Welcome to OLE Automation!

OLE Automation™ is a powerful tool you can use to customize FlowCharter ™ to meet your own specific needs. The extensive power and flexibility of Automation give you endless control over FlowCharter.

OLE Automation can provide seamless integration with outside applications. You can write automation programs that use FlowCharter information to perform tasks in other applications or use data from other applications to create and manipulate FlowCharter charts.

OLE Automation is one of the programs in the FlowCharter package. Together, they provide you with easy, efficient, and powerful Office-compatible tools.

The Help system is designed to let you move back and forth between OLE Automation Help and FlowCharter Help.

{button Related Topics,PI(`',`IDH_RT_Welcome')}

Help on Help
Accessing OLE Automation Help from FlowCharter Help
FlowCharter 7

Objects Collection

Description The Objects collection is below the Chart object. Below the Objects collection are the Object objects. You can have multiple Object objects in the collection.

Properties	Methods
Application	Item
Count	ItemFromAll
Parent	ItemFromAttachments
	ItemFromLines
	ItemFromFieldValue
	ItemFromNumber
	ItemFromShapes
	ItemFromSelection
	ItemFromText
	ItemFromUniqueID
	ResetSearch

{button Related Topics,PI(``,`IDH_RT_ABC_Objects_Collection')}

Count Property

Usage *Collection*.**Count**

Description The **Count** property returns the number of items in a collection. The collections in OLE Automation are the Objects collection, Charts collection, FieldTemplates collection, FieldValues collection, and Menu collection. You often use the **Count** property in a loop along with the **Item** method and one of the ItemFrom methods to search through a collection. The **Count** property is read only.

Data Type Long

Value The number of items in a collection

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Count_Property')}

Count Property Example

This example uses the **Count** property of the Objects collection to find how many objects are in the chart.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Add a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Set Everything = Chart.Objects                       ' Get all objects in the chart
ABC.MsgBox "The current chart contains " + Everything.Count + " objects."
```

{button Other examples,PI(`',`IDH_RT_Count_Property_Example')}

Example 1
Example 2

Item Method (Objects Collection)

Usage *ObjectsCollection*.**Item** (*Count*)
The *Count* element is the index of the item within the collection.

Description Use the **Item** method of the Objects collection to access Object objects within the Objects collection.

Data Type Object

Value The next valid Object object in the collection. If that object does not exist, the method returns Null.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Item_Method_ABC_Collection')}

Finding the Total Number of Objects
Example

Item Method (Charts Collection)
Item Method (FieldTemplates Collection)
Item Method (FieldValues Collection)
Item Method (Menu Collection)

Objects Collection

Item Method (Objects Collection) Example

This example uses the **Item** method of the Objects collection to turn all shapes in a chart red.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object
Dim CurrentShape As Object, CurrentItem As Object
Dim X, Y

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

For X = 1 To 5
   Set CurrentShape = Chart.DrawShape                  ' Draw a shape
Next X

ABC.MsgBox "Click OK to turn all items in the chart red."
Set Everything = Chart.Objects
For Y = 1 To Everything.Count                          ' For all objects in the chart
   Set CurrentItem = Everything.Item(Y)                ' Get the next item
   CurrentItem.Color = ABC.RED                         ' Make the item red
Next Y
```

ItemFromAll Method

Usage *ObjectsCollection*.**ItemFromAll**

Description You use the **ItemFromAll** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromAll** method returns successive objects from the Objects collection.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ItemFromAll_Method')}

Finding Objects in a Chart
Example

Item Method (Objects Collection)
ItemFromAttachments Method
ItemFromFieldValue Method
ItemFromLines Method
ItemFromNumber Method
ItemFromSelection Method
ItemFromShapes Method
ItemFromText Method
ItemFromUniqueID Method
ResetSearch Method

Valid Property

Objects Collection

ItemFromAll Method and Type Property (Object Object) Example

This example uses the **ItemFromAll** method of the Objects collection and the **Type** property of the Object object to identify the types of the items in a chart.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object
Dim OriginalColor As Long

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Add a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Set Everything = Chart.Objects                       ' Get all objects in the chart
Do
   Set Current = Everything.ItemFromAll              ' Check each item
   OriginalColor = Current.Color                     ' Remember the object's original color
   Current.Color = ABC.MakeRGB(255, 64, 0)           ' Make the current object orange
   Select Case Current.Type                          ' Determine the item's type
      Case 0
         ABC.MsgBox "The orange object is a shape."
      Case 1
         ABC.MsgBox "The orange object is a line."
      Case 2
         ABC.MsgBox "The orange object is text."
      Case 3
         ABC.MsgBox "The current object is a bitmap."
      Case 4
         ABC.MsgBox "The current object is an OLE client object."
      Case 5
         ABC.MsgBox "The orange object is a master item."
   End Select
   Current.Color = OriginalColor                     ' Restore the original color
Loop While Current.Valid
```

ItemFromAttachments Method

Usage *ObjectsCollection*.**ItemFromAttachments** (*ObjectWithAttachment1* [, *ObjectWithAttachment2*])
The *ObjectWithAttachment1* element is any object that you want to include in the search.
The *ObjectWithAttachment2* element, which is optional, is any object that you want to include in the search.

Description You use the **ItemFromAttachments** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromAttachments** method returns (from the Objects collection) successive shape, text, or line objects that are attached to the one or two objects you specify.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ItemFromAttachments_Method')}

Finding Objects in a Chart
Example

Item Method (Objects Collection)
ItemFromAll Method
ItemFromFieldValue Method
ItemFromLines Method
ItemFromNumber Method
ItemFromSelection Method
ItemFromShapes Method
ItemFromText Method
ItemFromUniqueID Method
ResetSearch Method

Valid Property

Objects Collection

ItemFromAttachments Method Example

This example uses the **ItemFromAttachments** method of the Objects collection to find a line that has text attached to it.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object, Text1 As Object
Dim Count As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
Set Chart = ABC.ActiveChart                            ' Get the active chart
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart

Set Shape1 = Chart.DrawShape("Connector")              ' Draw shapes
Set Shape2 = Chart.DrawShape("Connector")
Set Line1 = Chart.DrawLine(Shape1, Shape2)             ' Draw a line connecting the shapes
Line1.Repaint                                          ' Refresh the screen
Set Text1 = Chart.DrawTextBlock("Going my way?")       ' Create a text object
Text1.Font.Opaque = True                               ' Make the text background opaque
Line1.Line_.AttachText Text1                           ' Attach the text to the line

Set Everything = Chart.Objects                         ' Get all objects in the chart
Set Current = Everything.ItemFromAttachments(Text1)          ' Find item with text attached
Current.Color = ABC.Red                                ' Make the item red
ABC.MsgBox ("The red object has text attached.")
```

ItemFromLines Method

Usage *ObjectsCollection*.**ItemFromLines**

Description You use the **ItemFromLines** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromLines** method returns, from the Objects collection, successive lines.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ItemFromLines_Method')}

Finding Objects in a Chart
Example

Item Method (Objects Collection)
ItemFromAll Method
ItemFromAttachments Method
ItemFromFieldValue Method
ItemFromNumber Method
ItemFromSelection Method
ItemFromShapes Method
ItemFromText Method
ItemFromUniqueID Method
ResetSearch Method

Valid Property

Objects Collection

ItemFromLines Method, Source Property, and Destination Property Example

This example uses the **ItemFromLines** method of the Objects collection and the **Source** property and **Destination** property of the Line_ object to find a line, its source shape, and its destination shape. This example assumes that two shapes connected by a line already exist on the current chart.

```
Dim ABC As Object, Chart As Object
Dim Line1 As Object
Dim StartShape As Object, EndShape As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Add a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.Select (1)                                   ' Select the lines in the chart
Set Line1 = Chart.Objects.ItemFromLines            ' Assign the line to the variable Line1
Set StartShape = Line1.Line_.Source                ' Find the line's source shape
Set EndShape = Line1.Line_.Destination             ' Find the line's destination shape

StartShape.Text = "Source"                         ' "Source" in source shape
EndShape.Text = "Destination"                      ' "Destination" in destination shape
```

ItemFromFieldValue Method

Usage *ObjectsCollection*.**ItemFromFieldValue** (*FieldTemplateObject, Value*)
The *FieldTemplateObject* element is any FieldTemplate object that you want to examine.
The *Value* element is the value of the FieldTemplate object that you are searching for. The *Value* element is a double or a string, as appropriate for the FieldTemplate object.

Description You use the **ItemFromFieldValue** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromFieldValue** method returns, from the Objects collection, successive objects that contain a field with the value you specify.

Flow Equivalent None

Finding Objects in a Chart
Knowing When Data Fields Change
Example

Item Method (Objects Collection)
ItemFromAll Method
ItemFromAttachments Method
ItemFromLines Method
ItemFromNumber Method
ItemFromSelection Method
ItemFromShapes Method
ItemFromText Method
ItemFromUniqueID Method
ResetSearch Method

Valid Property

Objects Collection

ItemFromFieldValue Method and MsgBox Method Example

This example uses the **ItemFromFieldValue** method of the Objects collection to find the correct answer to a guessing game. It uses the **MsgBox** method of the Application object to give the answer.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, NewShape As Object
Dim Count As Single, Answer As Single
Dim NumberIn As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Magic Number")          ' Create a field
Field1.Type = 5                                        ' Make the field's type number

For Count = 1 To 3
   Set NewShape = Chart.DrawShape("Operation")     ' Draw a shape
   NumberIn = Int(InputBox("Enter a number between 1 and 10."))     ' Accept user input
   NewShape.FieldValues.Item("Magic Number").Value = NumberIn     ' Assign to field
Next Count

Randomize
Answer = Int(10 * Rnd + 1)                             ' Randomly generate an integer
Answer = CDbl(Answer)
Set Everything = Chart.Objects                         ' Get all objects in the chart
Do
' Find field value equal to Answer
   Set Current = Everything.ItemFromFieldValue(Field1, Answer)
   Current.Text = "You win!"                           ' Enter text into the shape
   Current.Color = ABC.Red                             ' Make the shape red
Loop While Current.Valid
ABC.MsgBox ("Thanks for playing! The correct answer was " + CStr(Answer) + ".")
```

ItemFromNumber Method

Usage *ObjectsCollection*.**ItemFromNumber** (*Value*)
The *Value* element is a string or double that is the number or identifier of the shape you are searching for.

Description You use the **ItemFromNumber** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromNumber** method returns, from the Objects collection, successive shapes with the number you specify.

Flow Equivalent None

Finding Objects in a Chart
Example

Item Method (Objects Collection)
ItemFromAll Method
ItemFromAttachments Method
ItemFromFieldValue Method
ItemFromLines Method
ItemFromSelection Method
ItemFromShapes Method
ItemFromText Method
ItemFromUniqueID Method
ResetSearch Method

Valid Property

Objects Collection

ItemFromNumber Method Example

This example uses the **ItemFromNumber** method of the Objects collection to find a specific shape by its number.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Count As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
Set Chart = ABC.ActiveChart                        ' Get the active chart
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Add a new chart

Chart.NextNumber = 100                             ' Set the next shape number used
For Count = 1 To 4                                 ' Draw four operation shapes
   Set NewShape = Chart.DrawShape("Operation")
Next Count

Set Everything = Chart.Objects                     ' Get all objects in the chart
Set Current = Everything.ItemFromNumber(102)       ' Get shape 102
Current.Shape.FillPattern = 22                     ' Fill the shape with a pattern
```

ItemFromShapes Method

Usage *ObjectsCollection*.**ItemFromShapes**

Description You use the **ItemFromShapes** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromShapes** method returns successive shapes from the Objects collection.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ItemFromShapes_Method')}

Finding Objects in a Chart
Selecting Shapes
Example

Item Method (Objects Collection)
ItemFromAll Method
ItemFromAttachments Method
ItemFromFieldValue Method
ItemFromLines Method
ItemFromNumber Method
ItemFromSelection Method
ItemFromText Method
ItemFromUniqueID Method
ResetSearch Method

Valid Property

Objects Collection

ItemFromShapes Method, IsLinked Property, Number Property, and LinkedChartName Property Example

This example uses the **ItemFromShapes** method of the Objects collection, the **IsLinked** property, **Number** property, and **LinkedChartName** property of the Shape object to select shapes, find the shapes that are linked, and describe them in a message box. For the program to work usefully, the chart must contain shapes, with at least one of the shapes linked.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim CurrentShape As Object
Dim SelectedShapes As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set SelectedShapes = Chart.Objects

Do
   Set CurrentShape = SelectedShapes.ItemFromShapes        ' Check all shapes in the chart
   If CurrentShape.Shape.IsLinked Then                     ' If shape is linked, display a message
        ABC.MsgBox "Shape #" + CurrentShape.Shape.Number + " is linked to " +
CurrentShape.Shape.LinkedChartName
   End If
Loop While CurrentShape.Valid
```

ItemFromSelection Method

Usage *ObjectsCollection*.**ItemFromSelection**

Description You use the **ItemFromSelection** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromSelection** method returns, from the Objects collection, successive selected objects.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_ItemFromSelection_Method')}

ItemFromSelection Method, Top Property, Left Property, Right Property, and Bottom Property Example 1

This example uses the **ItemFromSelection** method of the Objects collection and the **Top** property of the Object object to find objects and put all their upper edges in the same place. If you wish, you can substitute the **Left** property, **Right** property, or **Bottom** property for the **Top** property to achieve a similar effect.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim SelectedItems As Object                    ' Everything in the chart
Dim CurrentItem As Object                      ' Currently selected shape

Set ABC = CreateObject("ABCFlow.application")' Start ABC
ABC.Visible = True                             ' Make ABC visible
ABC.New                                        ' create new chart
Set Chart = ABC.ActiveChart                    ' Get the active chart

Chart.drawpositionx = 2                        ' Set where next shape will be
Chart.drawpositiony = 2

Set Shape1 = Chart.DrawShape("Operation")      ' Draw a shape

Chart.drawpositionx = 5                        ' Set where next shape will be
Chart.drawpositiony = 4

Set Shape2 = Chart.DrawShape("Decision")       ' Draw a shape

ABC.MsgBox ("Click on ok to move the shapes.")

Set SelectedItems = Chart.Objects              ' Use all shapes in the current chart

Chart.Select (0)                               ' Select all shapes in the chart

Do
    Set CurrentItem = SelectedItems.ItemFromSelection
    CurrentItem.Top = 1                        ' Place top edge of items at 1 inch
Loop While CurrentItem.Valid
```

ItemFromSelection Method, Valid Property, CenterX Property, and CenterY Property Example 2

This example uses the **ItemFromSelection** method of the Objects collection and the **Valid** property, **CenterX** property, and **CenterY** property of the Object object to find selected objects, ensure that only valid objects are acted on, and find and report the center of the objects both horizontally and vertically. To use this code example, you must open FlowCharter, place some objects in the chart and select some of the objects.

```
Dim ABC As Object, Chart As Object
Dim CurrentItem As Object
Dim Center_X, Center_Y As String
Dim SelectedItems As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set SelectedItems = Chart.Objects                      ' Get all objects in the chart

Set CurrentItem = SelectedItems.ItemFromSelection      ' Get the first selected object
Do While CurrentItem.Valid                             ' Loop through all selected objects
   Center_X = CStr(CurrentItem.CenterX)                ' Horizontal center of object
   Center_Y = CStr(CurrentItem.CenterY)                ' Vertical center of object
   ABC.MsgBox "X Value = " + Center_X                  ' Display the X coordinate
   ABC.MsgBox "Y Value = " + Center_Y                  ' Display the Y coordinate
   Set CurrentItem = SelectedItems.ItemFromSelection         ' Loop through all objects
Loop
```

ItemFromText Method

Usage *ObjectsCollection*.**ItemFromText** (*Text*)
The *Text* element is the text string you are searching for.

