvertToGroup
Copy
Cut
Delete
DeleteRow
DeleteSection
DeselectAll
DrawLine
DrawOval
DrawRectangle
Drop
Duplicate
EndTransaction
Export
FlipHorizontal
FlipVertical
Fragment
GlueTo
GlueToPos
Group
Import
LoadFromFile
Open
OpenEx
ParaProps
Paste
Print
PutShape
Quit
Redo
Remove
RemoveFromGroup
ReverseEnds
Rotate90
Run
Save
SaveAs
SaveAsEx
SaveToFile
SaveWorkspaceAs
Select
SelectAll

SendBackward
SendToBack
SetBegin
SetCenter
SetCustomMenus
SetCustomToolbars
SetEnd
Trigger
Undo
Ungroup
Union
UpdateUI

AccelItems property

Applies to: AccelTable

Summary: Returns the AccelItems collection of an AccelTable object.

Syntax: objRet = object.**AccelItems**

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

See also: AccelItem object

Example for AccelItems

AccelTables property

Applies to: UI Object

Summary: Returns the AccelTables collection of a UI object.

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.

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

Example for AccelTables

Action property

Applies to: Event

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

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 what happens in response to the event. Visio 4.0 supports only one action code:

visActCodeRunAddon = 1

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 "run add-on," the target is the name of the add-on to run, and the arguments are sent to the add-on when it is run.

See also: Event property, EventInfo property, EventList object, Target property, TargetArgs property

Example for Action

ActionText property

Applies to: MenuItem, StatusBarItem, ToolbarItem

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

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 also inserted into error messages. The string is also 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

Example for ActionText

Activate method

Applies to: Window

Summary: Activates the indicated window.

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 can be active. Activating a window can change the objects returned by the ActiveWindow, ActivePage, and ActiveDocument properties.

See also: ActiveWindow property

Example for Activate

Example

ActiveDocument property

Applies to: Application

Summary: Returns the active Document object.

Syntax: objRet = object.**ActiveDocument**

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

Remarks: When no document is active, ActiveDocument returns Nothing. To verify that a document is open, make sure Application.Documents.Count is greater than zero, or use the Is operator to compare the ActiveDocument property with Nothing.

See also: ActivePage property, ActiveWindow property

Close Copy Print

Example for ActiveDocument

```
'Get ActiveDocument Example
'
Dim appVisio As Object, Dim doc As Object

Set appVisio = GetObject(,"visio.application")  '-- Get Visio Instance

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

If appVisio.Documents.Count > 0 Then              '-- More Than One Doc...
  Set doc = appVisio.ActiveDocument               '--   Get Document
End If

'Alternate Example

If Not(appVisio.ActiveDocument Is Nothing) Then   '-- A Document Exists...
  Set doc = appVisio.ActiveDocument               '--   Retrieve It
End If
```

Example

ActivePage property

Applies to: Application

Summary: Returns the active Page object.

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

Close Copy Print

Example for ActivePage

```
Dim appVisio As Object, pag As Object

Set appVisio = GetObject(,"visio.application")       '-- Get Instance Of Visio

If Not(appVisio.ActivePage Is Nothing) Then          '-- Drawing Page
Active...
    Set pag = appVisio.ActivePage                    '--   Retrieve Page
End If
```

ActiveWindow property

Applies to: Application

Summary: Returns the active Window object.

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

Example for ActiveWindow

Add method

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

Summary: Adds a new object to the indicated collection.

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 Attributes, Entities, and EntityApps 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.

The Add method for the Addons, Documents, 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 Addons collection and returns an Addon object if the string expression specifies an .EXE file, or nothing if it specifies a .VSL file.

Documents collection: creates a new drawing based on the specified .VST file. This is equivalent to choosing New from the File menu. A null string ("") specifies a new drawing based on no template. 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. If no path is indicated, Visio searches directories designated in the application's TemplatePath.

If you specify a .VSD or .VSS file, Add opens a copy of that file. This is equivalent to selecting Copy in the Open section of the Open dialog box or using the OpenEx method with a visOpenCopy flag.

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

Layer object: the Add method assigns the specified Shape object to the layer. It has 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.

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

```
Set eventObj = eventList.Add(event,action,target,targetArgs)
```

The arguments set the initial values of the Event object's Event, Action, Target, and TargetArgs properties.

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, AddAt method, AddAtID method, BasedOn property, Event property, IncludesFill property, IncludesLine property, IncludesText property, Layer property, Name property, Open method, OpenEx method, Target property, TargetArgs property

Example for Add

AddAt method

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

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

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.

See also: Add method, AddAtID method

Close Copy Print

Example for AddAt

AddAtID method

Applies to: AccelTables, MenuSets, StatusBars, ToolbarSets

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

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 defined in 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: AccelTables object, Add method, AddAt method, MenuSets object, StatusBars object, ToolbarSets object

Close Copy Print

Example for AddAtID

AddCustomField method

Applies to: Characters

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

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 from the Insert menu to insert a custom formula field in text. To add a field of any other category, use the AddField method.

The following constants for field formats are defined in 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

Close Copy Print

Example for AddCustomField

AddField method

Applies to: Characters

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

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 defined in 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

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

Close Copy Print

Example for AddField

AddGuide method

Applies to: Page

Summary: Adds a guide to a drawing page.

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 defined in VISCONST.BAS are valid values for the type of guide to add:

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

Example for AddGuide

AddNamedRow method

Applies to: Shape

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

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 can be 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 cells in named rows, search Visio online Help for User.Row or Prop.Row.

See also: Cells property, CellsSRC property, RowName property

Close Copy Print

Example for AddNamedRow

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.

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

Example for AddonArgs

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.

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

Element	Description
object	The object that runs the add-on
addonStr	The name of the add-on to be run

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.

See also: AddonArgs property, CmdNum property

Example for AddonName

AddonPaths property

Applies to: Application

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

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: 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.

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.

See also: DrawingPaths property, FilterPaths property, HelpPaths property, StartupPaths property, StencilPaths property, TemplatePaths property

Example for AddonPaths

Addons property

Applies to: Application

Summary: Returns the Addons collection of an Application object.

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, AddonPaths property

Example for Addons

Close

AddRow method

Applies to: Shape

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

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 defined in 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.

The following row constants are defined in VISCONST.BAS:

```
visRowFirst = 0                 'first logical row in any section
visRowLast = -2                 'last logical row in any section
visRowNone = -1                 'unspecified row
visRowXFormOut = 1
visRowLine = 2
visRowFill = 3
visRowXForm1D = 4
visRowEvent = 5
visRowGuide = 7
visRowStyle = 8
visRowForeign = 9
visRowPage = 10
visRowText = 11
visRowTextXForm = 12
visRowXFormIn = 1
visRowAlign = 14
visRowLock = 15
visRowData123 = 16
visRowMisc = 17
```

visRowRulerGrid = 18
visRowMember = 0
visRowField = 0
visRowCharacter = 0
visRowParagraph = 0
visRowTab = 0
visRowScratch = 0
visRowExport = 0
visRowControl = 0
visRowComponent = 0
visRowVertex = 1
visRowAction = 0
visRowLayer = 0
visRowFormat = 0

The following constants for Geometry section row tags are defined in VISCONST.BAS:

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

Close

Example 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.

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 defined in 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.

The following row constants are defined in VISCONST.BAS:

```
visRowFirst = 0                 'first logical row in any section
visRowLast = -2                 'last logical row in any section
visRowNone = -1                 'unspecified row
visRowXFormOut = 1
visRowLine = 2
visRowFill = 3
visRowXForm1D = 4
visRowEvent = 5
visRowGuide = 7
visRowStyle = 8
visRowForeign = 9
visRowPage = 10
visRowText = 11
visRowTextXForm = 12
visRowXFormIn = 1
visRowAlign = 14
visRowLock = 15
visRowData123 = 16
```

visRowMisc = 17
visRowRulerGrid = 18
visRowMember = 0
visRowField = 0
visRowCharacter = 0
visRowParagraph = 0
visRowTab = 0
visRowScratch = 0
visRowExport = 0
visRowControl = 0
visRowComponent = 0
visRowVertex = 1
visRowAction = 0
visRowLayer = 0
visRowFormat = 0

The following constants for Geometry section row tags are defined in VISCONST.BAS:

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

See also: AddRow method

Close

Example for AddRows

Close

AddSection method

Applies to: Shape

Summary: Adds a new section to a ShapeSheet.

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 defined in VISCONST.BAS:

```
visSectionFirst = 0                'first logical section
visSectionLastComponent = 250
visSectionLast = 252               'last logical section
visSectionText = 253
visSectionForeign = 254
visSectionNone = 255               'unspecified logical section
visSectionObject = 1
visSectionMember = 2
```

visSectionCharacter = 3
visSectionParagraph = 4
visSectionTab = 5
visSectionScratch = 6
visSectionExport = 7
visSectionTextField = 8
visSectionControls = 9
visSectionFirstComponent = 10

See also: AddRow method, CellsSRC property, DeleteSection method

Close

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

```
Sub BuildShape ()
' Demonstrates working with shape sheet sections and rows
'
  Dim appVisio As Object, DrawPage As Object, shp As Object
  Dim cell As Object
  Dim BowCell As String, BowFormula As String
  Dim idxInnerRect As Integer, I As Integer

  BowCell = "Scratch.X1": BowFormula = "=Min(Width, Height) / 5"

  Set appVisio = CreateObject("visio.application")

  If appVisio.ActiveDocument Is Nothing Then
    appVisio.Documents.Add ("")
  End If

  Set DrawPage = appVisio.ActivePage

  If DrawPage Is Nothing Then
    Set DrawPage = appVisio.ActiveDocument.Pages(1)
  End If

' Now that we have a drawing page we first draw a rectangle.

  Set shp = DrawPage.DrawRectangle(1, 5, 5, 1)

' Now we add a scratch section, add a row to the scratch, and then
' place the bowing value into Scratch.X1.

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

  Set cell = shp.Cells(BowCell)                      ' Get bowing cell
  cell.Formula = BowFormula                          ' Set up offset

' Next we bow in the original rectangle's lines by changing each row
' to an arc and entering the bow value.

  For I = 1 To 4
    shp.RowType(visSectionFirstComponent, visRowVertex + I) = visTagArcTo

    Set cell = shp.CellsSRC(visSectionFirstComponent, visRowVertex + I, 2)
    cell.Formula = "-" & BowCell
  Next I

' Lastly we create the inner rectangle.  We add the new geometry (component)
' section and four line segments within it.  Then we draw the rectangle
' inside the now-bowed edges.

  idxInnerRect = visSectionFirstComponent + 1        ' Inner rectangle section
index
```

```
  shp.AddSection idxInnerRect               ' Add inner rect section
  shp.AddSection idxInnerRect + 1           ' Another for the heck of it!
  shp.DeleteSection idxInnerRect + 1        ' But we don't really want it!

  shp.AddRow idxInnerRect, visRowVertex, visTagComponent
  shp.AddRow idxInnerRect, visRowVertex + 1, visTagMoveTo

  For I = 1 To 4
    shp.AddRow idxInnerRect, visRowLast, visTagLineTo
  Next I

  Set cell = shp.CellsSRC(idxInnerRect, 1, 0)   ' Start
    cell.Formula = "Width * 0 + " & BowCell     '   X
  Set cell = shp.CellsSRC(idxInnerRect, 1, 1)   '
    cell.Formula = "Height * 0 + " & BowCell    '   Y

  Set cell = shp.CellsSRC(idxInnerRect, 2, 0)   ' Bottom line
    cell.Formula = "Width * 1 - " & BowCell      '   X
  Set cell = shp.CellsSRC(idxInnerRect, 2, 1)   '
    cell.Formula = "Height * 0 + " & BowCell    '   Y

  Set cell = shp.CellsSRC(idxInnerRect, 3, 0)   ' Right line
    cell.Formula = "Width * 1 - " & BowCell      '   X
  Set cell = shp.CellsSRC(idxInnerRect, 3, 1)   '
    cell.Formula = "Height * 1 - " & BowCell     '   Y

  Set cell = shp.CellsSRC(idxInnerRect, 4, 0)   ' Top line
    cell.Formula = "Width * 0 + " & BowCell     '   X
  Set cell = shp.CellsSRC(idxInnerRect, 4, 1)   '
    cell.Formula = "Height * 1 - " & BowCell     '   Y

  Set cell = shp.CellsSRC(idxInnerRect, 5, 0)   ' Left line
    cell.Formula = "Geometry2.X1"               '   X
  Set cell = shp.CellsSRC(idxInnerRect, 5, 1)   '
    cell.Formula = "Geometry2.Y1"               '   Y
End Sub
```

AddToGroup method

Applies to: Window

Summary: Adds the selected shapes to the selected group.

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

Example for AddToGroup

AlignName property

Applies to: Master

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

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 defined in VISCONST.BAS show the possible alignment values:

visLeft = 1
visCenter = 2
visRight = 3

Example for AlignName

Alt property

Applies to: AccelItem

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

Syntax: object.**Alt** = boolVal
boolVal = object.**Alt**

Element	Description
object	An AccelItem object
boolVal	TRUE (-1) if Alt modifies Key in the accelerator; otherwise FALSE (0)

See also: Control property, Key property, Shift property

Example for Alt

Application property

Applies to: Addon, Addons, Application, Cell, Characters, Color, Colors, Connect, Connects, Document, Documents, Event, EventList, Font, Fonts, Layer, Layers, Master, Masters, Page, Pages, Selection, Shape, Shapes, Style, Styles, Window, Windows

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

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. Use the Application property to retrieve the instance of Visio that is associated with a particular object.

Example for Application

AreaIU property

Applies to: Shape

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

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

Close

Example for AreaIU

Arrange method

Applies to: Windows

Summary: Arranges the windows in the indicated Windows collection.

Syntax: object.[**Arrange**]

Element	Description
object	The collection of windows to arrange

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

Enclose Arrange in brackets to distinguish it from the Visual Basic keyword.

Example for Arrange

Close

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.

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

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

Close

Example for Attributes

```
*PutShape Method
```

Background property

Applies to: Page

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

Syntax: retVal = object.**Background**
object.**Background** = {True | False}

Element	Description
retVal	TRUE if the page is a background page; otherwise FALSE
object	The Page object to examine

See also: BackPage property

Close

Example for Background

BackPage property

Applies to: Page

Summary: Returns or sets the background page assigned to another page.

Syntax: objRet = object.**BackPage**
object.**BackPage** = stringExpression

Element	Description
objRet	A Page object that represents the background page
object	The Page object that has or gets the background page
stringExpression	The name of the background page

Remarks: If the indicated page has no background, BackPage returns Nothing. 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 (not to an object reference to the page).

See also: Background property

Close

Example for BackPage

BasedOn property

Applies to: Style

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

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, TextBasedOnProperty

Close

Example for BasedOn

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.

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 start 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.

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 speed 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.

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.

Close

Example for BinaryData

BinaryLength property

Applies to: Entity

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

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.

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

BringForward method

Applies to: Selection, Shape

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

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

BringToFront method

Applies to: Selection, Shape

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

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

BuiltInMenus property

Applies to: Application

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

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 Visio's.

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

Example for BuiltInMenus

BuiltInToolbars property

Applies to: Application

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

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	The set of built-in toolbars to get

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 Visio's.

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.

fWhichToolbars should be one of the following values:

visToolbarMSOffice = 0
visToolbarLotusSS = 1
visToolbarNovellPO = 2

See also: BuiltInMenus property, CustomToolbarsFile property, Flavor property, SaveToFile method, SetCustomToolbars method

Close

Example for BuiltInToolbars

Caption property

Applies to: Menu, MenuItem, MenuSet, StatusBar, Toolbar, ToolbarSet

Summary: Gets or sets the caption for the indicated object.

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. The \t character inserts a tab in the string and is used to align text in columns on menus. The \a character aligns the text that follows it flush right on the menu or menu bar. 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, Toolbar, or ToolbarSet object.

See also: AccelItem object, ActionText property, CmdNum property

Example for Caption

CellExists property

Applies to: Shape

Summary: Returns TRUE if the indicated ShapeSheet cell exists in the scope of the search.

Syntax: boolRet = object.**CellExists**(stringExpression, fExistsLocally)

Element	Description
boolRet	0 if cell exists, -1 if it does not
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

Close

Example for CellExists

Close

Cells property

Applies to: Shape, Style

Summary: Returns a Cell object that represents the specified ShapeSheet cell.

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: If the specified cell does not exist, an error is generated.

For information about a particular ShapeSheet cell, see the Visio online Help.

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.

Suppose "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:

```
cellobj = shpobj.cells("User.MyRowsName")
```

The prompt cell in MyRowsName could be accessed via:

```
cellobj = shpobj.cells("User.MyRowsName.Prompt")
```

Suppose MyRowsName were in the Custom Properties section instead of the User-defined Cells section. The zero'th (value) cell would be accessed using:

```
cellobj = shpobj.cells("Prop.MyRowsName")
```

Other cells in the row could be accessed using:

```
cellobj = shpobj.cells("Prop.MyRowsName.xxx")
```

where xxx is one of: Label, Prompt, SortKey, Type, Format, Invisible, Ask

You can access cells in a named row via:

```
cellobj = shpobj.cells("User.somename.cellname")
```

This can be used to get any cell in the row.

To access cells in custom properties use "Prop" instead of "user".

See also: AddNamedRow method, CellsSRC property, RowName property, ShapeSheet Cells

Close

Example for Cells

```
*AddSection Method
```

CellsC property

Applies to: Layer

Summary: Returns a Cell object that represents the specified ShapeSheet cell.

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

Close

Example for CellsC

Close

CellsSRC property

Applies to: Shape

Summary: Returns a Cell object that represents the ShapeSheet cell identified by section, row, and column indices.

Syntax: objRet = object.**CellsSRC** (section, row, cell)

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: Use the CellsSRC property to access any shape formula by its section, row, and column indices. Constants for section, row, and column indices are defined in VISCONST.BAS.

See also: AddRow method, AddSection method, Cells property

Close

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.

Syntax: boolRet = object.**CellsSRCExists**(section,row,column,fExistsLocally)

Element	Description
boolRet	0 if the cell exists, -1 if it does not
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 defined in 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

Close

Example for CellsSRCExists

CenterDrawing method

Applies to: Page

Summary: Centers the shapes on a page.

Syntax: object.**CenterDrawing**

Element	Description
object	The page that contains the shapes to center

Remarks: Centering shapes does not change their positions relative to each other.

Close

Example for CenterDrawing

Characters property

Applies to: Shape

Summary: Returns a Characters object that represents the text of the indicated shape.

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 objectm, Text property

Close

Example for Characters

CharCount property

Applies to: Characters, Shape

Summary: Returns the number of characters in the indicated object.

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 method

Applies to: Characters

Summary: Sets the indicated character property of a Characters object to a new value.

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 defined in 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

Close

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.

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:

Symbol	Meaning
visBiasLeft	1
visBiasRight	2

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 method, ParaPropsRow property, TabPropsRow property

Example for CharPropsRow

CharSet property

Applies to: Font

Summary: Returns the Windows character set for a Font object.

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

Close

Example for CharSet

ClearCustomMenus method

Applies to: Application, Document

Summary: Restores Visio's built-in menus in place of any custom menus that are in effect.

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 method

Applies to: Application, Document

Summary: Restores Visio's built-in toolbars in place of any custom toolbars that are in effect.

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

Close

Close method

Applies to: Document, Window

Summary: Closes the indicated window or document.

Syntax: object.[**Close**]

Element	Description
object	The Window or Document object to close

Remarks: If the indicated window is the only window open for a document and the document contains unsaved changes, a dialog box appears asking if you want to save the document. If you want to close the document without saving and without seeing the dialog box, set the Saved property of the Document object representing the document to True immediately before closing. Set the Saved property to True only if you are sure you want to close the document without saving changes.

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.

Enclose Close in brackets to distinguish it from the Visual Basic keyword.

Example for Close

```
Sub CloseShapeSheets (wins As Object)
' Assumes wins is a valid Windows collection object.

    Dim I As Integer

    I = wins.Count

    While I <> 0
        If wins(I).Type = visSheet Then
            wins(I).[Close]
            I = wins.Count
        Else
            I = I - 1
        End If
    Wend
End Sub
```

CmdNum property

Applies to: AccelItem, MenuItem, StatusBarItem, ToolbarItem

Summary: Gets or sets the command ID associated with an AccelItem, MenuItem, StatusBarItem, or ToolbarItem object.

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, CmdNum should be set to 0. CmdNum can also be 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 prefixed in VISCONST.BAS with visCmd.

See also: AccelItem object, AddonName property, IsHierarchical property

Example for CmdNum

CntrlID property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the control ID for a ToolbarItem or StatusBarItem object.

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 identifies a toolbar or status bar button supplied with Visio. CntrlID should be 0 for a custom toolbar or status bar button.