Description You use the **ItemFromText** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromText** method returns, from the Objects collection, successive objects that contain the text you specify.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ItemFromText_Method')}

Finding Objects in a Chart
Example

Item Method (Objects Collection)
ItemFromAll Method
ItemFromAttachments Method
ItemFromFieldValue Method
ItemFromLines Method
ItemFromNumber Method
ItemFromSelection Method
ItemFromShapes Method
ItemFromUniqueID Method
ResetSearch Method

Valid Property

Objects Collection

ItemFromText Method Example

This example uses the **ItemFromText** method of the Objects collection to find a specific shape by its text.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
Set Chart = ABC.ActiveChart                        ' Get the active chart
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Add a new chart

Set Shape1 = Chart.DrawShape                       ' Draw a shape
Shape1.Text = "Roses are red"                      ' Enter text in the shape
Set Shape2 = Chart.DrawShape                       ' Draw a shape
Shape2.Text = "Violets are blue"                   ' Enter text in the shape
Shape2.Font.Color = ABC.White                      ' Set the text color

Set Everything = Chart.Objects                     ' Get all objects in the chart
Set Current = Everything.ItemFromText("Violets")   ' Get the shape containing "Violets"
Current.Shape.FillColor = ABC.MakeRGB(64, 0, 127)  ' Fill the shape with purple
```

ItemFromUniqueID Method

Usage *ObjectsCollection*.**ItemFromUniqueID** (*UniqueID*)
The *UniqueID* element is the unique identification number of the object you are searching for.

Description You use the **ItemFromUniqueID** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

Value The **ItemFromUniqueID** method returns, from the Objects collection, the object with the unique identification number you specify.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ItemFromUniqueID_Method')}

ItemFromUniqueID Method Example

This example uses the **ItemFromUniqueID** method of the Objects collection to identify a specific object.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, FirstYellow As Object
Dim Shape3 As Object, SecondYellow As Object
Dim Everything As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawSpacingX = .5                                ' Draw shapes 0.5" apart horizontally
Set Shape1 = Chart.DrawShape("Decision")               ' Draw a Decision shape
Shape1.Color = ABC.Red                                 ' Make the shape red
Set FirstYellow = Chart.DrawShape("Decision")          ' Draw a Decision shape
FirstYellow.Color = ABC.Yellow                         ' Make the shape yellow
Set Shape3 = Chart.DrawShape("Decision")               ' Draw a Decision shape
Shape3.Color = ABC.Blue                                ' Make the shape blue

Chart.Select (0)                                       ' Select all shapes
Chart.Duplicate                                        ' Make a copy of all shapes

ABC.MsgBox "Now let's get the second yellow shape."

Set Everything = Chart.Objects                         ' Get all objects in the chart
' Get the yellow shape from the duplicate set
Set SecondYellow = Everything.ItemFromUniqueID(FirstYellow.UniqueID + 3)
FirstYellow.ToBack                                     ' Move the shape behind other shapes
SecondYellow.ToFront                                   ' Move the shape in front of others
SecondYellow.Text = "UniqueID #" + CStr(SecondYellow.UniqueID)      ' Display the ID
SecondYellow.Shape.FitShapeToText                      ' Enlarge the shape so the text fits
```

ResetSearch Method

Usage *ObjectsCollection*.**ResetSearch**

Description You use the **ResetSearch** method to reset all searches that use the ItemFrom methods to the beginning of the items in the chart**.**

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ResetSearch_Method')}

ResetSearch Method Example

This example uses the **ResetSearch** method of the Objects collection to start a subsequent search through shapes from the beginning.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Count As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.New                                              ' Add a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart
ABC.Visible = True                                   ' Make ABC visible
Chart.View = 2                                       ' Set the view to Used Pages

Chart.NextNumber = 100                               ' Set the next shape number used
For Count = 1 To 10
   Set NewShape = Chart.DrawShape                    ' Draw a shape
   NewShape.Shape.FillPattern = Count                ' Fill the shape with a pattern
Next Count

Set Everything = Chart.Objects                       ' Get all objects in the chart
Do
   Set Current = Everything.ItemFromShapes           ' Get the next shape
   Current.Shape.FillColor = ABC.Red                 ' Make the shape's fill red
Loop Until Current.Shape.FillPattern = 5             ' Search until you find pattern 5
ABC.MsgBox ("The first search reached fill pattern 5.")

Everything.ResetSearch                               ' Start next search from beginning
Do
   Set Current = Everything.ItemFromShapes           ' Get the next shape
   Current.Shape.BorderColor = ABC.Blue              ' Make the shape's border blue
Loop Until Current.Shape.Number = "108"              ' Search until you find shape 108
ABC.MsgBox ("The second search started over in order to find shape 108.")
```

Object Object

Description

The Object object is contained in the Object collection. You can have multiple Object objects in the collection. Below the Object object are the Shape, Line_, TextBlock, OLE, Font, and FieldValues objects. Each Object object can have multiple FieldValue objects, but only one Shape, Line_, TextBlock, OLE, and Font object for each Object object. Note that the Shape object and Line_ object are mutually exclusive. If the Object object is a shape, the Line_ object is a meaningless placeholder.

Properties	Methods
Application	ApplyDefaults
Bottom	Clear_
CenterX	Duplicate
CenterY	Repaint
Color	RestorePicture
FieldValues	ToBack
FlippedHorizontal	ToFront
FlippedVertical	
Font	
Height	
Left	
Line_	
OLE	
Parent	
Right	
Rotation	
Selected	
Shape	
StretchType	
Text	
TextAlignment	
TextBlock	
TextLF	
Top	
Type	
UniqueID	
Valid	
Width	

Application Property

Usage	*ObjectObject*.**Application**
Description	You use the **Application** property to find the running Application object. You can find the object to which the master items apply and which master items to display. The **Application** property is read only.
Data Type	Object
Value	The running application
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Application_Property')}

Application Property Example

This example finds the **Application** property value of the Object object. The other **Application** property values are found in similar ways.

```
Dim ABC As Object
Dim Objects_Parent As Object
Dim Chart As Object
Dim Objects As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                'Create a new chart

Set Chart = ABC.ActiveChart
Set Objects = Chart.Objects

Set Objects_Parent = Objects.Application               ' Set the Objects Object
ABC.MsgBox "The running application is " + Objects_Parent
```

Bottom Property (Object Object)

Usage	*ObjectObject*.**Bottom** = *Distance*
Description	The **Bottom** property lets you find or set the location of the bottom of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units for measuring the distance using the **Units** property. The **Bottom** property is read/write.
Data Type	Double
Value	The location of the bottom of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Bottom_Property_ABC_Object')}

Moving Objects

Example

Bottom Property (Application Object)

CenterX Property

CenterY Property

Left Property (Object Object)

Right Property (Object Object)

Top Property (Object Object)

Units Property (Chart Object)

Units Property (Preferences Object)

Object Object

CenterX Property

Usage	*ObjectObject*.**CenterX** = *Distance*
Description	The **CenterX** property lets you find or set the center of the object. The property does not affect the size of the object. You set the units used to measure the distance using the **Units** property. The **CenterX** property is read/write.
Data Type	Double
Value	The center of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_CenterX_Property')}

Moving Objects

Example

Bottom Property (Application Object)

CenterY Property

Left Property (Object Object)

Right Property (Object Object)

Top Property (Object Object)

Units Property (Chart Object)

Units Property (Preferences Object)

Object Object

CenterY Property

Usage	*ObjectObject*.**CenterY** = *Distance*
Description	The **CenterY** property lets you find or set the center of the object. The property does not affect the size of the object. You set the units used to measure the distance using the **Units** property. The **CenterY** property is read/write.
Data Type	Double
Value	The center of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_CenterY_Property')}

Moving Objects

Example

Bottom Property (Application Object)

CenterX Property

Left Property (Object Object)

Right Property (Object Object)

Top Property (Object Object)

Units Property (Chart Object)

Units Property (Preferences Object)

Object Object

Color Property (Object Object) {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage	*ObjectObject*.**Color** = *Color*
Description	The **Color** property of the Object object lets you find or set the fill color for shapes, the color of lines, or the color of text (see the **MakeRGB** method). Using the **Color** property for shapes is the same as using the **FillColor** property. Using the **Color** property for lines finds the stem color and sets the color of the stem and both ends. The **Color** property is read/write.
Data Type	Long
Value	The color for a shape, line, or text object
Flow Equivalent	The **Color** property of the Object object is equivalent to selecting a shape, a line, or text, clicking the Fill Color button on the formatting toolbar, and clicking the color you want.

{button Related Topics,PI(`',`IDH_RT_Color_Property_ABC_Object_Method')}

Setting Shape Colors

Setting Line Colors

Setting Text Colors

Formatting Shape Numbers

Fill, Border, and Shadow Colors

Text Color

Example

BasicColor Method

BorderColor Property

Color Property (Font Object)

Color Property (Line Object)

FillColor Property

MakeRGB Method

ShadowColor Property

Object Object

Color Property, Height Property, Width Property (Object Object), and FillColor Property Example

This example uses the **Color** property, the **Height** property, and the **Width** property of the Object object and the **FillColor** property of the Shape object to set the color, width, and height of shapes.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim NewShape1 As Object, NewShape2 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                'Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set NewShape1 = Chart.DrawShape("Decision")            ' Draw a Decision shape
NewShape1.Color = ABC.RED                              ' Make the shape red
NewShape1.Height = 1                                   ' Make the shape 1 inch high
NewShape1.Width = 2                                    ' Make the shape 2 inches wide

Set NewShape2 = Chart.DrawShape("Operation")           ' Draw an Operation shape
NewShape2.Shape.FillColor = ABC.MakeRGB(0, 0, 255)     ' Make the shape blue
NewShape2.Height = .5                                  ' Make the shape 1/2 inch high
NewShape2.Width = 1                                    ' Make the shape 1 inch wide
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

FieldValues Property

Usage	*ObjectObject*.**FieldValues**
Description	The **FieldValues** property lets you find the data fields included in the FieldValues collection. The **FieldValues** property is read only, but all the properties from the object it returns are read/write.
Data Type	Collection object
Value	The fields included in the FieldValues collection
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_FieldValues_Property')}

FieldValues Property Example

This example uses the **FieldValues** property of the Object object to enter text in a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Client")        ' Create a field
Field1.Type = 0                                        ' Make the field's type text
Set Shape1 = Chart.DrawShape                           ' Draw a shape
Shape1.FieldValues.Item("Client").Value = "John P. Cliché"   ' Enter text in the field
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Font Property

Usage	*ObjectObject*.**Font**
Description	The **Font** property lets you find the font object for text. The **Font** property is read only, but all the properties from the object it returns are read/write.
Data Type	Object
Value	The Font object for text
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Font_Property')}

Text Typeface and Size

Example

Font Object

Object Object

Font Property Example

This example uses the **Font** property of the Object object to change text attributes.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object

Set ABC = CreateObject("ABCFlow.application")                ' Start ABC
ABC.Visible = True                                           ' Make ABC visible
ABC.New                                                      ' Create a new chart
Set Chart = ABC.ActiveChart                                  ' Get the active chart

Set Text1 = Chart.DrawTextBlock("OLE Automation is fun!")    ' Create a line of text
Text1.Font.Italic = True                                     ' Make the text italic
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Height Property (Object Object)

Usage	*ObjectObject*.**Height** = *Height*
Description	The **Height** property lets you find or set the height of the object. You set the units used to measure the height using the **Units** property. The **Height** property is read/write.
Data Type	Double
Value	The height of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Height_Property_ABC_Object')}

Resizing Objects

Example

Height Property (Application Object)

Height Property (PageLayout Object)

StretchType Property

Units Property (Chart Object)

Units Property (Preferences Object)

Width Property (Object Object)

Object Object

Left Property (Object Object)

Usage	*ObjectObject*.**Left** = *Distance*
Description	The **Left** property lets you find or set the location of the left side of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units for measuring the distance using the **Units** property. The **Left** property is read/write.
Data Type	Double
Value	The location of the left side of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Left_Property_ABC_Object')}

Moving Objects

Example

Bottom Property (Object Object)

CenterX Property

CenterY Property

Left Property (Application Object)

Right Property (Object Object)

Top Property (Object Object)

Units Property (Chart Object)

Units Property (Preferences Object)

Object Object

Line_ Property

Usage	*ObjectObject*.**Line_**
Description	The **Line_** property lets you find the line objects. The **Line_** property is read only, but all the properties from the object it returns are read/write.
Data Type	Object
Value	The Line_ object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Line_Property')}

Line Property Example

This example uses the **Line_** property of the Object object to change the style of a line stem.

```
Dim ABC As Object, Chart As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Line1 = Chart.DrawFreeLine(5, 2)               ' Draw a plain line
Line1.Line_.StemStyle = 4                          ' Change the stem style
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Parent Property

Usage	*Object*.**Parent**
Description	You use the **Parent** property to find the parent object of an object. For example, the parent of the Application object is the running FlowCharter application. The parent of the Objects collection is the chart object in which the objects reside. The **Parent** property is read only.
Data Type	Object
Value	The parent of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Parent_Property')}

Parent Property Example

This example uses the **Parent** property of the Application object to put the parent of the application into a variable. The **Parent** properties of the other objects and collections work the same way.

```
Dim ABC As Object
Dim App_Parent As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart

Set App_Parent = ABC.Parent                            ' Set the collection of open ABC charts
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

The following code tests the parent property of the Menu Collection.