Valid control IDs are prefixed in VISCONST.BAS with visCtrlID.

See also: CntrlType property, TypeSpecific1 property, TypeSpecific2 property

Example for CntrlID

CntrlType property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the control type of an item in a toolbar or status bar.

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 defined in 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

Colors property

Applies to: Document

Summary: Returns the Colors collection of a Document object.

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

Close

Example for Colors

Column property

Applies to: Cell

Summary: Returns the column index of a cell.

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

Close

Example for Column

Combine method

Applies to: Window

Summary: Creates a new shape by combining the selected shapes.

Syntax: object.**Combine**

Element	Description
object	The Window 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 original shapes are deleted.

See also: Fragment method, Union method

Close

Example for Combine

Connects property

Applies to: Shape

Summary: Returns the Connects collection for the indicated shape.

Syntax: objRet = object.**Connects**

Element	Description
objRet	The Connects collection of the Shape object
object	The Shape object that owns the collection

See also: Connect object, Connects object

Example for Connects

ContainingMaster property

Applies to: Selection, Shape, Shapes

Summary: Returns the Master object that contains the indicated object.

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.

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.

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 at the top level of a Page or Master object, ContainingShape returns the page sheet of that page or master.

If the ShapeObject is the page sheet of a page or master, ContainingShape returns Nothing.

If the Shape object isn't contained in another shape, ContainingShape returns Nothing.

See also: ContainingMaster property, ContainingPage property, PageSheet property, Shape object, Shapes property

Close

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.

Syntax:

retVal = object.**Control**
object.**Control** = expression
boolVal = object.**Control**
object.**Control** = boolVal

Element	Description
retVal	The current control value for an Entity object
object	The Entity or AccelItem object that has or gets the control value
expression	The new control value for an Entity object
boolVal	TRUE (-1) if the Control key modifies the key in an AccelItem object; otherwise FALSE (0)

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.

Example for Control

ConvertToGroup method

Applies to: Selection, Shape

Summary: Converts a selection or an object from another application (a linked or embedded object) to a group.

Syntax: object.**ConvertToGroup**

Element	Description
object	The Shape or Selection object to convert

Example for ConvertToGroup

Close

Copy method

Applies to: Characters, Selection, Shape, Window

Summary: Copies the specified object, selection, or text range to the Windows Clipboard.

Syntax: object.**Copy**

Element	Description
object	The object to copy

Remarks: Use the Copy method to copy a shape, selection, or text range using the Windows Clipboard.

For 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 applied to a Characters object, Copy places the text range represented by that object onto the Clipboard.

To make copies of a shape without using the Clipboard, use the Duplicate method.

See also: Cut method, Delete method, Duplicate method, Paste method

Close

Example for Copy, Cut, Paste

```
Sub TestEdits ()
' Demonstrate use of Copy/Paste/Cut methods.

  Dim appVisio As Object, DrawPage As Object, shp As Object

  Set appVisio = CreateObject("visio.application")

' First, set up Visio so that we have a document and a page to draw on.

  appVisio.Documents.Add("")
  Set DrawPage = appVisio.ActivePage

'   Next, draw a demonstration rectangle.  It is then copied, pasted,
' and cut.  The copy is then moved to the bottom left
' corner of the page by setting its PinX and PinY formulas.  It is assumed
that
' the pin is located at the center of the rectangle when it is drawn.  Also,
' since the copied shape was the last one drawn it can be retrieved from the
' end of the shape collection.

  Set shp = DrawPage.DrawRectangle(1, 5, 5, 1)

  shp.Copy                                '-- Copy Shape To Clipboard
  DrawPage.Paste                          '-- Paste Copy Into Drawing Page
  shp.Cut                                 '-- Remove Original

  Set shp = DrawPage.Shapes.Item(DrawPage.Shapes.Count)

  shp.PinX = "Width * 0.5"                '-- Set New PinX and PinY
  shp.PinY = "Height * 0.5"               '--
End Sub
```

Close

Count property

Applies to: AccelItems, AccelTables, Addons, Attributes, Colors, Connects, Documents, Entities, EntityApps, EventList, Fonts, Layers, Masters, MenuItems, Menus, MenuSets, Pages, Selection, Shapes, StatusBarItems, StatusBars, Styles, ToolbarItems, Toolbars, ToolbarSets, Windows

Summary: Returns the number of objects in a collection.

Syntax: intRet = object.**Count**

Element	Description
intRet	The number of objects in the collection
object	The collection to examine

Remarks: Use the Count property to enumerate a collection object as shown below.

Close

Example for Count

```
Sub UpdateForm ()
' Fills a list box with the name of every open Visio document.  Paste this
into the
' declarations section of a form and create a list box control named
' ctlDocList on the form

  On Error Goto lblErr

  Dim I As Integer
  Dim appVisio As Object, docsDocList As Object
  Dim pag As Object, doc As Object

  ' ctlDocList is the name of the list box to receive the document names
  ctlDocList.Clear

  Set appVisio = GetObject(,"visio.application")
  Set docsDocList = appVisio.Documents

  If docsDocList.Count <> 0 Then
    For I = 1 To docsDocList.count          '-- For Each Document...
      Set doc = docsDocList.Item(I)        '--   Get Next Document

      ctlDocList.AddItem doc.Name           '--   Add Its Name To List
    Next I

    ctlDocList.ListIndex = 0                '-- Default To First Document
  Else
    MsgBox ("There are no documents loaded in Visio!")
  End If

lblErr:
  Exit Sub
End Sub
```

Creator property

Applies to: Document

Summary: Returns or sets the value of the Creator field in a document's properties.

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.

See also: Description property, Keywords property, Subject property, Title property

Example for Creator

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.

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

Close

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.

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.

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: CustomMenus property, CustomMenusFile property, CustomToolbarsFile property

Close

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.

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

Close

Example for CustomToolbarsFile

Close

Cut method

Applies to: Characters, Selection, Shape, Window

Summary: Deletes the indicated object, selection, or text range and places it on the Windows Clipboard.

Syntax: object.**Cut**

Element	Description
object	The object to cut

Remarks: For a Window object, Cut deletes the shapes that are selected in that window. When applied to 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.

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.

See also: Data2 property, Data3 property

Example for Data1

Data2 property

Applies to: Shape

Summary: Returns or sets the value of the Data2 field for the indicated shape.

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.

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

Data3 property

Applies to: Shape

Summary: Returns or sets the value of the Data3 field for a shape.

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.

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

Close

DefaultValue property

Applies to: Attribute

Summary: Returns or sets the default value of an Attribute object.

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

Close

Example for DefaultValue

```
*PutShape Method
```

Close

Delete method

Applies to: AccelItem, AccelTable, Attribute, Entity, EntityApp, Event, Layer, Master, Menu, MenuItem, MenuSet, Page, Selection, Shape, StatusBar, StatusBarItem, Style, Toolbar, ToolbarItem, ToolbarSet, UI Object, Window

Summary: Deletes the indicated object or selection.

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: Enclose Delete in brackets to distinguish it from the Visual Basic keyword.

For a Window object, Delete deletes the selected shapes in that window. The Delete method is equivalent to choosing the Clear command from the Edit menu in Visio.

For 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.

For 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 UpdatePageNames in the Delete Pages dialog box.

Sometimes attempting to delete a page causes runtime error 1512: "Visio cannot delete the page because it is open in another Visio window." This error indicates that another object-- not necessarily a page or window object-- is associated with the page you are trying to delete. For example, the following statements may generate the error:

```
Set docObj = appVisio.ActiveDocument
appVisio.ActivePage.[Delete] 1                    'Sometimes fails
```

To prevent the error, set all objects that may be associated with the page to Nothing before attempting to delete the page. For example:

```
Set docObj = appVisio.ActiveDocument
Set docObj = Nothing
appVisio.ActivePage.[Delete] 1                    'Should not fail
```

See also: Copy method, Cut method, Duplicate method, Paste method

Close

Example for Delete, Duplicate

```
Sub DupAndDel ()
' Demonstrates duplicating and deleting objects
'
  Dim appVisio As Object, DrawPage As Object
  Dim shpOriginal As Object, shpDuplicate As Object

  Set appVisio = CreateObject("visio.application")
  Set DrawPage = appVisio.Documents.Add("").Pages(1)
  Set shpOriginal = DrawPage.DrawLine(1, 1, 5, 5)

  shpOriginal.Duplicate

  Set shpDuplicate = DrawPage.Shapes(2)

  shpDuplicate.Cells("BeginY") = "2"

  shpOriginal.[Delete]
End Sub
```

Close

DeleteRow method

Applies to: Shape

Summary: Deletes a row from a section in a ShapeSheet.

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: Use the DeleteRow method to remove one row at a time from a ShapeSheet section. 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

Close

Example for DeleteRow

```
*AddSection Method
```

Close

DeleteSection method

Applies to: Shape

Summary: Deletes a section from a ShapeSheet.

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

Close

Example for DeleteSection

```
*AddSection Method
```

Description property

Applies to: Document

Summary: Returns or sets the value of the Description field in a document's properties.

Syntax: strRet = object.**Description**
object.**Description** = 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 Description property is equivalent to filling in the Description in the Properties dialog box.

See also: Creator property, Keywords property, Subject property, Title property

Example for Description

DeselectAll method

Applies to: Selection, Window

Summary: Deselects all shapes in a window or selection.

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

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.

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.

Close

Example for Document

Close

Documents property

Applies to: Application

Summary: Returns the Documents collection for an instance of Visio.

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

Close

Example for Documents

```
' Retrieve document collection

Dim appVisio As Object, docsDocList As Object

Set appVisio = CreateObject("visio.application")
Set docsDocList = appVisio.Documents
```

DrawingPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for drawings.

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: 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.

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.

See also: AddonPaths property, FilterPaths property, HelpPaths property, StartupPaths property, StencilPaths property, TemplatePaths property

Close

Example for DrawingPaths

Close

DrawLine method

Applies to: Page

Summary: Draws a line on a page.

Syntax: objRet = object.**DrawLine** (x1, y1, x2, y2)

Element	Description
objRet	A Shape object that represents the new line
object	The Page object 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: This method is equivalent to using the line tool in Visio. The arguments are in page units.

See also: DrawOval method, DrawRectangle method

Example for DrawLine

```
*DrawRectangle Method
```

Close

DrawOval method

Applies to: Page

Summary: Draws an ellipse within the specified area.

Syntax: retVal = object.**DrawOval** (x1, y1, x2, y2)

Element	Description
retVal	A Shape object that represents the new ellipse
object	The Page object 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: This method is equivalent to using the ellipse tool in Visio. The arguments are in page units.

See also: DrawLine method, DrawRectangle method

Example for DrawOval

*DrawRectangle Method

Close

DrawRectangle method

Applies to: Page

Summary: Draws a rectangle within the specified area.

Syntax: objRet = object.**DrawRectangle** (x1, y1, x2, y2)

Element	Description
objRet	A Shape object that represents the new rectangle
object	The Page object 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: This method is equivalent to using the rectangle tool in Visio. The arguments are in page units.

See also: DrawLine method, DrawOval method

Close

Example for DrawLine, DrawOval, DrawRectangle

```
Sub TestDrawMethods ()
' Demonstrates use of Visio's drawing methods.

    Dim appVisio As Object
    Dim pag As Object
    Dim shp As Object

    Set appVisio = CreateObject(visApi)
    Set pag = appVisio.Documents.Add("").Pages(1)

    Set shp = pag.DrawRectangle(1, 4, 4, 1)
    shp.FillStyle = "Blue fill"

    Set shp = pag.DrawOval(1.5, 10.5, 7.5, 6.5)
    shp.FillStyle = "Red fill"

    Set shp = pag.DrawLine(5, 4, 7.5, 1)
End Sub
```

Close

Drop method

Applies to: Document, Page, Shape

Summary: Drops a Shape object into a stencil, drawing page, or group.

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
y	The y-coordinate at which to place the dropped shape's center of rotation

Remarks: 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.

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, Open method

Close

Example for Drop, Group

```
Sub TestDropMethod ()
' Demonstrates the drop method with Page.Drop, Document.Drop and Shape.Drop

    Dim appVisio As Object
    Dim shp1 As Object
    Dim shp2 As Object
    Dim shp3 As Object
    Dim mst As Object

    Set appVisio = CreateObject("visio.application")     '-- Create Visio
Instance
    appVisio.Documents.Add ("")                          '-- Create Blank
Document

    Set shp1 = appVisio.ActivePage.DrawRectangle(1, 2, 2, 1)
    Set shp2 = appVisio.ActivePage.DrawRectangle(1, 4, 2, 3)

  ' Example of Page.Drop
  '
    appVisio.ActivePage.Drop shp1, 3.5, 3.5
    Set shp3 = appVisio.ActivePage.Shapes(3)

  ' Example of Document.Drop - Creates a master
  '
    Set mst = appVisio.ActiveDocument.Drop(shp3, 0, 0)

  ' Example of Shape.Drop
  '
    appVisio.ActiveWindow.Select shp1, visSelect
    appVisio.ActiveWindow.Select shp2, visSelect
    appVisio.ActiveWindow.Group

    Set shp1 = appVisio.ActivePage.Shapes(2)

    shp1.Drop shp3, 3, 3
End Sub
```

Close

Duplicate method

Applies to: Selection, Shape, Window

Summary: Duplicates the specified object or selection.

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. The Duplicate method is equivalent to choosing the Duplicate command from the Edit menu in Visio. For a Shape object, Duplicate duplicates the shape. For 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

Summary: Returns true if the Addon object is currently enabled, or false if it is disabled.

Syntax: boolVal = object.**Enabled**

Element	Description
boolVal	0 if the add-on is disabled, -1 if it is enabled
object	The Addon object to examine

Remarks: 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.

Example for Enabled

End property

Applies to: Characters

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

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

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

Remarks: The End property determines the end of the text range represented by a Characters object. 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

Example for End

EndTransaction method

Applies to: ShapeData

Summary: Ends a transaction for the ShapeData object.

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.

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

Close

EntityApps property

Applies to: ShapeData

Summary: Returns the EntityApps collection for a given shape.

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

Close

Example for EntityApps

*PutShape Method

Error property

Applies to: Cell

Summary: Returns the error code generated by the last evaluation of the indicated cell's formula.

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 defined in 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.

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. An event also specifies the target of the action and arguments to send to the target. An add-on can cause an event's action to occur by using the Trigger method.

Visio 4.0 supports these event codes:

visEvtCodeDocCreate = 1
visEvtCodeDocOpen = 2
visEvtCodeShapeDelete = 801

An event code of visEvtCodeDocCreate triggers the event when another document is created from that document. Often, the event will be defined in a template file (.VST), which means the event is triggered when a new document based on that template is created.

An event code of visEvtCodeDocOpen triggers the event whenever the document is opened.

An event code of visEvtCodeShapeDelete triggers the event whenever a shape or shapes is deleted. In this case, the EventInfo property may contain a list of shapes that were deleted.

See also: Action property, 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.

Syntax: strRet = object.**EventInfo**(eventID)

Element	Description
strRet	Additional information about the event
object	The Application object to examine
eventID	The ID of the event to examine, or visEvtIDMostRecent

Remarks: When an Event object's action is performed, Visio passes an argument string to the target of the event. Because a limited amount of information can be passed in the target argument string, Visio records additional information for some events. To obtain this information, the event target can get the EventInfo property. If an event does not record extra information, EventInfo returns Nothing.

The target argument string is in the form "/eventid=<number>". To ensure that the information returned by EventInfo is associated with the same event as the target argument string, pass <number> as an argument to EventInfo. If Visio no longer has information for the specified event, EventInfo generates an error.

To get information about the most recent event, pass visEventIDMostRecent as an argument to EventInfo. If a target queries EventInfo immediately after being triggered, the most recent event and the event identified in the argument string are usually the same. If the target is an add-on implemented by an .EXE file, this may not always be the case, because the .EXE and Visio are separate tasks that aren't modal with respect to each other.

If the Event property of the Event object is visEvtCodeDocCreate or visEvtCodeDocOpen, EventInfo returns Nothing. If the Event property is visEvtCodeShapeDelete, the TargetArgs property contains the index of the Page object or Master object that contains the deleted shape or shapes.

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

Note that by the time you receive the visEvtCodeShapeDelete event, the shapes are already gone.

See also: Action property, Event object, Event property, NameID property, Target property, TargetArgs property, UniqueID property

Example for EventInfo

EventList property

Applies to: Document, Event

Summary: Returns the EventList collection of an object or the EventList collection that contains an Event object.

Syntax: objRet = object.**EventList**

Element	Description
objRet	The EventList collection
object	The Document that owns the collection or the Event object that belongs to the collection

See also: EventList object

Example for EventList

Export method

Applies to: Page, Selection, Shape

Summary: Exports the indicated object from Visio.

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, 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.

If specified file already exists, it is replaced without prompting the user.

See also: Import method

Close

Example for Export

FieldCategory property

Applies to: Characters

Summary: Returns the field category for the field represented by the indicated object.

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 defined in VISCONST.BAS.

See also: AddField method, FieldCode property, FieldFormat property, FieldFormula property

Close

Example for FieldCategory

FieldCode property

Applies to: Characters

Summary: Returns the field code for the field represented by the indicated object.

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 constants are defined in 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.

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 defined in VISCONST.BAS.

See also: AddCustomField method, AddField method, FieldCategory property, FieldCode property, FieldFormula property

Close

Example for FieldFormat

FieldFormula property

Applies to: Characters

Summary: Returns the formula of the custom field represented by the indicated object.

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

Close

Example for FieldFormula

FillBasedOn property

Applies to: Style

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

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, TextBasedOnProperty

Close

Example for FillBasedOn

FillStyle property

Applies to: Selection, Shape

Summary: Returns or sets the fill style for an object

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

Close

Example for FillStyle

FillStyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a fill style to an object while preserving local formatting.

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

Close

Example for FillStyleKeepFmt

FilterPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for import and export filters.

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: 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.

For example, if Visio's executable file is installed in c:\Visio, and FilterPaths is "Filters;d:\Filters", Visio looks for filters in both c:\Visio\Filters and d:\Filters.

See also: AddonPaths property, DrawingPaths property, HelpPaths property, StartupPaths property, StencilPaths property, TemplatePaths property

Example for FilterPaths

Flags property

Applies to: Color

Summary: Gets or sets the flags that specify how a Color object is used.

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: Determines whether custom toolbars are based on Microsoft Office, Lotus SmartSuite, or Novell PerfectOffice product suites.

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 Flavor property can have one of the following values:

visToolBarMSOffice = 0
visToolBarLotusSS = 1
visToolBarNovellPO = 2

See also: BuiltInToolbars property

Example for Flavor

FlipHorizontal method

Applies to: Selection, Shape

Summary: Flips an object horizontally.

Syntax: object.**FlipHorizontal**

Element	Description
object	The Shape or Selection object to flip

See also: FlipVertical method, ReverseEnds method, Rotate90 method

Close

Example for FlipHorizontal

FlipVertical method

Applies to: Selection, Shape

Summary: Flips an object vertically.

Syntax: object.**FlipVertical**

Element	Description
object	The Shape or Selection object to flip

See also: FlipHorizontal method, ReverseEnds method, Rotate90 method

Close

Example for FlipVertical

Fonts property

Applies to: Document

Summary: Returns the Fonts collection of a Document object.

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

Close

Example for Fonts

Close

Formula property

Applies to: Cell

Summary: Returns or sets the formula for a Cell object.

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

Close

Example for Formula

```
*AddSection Method
```

FormulaForce property

Applies to: Cell

Summary: Sets the formula in a Cell object, even if the formula is protected with a GUARD function.

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

Close

Example for FormulaForce

Fragment method

Applies to: Window

Summary: Breaks selected shapes into smaller shapes.

Syntax: object.**Fragment**

Element	Description
object	The Window that contains the shapes to fragment

Remarks: The Fragment method is equivalent to choosing the Fragment command from the Operations submenu on the Shape menu in Visio.

See also: Combine method, Union method

Example for Fragment

Close

FromCell property

Applies to: Connect

Summary: Returns the cell from which a connection originates.

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

Close

Example for FromCell, FromPart, FromSheet, ToCell, ToPart, ToSheet

```
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
```

Close

FromPart property

Applies to: Connect

Summary: Returns the part of a shape from which a connection originates.

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 defined in 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

Close

Example for FromPart

```
*FromCell Property
```

Close

FromSheet property

Applies to: Connect, Connects

Summary: Returns the shape from which connections originate.

Syntax: objRet = object.**FromSheet**

Element	Description
objRet	The shape from which the connections originate
object	The Connect object or Connects collection to examine

See also: GlueTo method, ToSheet property

Close

Example for FromSheet

*FromCell Property

FullName property

Applies to: Document

Summary: Returns the name of a document, including the drive and path.

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.

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

Close

Example for GeometryCount

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.

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 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.)

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 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.