```
Dim ABC As Object, Menu As Object, TitleCap As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True   ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)    ' Add a new menu

Set TitleCap = Menu.Parent
TitleCap.Caption = "A new menu item called test has been added."
```

Note: The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

Note: Visual Basic requires different FlowCharter custom controls, depending on the version of Visual Basic that you are using. (Visual Basic 4.0 is not backward-compatible with older versions of the controls, those with VBX file extensions.)

If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

Right Property (Object Object)

Usage	*ObjectObject*.**Right** = *Distance*
Description	The **Right** property lets you find or set the location of the right side of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units used for measuring the distance using the **Units** property. The **Right** property is read/write.
Data Type	Double
Value	The location of the right side of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Right_Property_ABC_Object')}

Moving Objects

Example

Bottom Property (Application Object)

CenterX Property

CenterY Property

Left Property (Object Object)

Right Property (Application Object)

Top Property (Object Object)

Units Property (Chart Object)

Units Property (Preferences Object)

Object Object

StretchType Property

Usage *ObjectObject*.**StretchType** = *Value*

Description The **StretchType** property lets you find or set the type of stretching for the object. You can set it to normal (opposite sides both move as when you stretch in FlowCharter) or so that one side is fixed. If the user stretches with one side fixed, it is the same as if he or she pressed **CTRL** while stretching. If you resize using OLE Automation, then the top and left sides are held fixed as if you were stretching from the right or bottom center handle and holding the **CTRL** key. The **StretchType** property is read/write.

Data Type Integer

Value The values for the stretch types are in the following table.

Value	**Meaning**
0	Normal
1	Fixed sides

Flow Equivalent None

StretchType Property and DrawDirection Property Example

This example uses the **StretchType** property of the Object object and the **DrawDirection** property of the Chart object to set the type of stretching for an object.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawDirection = 2                                ' Draw new shapes down the page
Set Shape1 = Chart.DrawShape                           ' Draw the first shape
Set Shape2 = Chart.DrawShape                           ' Draw the second shape

Shape1.StretchType = 0                                 ' Use Normal stretch type
Shape1.Text = "Normal"                                 ' Enter text in the shape
Shape1.Color = ABC.Cyan                                ' Apply a color to the shape
ABC.MsgBox "You can resize the cyan object normally."
Shape2.StretchType = 1                                 ' Use OppositeSideFixed stretch type
Shape2.Text = "Opposite Side Fixed"                    ' Enter text in the shape
Shape2.Color = ABC.Yellow                              ' Apply a color to the shape
ABC.MsgBox "You can resize each side of the yellow object independently."
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Remove ABC. from MsgBox methods.

Selected Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Selecting_Objects');CW(`concfull')}

Usage	*ObjectObject*.**Selected** = {True | False}
Description	You use the **Selected** property to find or set whether an object is selected. The **Selected** property is read/write.
Data Type	Integer (Boolean)
Value	True means the object is selected; False means the object is not selected.
Flow Equivalent	The **Selected** property is equivalent to clicking an object to select it.

{button Related Topics,PI(`',`IDH_RT_Selected_Property')}

Selected Property Example

This example uses the **Selected** property of the Object object to select a shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
Set Chart = ABC.ActiveChart                            ' Get the active chart
ABC.Visible = True                                     ' Make ABC Visible
ABC.New                                                ' Create a new chart

Set Shape1 = Chart.DrawShape                           ' Draw a shape
Shape1.Shape.FillColor = ABC.MakeRGB(255, 0, 0)        ' Make the shape red
Set Shape2 = Chart.DrawShape                           ' Draw a shape

Set Everything = Chart.Objects                         ' Get all items in the chart
Do
    Set Current = Everything.ItemFromShapes            ' Get the next shape in the chart
    If Current.Shape.FillColor = ABC.MakeRGB(255, 0, 0) Then
        Current.Selected = True                        ' Select the red shape
    End If
Loop While Current.Valid                               ' Continue for all shapes
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Change Loop While to another iterative loop that can be found in Visual Basic Script.

Shape Property

Usage	*ObjectObject*.**Shape**
Description	You use the **Shape** property to find the shape object. The **Shape** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	The Shape object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Shape_Property')}

Example

Shape Object

Object Object

Shape Property Example

This example uses the **Shape** property of the Object object to set the color of a shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape                           ' Draw a shape
Shape1.Shape.FillColor = ABC.MakeRGB(0, 127, 127)      ' Access the Shape Object property FillColor
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Text Property (Object Object) {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage	*ObjectObject*.**Text** = *TextString*
Description	You use the Text property of the Object object to add or read text inside any shape or text block. If you wish to preserve Returns when reading the text, you should use the **TextLF** property. The **Text** property is read/write.
Data Type	String
Value	The text inside a shape
Flow Equivalent	The **Text** property of the Object object is equivalent to typing while a shape is selected.

{button Related Topics,PI(`',`IDH_RT_Text_Property_ABC_Object')}

Adding Text to Shapes

Adding Text to a Shape

Example

FitShapeToText Method

Text Property (Menu Collection)

Text Property (MenuItem Object)

TextLF Property

TextBlock Object

Object Object

Text Property (Object Object) Example

This example uses the **Text** property of the Object object to add text to a shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape("Document")               ' Draw a Document shape
Shape1.Text = "I love chocolate!"                      ' Add text to the shape
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

TextAlignment Property

Usage *ObjectObject*.**TextAlignment** = *AlignmentChoice*

Description You use the **TextAlignment** property to align the text inside shapes and in text blocks or to find the alignment. The **TextAlignment** property is read/write.

Data Type Integer

Value The TextAlignment property uses the following values to represent combinations of vertical and horizontal alignment.

Value	Vertical	Horizontal
0	Top	Left
1	Top	Center
2	Top	Right
3	Middle	Left
4	Middle	Center
5	Middle	Right
6	Bottom	Left
7	Bottom	Center
8	Bottom	Right

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_TextAlignment_Property')}

Text Alignment

Example

Bold Property

Color Property (Font Object)

Italic Property

Opaque Property

Size Property

Underline Property

TextBlock Object

Object Object

TextAlignment Property Example

This example uses the **TextAlignment** property of the Object object to set the text alignment in a shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")                ' Start ABC
ABC.Visible = True                                           ' Make ABC visible
ABC.New                                                      ' Create a new chart
Set Chart = ABC.ActiveChart                                  ' Get the active chart

Set Shape1 = Chart.DrawShape("Operation")                    ' Draw an Operation shape
Shape1.Text = "This text belongs in the upper right corner"  ' Add text to the shape
Shape1.Shape.FitShapeToText                                  ' Enlarge the shape so the text fits
Shape1.TextAlignment = 2                                     ' Align the text within the shape
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Top Property (Object Object)

Usage	*ObjectObject*.**Top** = *Distance*
Description	The **Top** property lets you find or set the location of the top of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units to measure the distance using the **Units** property. The **Top** property is read/write.
Data Type	Double
Value	The location of the top of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Top_Property_ABC_Object')}

Moving Objects

Example

Bottom Property (Object Object)

CenterX Property

CenterY Property

Left Property (Object Object)

Right Property (Object Object)

Top Property (Application Object)

Units Property (Chart Object)

Units Property (Preferences Object)

Object Object

Type Property (Object Object)

Usage *ObjectObject*.**Type**

Description The **Type** property lets you find the type of object. The **Type** property is read only.

Data Type Integer

Value The values for the types are in the following table.

Object Type	Description
0	Shape
1	Line
2	Text
3	Bitmap
4	OLE client object
5	Master

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_Type_Property_ABC_Object')}

UniqueID Property

Usage	*ObjectObject*.**UniqueID**
Description	The **UniqueID** property lets you find the ID for an object. You can use the ID to choose an object in the Objects collection using, for example, the **ItemFromUniqueID** method. The identifier is unique for each object in each chart. If you wish, you could create a database containing the **UniqueID** property values for all the objects in a chart to make it easy to identify and act on each of them. A **UniqueID** is never reused in a chart even if you delete the object. The **UniqueID** property is read only.
Data Type	Double
Value	The unique ID of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_UniqueID_Property')}

Identifying an Object

Example

ItemFromUniqueID Method

ShapeName Property
Type Property (Object object)

Object Object

UniqueID Property Example

This example uses the **UniqueID** property of the Object object to find the unique identifier for an object.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape                           ' Draw a shape
ABC.MsgBox "The shape's unique ID is " & Shape1.UniqueID & "."       ' Display the shape's ID
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Remove ABC. from the MsgBox method.

Valid Property

Usage	*ChartObject*.**Valid** *ObjectObject*.**Valid**
Description	You use the **Valid** property in the While part of a Do While loop to check that the ItemFrom methods are returning valid objects. The **Valid** property is read only.
Data Type	Integer (Boolean)
Value	True means the object is valid; False means the object is not valid.
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Valid_Property')}

Valid Property and Type Property (Chart Object) Example

This example uses the **Valid** property and the **Type** property of the Chart object to find valid charts and display their types.

```
Dim ABC As Object, Chart As Object
Dim Path1 As String
Dim File1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.CloseAll                                           ' Close all open charts

Path1 = ABC.Path + "\Samples\Quality.abc"              ' Path of file to be opened
Set File1 = ABC.Open(Path1)                            ' Open chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

If Chart.Valid Then                                    ' If the current chart is valid
    Chart.Minimize                                     ' minimize its window and
    ABC.MsgBox Path1 + " is a " + Chart.Type + " type of chart."        ' post message with type
    ABC.MsgBox "The minimized chart is a valid chart."
Else
    MSG1 = " was not found. Please enter a valid sample file name in the code and try again."
    ABC.MsgBox (Path1 + MSG1)
End If
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Remove ABC. from the MsgBox methods.

Width Property (Object Object)

Usage	*ObjectObject*.**Width** = *Width*
Description	The **Width** property lets you find or set the width of the object. You set the units used to measure the width using the **Units** property. The **Width** property is read/write.
Data Type	Double
Value	The width of the object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Width_Property_ABC_Object')}

Resizing Objects

Example

Height Property (Object Object)

StretchType Property

Units Property (Chart Object)

Units Property (Preferences Object)

Width Property (Application Object)

Width Property (PageLayout Object)

Object Object

Duplicate Method (Object Object)

Usage	*ObjectObject*.**Duplicate**
Description	The **Duplicate** method of the Object object makes a duplicate of the selected object and returns the duplicate object.
Data Type	Object
Value	The duplicate object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Duplicate_Method_ABC_Object')}

Duplicating Objects

Speeding Actions

Example

Copy Method

Duplicate Method (Chart Object)

Paste Method

Object Object

Duplicate Method (Object Object), NumberShown Property, and Renumber Method Example

This example uses the **Duplicate** method of the Object object and the **NumberShown** property and **Renumber** method of the Shape object to duplicate a shape, show the number on the shape, and increment the number of the shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, NewShape As Object
Dim Count As Double

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.NextNumber = 100
Set Shape1 = Chart.DrawShape("Terminal")           ' Draw a Terminal shape
For Count = 1 To 10                                ' To make 10 copies of the shape
    Set NewShape = Shape1.Duplicate                ' Duplicate the last shape
    NewShape.CenterY = Count / 2                   ' Move the new shape down
    NewShape.Shape.NumberShown = True              ' Show the shape number
    NewShape.Shape.Renumber                        ' Increment the shape number
Next Count
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Repaint Method

Usage *ChartObject*.**Repaint**
ObjectObject.**Repaint**

Description You use the **Repaint** method to repaint the entire chart after a series of actions with the **NoRepaint** property set to True.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_Repaint_Method')}

Repaint Method and NoRepaint Property Example

This example uses the **NoRepaint** property and **Repaint** method of the Chart object to turn off repainting the screen with each change and then repaint the screen after the operations are finished.

```
Dim ABC As Object, Chart As Object, Obj1 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.NoRepaint = True                                 ' Do not repaint screen
Set Obj1 = Chart.DrawShape("Operation")                ' Draw Operation shape
Obj1.Text = "Unit 1"                                   ' Add text to shape

Chart.Select (0)                                       ' Select Shape

For X = 1 To 3
    Chart.Duplicate                                    ' Duplicate shape three times
Next X

Chart.Repaint                                          ' Repaint screen
Chart.NoRepaint = False                                ' Restore repainting screen
```

{button Other example,JI(`>example',`IDH_DeleteLines_Method_Example')}

RestorePicture Method (Object Object)

Usage *ObjectObject*.**RestorePicture**

Description The **RestorePicture** method of the Object object lets you restore bitmap and OLE client objects to their original size.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_RestorePicture_Method_ABC_Object')}

Resizing Objects

Example

Object Object

RestorePicture Method (Object Object) Example

This example uses the **RestorePicture** method of the Object object to restore bitmaps and OLE objects to their original sizes. For the program to have any effect, you must have a resized bitmap or OLE client object in the chart.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Everything = Chart.Objects                         ' Get all items in the chart
Do
    Set Current = Everything.ItemFromAll               ' Choose the next item
    Current.RestorePicture                             ' Return bitmaps and OLE objects to their original
size
Loop While Current.Valid
```

ToBack Method (Object Object){button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Send_to_Back_Command');CW(`concfull')}

Usage	*ObjectObject***.ToBack**
Description	You use the **ToBack** method of the Object object to move the object to the back.
Flow Equivalent	The **ToBack** method is equivalent to clicking the Send to Back command on the Order submenu on the Arrange menu.

{button Related Topics,PI(`',`IDH_RT_ToBack_Method_ABC_Object')}

Changing the Display Order of Objects

Example

ToBack Method (Chart Object)

ToFront Method (Chart Object)

ToFront Method (Object Object)

Object Object

ToBack Method and ToFront Method (Object Object) Example

This example uses the **ToBack** method and **ToFront** method of the Object object to move the shape in front of and behind other shapes.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object, Shape3 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawSpacingX = .5                                ' Draw shapes 0.5" apart horizontally
Set Shape1 = Chart.DrawShape                           ' Draw a shape
Shape1.Color = ABC.Red                                 ' Make the shape red
Set Shape2 = Chart.DrawShape                           ' Draw a shape
Shape2.Color = ABC.Yellow                              ' Make the shape yellow
Set Shape3 = Chart.DrawShape                           ' Draw a shape
Shape3.Color = ABC.Blue                                ' Make the shape blue
ABC.MsgBox "The yellow shape will move to the back when you click OK."
Shape2.ToBack                                          ' Move shape behind other shapes
ABC.MsgBox "This time the yellow shape will move to the front when you click OK."
Shape2.ToFront                                         ' Move shape in front of other shapes
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Remove ABC. from the MsgBox method.

ToFront Method (Object Object)

{button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Bring_to_Front_Command');CW(`concfull')}

Usage *ObjectObject*.**ToFront**

Description You use the **ToFront** method of the Object object to move the object to the front.

Flow Equivalent The **ToFront** method is equivalent to to clicking the Bring to Front command on the Order submenu on the Arrange menu.

{button Related Topics,PI(`',`IDH_RT_ToFront_Method_ABC_Object')}

Changing the Display Order of Objects

Example

ToBack Method (Chart Object)

ToBack Method (Object Object)

ToFront Method (Chart Object)

Object Object

OLE Property

Usage	*ObjectObject*.**OLE**
Description	The **OLE** property lets you find or set the properties and methods associated with OLE objects. The **OLE** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	An OLE object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_OLE_Property')}

Using OLE Client Objects

Example

DoVerb Method

ObjectType Property

OLE Object

Object Object

OLE Property and DoVerb Method Example

This example uses the **OLE** property of the Object object and the **DoVerb** method of the OLE object to execute an OLE verb.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object
Dim PaintHandle
Dim Pasted

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
ABC.Visible = True                                     ' Make ABC visible

Clipboard.Clear                                        ' Empty the Clipboard
PaintHandle = Shell("PBRUSH.EXE", 1)                   ' Run Paintbrush
SendKeys "%FO", 1                                      ' Send File/Open to Paintbrush
SendKeys "c:\abc\samples\mgxlogo.bmp{ENTER}", 1
SendKeys "%EF", 1                                      ' Edit/Paste From
SendKeys "c:\abc\samples\mgxlogo.bmp{ENTER}", 1
SendKeys "%EC", 1                                      ' Copy to the Clipboard
Pasted = Chart.Paste                                   ' Paste the OLE object into ABC
If Not Pasted Then                                     ' If nothing was pasted, post an error message and
stop the code
   ABC.MsgBox "Either Paintbrush or the MGXLOGO.BMP file was not found. Please edit the code and try again."
   Exit Sub
End If

ABC.Visible = True                                     ' Bring ABC to the front
Chart.View = 1                                         ' View the current page in ABC
Set Everything = Chart.Objects                         ' Get all items in the chart
Do
   Set Current = Everything.ItemFromAll                ' Choose the next item
   If Current.Type = 4 Then                            ' If an OLE object is found
      ABC.MsgBox "An OLE object exists on the chart. Click OK to execute the OLE verb."
      Current.OLE.DoVerb                               ' execute its default verb
      Exit Sub                                         ' and stop the code
   End If
Loop While Current.Valid
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Remove ABC. from the MsgBox method.

TextBlock Property

Usage	*ObjectObject*.**TextBlock**
Description	The **TextBlock** property lets you find the properties of a block of text. The **TextBlock** property is read only, but all the properties from the object it returns are read/write.
Data Type	Object
Value	The properties of a block of text
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_TextBlock_Property')}

TextBlock Property and AttachedToLine Property Example

This example uses the **TextBlock** property of the Object object and the **AttachedToLine** property of the TextBlock object to find a line that has text attached to it and turn the line red.

```
Dim ABC As Object, Chart As Object
Dim NewLine1 As Object, NewLine2 As Object
Dim NewText1 As Object, NewText2 As Object
Dim LineWithText As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                          ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                 ' Get the active chart

Set NewText1 = Chart.DrawTextBlock("Attached text") ' Draw text objects
Chart.DrawPositionX = 4                            ' Set a horizontal drawing position
Set NewText2 = Chart.DrawTextBlock("Unattached text")
Chart.DrawPositionX = 1                            ' Set a horizontal drawing position
Set NewLine1 = Chart.DrawFreeLine(3, 4)     ' Draw a line
Chart.DrawPositionX = 4                            ' Set a horizontal drawing position
Set NewLine2 = Chart.DrawFreeLine(5, 7)     ' Draw a line
ABC.MsgBox "Let's attach some text to a line."
NewLine2.Line_.AttachText NewText1          ' Attach a text object to a line
If NewText2.TextBlock.AttachedToLine = 0 Then      ' Check to see if the text is attached
Set LineWithText = NewLine2.Line            ' Get the line with text attached
LineWithText.StemColor = ABC.RED            ' Make the line red
ABC.MsgBox "The red line has text attached to it."
Else
ABC.MsgBox "The text did not align."
End If
ABC.MsgBox "All done!"
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Remove ABC. from the MsgBox method.

TextLF Property

Usage	*ObjectObject*.**TextLF** = *TextString*
Description	You use the TextLF property of the Object object to add or read text inside any shape or text block. When adding text, the property is identical to the Text property. When reading text, the property does not substitute spaces for Returns as the Text property does. If you do not wish to preserve Returns, you should use the Text property. The TextLF property is read/write.
Data Type	String
Value	The text inside a shape with the Returns preserved
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_TextLF_Property')}

Adding Text to Shapes

Adding Text to a Shape

Example

FitShapeToText Method

Paste Method

Text Property

Text Property (Menu Collection)

Text Property (MenuItem Object)

TextBlock Object

Object Object

TextLF Property Example

This example uses the **TextLF** property of the Object object to read text from a shape, preserving the Returns in it.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, ShapeText As String

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape("Document")               ' Draw a Document shape
Shape1.Text = "I love chocolate " + CHR$(13) + "a whole lot!"          ' Add text to the shape
ShapeText = Shape1.TextLF                              ' Read text, preserving Returns
```

Note

- To do this in Living FlowChart script, change ABC. to Application. Remove ABC. from the MsgBox method.

Application Object

Description The Application object is at the top of the OLE Automation hierarchy. It is the interface to OLE Automation. There can be only one Application object at a time running in your system. Below the FlowCharter Application object are the Charts collection, Preferences object, and Menus collection. You can have multiple Charts collections and Menus collections, but only one Preferences object.

Properties	Methods
ActiveChart	Activate
Application	AddMenu
Bottom	ArrangeIcons
Caption	BasicColor
Charts	CascadeCharts
DefaultFilePath	ChartTypeShutdown
FieldViewerVisible	CloseAll
FieldViewerWindowHandle	CreateAddOn
FullName	Help
Height	HidePercentGauge
Hourglass	Hint
Left	MakeRGB
Name	Maximize
NoteViewerVisible	Minimize
NoteViewerWindowHandle	MsgBox
OperatingSystem	New
Parent	NewFromTemplate
Path	Open
PercentGaugeValue	Quit
Preferences	PercentGauge
Printer	PercentGaugeCancelled
Right	RegisterEvent
ShapePaletteVisible	RemoveAddOn
ShapePaletteWindowHandle	RemoveMenu
StatusBar	Restore
StatusBarVisible	TileCharts
Top	Undo
UndoAvailable	UnRegisterEvent
Version	
Visible	
Width	
WindowHandle	
ZoomWindowVisible	

ActiveChart Property

Usage *ApplicationObject*.**ActiveChart**

Description You use the **ActiveChart** property to find the active Chart object in the Application. This is the simplest way to be sure that you are operating on the current chart. The **ActiveChart** property is read only.

Data Type Object

Value The currently active chart

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ActiveChart_Property')}

Activating a Chart
Example

Activate Method (Application Object)
Name Property (Application Object)
Application Object

Bottom Property (Application Object)

Usage *ApplicationObject*.**Bottom** = *PositionInPixels*

Description The **Bottom** property lets you specify the position of the bottom of the FlowCharter window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The **Bottom** property is read/write.

Data Type Long

Value The number of pixels from the bottom of the screen to the bottom of the FlowCharter window

Flow Equivalent none

Displaying the Field Viewer, Notes Viewer, and Shape Palette Example

Bottom Property (Object Object)
Height Property (Application Object)
Left Property (Application Object)
Right Property (Application Object)
Top Property (Application Object)
Width Property (Application Object)

Application Object

Caption Property

Usage *ApplicationObject*.**Caption** = *Title*

Description The **Caption** property lets you customize FlowCharter by changing what it says in the title bar. Set the **Caption** property to "" to restore the standard FlowCharter caption ("Micrografx FlowCharter 7"). The **Caption** property is read/write.

Data Type String

Value The text in the title bar of FlowCharter

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_Caption_Property')}

Charts Property

Usage *ApplicationObject*.**Charts**

Description The **Charts** property lets you find the charts included in the Charts collection. The **Charts** property is read only, but all the properties from the object it returns are read/write.

Data Type Collection object

Value The **Charts** property returns the charts included in the Charts collection.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Charts_Property')}

Charts Property Example

This example uses the **Charts** property of the Application object to put the chart collection into a variable.

```
Dim ABC As Object
Dim Application_Chart As Object

Set ABC = CreateObject("ABCFlow.application")    ' Start ABC
ABC.Visible = True                               ' Make ABC visible
ABC.New                                          ' Create a new chart

Set Application_Chart = ABC.Charts               ' Set the collection of open ABC charts
```

DefaultFilePath Property

Usage *ApplicationObject*.**DefaultFilePath** = *Path*
The *Path* element is the default file path.

Description You use the **DefaultFilePath** property to find or set the default path for all files that are opened or saved. The **DefaultFilePath** property is read/write.

Data Type String

Value The default file path

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DefaultFilePath_Property')}

Assorted Application Object Properties Example 1

This example uses properties of the Application object to find and display the default file path, path, operating system, current printer, and whether Undo is available.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Dim App_File_Path As String
App_File_Path = ABC.DefaultFilePath                  ' Get application default path
ABC.MsgBox "Application Default File Path is " + App_File_Path' Display

Dim EXE_Path As String
EXE_Path = ABC.Path                                  ' Get path to ABC.EXE (executable file)
ABC.MsgBox "The ABC.EXE path is " + EXE_Path         ' Display

Dim Operating_System As String
Operating_System = ABC.OperatingSystem               ' Get operating system
ABC.MsgBox "ABC is running on " + Operating_System          ' Display

Dim ABC_Printer As String
ABC_Printer = ABC.Printer                            ' Get current printer
ABC.MsgBox "Current Printer is " + ABC_Printer       ' Display

Dim Undo_Status As Integer
Undo_Status = ABC.UndoAvailable                      ' Get undo status
Select Case Undo_Status                              ' Display
   Case True
        ABC.MsgBox "Undo is available."
   Case Else
        ABC.MsgBox "Undo is unavailable."
End Select
```

FieldViewerVisible Property {button Flow Equivalent,JI(`FLOW.HLP>procedur','IDH_To_show_and_hide_the_Field_Viewer');CW(`concfull')}

Usage *ApplicationObject*.**FieldViewerVisible** = {True | False}

Description The **FieldViewerVisible** property lets you show or hide the FlowCharter Field Viewer. The **FieldViewerVisible** property is read/write.

Data Type Integer (Boolean)

Value True makes the Field Viewer visible; False makes it invisible.

Flow Equivalent The **FieldViewerVisible** property is equivalent to clicking Field Viewer on the View menu.

{button Related Topics,PI(``,`IDH_RT_FieldViewerVisible_Property')}

Displaying the Field Viewer, Notes Viewer, and Shape Palette
Opening the Field Viewer
Example

NoteViewerVisible Property
ShapePaletteVisible Property
Visible Property (Menu Collection)

Application Object

Assorted Application Object Properties Example 2

This example uses properties of the Application object to see windows belonging to FlowCharter and find their window handles.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Chart.DrawPositionX = 2                              ' Set drawing position
Chart.DrawPositionY = 2                              ' (Default is inches)

Chart.DrawShape ("delay")                            ' Create a shape

Chart.FieldTemplates.Add("Inventory")                ' Create a field for the shape

ABC.FieldViewerVisible = True                        ' Make field viewer visible
Dim Field_Viewer_Window_Handle As Long
Field_Viewer_Window_Handle = ABC.FieldViewerWindowHandle        ' Get Window Handle
ABC.MsgBox "Field Viewer Window Handle = " + CStr(Field_Viewer_Window_Handle)
ABC.FieldViewerVisible = False                       ' Make field viewer invisible

ABC.NoteViewerVisible = True                         ' Make note viewer visible
Dim Note_Viewer_Window_Handle As Long
Note_Viewer_Window_Handle = ABC.NoteViewerWindowHandle          ' Get Window Handle
ABC.MsgBox "Note Viewer Window Handle = " + CStr(Note_Viewer_Window_Handle)
ABC.NoteViewerVisible = False                        ' Make note viewer invisible

ABC.ShapePaletteVisible = True                       ' Make shape palette visible
Dim Shape_Palette_Window_Handle As Long
Shape_Palette_Window_Handle = ABC.ShapePaletteWindowHandle      ' Get Window Handle
ABC.MsgBox "Shape Palette Window Handle = " + CStr(Shape_Palette_Window_Handle)
ABC.ShapePaletteVisible = False                      ' Make shape palette invisible
```

FieldViewerWindowHandle Property

Usage *ApplicationObject*.**FieldViewerWindowHandle**

Description The **FieldViewerWindowHandle** property lets you find the handle to the window of the Field Viewer. If the window is not visible, its value is Null. The **FieldViewerWindowHandle** property is read only.

Data Type Long

Value The handle to the Field Viewer window. If the window is not visible, the value is Null.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_FieldViewerWindowHandle_Property')}

Window Handles
Example

NoteViewerWindowHandle Property
ShapePaletteWindowHandle Property
WindowHandle Property

Application Object

FullName Property (Application Object)

Usage *ApplicationObject*.**FullName**

Description The **FullName** property of the Application object lets you find the FlowCharter path, including the executable filename. To get the path without the executable file name, use the **Path** property. The **FullName** property is read only.

Data Type String

Value The fully qualified path of the FlowCharter program that is running, including the name of the executable file

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_FullName_Property_Application_Object')}

Getting FlowCharter System Information
Example

FullName Property (Chart Object)
Name Property (Application Object)
OperatingSystem Property
Path Property
Version Property

Application Object

FullName Property and Name Property (Application Object) Example

This example uses the **FullName** property and **Name** property of the Application object to find and display the full name and name of the running application.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.New                                              ' Create a new chart

ABC.MsgBox "Application Full Name = " + ABC.FullName        ' Display

ABC.MsgBox "Default Application Name = " + ABC.Name        ' Display
```

Height Property (Application Object)

Usage *ApplicationObject*.**Height** = *HeightInPixels*

Description The **Height** property lets you specify the position of the height of the FlowCharter window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The **Height** property is read/write.

Data Type Long

Value The height of the FlowCharter window in pixels

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Height_Property_Application_Object')}

Positioning and Resizing the FlowCharter Window
Example

Bottom Property (Object Object)
Height Property (Object Object)
Height Property (PageLayout Object)
Left Property (Application Object)
Right Property (Application Object)
Top Property (Application Object)
Width Property (Application Object)

Application Object

Height, Width Property (Application Object) Example

This example uses the **Height** property and **Width** property of the Application object to find and display the height and width of the FlowCharter window.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

ABC.MsgBox "Application Height = " + ABC.Height    ' Display

ABC.MsgBox "Application Width = " + ABC.Width      ' Display
```

Left Property (Application Object)

Usage *ApplicationObject*.**Left** = *PositionInPixels*

Description The **Left** property lets you specify the position of the left side of the FlowCharter window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The **Left** property is read/write.

Data Type Long

Value The number of pixels from the left of the screen to the left side of the FlowCharter window

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Left_Property_Application_Object')}

Positioning and Resizing the FlowCharter Window
Example

Bottom Property (Application Object)
Height Property (Application Object)
Left Property (Object Object)
Right Property (Application Object)
Top Property (Application Object)
Width Property (Application Object)

Application Object

Name Property (Application Object)

Usage *ApplicationObject*.**Name**

Description The **Name** property always equals "FlowCharter" for compatibility with all FlowCharter products. The **Name** property is read only.

Data Type String

Value Always equals "FlowCharter"

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Name_Property_Application_Object')}

Getting FlowCharter System Information
Example

FullName Property (Application Object)
Name Property (Chart Object)
Name Property (FieldTemplate Object)
Name Property (FieldValue Object)
Name Property (Font Object)
OperatingSystem Property
Path Property
Version Property

Application Object

NoteViewerVisible Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_NOTES');CW(`concfull')}

Usage *ApplicationObject*.**NoteViewerVisible** = {True | False}

Description You use the NoteViewerVisible property to find or set whether the Note window is open or closed. The NoteViewerVisible property is read/write.

Data Type Integer (Boolean)

Value True makes the Note Viewer visible; False makes it invisible.

Flow Equivalent The **NoteViewerVisible** property is equivalent to clicking Note on the View menu.

{button Related Topics,PI(`',`IDH_RT_NoteViewerVisible_Property')}

NoteViewerWindowHandle Property

Usage *ApplicationObject*.**NoteViewerWindowHandle**

Description The **NoteViewerWindowHandle** property lets you find the handle to the window of the Note Viewer. If the window is not visible, its value is Null. The **NoteViewerWindowHandle** property is read only.

Data Type Long

Value The handle to the Note Viewer window. If the window is not visible, the value is Null.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_NoteViewerWindowHandle_Property')}

Window Handles
Example

FieldViewerWindowHandle Property
ShapePaletteWindowHandle Property
WindowHandle Property

Application Object

OperatingSystem Property

Usage *ApplicationObject*.**OperatingSystem**

Description The **OperatingSystem** property lets you find the operating system under which FlowCharter is running. The **OperatingSystem** property is read only.

Data Type String

Value The operating system under which the FlowCharter program is running. For example, it equals "DOS 6.21;Windows 3.11" if you are running those versions.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_OperatingSystem_Property')}

Getting FlowCharter System Information
Example

FullName Property (Application Object)
Name Property (Application Object)
Path Property
Version Property

Application Object

Path Property

Usage *ApplicationObject*.**Path**

Description The **Path** property lets you find the FlowCharter application path, excluding the executable filename. The path does not include a final back slash (\). To get the path with the executable file name, use the **FullName** property. The **Path** property is read only.

Data Type String

Value The fully qualified path of the FlowCharter program that is running, excluding the name of the executable file

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_Path_Property')}

Getting FlowCharter System Information
Example

FullName Property (Application Object)
Name Property (Application Object)
OperatingSystem Property
Version Property

Application Object

PercentGaugeValue Property

Usage *ApplicationObject*.**PercentGaugeValue** = *PercentageDone*

Description The **PercentGaugeValue** property lets you set the value in the Percent Gauge dialog box you created using the **PercentGauge** method. The **PercentGaugeValue** property is read/write.

Data Type Integer

Value The value of the percent gauge

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_PercentGaugeValue_Property')}

Preferences Property

Usage *ApplicationObject*.**Preferences**

Description The **Preferences** property lets you find the Preferences object. The **Preferences** property is read only, but all the properties from the object it returns are read/write.

Data Type Object

Value The Preferences object

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Preferences_Property')}

Preferences Property Example

This example uses the **Preferences** property of the Application object to put the preferences collection into a variable.