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

Close

Example for GlueToPos

Green property

Applies to: Color

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

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

Close

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.

Syntax: object.**Group**

Element	Description
object	The object to group

See also: AddToGroup method, ConvertToGroup method, RemoveFromGroup method, Ungroup method

Close

Example for Group

```
*Drop Method
```

Group property

Applies to: Entity

Summary: Specifies the Group type of an Entity object.

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 may only use 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.

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.

Example for Handle

Help property

Applies to: Shape

Summary: Sets or returns the help string for a shape.

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.

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. For a list of command IDs, see visCmd* in VISCONST.BAS.

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 Visio's built-in 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.

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 Visio's built-in user interface.

See also: HelpContextID property, HelpPaths property

Example for HelpFile

HelpPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for help files.

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: 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.

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.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, StartupPaths property, StencilPaths property, TemplatePaths property

Close

Example for HelpPaths

IconFileName property

Applies to: StatusBarItem, ToolbarItem

Summary: Sets a custom icon file to be used for an item in a toolbar or status bar.

Syntax: object.**IconFileName** = fileStr

Element	Description
object	The object that loads the icon file
fileStr	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.)

Close

Example for IconFileName

IconSize property

Applies to: Master

Summary: Returns or sets the size of a master icon.

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 defined in VISCONST.BAS show the possible values for IconSize:

visNormal = 1
visTall = 2
visWide = 3
visDouble = 4

Close

Example for IconSize

IconUpdate property

Applies to: Master

Summary: Determines whether a master icon is updated manually or automatically.

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 defined in VISCONST.BAS show the possible values for IconUpdate:

visAutomatic = 1
visManual = 0

See also: IconSize property

Close

Example for IconUpdate

Close

ID property

Applies to: Font

Summary: Returns the ID of a Font object.

Syntax: intVal = object.**ID**

Element	Description
intVal	The ID of the Font object
object	The Font object to examine

Remarks: 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 example below.

Note that the ID associated with a particular font will vary from system to system or as fonts are installed and removed on a given system.

See also: CharProps method, ItemFromID property

Close

Example for ID

To apply "Arial" to all the text of the shape shpObj:

```
    Dim fontObj As Object
    Set fontObj = shpObj.document.fonts("Arial")
    shpObj.characters.charprops(visCharacterFont) = fontObj.id
```

To determine the name of the font applied to the first character of shpObj's text:

```
    ' First character of text is "covered" by first row in shape's
    ' character properties section.
    Dim fontID As Integer
    Dim fontObj As Object
    Dim fontName As String
    fontID = shpObj.Cells(visSectionCharacter,0,visCharacterFont)
    Set fontObj = shpObj.document.fonts.ItemFromID(fontID)
    fontName = fontObj.name
```

Import method

Applies to: Page

Summary: Imports a file into Visio.

Syntax: objRet = object.**Import**(stringExpression)

Element	Description
objRet	A Shape object that represents the new shape imported from the file
object	The Page object 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. 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

IncludesFill property

Applies to: Style

Summary: Indicates whether the style includes fill attributes.

Syntax: boolRet = object.**IncludesFill**
object.**IncludesFill** = intExpression

Element	Description
boolRet	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

Close

Example for IncludesFill

IncludesLine property

Applies to: Style

Summary: Indicates whether the style includes line attributes.

Syntax: boolRet = object.**IncludesLine**
object.**IncludesLine** = intExpression

Element	Description
boolRet	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

Close

Example for IncludesLine

IncludesText property

Applies to: Style

Summary: Indicates whether the style includes text attributes.

Syntax: boolRet = object.**IncludesText**
object.**IncludesText** = intExpression

Element	Description
boolRet	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

Close

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.

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

Close

Example for Index

InPlace property

Applies to: Document

Summary: Determines whether or not a Document object is open in place.

Syntax: boolVal = object.**InPlace**

Element	Description
boolVal	TRUE if the Document object is open in place; otherwise FALSE
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

InstanceHandle property

Applies to: Application

Summary: Returns the instance handle of the Application object.

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.

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

IsConstant property

Applies to: Cell

Summary: Returns TRUE if the formula of the cell is a constant expression.

Syntax: boolRet = object.**IsConstant**

Element	Description
boolRet	-1 if the object's formula is a constant, 0 if it is not
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.

Syntax: boolRet = object.**IsField**

Element	Description
boolRet	TRUE if the object represents only the expanded text of a field; otherwise FALSE
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

Close

Example for IsField

IsHierarchical property

Applies to: MenuItem

Summary: Indicates whether a MenuItem represents a hierarchical submenu.

Syntax: boolVal = object.**IsHierarchical**

Element	Description
boolVal	TRUE if the object represents a submenu; otherwise FALSE
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

IsInherited property

Applies to: Cell

Summary: Returns TRUE if the formula of the cell is inherited from a master or a style.

Syntax: boolRet = object.**IsInherited**

Element	Description
boolRet	-1 if the object's formula is inherited, 0 if it is not
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

Example for IsInherited

IsSeparator property

Applies to: MenuItem

Summary: Indicates whether a MenuItem object represents a separator on a menu.

Syntax: boolVal = object.**IsSeparator**

Element	Description
boolVal	TRUE (-1) if the menu item is a separator, FALSE (0) if it is not.
object	The MenuItem object to examine

Remarks: The CmdNum property of a MenuItem object that represents a separator is 0.

See also: CmdNum property

Close

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.

Syntax: boolRet = object.**IsVisio16**

Element	Description
boolRet	-1 if object is a Win16 instance, 0 if it is not
object	The Application object to examine

See also: IsVisio32 property

Close

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.

Syntax: boolRet = object.**IsVisio32**

Element	Description
boolRet	-1 if the object is a Win32 instance, 0 if it is not
object	The Application object to examine

See also: IsVisio16 property

Close

Example for IsVisio32

Close

Item property

Applies to: AccelItems, AccelTables, Addons, Attributes, Colors, Connects, Documents, Entities, EntityApps, EventList, Fonts, Layers, Masters, MenuItems, Menus, MenuSets, Pages, Selection, Shapes, StatusBarItems, StatusBars, Styles, ToolbarItems, Toolbars, ToolbarSets, Windows

Summary: Returns an object from a collection.

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, or Shapes 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")
```

See also: Index property, UniqueID property

Close

Example for Item

```
' Item method example.  To see a collection iteration using Count and Item
' look at the Shapes property example.
'

Dim appVisio As Object, docObj As Object, pagObj As Object

Set appVisio = CreateObject("visio.application")
Set docObj = appVisio.Documents.Add("")

' The following two lines are identical...
'
Set pagObj = docObj.Pages(1)                    '-- Default Method Syntax
Set pagObj = doc.Obj.Pages.Item(1)               '-- Regular Syntax
```

ItemAtID property

Applies to: AccelTables, MenuSets, StatusBars, ToolbarSets

Summary: Returns the AccelTable, MenuSet, StatusBar, or ToolbarSet object for the indicated ID within the collection.

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 in VISCONST.BAS.

See also: AddAtID method, Item property

Example for ItemAtID

Close

ItemFromID property

Applies to: Fonts

Summary: Returns a Font object that has the indicated ID in a Fonts collection.

Syntax: objRet = object.**ItemFromID**(fontID)

Element	Description
objRet	The Font object retrieved from the collection
object	The Fonts collection that contains the object
fontID	The ID of the font to retrieve

Remarks: 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.

See also: CharProps method, ID property

Close

Example for ItemFromID

To determine the name of the font applied to the first character of shpObj's text:

```
    ' First character of text is "covered" by first row in shape's
    ' character properties section.
    Dim fontID As Integer
    Dim fontObj As Object
    Dim fontName As String
    fontID = shpObj.Cells(visSectionCharacter,0,visCharacterFont)
    Set fontObj = shpObj.document.fonts.ItemFromID(fontID)
    fontName = fontObj.name
```

Key property

Applies to: AccelItem

Summary: Gets or sets the ASCII key code value for an accelerator.

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.

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.

See also: Creator property, Description property, Subject property, Title property

Close

Example for Keywords

Language property

Applies to: Application

Summary: The language ID of the version of the Visio instance represented by the Application object.

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.

Close

Example for Language

Layer property

Applies to: Shape

Summary: Returns the i'th layer to which a shape is assigned.

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: LayerCount property

Close

Example for Layer

LayerCount property

Applies to: Shape

Summary: Returns the number of layers to which a shape is assigned.

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

LayerName property

Applies to: Entity

Summary: Specifies the layer name represented by an Entity object.

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.

Close

Example for LayerName

Layers property

Applies to: Master, Page

Summary: Returns the Layers collection of the indicated object.

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

Close

Example for Layers

LengthIU property

Applies to: Shape

Summary: Returns the length (perimeter) of the object in internal units.

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

Close

Example for LengthIU

LineBasedOn property

Applies to: Style

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

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

Close

Example for LineBasedOn

LineStyle property

Applies to: Selection, Shape

Summary: Specifies the line style for an object.

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 applying 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

Close

Example for LineStyle

LineStyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a line style to an object while preserving local formatting.

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

Close

Example for LineStyleKeepFmt

LoadFromFile method

Applies to: UI Object

Summary: Loads a Visio UI object from a file.

Syntax: object.**LoadFromFile**(stringExpression)

Element	Description
stringExpression	The name of the file to load
object	The UI object to receive data from the file

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

LocalName property

Applies to: Cell

Summary: Returns the local name of a cell.

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.

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: RealValue property, ShortValue property

Example for LongValue

Master property

Applies to: Layer, Layers, Shape

Summary: Returns the master from which the Shape object was created or the master that contains the Layer or Layers object.

Syntax: objRet = object.**Master**

Element	Description
objRet	A Master object that represents the object's master
object	The object to examine

Remarks: If the Shape object is not an instance of a master, its Master property returns Nothing.

If the Layer or Layers object is from a page rather than 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 Shape object is a guide, its Master property returns Nothing.

See also: Page property

Close

Example for Master

Close

Masters property

Applies to: Document

Summary: Returns the Masters collection for the indicated document's stencil.

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

Close

Example for Masters

```
' Print all master names in the current document to the debug window.
' Open a document before using.
'
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
```

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.

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.

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 its submenu items. Otherwise, its MenuItems collection is empty.

Example for MenuItems

Menus property

Applies to: MenuSet

Summary: Returns the Menus collection of a MenuSet object.

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.

Example for Menus

MenuSets property

Applies to: UI Object

Summary: Returns the MenuSets collection of a UI object.

Syntax: objRet = object.Menus

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

Close

Example for MenuSets

MiniHelp property

Applies to: MenuItem

Summary: Gets or sets the string that appears in the status bar when a menu item is selected.

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, it uses the default minihelp text from Visio's built-in user interface.

Close

Example for MiniHelp

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.

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

Example for Name

NameID property

Applies to: Shape

Summary: Returns the unique identifier for an object.

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. The identifier has the following form:

sheet.X

where X is a number from 1 to 4095.

See also: EventInfo property, Name property

Close

Example for NameID

OnDataChangeDelay property

Applies to: Application

Summary: Controls whether a container application updates a Visio object that is in place in the container.

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.

Syntax: retVal = object.**OneD**
object.**OneD** = {True | False}

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

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.

Close

Example for OneD

Close

Open method

Applies to: Documents

Summary: Opens an existing Visio file.

Syntax: objRet = object.**Open** (stringExpression)

Element	Description
objRet	A Document object that represents the file that was opened
object	The Documents collection to receive the opened file
stringExpression	The name of the file

Remarks: The Open method 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.

See also: Add method, Drop method, OpenEx method

Close

Example for Open

```
' Demonstrate methods for opening files.
'
Sub OpenDoc ()
  Dim appVisio As Object, CurDoc As Object

  Set appVisio = CreateObject("visio.application")

' The next line opens a blank document (not based on a template).  The
' Name property is synthesized because the document is unsaved by default.