```
Dim ABC As Object
Dim App_Preferences As Object

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True  ' Make ABC visible
ABC.New                                       ' Create new chart

Set App_Preferences = ABC.Preferences         ' Get the Preferences Object
App_Preferences.ShowRulers = False            ' Turn off the Rulers
ABC.MsgBox ("Click OK to turn the Rulers back on.")
App_Preferences.ShowRulers = True             ' Turn the Rulers on
```

Printer Property

Usage *ApplicationObject*.**Printer** = {*PrinterName* | *PrinterPort*}

Description The **Printer** property lets you find or set the current printer. When you read the value of the **Printer** property, it returns the current printer and port. For example, it might return "HP LaserJet III on LPT2:." When you set the value, the program uses a "loose matching" routine that starts at the beginning of the string. For example, setting the Printer property to "HP Laser" or "LPT2" chooses "HP LaserJet III on LPT2:" if that is the printer on LPT2:. If more than one printer matches the value you set, the first one alphabetically is used.

Data Type String

Value The current printer

Flow Equivalent The **Printer** property is equivalent to clicking Printer Setup on the File menu and clicking the printer you want to use.

{button Related Topics,PI(``,`IDH_RT_Printer_Property')}

Right Property (Application Object)

Usage *ApplicationObject*.**Right** = *PositionInPixels*

Description The **Right** property lets you specify the position of the right side of the FlowCharter window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The **Right** property is read/write.

Data Type Long

Value The number of pixels from the right of the screen to the right side of the FlowCharter window

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Right_Property_Application_Object')}

Positioning and Resizing the FlowCharter Window
Example

Bottom Property (Application Object)
Height Property (Application Object)
Left Property (Application Object)
Right Property (Object Object)
Top Property (Application Object)
Width Property (Application Object)

Application Object

ShapePaletteVisible Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Shape_Palettes_Command');CW(`concfull')}

Usage *ApplicationObject*.**ShapePaletteVisible** = {True | False}

Description You use the ShapePaletteVisible property to find or set whether the Shape Palette window is open or closed. The ShapePaletteVisible property is read/write.

Data Type Integer (Boolean)

Value True makes the Shape Palette visible; False makes it invisible.

Flow Equivalent The **ShapePaletteVisible** property is equivalent to clicking Shape Palettes on the View menu.

{button Related Topics,PI(``,`IDH_RT_ShapePaletteVisible_Property')}

ShapePaletteWindowHandle Property

Usage *ApplicationObject*.**ShapePaletteWindowHandle**

Description The **ShapePaletteWindowHandle** property lets you find the handle to the window of the Shape Palette. If the window is not visible, its value is Null. The **ShapePaletteWindowHandle** property is read only.

Data Type Long

Value The handle to the Shape Palette window. If the window is not visible, the value is Null.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ShapePaletteWindowHandle_Property')}

Window Handles
Example

FieldViewerWindowHandle Property
NoteViewerWindowHandle Property
WindowHandle Property

Application Object

StatusBar Property

Usage *ApplicationObject*.**StatusBar** = *StatusBarText*

Description The **StatusBar** property lets you customize FlowCharter by changing what it says in the status bar. You can restore the normal status bar hints by setting the **StatusBar** property to "". To set a temporary message in the hint line, use the **Hint** method. The **StatusBar** property is read/write.

Data Type String

Value The text in the status bar

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_StatusBar_Property')}

Top Property (Application Object)

Usage *ApplicationObject*.**Top** = *PositionInPixels*

Description The **Top** property lets you specify the position of the top of the FlowCharter window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The **Top** property is read/write.

Data Type Long

Value The number of pixels from the top of the screen to the top of the FlowCharter window

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Top_Property_Application_Object')}

Positioning and Resizing the FlowCharter Window
Example

Bottom Property (Application Object)
Height Property (Application Object)
Left Property (Application Object)
Right Property (Application Object)
Top Property (Object Object)
Width Property (Application Object)

Application Object

Top Property (Application Object) Example

This example uses the **Top** property, **Bottom** property, **Left** property, and **Right** property of the Application object to find and display the location of the FlowCharter window.

```
Dim ABC As Object
Dim Chart As Object
Dim App_Top As Long, App_Bottom As Long, App_Left As Long, App_Right As Long

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

App_Top = ABC.application.Top
App_Bottom = ABC.application.Bottom
App_Left = ABC.application.Left
App_Right = ABC.application.Right

ABC.MsgBox "ABC's window border location is: Top = " + CStr(App_Top) + ", Bottom = " +
CStr(App_Bottom) + ", Left = " + CStr(App_Left) + ", and Right = " + CStr(App_Right)
```

UndoAvailable Property

Usage *ApplicationObject*.**UndoAvailable**

Description The **UndoAvailable** property lets you find if there is anything to undo. The **UndoAvailable** property is read only.

Data Type Integer (Boolean)

Value True means something is available to undo; False means nothing is available to undo.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ItemFromSelection_Method')}

Version Property

Usage *ApplicationObject*.**Version**

Description The **Version** property lets you find the version of the OLE automation application object that is running. Note that this is not the version number of the FlowCharter application, but rather the version number of the OLE Automation API set. For example, for FlowCharter 7.0, the **Version** property returns 7.0. The **Version** property is read only.

Data Type String

Value The version of the OLE Automation application object that is running. For example, it equals "7.0" if you are running FlowCharter7.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Version_Property')}

Getting FlowCharter System Information Example

FullName Property (Application Object)
Name Property (Application Object)
OperatingSystem Property
Path Property

Application Object

Object Object Properties Example

This example uses properties of the Object object to determine the application's version, if the application is visible, what the application's window handle is, the message in the application's status bar, and the application's caption in the title bar.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Dim App_Version As String                              ' Get application version
App_Version = ABC.Version
ABC.MsgBox "Application version = " + App_Version      ' Display

Dim App_Visible As Integer                             ' Get application visibility state
App_Visible = ABC.Visible
Select Case App_Visible                                ' Display
   Case True
      ABC.MsgBox "Application is visible."
   Case Else
      ABC.MsgBox "Application is not visible."
End Select

Dim App_Window_Handle As Long                          ' Get application window handle
App_Window_Handle = ABC.WindowHandle
ABC.MsgBox "Application window handle = " + Hex$(App_Window_Handle)     ' Display

Dim App_Status_Bar As String                           ' Get application status bar message
App_Status_Bar = ABC.StatusBar
ABC.MsgBox "Application status bar = " + App_Status_Bar      ' Display

Dim App_Caption As String                              ' Get application caption
App_Caption = ABC.Caption
ABC.MsgBox "Application caption = " + App_Caption      ' Display
```

Visible Property (Application Object)

Usage *ApplicationObject*.**Visible** = {True | False}

Description If you set the **Visible** property to True, the application is visible. If you set the **Visible** property to False, the application is still running, but it is not visible. You cannot switch to it using **Alt+Tab**, and it is not shown in the Task List dialog box that appears when you press **Ctrl+Esc**. The value False is the default, so you must begin all your programs by setting it to True. The **Visible** property is read/write.

Data Type Integer (Boolean)

Value True makes FlowCharter visible. False makes FlowCharter not visible; you cannot switch to it using **Alt+Tab**, and it is not shown in the Task List dialog box that appears when you press **Ctrl+Esc**.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Visible_Property_Application_Object')}

Starting FlowCharter
Example

Activate Method (Application Object)
Activate Method (Chart Object)

Application Object

Width Property (Application Object)

Usage *ApplicationObject*.**Width** = *WidthInPixels*

Description The **Width** property lets you specify the position of the width of the FlowCharter window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The **Width** property is read/write.

Data Type Long

Value The width of the FlowCharter window in pixels

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Width_Property_Application_Object')}

Positioning and Resizing the FlowCharter Window
Example

Bottom Property (Application Object)
Height Property (Application Object)
Left Property (Application Object)
Right Property (Application Object)
Top Property (Application Object)
Width Property (Object Object)
Width Property (PageLayout Object)

Application Object

WindowHandle Property

Usage *ApplicationObject*.**WindowHandle**
*ChartObject.***WindowHandle**

Description The **WindowHandle** property lets you find the handle to the window of FlowCharter or of a chart. If the window is not visible, its value is Null. The **WindowHandle** property is read only.

Data Type Long

Value The handle to the window of FlowCharter or the chart. If the window is not visible, the value is Null.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_WindowHandle_Property')}

Window Handles
Example

FieldViewerWindowHandle Property
NoteViewerWindowHandle Property
ShapePaletteWindowHandle Property

Application Object

WindowHandle Property Example

This example uses the **WindowHandle** property of the Chart object to find a chart's window handle.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

ABC.MsgBox "The window handle for this chart is " + Chart.WindowHandle + "."
```

{button Other Example,JI(`',`IDH_Version_Property_Example')}

Activate Method (Application Object)

Usage *ApplicationObject*.**Activate**

Description You bring FlowCharter to the front using the **Activate** method of the Application object. You usually have to do this only after the user has done something that moves FlowCharter to the back, such as clicking another application that is visible on the screen or switching to another application using **ALT+TAB** or **CTRL+ESC**.

Flow Equivalent none

{button Related Topics,PI(`',`IDH_RT_Activate_Method_Application_Object')}

Bringing FlowCharter or a Chart to the Front
Example

Activate Method (Chart Object)
Visible Property (Application Object)

Application Object

Activate Method (Application Object) Example

This example uses the **Activate** method of the Application object to activate FlowCharter.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible

ABC.Activate                                       ' Activate ABC
ABC.New                                            ' Create a new chart
```

ArrangeIcons Method {button Flow Equivalent,JI(`FLOW.HLP',`IDH_Arrange_All_Command');CW(`concfull')}

Usage *ApplicationObject*.**ArrangeIcons**

Description When you have several FlowCharter chart windows minimized to icons, you can arrange them at the bottom of the FlowCharter window using the **ArrangeIcons** method.

Flow Equivalent The **ArrangeIcons** method is equivalent to clicking Arrange All on the Window menu for a chart that is minimized.

{button Related Topics,PI(``,`IDH_RT_ArrangeIcons_Method')}

ArrangeIcons Method Example

This example uses the **ArrangeIcons** method of the Application object to arrange FlowCharter icons. For this call to have any visible effect, one or more chart windows must be minimized and be moved from their original positions.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible

ABC.ArrangeIcons                                   ' Arrange icons
```

BasicColor Method

Usage *ApplicationObject*.**BasicColor** (*Color*)
The *Color* element is an integer representing one of the sixteen standard VGA colors.

Description The **BasicColor** method lets you set colors from the sixteen VGA colors. The method returns the color as a long decimal value. You cannot change the values in the **BasicColor** method. For example, you cannot make BasicColor(10) yield the color purple.

Data Type Long

Value The **BasicColor** method returns one of the following values, based on the *Color* element.

Color	**BasicColor**	
Element	**Value**	**Result**
0	16777215	White
1	0	Black
2	255	Red
3	65280	Green
4	16711680	Blue
5	65535	Yellow
6	16711935	Magenta
7	16776960	Cyan
8	12632256	Gray
9	127	Dark Red
10	32512	Dark Green
11	8323072	Dark Blue
12	326397	Dark Yellow
13	8323199	Dark Magenta
14	8355584	Dark Cyan
15	8355711	Dark Gray

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_BasicColor_Method')}

BasicColor Method Example

This example uses the **BasicColor** method of the Application object to arrange FlowCharter icons. For this call to function, FlowCharter must be active and chart windows must be minimized.

```
Dim ABC As Object
Dim Basic_Color As Long
Dim User_Input As String

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

' Get Basic Color for input value from 0 - 15
a:
User_Input = InputBox$("Please Enter a Windows color with a value from 0 - 15", "Basic Color")

Select Case User_Input
   Case "0"
        ABC.MsgBox "Entered color is WHITE."
   Case "1"
        ABC.MsgBox "Entered color is BLACK."
   Case "2"
        ABC.MsgBox "Entered color is RED."
   Case "3"
        ABC.MsgBox "Entered color is GREEN."
   Case "4"
        ABC.MsgBox "Entered color is BLUE."
   Case "5"
        ABC.MsgBox "Entered color is YELLOW."
   Case "6"
        ABC.MsgBox "Entered color is MAGENTA."
   Case "7"
        ABC.MsgBox "Entered color is CYAN."
   Case "8"
        ABC.MsgBox "Entered color is GRAY."
   Case "9"
        ABC.MsgBox "Entered color is DK_RED."
   Case "10"
        ABC.MsgBox "Entered color is DK_GREEN."
   Case "11"
        ABC.MsgBox "Entered color is DK_BLUE."
   Case "12"
        ABC.MsgBox "Entered color is DK_YELLOW."
   Case "13"
        ABC.MsgBox "Entered color is DK_MAGENTA."
   Case "14"
        ABC.MsgBox "Entered color is DK_CYAN."
   Case "15"
        ABC.MsgBox "Entered color is DK_GRAY."
   Case Else
```

```
        ABC.MsgBox "Unrecognized entry. Please try again."
        GoTo a:
End Select

Basic_Color = ABC.BasicColor(User_Input)

ABC.MsgBox "Long conversion of color is " + CStr(Basic_Color)' Display return value
```

CascadeCharts Method

Usage *ApplicationObject*.**CascadeCharts**

Description When you have several FlowCharter chart windows open, you can arrange them in the FlowCharter window using the **CascadeCharts** method. The charts are arranged so the title bar of each one is visible.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_CascadeCharts_Method')}

Arranging FlowCharter Charts Example

TileCharts Method

Application Object

CascadeCharts Method Example

This example uses the **CascadeCharts** method of the Application object to cascade all open charts.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible

For I = 1 To 5                                     ' Create a series of new charts
    ABC.New
Next I

ABC.CascadeCharts                                  ' Cascade all open charts
```

CloseAll Method {button Flow Equivalent,JI(`FLOW.HLP',`IDH_Close_All_Command');CW(`concfull')}

Usage *ApplicationObject*.**CloseAll**
ChartsCollection.**CloseAll**

Description You use the **CloseAll** method to close all charts in the FlowCharter workspace. If changes have been made, the user is not prompted to save changed charts.

Flow Equivalent The **CloseAll** method is equivalent to clicking Close All on the File menu, except that the user is not prompted to save changes to changed charts.

{button Related Topics,PI(``,`IDH_RT_CloseAll_Method')}

CloseAll Method Example

This example uses the **CloseAll** method of the Application object to close all charts.

```
Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible

For I = 1 To 5                                     ' Create a series of new charts
   ABC.New
Next I

ABC.CloseAll                                       ' Close all charts
```

{button Other Example,JI(`',`IDH_RT_CloseAll_Method_Example')}

Help Method

Usage *ApplicationObject*.**Help** [*HelpFileName*] {, [*ContextID*] | [*HelpContext*]}
The *HelpFileName* element, an optional string, is the name of a Windows help file. Quotation marks should be used whenever long filenames or long pathnames are used.
The *ContextID* element, an optional integer, is a context ID.
The *HelpContext* element, an optional string, specifies the help context to display. In Help, you see the *HelpContext* elements in the top list box when you click the Search button.

Description The **Help** method lets you run a help file. The first element specifies the help file to run. If you omit the first element, the help file shipped with FlowCharter runs. The second element is either a context ID (an integer) or a help context (a string) to call a particular topic in the help file. If you omit the element, the Contents of the help file appears.

Flow Equivalent If you use the **Help** method to run the help file that ships with FlowCharter, the method is equivalent to pressing **F1** in the proper context clicking FlowCharter Help on the Help menu. If you are running a help file that you created, there is no FlowCharter equivalent.

{button Related Topics,PI(`',`IDH_RT_Help_Method')}

Help Method Example

This example uses the **Help** method of the Application object to run the FlowCharter help at a specific help topic.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC

ABC.Help , 57345                                     ' Run ABC Help at Glossary topic
```

HidePercentGauge Method

Usage *ApplicationObject*.**HidePercentGauge**

Description The **HidePercentGauge** method lets you close the Percent Gauge dialog box you created using the **PercentGauge** method.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_HidePercentGauge_Method')}

HidePercentGauge Method Example

This example uses the **HidePercentGauge** method of the Application object to remove a Percent Gauge dialog box.

```
Dim ABC As Object
Dim ABCGauge As Single

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart

ABC.PercentGauge                                   ' Create gauge

ABC.HidePercentGauge                               ' Remove gauge

ABC.MsgBox "Percentage Gauge Hidden"
```

New Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_New_Command');CW(`concfull')}

Usage *ApplicationObject*.**New**

Description You use the **New** method to create a new chart with default attributes. This opens a new chart window.

Data Type Object

Value The chart object

Flow Equivalent The **New** method is equivalent to clicking New on the File menu.

{button Related Topics,PI(`',`IDH_RT_New_Method')}

New Method Example

This example uses the **New** method of the Application object to create a new chart.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible

ABC.New                                            ' Create a new chart
```

NewFromTemplate Method

Usage *ApplicationObject*.**NewFromTemplate** (*TemplateName*)
The *TemplateName* element is the path and name of the template to use to create the chart. Quotation marks should be used whenever long filenames or long pathnames are used.

Description You use the **NewFromTemplate** method to create a new chart based on the specified chart template name. If *TemplateName* file cannot be loaded for any reason, the returned Chart.Valid is False.

Data Type Object

Value The chart that is created

Flow Equivalent The **NewFromTemplate** method is equivalent to clicking Open on the File menu, choosing file type AFT, then saving the chart as file type FLO.

{button Related Topics,PI(`',`IDH_RT_NewFromTemplate_Method')}

NewFromTemplate Method Example

This example uses the **NewFromTemplate** method of the Application object to create a new file using a template.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

ABC.NewFromTemplate (ABC.Path + "\Template\Living FlowCharts\Basic.aft")              ' Open
FlowCharter template file
```

Open Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_OPEN');CW(`concfull')}

Usage *ChartsCollection*.**Open** (*PathName* [, *AsReadOnly*])
ApplicationObject.**Open** (*PathName* [, *AsReadOnly*])
The *PathName* element is the path and name of the chart to open. Quotation marks should be used whenever long filenames or long pathnames are used.
The *AsReadOnly* element opens the chart as read only.

Description The **Open** method in the Charts collection and Application object work the same way and have the same effect. You use the **Open** method to open a chart. If the chart is already open, the **Open** method moves the chart to the front. You can optionally specify that the chart is to be opened read only.

You can open the following file types.

- Charts (filenames ending with an ABC, AF3 or FLO extension; files that contain the shapes, lines, and text that comprise your charts)
- Templates (filenames ending with an AFT extension; files that hold object attributes and page layouts used by your charts)

Data Type Object. The *AsReadOnly* element is an integer (Boolean)

Value The Chart object

Flow Equivalent The **Open** method is equivalent to clicking Open on the File menu, clicking the drive and directory that contain the file you want to open, clicking the file you want to open, and clicking OK.

{button Related Topics,PI(``,`IDH_RT_Open_Method')}

Opening Charts
Example

DefaultFilePath Property

Application Object
Charts Collection

Open Method Example

This example uses the **Open** method of the Application object to open a file read/write and then open a file read only.

```
Dim ABC As Object
Dim ABC_Read_Only As Single

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart

ABC_Read_Only = False                                  ' Set parameter to read/write access

ABC.Open ABC.Path + "\Tutorial\National.flo", ABC_Read_Only    ' Open file read/write

ABC_Read_Only = True                                   ' Set parameter to read only access

ABC.Open ABC.Path + "\Tutorial\Orgchart.flo", ABC_Read_Only    ' Open file read only
```

{button Other Example,JI(`',`IDH_Open_Method_Example2')}

Open Method, Item Method, and Count Property Example

This example uses the **Open** method, **Count** property, and **Item** method of the Charts collection to open and print charts.

```
Dim ABC As Object, Charts As Object, Chart As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

ABC.Open (ABC.Path + "\Samples\Dataflow.flo")      ' Open charts to add to collection
ABC.Open (ABC.Path + "\Samples\Network.flo")
ABC.Open (ABC.Path + "\Samples\Orgdemo.flo")

For X = 1 To ABC.Charts.Count                      ' Iterate through all charts
   Set Chart = ABC.Charts.Item(X)                  ' Get a chart
   Chart.PrintOut                                  ' Print the chart
Next X
```

Quit Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_EXIT');CW(`concfull')}

Usage *ApplicationObject*.**Quit**

Description The **Quit** method closes FlowCharter. It does not prompt the user to save changes to open files. Before you close FlowCharter, you should save the files you want to be saved.

Flow Equivalent The **Quit** method is equivalent to clicking Exit on the File menu.

{button Related Topics,PI(`',`IDH_RT_Quit_Method')}

Quit Method Example

This example uses the **Quit** method of the Application object to close FlowCharter.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible

For I = 1 To 3                                     ' Create a series of new charts
    ABC.New
Next I

ABC.Quit                                           ' Close ABC
```

MakeRGB Method

Usage *ApplicationObject*.**MakeRGB** (*Red*, *Green*, *Blue*)
The *Red*, *Green*, and *Blue* elements are integers that define the RGB components of color.

Description The **MakeRGB** method lets you set colors from a palette of over sixteen million colors. You specify the color as quantities of red, green, and blue, with each color a number from 0 (no color) through 255 (solid color).

Data Type Long

Value Returns the decimal equivalent of a six-digit, hexadecimal value. The following table shows some of the values of the red, green, and blue components and their equivalent in decimal and hexadecimal.

Color	MakeRGB	Decimal	Hex
White	(255,255,255)	16777215	FFFFFF
Black	(0,0,0)	0	0
Red	(255,0,0)	255	FF
Green	(0,255,0)	65280	FF00
Blue	(0,0,255)	16711680	FF0000
Yellow	(255,255,0)	65535	FFFF
Magenta	(255,0,255)	16711935	FF00FF
Cyan	(0,255,255)	16776960	FFFF00
Gray	(192,192,192)	12632256	C0C0C0
Dark Red	(127,0,0)	127	7F
Dark Green	(0,127,0)	32512	7F00
Dark Blue	(0,0,127)	8323072	7F0000
Dark Yellow	(127,127,0)	326397	7F7F
Dark Magenta	(127,0,127)	8323199	7F007F
Dark Cyan	(0,127,127)	8355584	7F7F00
Dark Gray	(127,127,127)	8355711	7F7F7F

Flow Equivalent None

RGB Values
Text Color
Example

BasicColor Method

Application Object

MakeRGB Method Example

This example uses the **MakeRGB** method of the Application object to find a color value.

```
Dim ABC As Object
Dim Red_Green_Blue As Long

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.New                                            ' Create a new chart

Red_Green_Blue = ABC.MakeRGB(255, 255, 255)        ' Find color value
ABC.MsgBox CStr(Red_Green_Blue)
```

PercentGauge Method

Usage *ApplicationObject*.**PercentGauge** [*TitleBar*] [, *TextLine1*] [, *TextLine2*]
The optional *TitleBar* element is the name that goes in the title bar.
The optional *TextLine1* element is the first line of text above the gauge.
The optional *TextLine2* element is the second line of text above the title bar.

Description The **PercentGauge** method lets you create a percent gauge, with its value set to 0.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_PercentGauge_Method')}

PercentGauge Method, PercentGaugeValue Property Example

This example uses the **PercentGauge** method and **PercentGaugeValue** property of the Application object to create and increment a gauge.

```
Dim ABC As Object
Dim ABCGauge As Single

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

ABC.PercentGauge                                     ' Make gauge visible

For I = 1 To 100                                     ' Incrementally increase gauge value
   ABC.PercentGaugeValue = ABC.PercentGaugeValue + 1
Next I
```

PercentGaugeCancelled Method

Usage *ApplicationObject*.**PercentGaugeCancelled**

Description The **PercentGaugeCancelled** method lets you determine whether the user has clicked the Cancel button in the Percent Gauge dialog box you created using the PercentGauge method.

Data Type Integer (Boolean)

Value True means the user clicked the Cancel button; False means the user did not click the Cancel button.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_PercentGaugeCancelled_Method')}

PercentGaugeCancelled Method Example

This example uses the **PercentGaugeCancelled** method of the Application object to detect if the user has clicked the Cancel button on the Percent Gauge dialog box.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim X

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawSpacingX = .25                               ' Set horizontal draw spacing
ABC.PercentGauge "My Percent Gauge"; "Draws Decision shapes"; "and changes their colors."
                                                       ' Create a percent gauge

For X = 1 To 25
   Set NewShape = Chart.DrawShape("Decision")          ' Draw a shape
   NewShape.Shape.FillColor = ABC.MakeRGB(127, 0, X * 10)  ' Set the shape color
   ABC.PercentGaugeValue = X * 4                       ' Increment the percent gauge
   If ABC.PercentGaugeCancelled Then GoTo StopItNow          ' If the Cancel button
                                                       ' is pressed, stop
Next X

ABC.HidePercentGauge                                   ' Remove the percent gauge
Exit Sub                                               ' Stop the subroutine

StopItNow:
ABC.HidePercentGauge                                   ' Remove the percent gauge
NewShape.Text = "Cancel pressed!"                      ' Place text in the last shape drawn
Exit Sub                                               ' Stop the subroutine
```

TileCharts Method

Usage *ApplicationObject*.**TileCharts**

Description When you have several FlowCharter chart windows open, you can arrange them in the FlowCharter window using the **TileCharts** method. The charts are arranged so that a portion of each is visible.

Flow Equivalent The **TileCharts** method is equivalent to clicking Arrange All on the Window menu for a chart that is restored.

Arranging FlowCharter Charts Example

CascadeCharts Method

Application Object

TileCharts Method Example

This example uses the **TileCharts** method of the Application object to tile the open charts.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible

For I = 1 To 5                                     ' Create a series of new charts
    ABC.New
Next I

ABC.TileCharts                                     ' Tile all open charts
```

Undo Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Undo');CW(`concfull')}

Usage *ApplicationObject*.**Undo**

Description You use the **Undo** method to undo the last FlowCharter action. You can find out if there is anything to undo using the **UndoAvailable** property.

Flow Equivalent The **Undo** method is equivalent to clicking Undo on the Edit menu.

{button Related Topics,PI(`',`IDH_RT_Undo_Method')}

Undo Method Example

This example uses the **Undo** method of the Application object to undo a user action.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Msg1 As String, Msg2 As String, Msg3 As String

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Set NewShape = Chart.DrawShape("Operation")          ' Draw a shape
NewShape.Color = ABC.MakeRGB(0, 0, 255)              ' Make the shape blue
Msg1 = "A blue shape has just been drawn. "
Msg2 = "Please move this message out of the way and move or delete the shape. "
Msg3 = "Then click OK in this message. The code will undo whatever you just did."
ABC.MsgBox Msg1 + Msg2 + Msg3

If ABC.UndoAvailable Then                            ' If undo is available
        ABC.Undo                                     ' Undo the last action
   Else
        ABC.MsgBox "There is nothing to undo!"
End If
```

MsgBox Method

Usage

ApplicationObject.**MsgBox** *MessageText* [,*BoxType*] [,*BoxTitle*]
The *MessageText* element is the message that goes in the dialog box.
The optional *BoxType* element defines the type of dialog box. If you omit this element, the value is 0.
The optional *BoxTitle* element sets the title bar text of the dialog box. If you omit this element, the title of the dialog box is "Micrografx FlowCharter 7."

Description

The **MsgBox** method lets you post a dialog box. The method is similar to the MsgBox function used in the Visual Basic programming language.

Data Type

Integer

Value

The following table shows the value returned according to the button that the user selected. Note: In Visual Basic, these values have constants associated with them, such as IDOK. Those constants are not available for OLE Automation.

Button Selected	Value
OK	1
Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

The following table shows the values available for the *BoxType* element, which is optional. The value of the second element can be the sum of values from the table. Note: In Visual Basic, these values have constants associated with them, such as MB_OK. Those constants are not available for OLE Automation.

Value	Effect
0	Display OK button only
1	Display OK and Cancel buttons
2	Display Abort, Retry, and Ignore buttons
3	Display Yes, No, and Cancel buttons
4	Display Yes and No buttons
5	Display Retry and Cancel buttons
16	Display stop icon 
32	Display question mark icon 
48	Display exclamation point icon 
64	Display information icon 
0	First button is the default
256	Second button is the default
512	Third button is the default
0	The dialog box is application modal, so FlowCharter is suspended until

	the user responds to the dialog box
4096	The dialog box is system modal, so all applications are suspended until the user responds to the dialog box

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_MsgBox_Method')}

RegisterEvent Method

Usage *ApplicationObject*.**RegisterEvent** *OCXName*,, *IdString*, *EventName* [, *ChartType*]

The *OCXName* element identifies the OLE Automation control to which the registered events apply. Unless you have changed the OLE Automation control's **Name** property from its default setting, OCX*Name* is ABC1. If you are using ABCAUTO.VBX from Visual Basic 3.0, use ABC1.VBX for the *OCXName* parameter.

IdString identifies the Visual Basic form on which the FlowCharter control is located. It is normally the **Caption** property setting of the form.

The *EventName* element is the name of the event being registered. This name must be enclosed in quotes.

The *ChartType* element, which is optional, lets you register the event for only a particular type of chart. You set a chart's type with the **Type** method of the Chart object. If you omit the *ChartType*, the registered events apply to all charts.

Note: This method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX.

Description The **RegisterEvent** method lets you register an event procedure. If you do not register an event, OLE Automation does not respond when the user of your program performs the event.

Data Type Integer (Boolean)

Value True means the event was successfully registered; False means it was not.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_RegisterEvent_Method')}

Registering Event Procedures
Example

Type Property (Chart Object)
TypeRequiresEXE Property
TypeUsesEXE Property
UnRegisterEvent Method

Application Object

RegisterEvent, UnRegisterEvent Method Example

This example uses the **RegisterEvent** method and **UnRegisterEvent** method of the Application object to register an event. The code must be in three different places in Visual Basic.

The first section of code goes in Form_Load or in a Command button.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Msg1 As String, Msg2 As String

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Add a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

' Begin accepting user actions
ABC.RegisterEvent ABC1, Form1.Caption, "ObjectMovedNOTIFY"
ABC.RegisterEvent ABC1, Form1.Caption, "ObjectLineAttachNOTIFY"

Msg1 = "After you click OK, two shapes will appear. Please move either shape once. "
Msg2 = "This application will know when you have moved one, and you will see it change."
ABC.MsgBox Msg1 + Msg2
Set Shape1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Shape2 = Chart.DrawShape("Decision")
Shape1.Text = "Move me!"                             ' Place text in the shapes
Shape2.Text = "No! Move me!"
Shape2.Shape.FitShapeToText                          ' Resize the shape so its text fits

End Sub
```

The second section of code goes in ABC1_ObjectMovedNOTIFY.

```
Dim ABC As Object
Set ABC = Object.Application

Object.Color = ABC.MakeRGB(255, 0, 0)   ' Turn shape Red
                                        ' Place text in the shape
Object.Text = "Eeek! I've been moved! Please draw a line from me to the other
shape."
Object.Shape.FitShapeToText                    ' Resize the shape so its text fits
```

The third section of code goes in ABC1_ObjectLineAttachedNOTIFY.