  Set CurDoc = appVisio.Documents.Add("")
  Debug.Print "             NewDoc : "; CurDoc.Name

' Open a new document based on a template.

  Set CurDoc = appVisio.Documents.Add("c:.vst")
  Debug.Print "  Based On Template : "; CurDoc.Name

' Open a document in original mode.

  Set CurDoc = appVisio.Documents.Open("c:.vst")
  Debug.Print "           Open Doc : "; CurDoc.Name
End Sub
```

OpenEx method

Applies to: Documents

Summary: Opens an existing Visio file.

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 specifiy how the document is to be opened. 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

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.

Syntax: objRet = object.**Page**
object.**Page** = stringExpression

Element	Description
objRet	The Page object in the window or the Page object that contains the layer or layers
object	The Window object, Layer object, or Layers collection
stringExpression	The name of the page to display in the indicated window

Remarks: Setting a window's Page property makes the specified page the active 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

Close

Example for Page

Close

Pages property

Applies to: Document

Summary: Returns the Pages collection of a document.

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

Close

Example for Pages

```
' Prints the names of a document's pages.  Make sure a document is open.
'
Sub PrintPageNames ()
  Dim I As Integer
  Dim appVisio As Object, CurDoc As Object, PageList As Object

  Set appVisio = GetObject(, "visio.application")
  Set CurDoc = appVisio.ActiveDocument

  If CurDoc Is Nothing Then Exit Sub

  Set PageList = CurDoc.Pages

  Debug.Print "Page names for document : "; CurDoc.Name

  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.

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 method 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")
```

Close

Example for PageSheet

PaletteEntry property

Applies to: Color

Summary: Gets or sets the red, green, blue, and flags components of the color.

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

ParaProps method

Applies to: Characters

Summary: Sets the indicated paragraph property of a Characters object to a new value.

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 defined in 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: ParaPropsRow property

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.

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:

Symbol	Value
visBiasLeft	1
visBiasRight	2

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 method, TabPropsRow property

Close

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.

Syntax: objRet = object.**Parent**

Element	Description
objRet	The parent of the indicated object
object	The object to examine

Remarks: If a Shape object is a member of a group, the parent is that group. Otherwise, the returned object will be a page or a Master object.

Example for Parent

Close

Paste method

Applies to: Characters, Page, Window

Summary: Pastes the contents of the Windows Clipboard into the indicated object.

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

Close

Example for Paste

```
*Copy Method
```

Path property

Applies to: Document

Summary: Returns the drive and folder path of a document, without the document's filename.

Syntax: strRet = object.**Path**

Element	Description
strRet	The path of the indicated document
object	The Document object to examine

Remarks: 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

PitchAndFamily property

Applies to: Font

Summary: Returns the pitch and family code for a Font object.

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

Close

Example for PitchAndFamily

Print method

Applies to: Document, Page

Summary: Prints the contents of an object to the default printer.

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.

Enclose Print in brackets to distinguish it from the Visual Basic keyword.

Close

Example for Print

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.

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

Close

ProcessID property

Applies to: Application

Summary: Returns a unique process ID for the indicated instance of Visio.

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

```
Sub ShowProcessID ()
    Dim vis As Object

    Set vis = CreateObject("visio.application")

    Debug.Print "Visio Process ID : "; vis.ProcessID
End Sub
```

ProfileName property

Applies to: Application

Summary: Returns the name of the Visio application object's profile (.ini) file.

Syntax: strRet = object.**ProfileName**

Element	Description
strRet	The name of Visio's profile (.ini) file
object	An Application object

Example for ProfileName

Prompt property

Applies to: Attribute, Master

Summary: Returns or sets the prompt string for the indicated object.

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.

Syntax: boolRet = object.**PromptForSummary**
object.**PromptForSummary** = intExpression

Element	Description
boolRet	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

Close

PutShape method

Applies to: ShapeData

Summary: Sets the shape to be used by the ShapeData object.

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 you want to attach the database to.

Close

Example for Attributes, DefaultValue, EntityApps, PutShape, Value

```
Sub DatabaseTest ()
    Dim ShapeData As Object
    Dim shp As Object
    Dim Visio As Object
    Dim Attribs As Object
    Dim Attrib As Object
    Dim EntityApps As Object
    Dim EntityApp As Object
    Dim Entities As Object
    Dim Entity As Object

    ' First we get Visio and the Shape database

    Set ShapeData = CreateObject("Visio.ShapeDatabase")
    Set Visio = CreateObject("visio.application")

    Set shp = Visio.Documents.Add("").Pages(1).DrawRectangle(1, 2, 2, 1)

    ' Before using the Shape database we must initalize it using
    ' the PutShape property.  To speed things up, we use the
    ' BeginTransaction action property to force the database to
    ' delay all reads and writes until a matching EndTransaction.

    ShapeData.PutShape shp
    ShapeData.BeginTransaction

    ' First we demonstrate working with attribute data

    Set Attribs = ShapeData.Attributes
    Set Attrib = Attribs.Add("Part #")
    Attrib.Value = "XYZ-123"
    Attrib.DefaultValue = "xxx-xxx"
    Attrib.Prompt = "Please enter the part #"

    ' Next we demonstrate Extended Entity Data.  To add Extended Entity
    ' Data to a shape, you must first create an EntityApp object
    ' to represent your application.

    Set EntityApps = ShapeData.EntityApps
    Set EntityApp = EntityApps.Add("TestApp")

    ' Now that we have an EntityApp, we can retrieve its Entities
    ' collection and add an Entity.  For our test we will add a
    ' vector Entity.

    Set Entities = EntityApp.Entities
    Set Entity = Entities.Add

    Entity.VectorX = 1.23
    Entity.VectorY = -423.1002
    Entity.VectorZ = 12.932
```

```
    ' To ensure that our transactions above are written back to the
    ' shape, we must call EndTransaction.  Failing to do so would
    ' cause any changes to be ignored.

    ShapeData.EndTransaction
End Sub
```

Quit method

Applies to: Application

Summary: Closes the indicated instance of Visio.

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.

Example for Quit

ReadOnly property

Applies to: Document

Summary: Indicates whether the file was opened as read-only.

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.

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.

Close

Example for RealValue

Red property

Applies to: Color

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

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

Close

Example for Red

Close

Redo method

Applies to: Application

Summary: Reverses the most recent Undo command.

Syntax: object.**Redo**

Element	Description
object	The Application object in which to reverse Undo

Remarks: Use the Redo method to reverse the effect of the Undo 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

Close

Example for Redo, Undo

```
Sub TestDos ()
' Demonstrate use of Repeat, Undo, and Redo methods.
'
  Dim appVisio As Object, DrawPage As Object, shp As Object

  Set appVisio = CreateObject("visio.application")

'--   First set up Visio so that there is a document and a page
'-- to draw on.

  appVisio.Documents.Add ""
  Set DrawPage = appVisio.ActivePage

'--   Next, we draw a demonstration rectangle, delete it using Undo, and then
'-- redraw it using Redo.

  Set shp = DrawPage.DrawRectangle(1, 5, 5, 1)

  appVisio.Undo                          '-- Delete the shape
  appVisio.Redo                          '-- Bring it back

  appVisio.Quit
End Sub
```

Remove method

Applies to: Layer

Summary: Removes a shape from a layer.

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.

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.

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 a cell's value
newValue	The new value for the cell

Remarks: Use the Result method 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 defined in VISCONST.BAS:

```
visNumber = 32              'non-dimensional number
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
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
visRadians = 83             'angle in radians
visMin = 84                 'angle in minutes-seconds
visSec = 85                 'angle in seconds
```

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, ResultStr property

Example for Result

ResultForce property

Applies to: Cell

Summary: Sets a cell's value, even if the cell's formula is protected with the GUARD function.

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, ResultIU property, ResultIUForce property, ResultStr property

Example for ResultForce

ResultInt property

Applies to: Cell

Summary: Gets the value of a cell expressed as an integer, or sets the value to an integer value.

Syntax: intRet = object.**ResultInt**(units,roundFlag)
object.**ResultInt**(units) = newValue

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 or setting a cell's value
roundFlag	0 to truncate the value, non-zero to round it
newValue	The new value for the cell

Remarks: ResultInt is analogous to a cell's Result property. The difference is that ResultInt accepts or returns an integer for the value of the cell, whereas Result accepts or 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.

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.

If the cell's formula is protected with a GUARD function, use ResultIntForce.

See also: Result property, ResultIntForce property, ResultIU property, ResultStr property

Example for ResultInt

ResultIntForce 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.

Syntax: object.**ResultIntForce**(units) = newValue

Element	Description
object	The Cell object that gets the new value
units	The units to use when retrieving or setting a cell's value
newValue	The new value for the cell

Remarks: Use the ResultIntForce method 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 ResultInt.

See also: Result property, ResultInt property, ResultStr property

Close

Example for ResultIntForce

ResultIU property

Applies to: Cell

Summary: Returns or sets a cell's value in internal units.

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 method 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.

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, ResultForce property, ResultIU property, ResultStr property

Close

Example for ResultIUForce

ResultStr property

Applies to: Cell

Summary: Gets the value of a cell expressed as a string.

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. You can get the value in inches, then get the same value in centimeters, for example.

See also: Result property, ResultInt property

Example for ResultStr

ReverseEnds method

Applies to: Selection, Shape

Summary: Reverses an object by flipping it both horizontally and vertically.

Syntax: object.**ReverseEnds**

Element	Description
object	The Shape or Selection object to reverse

See also: FlipHorizontal method, FlipVertical method, Rotate90 method

Close

Example for ReverseEnds

Rotate90 method

Applies to: Selection, Shape

Summary: Rotates an object 90 degrees counterclockwise.

Syntax: object.**Rotate90**

Element	Description
object	The Shape or Selection object to rotate

See also: FlipHorizontal method, FlipVertical method, ReverseEnds method

Close

Example for Rotate90

Row property

Applies to: Cell, Layer

Summary: Returns the row index of a cell or layer.

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.

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.

See also: AddRow method, AddSection method, GeometryCount property, RowsCellCount property

Close

Example for RowCount

RowExists property

Applies to: Shape

Summary: Returns TRUE if the indicated ShapeSheet row exists.

Syntax: boolRet = object.**RowExists**(section,row,fExistsLocally)

Element	Description
boolRet	-1 if the row exists, 0 if it does not
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.

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 or Custom properties 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

Close

Example for RowName

RowsCellCount property

Applies to: Shape

Summary: Returns the number of cells in a row of a ShapeSheet section.

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 defined in VISCONST.BAS. For a list of constants, see the AddRow and AddSection methods.

See also: AddRow method, AddSection method, RowCount property

Example for RowsCellCount

Close

RowType property

Applies to: Shape

Summary: Returns or sets the type of a row in a Geometry section of a ShapeSheet.

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: Use the RowType property to change the type of a row in a Geometry section. 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

Close

Example for RowType