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.application")        ' Start ABC

ABC.MsgBox "This application will now stop receiving ABC events."

' Stop accepting user actions
ABC.UnRegisterEvent ABC1, "ObjectMovedNOTIFY"
```

```
ABC.UnRegisterEvent ABC1, "ObjectLineAttachNOTIFY"
```

UnRegisterEvent Method

Usage *ApplicationObject*.**UnRegisterEvent** *OCXName*.OCX, *EventName* [, *ChartType*]

The *OCXName*.OCX element identifies the OLE Automation control to which the registered events apply. Unless you have changed the OLE Automation control's **Name** property from its default setting, OC*XName* is ABC1. If you are using ABCAUTO.VBX and Visual Basic 3.0, use ABC1.VBX for the OCXName parameter.

The *EventName* element is the name of the event being unregistered. This name must be enclosed in quotes.

The *ChartType* element, which is optional, lets you unregister the event for only a particular type of chart. You set a chart's type with the **Type** method of the Chart object. If you omit the *ChartType*, the unregister applies to all charts.

Note: This method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX.

Description The **UnRegisterEvent** method lets you unregister an event procedure. If you do not unregister an event, OLE Automation continues to respond when the user of your program performs the event.

Data Type Integer (Boolean)

Value True means the event was successfully unregistered; False means it was not.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_UnRegisterEvent_Method')}

Registering Event Procedures
Example

RegisterEvent Method
Type Property (Chart Object)
TypeRequiresEXE Property
TypeUsesEXE Property

Application Object

CreateAddOn Method

Usage *ApplicationObject*.**CreateAddOn** *Position*, *HintName*, *ProgramFileName* , , [*MenuItem*]
The *Position* element specifies the position of the menu item. Use -1 for the first available position.
The *HintName* element is the name of the button. The text you enter is used in the hint line.
The *ProgramFileName* element is the name of the program to run, including the fully qualified path. If the path contains a long filename, the string must be contained within quote marks.
The fourth parameter is unused.

MenuItem is the name you want for the menu item. If no title is specified for the menu item, the hintline text is used. If there is no hint name, the name of the executable file is used (including extension.

Description The **CreateAddOn** method of the Menu collection lets you add menu items to the Add Ons submenu of the Tools menu.

Data Type Integer (Boolean)

Value True means the item was created successfully; False means it was not.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_CreateAddOn_Method')}

CreateAddOn, RemoveAddOn Method Example

This example uses the **CreateAddOn** method and **RemoveAddOn** method of the Application object to add a button to the toolbox and then remove it.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

AppPath = ABC.Path                                   ' Set path to ABC
SPath = AppPath + "\Samples\Billing.exe"             ' Set path to chart file
BPath = AppPath + "\Samples\Toolbar.ico"             ' Set path to icon file

Z = ABC.CreateAddOn(3, "Weekly Billing", , Bpath, "WkBilling")          ' Add an item

ABC.MsgBox "Notice that a menu item WkBilling, has been added to the Add Ons menu. Clicking OK
will remove the item."
Z = ABC.RemoveAddOn(3)                               ' Remove the item at position 3
```

RemoveAddOn Method

Usage *ApplicationObject*.**RemoveAddOn** {*Position* | *ProgramFileName*}
The *Position* element specifies the position of the menu item
The *ProgramFileName* element is the name of the menu item.

Description The **RemoveAddOn** method lets you remove a button that you have added to the FlowCharter toolbox.

Data Type Integer (Boolean)

Value True means the menu item was removed successfully; False means it was not.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_RemoveAddOn_Method')}

Adding Buttons
Example

CreateAddOn Method
RemoveMenu Method

Application Object

Restore Method (Application Object)

Usage *ApplicationObject*.**Restore**

Description The **Restore** method of the Application object lets you change the FlowCharter window to its previous size.

Flow Equivalent none

{button Related Topics,PI(`',`IDH_RT_Restore_Method_Application_Object')}

Minimizing, Maximizing, and Restoring a Window Example

Maximize Method (Application Object)
Minimize Method (Application Object)
Restore Method (Chart Object)

Application Object

Minimize Method (Application Object)

Usage *ApplicationObject*.**Minimize**

Description The **Minimize** method of the Application object lets you change the FlowCharter window to an icon.

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Minimize_Method_Application_Object')}

Minimizing, Maximizing, and Restoring a Window Example

Maximize Method (Application Object)
Minimize Method (Chart Object)
Restore Method (Application Object)

Application Object

Maximize Method (Application Object)

Usage *ApplicationObject*.**Maximize**

Description The **Maximize** method of the Application object lets you change the FlowCharter window to its maximum size.

Flow Equivalent none

{button Related Topics,PI(`',`IDH_RT_Maximize_Method_Application_Object')}

Minimizing, Maximizing, and Restoring a Window Example

Maximize Method (Chart Object)
Minimize Method (Application Object)
Restore Method (Application Object)

Application Object

Maximize, Minimize, Restore Method (Application Object) Example

This example uses the **Maximize** method, **Minimize** method, and **Restore** method of the Application object to minimize, restore, and maximize the FlowCharter window.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

ABC.MsgBox "Click OK to minimize ABC FlowCharter."
ABC.Minimize                                         ' Minimize ABC

ABC.MsgBox "Click OK to maximize ABC FlowCharter."
ABC.Maximize                                         ' Maximize ABC

ABC.MsgBox "Click OK to restore ABC FlowCharter to normal size."
ABC.Restore                                          ' Restore ABC
```

Hourglass Property

Usage *ApplicationObject*.**Hourglass** = {True | False}

Description The **Hourglass** property lets you change the pointer to a wait cursor or back to the FlowCharter pointer. The **Hourglass** property is read/write.

Data Type Integer (Boolean)

Value True makes the pointer a wait cursor; False makes the cursor the FlowCharter pointer.

Flow Equivalent None

Hourglass Property Example

This example uses the **Hourglass** property of the Application object to indicate that FlowCharter is busy while shapes are drawing.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim x

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

ABC.MsgBox "When you click OK an Hour Glass pointer will appear while shapes are drawn."
Chart.View = 1                                         ' View the current page
Chart.DrawSpacingX = .25                               ' Set horizontal draw spacing
ABC.Hourglass = True                                   ' Display wait cursor
For x = 1 To 25
   Set NewShape = Chart.DrawShape("Operation")         ' Draw a shape
   NewShape.Shape.FillColor = ABC.MakeRGB(200, (x * 4), x)  ' Set shape color
Next x
ABC.Hourglass = False                                  ' Do not display wait cursor
```

AddMenu Method

Usage *ApplicationObject*.**AddMenu** (*MenuName*, *OCXName.OCX*, *ProgramName* [, *ChartType*])

The *MenuName* element is the title of the menu.

The OCX*Name.OCX* element identifies the custom control used to send notification events to when the menu is used. Normally you use ABC1, which registers menus for the **AppMenuSUBCLASS** event. (If you are using Visual Basic 3.0, or are working in a 16-bit environment, use ABC1.VBX for this element.)

The *ProgramName* element is the name of the program adding the menu to FlowCharter. The easiest way to identify the program is using Form1.Caption.

The *ChartType* element, which is optional, lets you specify a chart type for the menu. A chart type is a hidden string field up to eight characters in length indicating the chart type. This field is never used within FlowCharter, but it is useful within a FlowCharter events OCX. For example, if two OLE Automation programs are running, you could change the fourth element to avoid conflicts.

Note: This method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX.

Description The **AddMenu** method lets you add a menu to FlowCharter. The menu is added to FlowCharter at the left of the Window menu, so you set the order of the menus by the order in which you create them. When the program ends (when the OCX or VBX shuts down), the menu is removed from FlowCharter.

Data Type Object

Value The Menu object

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_AddMenu_Method')}

Adding Menus
Example

AppendItem Method
DeleteAll Method
DeleteItem Method
InsertItem Method
RemoveMenu Method

Text Property (Menu Collection)
Visible Property (Menu Collection)

Application Object

AddMenu Method, AppendItem Method Example

This example uses the **AddMenu** method of the Application object and the **AppendItem** method of the Menu collection to create a menu and add two items to it.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)        ' Add a new menu item

Menu.AppendItem ("First Item")                       ' Append items to the new menu
Menu.AppendItem ("Second Item")
```

RemoveMenu Method

Usage *ApplicationObject*.**RemoveMenu** *MenuName*
The *MenuName* element is the title of the menu.

Description The **RemoveMenu** method lets you remove a menu you have added. You cannot remove the menus that FlowCharter starts with.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_RemoveMenu_Method')}

Adding Menus
Example

AddMenu Method
AppendItem Method
DeleteAll Method
DeleteItem Method
InsertItem Method

Text Property (Menu Collection)
Visible Property (Menu Collection)

Application Object

RemoveMenu Method Example

This example uses the **RemoveMenu** method of the Application object to remove a menu you have added.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

Set Menu = ABC.AddMenu("New Menu", ABC1, Form1.Caption)          ' Add a new menu

ABC.MsgBox "Notice a menu titled 'New Menu' has been added to your menu-bar. Clicking OK will remove the menu."
ABC.RemoveMenu ("New Menu")                          ' Remove the newly added menu
```

Hint Method

Usage *ApplicationObject*.**Hint** *HintText*
The *HintText* element is the text of the message you are placing in the hint line.

Description The **Hint** method lets you set a temporary status bar message. It stays in the hint line until the cursor moves over another item in FlowCharter that causes the hint line to change. To set a permanent message in the hint line, use the **StatusBar** property.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_Hint_Method')}

Providing Feedback
Example

AddMenu Method
CreateAddOn Method
MsgBox Method
PercentGauge Method
StatusBar Property

Application Object

Hint Method Example

This example uses the **Hint** method of the Application object to display a hint message.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set NewShape = Chart.DrawShape("Operation")        ' Draw a shape
NewShape.Text = "Look at the hint line!"           ' Place text in the shape
NewShape.Shape.ShadowStyle = 3                     ' Give the shape a shadow
' Display a hint line
ABC.Hint "Poltergeists are the principal form of supernatural manifestation."
```

ChartTypeShutdown Method

Usage *ApplicationObject*.**ChartTypeShutdown** *ChartType*, *ApplicationName*
The *ChartType* element is the type of chart that you want to close.
The *ApplicationName* element is the name of the application.

Description The **ChartTypeShutdown** method lets you have an external program (EXE) alert OLE Automation that it is shutting down for some reason, usually because a RegisterEvent call failed. You usually call the **ChartTypeShutdown** method during the loading of an AddOn menu if a RegisterEvent call failed. You can also call it from the Form.QueryUnload event indicating that the external program with SUBCLASSing behavior is shutting down. If you set either **TypeRequiresEXE** or **TypeUsesEXE** to True in a program, then you also must ensure that you close all charts of that **Type** when your program closes. You use the **ChartTypeShutdown** method of the Application object to close the charts.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ChartTypeShutdown_Method')}

Linking EXEs to Charts
Example

CreateAddOn Method
RegisterEvent Method
TypeRequiresEXE Property
TypeUsesEXE Property

Application Object

ChartTypeShutdown Method Example

This example uses the **ChartTypeShutdown** method of the Application object in two places. When the form loads, **ChartTypeShutdown** is activated if the Events being registered do not register (if an error occurs). Then the **ChartTypeShutdown** method completely unloads all events. When the form unloads, the **ChartTypeShutdown** method is used to completely unregister all events. For this to work, the user must have a form already created that contains this code.

The following code needs to be in the general declarations section of the Form.

```
Const APPNAME = "Sample Application"                ' Sets the APPNAME constant
Const CHARTTYPE = "TEST"                            ' Sets the CHARTTYPE constant
Dim ABC As Object
```

The following code needs to be in the [Form_Load] routine.

```
Form1.Caption = "Sample Application"                ' Set title bar of sample application

Set ABC = CreateObject("ABCFlow.application")       ' Start ABC
ABC.Visible = True                                  ' Make ABC visible

ABC.Caption = "Sample application is now running."  ' Change ABC title bar

ItFailed = False                                    ' Set ItFailed variable to False

' Register Events with ABC FlowCharter 4.0
If (Not ABC.RegisterEvent(ABC1, APPNAME, "ChartActivateNOTIFY")) Then ItFailed = True
If (Not ABC.RegisterEvent(ABC1, APPNAME, "ObjectMovedNOTIFY", CHARTTYPE)) Then ItFailed = True
If (Not ABC.RegisterEvent(ABC1, APPNAME, "ObjectMoveSUBCLASS", CHARTTYPE)) Then ItFailed =
True

' Check to see if ItFailed was set to True after registering the Events above
If (ItFailed) Then
   ABC.ChartTypeShutdown CHARTTYPE, APPNAME      ' Unregister all registered Events with ABC
FlowCharter 4.0
   ABC.MsgBox "Could not register events. Closing application."
   ABC.Caption = ""                              ' Change title bar to default
   End                                           ' Close application
End If
```

The following code needs to in the [Form_QueryUnload] routine.