```
*AddSection Method
```

Run method

Applies to: Addon

Summary: Runs the add-on represented by an Addon object.

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.

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.

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

Save method

Applies to: Document

Summary: Saves a document.

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.

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 SaveAs

SaveAsEx method

Applies to: Document

Summary: Saves a document.

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 specifiy 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

Saved property

Applies to: Document

Summary: Determines whether a document has any unsaved changes.

Syntax: retVal = object.**Saved**
object.**Saved** = boolVal

Element	Description
retVal	TRUE if the document has no unsaved changes; otherwise FALSE
object	The Document object that has or gets the setting
boolVal	TRUE to indicate the document is saved; FALSE 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 will be 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.

Close

Example for Saved

SavePreviewMode property

Applies to: Document

Summary: Determines whether Visio saves a preview of a document when the document is saved.

Syntax: boolRet = object.**SavePreviewMode**
object.**SavePreviewMode** = intExpression

Element	Description
boolRet	0 if the property is not set; otherwise -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

Remarks: This property corresponds to the Save Preview Picture checkbox in Visio's properties dialog box.

Example for SavePreviewMode

SaveToFile method

Applies to: UI Object

Summary: Saves the user interface represented by a UI object in a file.

Syntax: object.**SaveToFile**(fileName)

Element	Description
object	The UI object to save to the file
fileName	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 SaveToFile

SaveWorkspaceAs method

Applies to: Application

Summary: Saves the current workspace.

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 on Visio's File menu. The specified filename should have a .VSW extension.

Example for SaveWorkspaceAs

ScreenUpdating property

Applies to: Application

Summary: Determines whether or not the Visio screen is updated (redrawn) during a series of actions.

Syntax: boolRet = object.**ScreenUpdating**
object.**ScreenUpdating** = intExpression

Element	Description
boolRet	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 will turn screen updating back on when a user performs an operation.

Example for ScreenUpdating

Section property

Applies to: Cell

Summary: Returns the section index of a cell.

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.

Syntax: boolRet = object.**SectionExists**(section,fExistsLocally)

Element	Description
boolRet	0 if cell exists, -1 if it does not
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, CellsExists property, CellsSRC property, CellsSRCExists property, RowExists property

Close

Example for SectionExists

Close

Select method

Applies to: Selection, Window

Summary: Selects or deselects an object.

Syntax: object.**Select** addObj, selectType

Element	Description
object	The Window or Selection object that contains the shapes
addObj	The Shape object to add to selection
selectType	The type of selection to make

Remarks: The following constants defined in VISCONST.BAS show valid values for selection types:

visDeselect = &H1
visSelect = &H2
visSubSelect = &H3
visDeselectAll = &H100

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

Close

Example for Select

```
Sub SelectDemo ()
' Demonstrates the use of the Selection method
'
    Const MAX_SHAPES = 6
    ReDim shps(1 To MAX_SHAPES) As Object

    Dim I As Integer, bCreate As Integer
    Dim vis As Object, doc As Object
    Dim pag As Object, win As Object, shp As Object

    ' Get Visio instance.
    '
    On Error GoTo lblCatchErr

    bCreate = False
    Set vis = GetObject(, "visio.application")

    On Error GoTo lblErr

    If vis Is Nothing Then
        bCreate = True
        Set vis = CreateObject("visio.application")
    End If

    ' Set up our objects.  Need a blank document, its first page, and the
    ' active window.
    '
    Set doc = vis.Documents.Add("")
    Set pag = doc.Pages(1)
    Set win = vis.ActiveWindow

    ' Draw some shapes to select.
    '
    For I = 1 To MAX_SHAPES
        Set shps(I) = pag.DrawRectangle(I, I + 1, I + 1, I)
    Next I

    ' Set up a group for testing sub-selections by selecting the first
    ' three shapes on the page, grouping them, and storing the group in shp.
    ' Although the first three shapes are grouped, the array shps() still
contains
    ' them.
    '
    win.DeselectAll

    For I = 1 To 3
        win.Select shps(I), visSelect
    Next I

    win.Group

    Set shp = pag.Shapes(pag.Shapes.Count)
```

```
    ' Now, we have (MAX_SHAPES - 3) shapes on the page with three shapes in
the group.
    ' Sub selection is accomplished by having the parent shape selected first
    ' or one of the groups shapes already subselected.
    '
    win.Select shp, visDeselectAll + visSelect           ' Select parent
    win.Select shps(1), visSubSelect
    win.Select shps(3), visSubSelect

    ' Next, select just one shape.  At this point two shapes are
    ' subselected but we 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.
    '
    win.Select shps(MAX_SHAPES), visDeselectAll + visSelect
    win.Select shps(MAX_SHAPES - 1), visSelect
    win.Select shp, visSelect

    ' Now remove just one shape from the window selection.
    '
    win.SelectAll
    win.Select shps(MAX_SHAPES - 1), visDeselect

    ' Close the document and quit Visio only if a new instance was created.
    '
    pag.Document.[Close]
    If bCreate Then vis.Quit
    Exit Sub

lblCatchErr:
    Resume Next

lblErr:
    MsgBox "An error occurred."
    Exit Sub
End Sub
```

SelectAll method

Applies to: Selection, Window

Summary: Selects all possible shapes in a window or selection.

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

Close

Example for SelectAll

Selection property

Applies to: Window

Summary: Returns a Selection object.

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

SendBackward method

Applies to: Selection, Shape

Summary: Moves a shape or selected shapes back one position in the z-order.

Syntax: object.**SendBackward**

Element	Description
object	The Shape or Selection object to send backward

See also: BringForward method, BringToFront method, SendToBack method

Example for SendBackward

SendToBack method

Applies to: Selection, Shape

Summary: Moves the shape or selected shapes to the back of the z-order.

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

SetBegin method

Applies to: Shape

Summary: Moves the begin point of a 1-D shape to the coordinates represented by x and y.

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 center of rotation is positioned at the coordinates represented by x and y.

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: 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.

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 (that is, by using CreateObject instead of getting the appropriate property of an Application or Document object), SetCustomMenus returns an error.

See also: SetCustomToolbars method

Close

Example for SetCustomMenus

SetCustomToolbars method

Applies to: Application, Document

Summary: Replaces the current built-in or custom toolbars of an Application or Document object.

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 (that is, by using CreateObject instead of getting the appropriate property of an Application or Document object), SetCustomToolbars returns an error.

See also: SetCustomMenus method

Close

Example for SetCustomToolbars

SetEnd method

Applies to: Shape

Summary: Moves the end point of a 1-D shape to the coordinates represented by x and y.

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

SetID property

Applies to: AccelTable, MenuSet, StatusBar, ToolbarSet

Summary: Returns the set ID of the object in its collection.

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 defined in 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

Shape property

Applies to: Cell, Characters, ShapeData

Summary: Returns the Shape object that owns a Cell or Characters object or that is associated with a ShapeData object.

Syntax: objRet = object.**Shape**

Element	Description
objRet	The Shape object that contains or is associated with the object
object	The object to examine

Close

Example for Shape

Close

Shapes property

Applies to: Master, Page, Shape

Summary: Returns the Shapes collection for an object.

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

Close

Example for Shapes

```
Sub DumpShapeNames ()
' Print all shape names on page 1 of the current document to the debug window.
' Open a document before using this procedure.

    On Error Goto lblErr

    Dim I As Integer, iShapeCount As Integer
    Dim appVisio As Object, CurDoc As Object, ShapeColl As Object

    Set appVisio = GetObject(, "visio.application")
    Set CurDoc = appVisio.ActiveDocument

    If CurDoc Is Nothing Then
        MsgBox "No Document Loaded"
        Exit Sub
    End If

    Set ShapeColl = CurDoc.Pages.Item(1).Shapes

    Debug.Print "Shape Name Dump For Document : "; CurDoc.Name
    Debug.Print "                          Page : "; CurDoc.Pages.Item(1).Name

    iShapeCount = ShapeColl.Count

    If iShapeCount > 0 Then
        For I = 1 To iShapeCount
            Debug.Print "  "; ShapeColl.Item(I).Name
        Next I
    Else
        Debug.Print "  No Shapes On Page"
    End If

    Exit Sub
lblErr:
    MsgBox "Visio not running."
    Exit Sub
End Sub
```

Shift property

Applies to: AccelItem

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

Syntax: object.**Shift** = boolVal
boolVal = object.**Shift**

Element	Description
object	The AccelItem object that has or gets the setting
boolVal	TRUE (-1) if Shift should modify 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.

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: LongValue property, RealValue property

Example for ShortValue

Spacing property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the spacing before and after a toolbar or status bar item.

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 defined in VISCONST.BAS:

visCtrlSpacingNONE% = &H0
visCtrlSpacingVARIABLE_BEFORE% = &H1
visCtrlSpacingVARIABLE_AFTER% = &H2
visCtrlSpacingFIXED_BEFORE% = &H4
visCtrlSpacingFIXED_AFTER% = &H8

Example for Spacing

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.

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

Element	Description
strRet	A list of directories
object	An Application object
pathsStr	A list of directories

Remarks: 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.

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.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, HelpPaths property, StencilPaths property, TemplatePaths property

Example for StartupPaths

StatusBarItems property

Applies to: StatusBar

Summary: Returns the StatusBarItems collection of a StatusBar object.

Syntax: objRet = object.**StatusBarItems**

Element	Description
objRet	The StatusBarItems collection of the StatusBar object
object	The StatusBar object that owns the collection

See also: StatusBar object, StatusBarItems object

Example for StatusBarItems

StatusBars property

Applies to: UI Object

Summary: Returns the StatusBars collection of a UI object.

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, UI object

Example for StatusBars

StencilPaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for stencils.

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: 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.

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.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, HelpPaths property, StartupPaths property, TemplatePaths property

Close

Example for StencilPaths

String property

Applies to: Entity

Summary: Specifies the String value contained in an Entity object.

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.

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.

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

StyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a style to an object while preserving local formatting.

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.

Close

Example for StyleKeepFmt

Styles property

Applies to: Document

Summary: Returns the Styles collection for a document.

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

Subject property

Applies to: Document

Summary: Returns or sets the value of the Subject field in a document's properties.

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.

See also: Creator property, Description property, Keywords property, Title property

Close

Example for Subject

SubType property

Applies to: Window

Summary: Returns the subtype of a Window object that represents a drawing window.

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 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

Close

Example for SubType

TableName property

Applies to: AccelTable

Summary: Gets or sets the name of an AccelTable object.

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.

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

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

Close

Example for TabPropsRow

Target property

Applies to: Event

Summary: Gets or sets the target of an event.

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.

Visio 4.0 supports only one action, which is to run an add-on. In this case, the Target property contains the name of the add-on to run.

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.

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

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 4.0 supports only one action, which is to run an add-on. In this case, the TargetArgs property contains the arguments to send to the add-on when it is run.

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.

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

Close

Example for Template

TemplatePaths property

Applies to: Application

Summary: Gets or sets the paths where Visio looks for templates.

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: 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.

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.

See also: AddonPaths property, DrawingPaths property, FilterPaths property, HelpPaths property, StartupPaths property, StencilPaths property

Close

Example for TemplatePaths

Text property

Applies to: Characters, Shape

Summary: Returns or sets the text of the object.

Syntax: strRet = object.**Text**
object.**Text** = stringExpression
object.**Text** = objectExpression

Element	Description
strRet	The current text of the Shape or Characters object
object	The Shape or Characters object that owns the text
stringExpression	New text for the Shape object
objectExpression	Another Shape or Characters object that represents new text for the Shape or Characters object

Remarks: The Text property of a Shape object returns the entire text of the shape. Fields are represented as 4-character escape sequences.

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. Fields are expanded to the number of characters that are visible in the drawing window.

For example, suppose a shape's text contains a field that displays the filename of a drawing. For a Shape object representing that shape, the Text property returns the 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 and guides have no Text property.

See also: Characters object

Example for Text

TextBasedOn property

Applies to: Style

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

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

Close

Example for TextBasedOn

TextStyle property

Applies to: Selection, Shape

Summary: Returns or sets the text style for an object.

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

Close

Example for TextStyle

TextStyleKeepFmt property

Applies to: Selection, Shape

Summary: Applies a text style to an object while preserving local formatting.

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

Close

Example for TextStyleKeepFmt

Title property

Applies to: Document

Summary: Returns or sets the value of the Title field in a document's properties.

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.

See also: Creator property, Description property, Keywords property, Subject property

Close

Example for Title

Close

ToCell property

Applies to: Connect

Summary: Returns the cell to which a connection is made.

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

Close

Example for ToCell

```
*FromCell Property
```

ToolbarItems property

Applies to: Toolbar

Summary: Returns the ToolbarItems collection of a Toolbar object.

Syntax: objRet = object.**ToolbarItems**

Element	Description
objRet	The ToolbarItems collection of the Toolbar object
object	The Toolbar object that owns the collection

See also: Toolbar object, ToolbarItems object

Close

Example for ToolbarItems

Toolbars property

Applies to: ToolbarSet

Summary: Returns the Toolbars collection of a ToolbarSet object.

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, ToolbarSet object

Close

Example for Toolbars

ToolbarSets property

Applies to: UI Object

Summary: Returns the ToolbarSets collection of a UI object.

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, UI object

Close

Example for ToolbarSets

Close

ToPart property

Applies to: Connect

Summary: Specifies the part of a shape to which a connection is made.

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 defined in 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

Close

Example for ToPart

```
*FromCell Property
```

Close

ToSheet property

Applies to: Connect

Summary: Returns a Shape object that represents the shape to which a connection is made.

Syntax: objRet = object.**ToSheet**

Element	Description
objRet	The shape to which the connection is made
object	The Connect object to examine

See also: FromSheet property, GlueTo method

Close

Example for ToSheet

```
*FromCell Property
```

Trigger method

Applies to: Cell, Event

Summary: Evaluates the formula of a cell or causes an event's action to be performed.

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, the string is passed in the command line string sent to the add-on.

See also: Event object

Example for Trigger

Type property

Applies to: Shape, Window

Summary: Returns the type of the object.

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 defined in VISCONST.BAS show possible values that Type returns.

The Type property of a Shape object returns one of the following:

visTypeStyle = 6
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 Type

TypeSpecific1 property

Applies to: StatusBarItem, ToolbarItem

Summary: Gets or sets the type of a toolbar or status bar item.

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 in 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 visCtrlTypePUSHBUTTON, TypeSpecific1 can be any constant prefixed with visStrID in VISCONST.BAS.

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 in VISCONST.BAS.

If CntrlType is visCtrlTypeMESSAGE, TypeSpecific1 can be any constant prefixed with visCtrlAlignment in 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.

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

Close

Example for TypeSpecific2

Close

Undo method

Applies to: Application

Summary: Reverses the most recent action, if the action can be reversed.

Syntax: object.**Undo**

Element	Description
object	The Application object in which to reverse the action

Remarks: You can reverse up to the last 10 actions, one action at a time. You can undo most actions but not all. Use the Redo method to reverse the effect of the Undo method.

See also: Redo method

Example for Undo

```
*Redo Method
```

Ungroup method

Applies to: Selection, Shape

Summary: Ungroups a group.

Syntax: object.**Ungroup**

Element	Description
object	The Shape or Selection object to ungroup

See also: AddToGroup method, Group method, RemoveFromGroup method

Example for Ungroup

Union method

Applies to: Window

Summary: Creates a new shape from the perimeter of selected shapes in a window.

Syntax: object.**Union**

Element	Description
object	The Window 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 original shapes are deleted.

See also: Combine method, Fragment method

Close

Example for Union

UniqueID property

Applies to: Master, Shape

Summary: Returns or clears the unique ID of the indicated object.

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.

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 defined in 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.

Close

Example for Units

UpdateUI method

Applies to: UI Object

Summary: Causes Visio to display changes to the user interface represented by a UI object.

Syntax: object.**UpdateUI**

Element	Description
object	The UI object that represents the user interface that was changed

Remarks: The UpdateUI method updates Visio's 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.

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

Close

Value property

Applies to: Attribute

Summary: Returns or sets the value of an Attribute object.

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

Close

Example for Value

```
*PutShape Method
```

VectorX property

Applies to: Entity

Summary: Specifies the X component of an Entity object.

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

Close

Example for VectorX

VectorY property

Applies to: Entity

Summary: Specifies the Y component of an Entity object.

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

Close

Example for VectorY

VectorZ property

Applies to: Entity

Summary: Specifies the Z component of an Entity object.

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

Close

Example for VectorZ

Close

Version property

Applies to: Application, Document

Summary: Returns the version of a running instance of Visio or determines the version of a saved document.

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 4.0 is "4.0".

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 defined in VISCONST.BAS. The versions Visio 4.0 can save are:

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 4.0 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 4.0 converts the in-memory representation of every document it opens to 4.0 format.

Example for Version

```
Sub ShowVersion ()
    Dim vis As Object
    Dim strVer As String
    Dim iDotPos As Integer

    Set vis = CreateObject("visio.application")

    strVer = vis.Version
    iDotPos = InStr(strVer, ".")

    Debug.Print "  Major Version : "; Left(strVer, iDotPos - 1)
    Debug.Print "  Minor Version : "; Right(strVer, Len(strVer) - iDotPos)
End Sub
```

WindowHandle property

Applies to: Application

Summary: Returns the window handle for an instance of Visio.

Syntax: retVal = object.**WindowHandle**

Element	Description
retVal	The window handle for Visio's main frame window (a 2-byte value)
object	The Application object that represents the frame window

Remarks: Use the WindowHandle property to obtain the HWND for Visio's main window. You can use 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: IsVisio16 property, IsVisio32 property, WindowHandle32 property

Example for WindowHandle

WindowHandle32 property

Applies to: Application

Summary: Returns the window handle for an instance of 32-bit Visio.

Syntax: retVal = object.**WindowHandle32**

Element	Description
retVal	The window handle for Visio's main frame window (4-byte value)
object	The Application object that represents the frame window

Remarks: Use the WindowHandle property to obtain the HWND for Visio's main window. You can use HWND in Windows API calls.

WindowHandle32 returns a 4-byte value. If an Application object represents 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.

See also WindowHandle which returns a 2-byte value. (In Win16 handles are 2-byte values. In Win32 they're 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.

See also: IsVisio16 property, IsVisio32 property, WindowHandle property

Example for WindowHandle32

Windows property

Applies to: Application

Summary: Returns the Windows collection for an instance of Visio.

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

Zoom property

Applies to: Window

Summary: Returns or sets the current display size (magnification factor) for a page in a window.

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.

Close

Example for Zoom

Close

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 connection−it behaves no differently from any other 1-D shape.

Visio is proud to present...

Programming *Visio*

Get real, get...

Project Manager	*Michele DeWilliam*
Editor	*Jim Pursell*
Writer	*Carol Buchmiller*
Reviewers	*Peter Roth* *Tom Booster* *Mike Frederick* *Mitch Boss* *Peter Mullen* *The Application Engineers* *Troy Sandal*
Database	*Troy Sandal*
Setup	*Dan Clay*
Help Guru	*Lori Schultz*
Production	*Jess Rice*
Programming	*Peter Roth* *Mike Frederick* *Michael Kallay* *Tom Booster* *Carol Buchmiller* *Troy Sandal*

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.