```
ABC.ChartTypeShutdown CHARTTYPE, APPNAME        ' Unregister all registered Events with ABC
FlowCharter 4.0

ABC.MsgBox "Unregistering events. Closing application."
ABC.Caption = ""                                ' Change ABC title bar
End                                             ' Close application
```

Properties and Methods by Task

Click the task to display the associated properies and methods:

Tasks: Setting up the Application

Tasks: Working with Chart Files

Tasks: Working with Objects

Tasks: Working with Shapes

Tasks: Working with Lines

Tasks: Working with Text

Tasks: Working with Data Fields

Tasks: Using Color

Tasks: Customizing Menus

Tasks: Special Programming Features

Tasks: Setting up the Application

Task **Properties/Methods**

Displaying the FlowCharter window

Visible, Activate, ArrangeIcons, CascadeCharts, TileCharts, Maximize, Minimize

Resizing and positioning the window

Bottom, Height, Left, Right, Top, Width, Restore

Changing and reading the title bar

Caption

Changing and reading the status bar

StatusBar, Hint, StatusBarVisible Property

Displaying the field viewer, notes viewer, and shape palette

FieldViewerVisible, NoteViewerVisible, ShapePaletteVisible

Getting FlowCharter system information

FullName, Name, Path, Version, OperatingSystem

Customizing preferences

AlignToRulers, ChannelAlignment, LineSpacingX, LineSpacingY, ShowRulers, SSSHorizontal, SSSVertical, SmartShapeSpacing, TouchAlignment, Preferences, LinkIndicator, LinkShadow, NoteIndicator, NoteShadow, ShowNodesOnLines, NumberFont, SetDefaults

Displaying help

Help

Closing FlowCharter

Quit

Choosing the target printer

Printer

Customizing the menu

AddMenu, RemoveMenu

Customizing the toolbox

CreateAddOn, RemoveAddOn

Setting the default path

DefaultFilePath

Tasks: Working with Chart Files

Task **Properties/Methods**

Creating new charts

New, NewFromTemplate, Add, AddFromTemplate

Opening charts

Open, DefaultFilePath

Saving charts

HasDiskFile, ReadOnly, Saved, RevertToSaved, Save, DefaultFilePath, Export Method

Closing charts

CloseChart, CloseAll

Activating a chart

Item, Count, ActiveChart, Name, Activate, Charts, FullName

Protecting charts

Protected, SetProtection

Linking charts

LinkedChartName, IsLinked, LinkFields, Link, LinkIndicator, LinkShadow

Launching applications

LaunchCommand, IsLaunched

Printing charts

PrintOut, PrintSelected, Printer, PrintBlankPages, PrintPreview Method

Redrawing a chart

NoRepaint, Repaint

Adjusting the page layout

Height, MarginBottom, MarginLeft, MarginRight, MarginTop, Orientation, PageHeight, PageLayout, PageOrder, PageWidth, PaperSize, PrintBlankPages, Width

Displaying Master Items

UpdateDateAndTime, ChartName, Date, DateStyle, Logo, LogoPathname, MasterItems, PageNumber, Range, Text1, Text2, Time, PageNumberShown, Text1Shown, Text2Shown, TimeShown, LogoShown, ChartNameShown, DateShown, HideAll, ShowAll

Viewing a chart

PageCount, ScrollLeft, ScrollTop, ScrollPage, ScrollPosition, ZoomPercentage, FullScreen, CancelFullScreen, ZoomWindowVisible Property

Using guidelines

GuidelinesOn, AddHorizontalGuideline, AddVerticalGuideline, ClearGuidelines

Sizing a chart window

Maximize, Minimize, Restore, View

Changing the view magnification

View, ZoomPercentage

Setting units of measure

Units

Doing a presentation

FullScreen, CancelFullScreen

Managing objects within a chart

Align Method, ImportShape Method, MakeSameSize Method, SpaceEvenly Method

Tasks: Working with Objects

Task **Properties/Methods**

Getting an object in a chart

Count, Item, ItemFromAll, ItemFromAttachments, ItemFromLines, ItemFromFieldValue, ItemFromNumber, ItemFromShapes, ItemFromSelection, ItemFromText, ItemFromUniqueID, ResetSearch, Valid

Selecting objects in a chart

Selected, Select, SelectShapeType

Deselecting objects in a chart

DeselectAll

Finding the number of selected objects

SelectedLineCount, SelectedObjectCount, SelectedOtherCount, SelectedShapeCount

Identifying an object

Type, UniqueID, ShapeName

Cutting, copying, and pasting objects

Copy, Cut, Paste, PasteLink, Duplicate, Clear_, PasteSpecial, ClipboardFormatAvailable

Executing an embedded or linked object

OLE, ObjectType, DoVerb

Inserting objects from a file

InsertObjectFromFile

Moving objects

Bottom, CenterX, CenterY, Left, Top, Right, FlippedHorizontal Property, FlippedVertical Property, Rotation Property

Resizing objects

Height, StretchType, Width

Restoring a bitmap

RestorePicture

Changing the display order of objects

ToBack, ToFront

Setting the current drawing position

DrawDirection, DrawPositionX, DrawPositionY

Redrawing an object

Repaint

Undoing a change

Undo, UndoAvailable

Formatting objects

ApplyDefaults Method

Tasks: Working with Shapes

Task **Properties/Methods**

Using the shape palette

CurrentShapePalette, ShapePaletteVisible, CurrentShape

Drawing shapes

DrawShape, DrawDirection, DrawPositionX, DrawPositionY, DrawSpacingX, DrawSpacingY, ShapeName, CurrentShape, SetDefaults

Connecting shapes with lines

DrawLine, Source, Destination, ReconnectSource, ReconnectDest, CurrentLineRouting

Moving shapes

Bottom, Left, Right, Top

Selecting shapes

SelectedShapeCount, DeselectAll, ItemFromShapes, Selected, SelectShapeType

Formatting shapes

Color, BorderColor, BorderStyle, BorderWidth, FillColor, FillPattern, ShadowColor, ShadowOffset, ShadowStyle, Shape, SetDefaults

Numbering shapes

NextNumber, NumberFont, Number, NumberShown, Renumber

Replacing shapes

ReplaceShape

Adding notes to shape

NoteIndicator, NoteShadow, NoteFont, NoteText, NoteTextLF

Adding text to shapes

TextAlignment, FitShapeToText, Text, TextLF, Font

Resizing shapes

Height, Width, StretchType

Tasks: Working with Lines

Task **Properties/Methods**

Drawing lines

DrawFreeLine, DrawLineToOneObject, DrawLine

Connecting existing lines to shape

Source, Destination, ReconnectSource, ReconnectDest, Line_

Setting line routing

CurrentLineRouting, Type

Formatting lines

Color, SourceArrowColor, StemColor, DestArrowColor, SourceArrowSize, StemWidth, DestArrowSize, StemStyle, SourceArrowStyle, DestArrowStyle, Line_, LineCrossoverStyle, LineCrossoverSize, SetDefaults, r

Displaying nodes on connecting lines

ShowNodesOnLines, LineCrossoverStyle, LineCrossoverSize, CrossoverSize Property, CrossoverStyle Property

Attaching text to lines

Line_, AttachText, AttachedToLine, UnattachFromLine, TextBlock

Deleting lines

DeleteLines, Clear_

Tasks: Working with Text

Task	Properties/Methods
Creating text blocks	DrawTextBlock, DrawPositionX, DrawPositionY
Adding text to a shape	Text, TextLF, FitShapeToText
Adding notes to a shape	NoteText, NoteTextLF, NoteFont, NoteIndicator, NoteShadow
Adding text to a line	DrawTextBlock, AttachText, AttachedToLine, UnattachFromLine
Formatting text	Size, Name, Bold, Italic, Underline, Strikethrough, Color, Opaque, TextAlignment, SetDefaults
Spell checking	Spelling
Replacing text	ReplaceText Method,

Tasks: Working with Data Fields

Task **Properties/Methods**

Adding data fields to a chart

Add, Name, Format, AccumulationMethod, Hidden, Type, FieldTemplates

Deleting data fields from a chart

DeleteField

Setting data field preference

FieldNamesHidden, FieldPlacement, FieldsOpaque, FieldsHoursPerDay, FieldsDaysPerWeek, FieldFont

Working with data field values

Value, Item, Accumulation, ItemFromFieldValue, Empty, FieldViewerVisible, FieldValues, Count, IsEmpty, FormattedValue

Viewing the legend

ShowLegend, Accumulation

Using linked field data

LinkFields, UpdateFields, LinkIndicator, LinkShadow, IsLinked, LinkedChartName, Link

Getting a data field

Count, Item, FieldTemplates

Tasks: Using Color

Task	Properties/Methods
Color representation	BasicColor, MakeRGB
Setting shape colors	FillColor, Color, BorderColor, ShadowColor, SetDefaults
Setting line colors	Color, SourceArrowColor, DestArrowColor, StemColor, SetDefaults
Setting text color	Color, SetDefaults

Tasks: Customizing Menus

Task	Properties/Methods
Adding a new menu	AddMenu
Getting a menu item	Count, Item
Adding a menu item	AppendItem
Deleting a menu item	DeleteItem, DeleteAll, Visible
Hiding a menu item	Visible
Changing menu text	Text
Disabling a menu item	Enabled
Checking a menu item	Checked

Tasks: Special Programming Features

Task	Properties/Methods
Displaying a wait hourglass	Hourglass
Displaying a percent gauge	PercentGaugeValue, PercentGauge, PercentGaugeCancelled, HidePercentGauge
Displaying a message	MsgBox
Using FlowCharter Events	RegisterEvent, UnRegisterEvent, ChartTypeShutdown
Sending mail	SendMail

Chart Object

Description The Chart object is below the Chart collection. You can have multiple Chart objects. Each Chart object is restricted to a single PageLayout and MasterItems object, but can have multiple FieldTemplate and Object objects.

Properties	Methods
Application	Activate
ClipboardFormatAvailable	AddHorizontalGuideline
CurrentLineRouting	AddVerticalGuideline
CurrentShape	Align
CurrentShapePalette	CancelFullScreen
DrawDirection	Clear_
DrawPositionX	ClearGuidelines
DrawPositionY	CloseChart
DrawSpacingX	Copy
DrawSpacingY	Cut
FieldFont	DeselectAll
FieldNamesHidden	DrawFreeLine
FieldPlacement	DrawLine
FieldsDaysPerWeek	DrawLineToOneObject
FieldsHoursPerDay	DrawShape
FieldsOpaque	DrawTextBlock
FieldTemplates	Duplicate
FullName	Export
GuidelinesOn	FullScreen
HasDiskFile	GroupAndLink
LineCrossoverSize	ImportShape
LineCrossoverStyle	InsertObjectFromFile
LinkIndicator	MakeSameSize
LinkShadow	Minimize
MasterItems	Maximize
Name	Paste
NextNumber	PasteLink
NextShapeHeight	PasteSpecial
NextShapeWidth	PrintOut
NoRepaint	PrintPreview
NoteIndicator	PrintSelected
NoteShadow	Repaint
NumberFont	ReplaceText
Objects	Restore
PageCount	RevertToSaved
PageLayout	Save
Parent	ScrollPage
Protected	ScrollPosition
ReadOnly	Select
Saved	SelectShapeType
ScrollLeft	SendMail
ScrollTop	SetDefaults

SelectedLineCount
SelectedObjectCount
SelectedOtherCount
SelectedShapeCount
ShowLegend
ShowNodesOnLines
Type
TypeRequiresEXE
TypeUsesEXE
Units
Valid
View
WindowHandle
ZoomPercentage
SetProtection
SpaceEvenly
Spelling
ToBack
ToFront
UpdateFields

{button Related Topics,PI(``,`IDH_RT_Chart_Object')}

ClipboardFormatAvailable Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_PASTESPECIALDB');CW(`concfull')}

Usage *ChartObject*.**ClipboardFormatAvailable** (*Format*)

Description The **ClipboardFormatAvailable** property lets you find if a format is available to be pasted from the Windows Clipboard. The **ClipboardFormatAvailable** property is read only.

Data Type Integer (Boolean)

Value True means the format is available; False means the format is not available.

The values for the formats are in the following table.

Format	**Description**
0	ABC Native
1	OLE Client Embed
2	ABC Rich Text
3	Rich Text Format (RTF)
4	Unformatted Text
5	Metafile
6	Device-Independent Bitmap
7	Bitmap
8	OLE Client Link

Flow Equivalent The **ClipboardFormatAvailable** property is equivalent to clicking Paste Special on the Edit menu and checking if a format is available.

{button Related Topics,PI(`',`IDH_RT_ClipboardFormatAvailable_Property')}

CurrentLineRouting Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage *ChartObject*.**CurrentLineRouting** = *LineRoutingValue*

Description You use the **CurrentLineRouting** property to find or set the type of routing for new lines. The **CurrentLineRouting** property is read/write.

Data Type Integer

Value The following table describes the values for the **CurrentLineRouting** property.

Value	Type of Line
0	Direct
1	Right angle
2	Curved
3	Organization chart
4	Cause-and-effect

Flow Equivalent The **CurrentLineRouting** property is equivalent to clicking the Direct Line, Right Angle Line, Curved Line Org Chart Line, or Cause and Effect Line tool in the Toolbox with no lines selected. You cannot change the type of routing for lines that have already been drawn.

{button Related Topics,PI(`',`IDH_RT_CurrentLineRouting_Property')}

Setting Line Routing
Example

Type Property (Line_Object)

Chart Object

CurrentShape Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Shape_Palette_Overview');CW(`concfull')}

Usage *ChartObject*.**CurrentShape** = *Name*

Description The **CurrentShape** property lets you find or set the current shape so that it is the next shape drawn when you draw a shape. When you are setting the value, you can define it loosely. For example, setting its value to "dec" chooses "Decision." The **CurrentShape** property is read/write.

Data Type String

Value The name of the next shape to be drawn

Flow Equivalent The **CurrentShape** property is equivalent to clicking the shape you want in the Shape palette.

{button Related Topics,PI(`',`IDH_RT_CurrentShape_Property')}

Choosing a Shape in the Palette
Example

CurrentShape Property
DrawShape Method

Chart Object

CurrentShape Property Example

This example uses the **CurrentShape** property and **DrawShape** method of the Chart object to set the type of shape to be created.

```
Dim ABC As Object, Chart As Object, Obj1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.CurrentShape = "Decision"                    ' Set shape to Decision diamond
Set Obj1 = Chart.DrawShape()                       ' Draw Decision shape
```

CurrentShapePalette Property {button Flow Equivalent,JI(`FLOW.HLP',`IDH_Shape_Palettes_Command');CW(`concfull')}

Usage *ChartObject*.**CurrentShapePalette** = *ShapePaletteName*

Description You use the CurrentShapePalette property to open a Shape Palette or determine the name of the current Shape Palette. The name of the Shape Palette appears in the title bar of the palette and is not related to the filename of the palette. The CurrentShapePalette property is read/write.

Data Type String

Value The name of the current shape palette

Flow Equivalent The **CurrentShapePalette** property is equivalent to clicking Shape Palettes on the View menu.

{button Related Topics,PI(``,`IDH_RT_CurrentShapePalette_Property')}

Using the Shape Palette
Example

CurrentShape Property
DrawShape Method

Chart Object

CurrentShapePalette Property Example

This example uses the **CurrentShapePalette** property of the Chart object to set the current shape palette.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.CurrentShapePalette = "Net - PC"             ' Change current shape palette
```

DrawDirection Property

Usage *ChartObject*.**DrawDirection** = *Direction*

Description The **DrawDirection** property lets you find or set the direction for placing new shapes. The **DrawDirection** property is read/write.

Data Type Integer

Value The **DrawDirection** property uses the values shown in the following table.

Value	Description
0	North
1	East
2	South
3	West
10	Stacked

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DrawDirection_Property')}

DrawPositionX Property

Usage *ChartObject*.**DrawPositionX** = *HorizontalDistance*

Description The **DrawPositionX** property lets you find or set the horizontal drawing position where you want to place the next object, text, or line. The position you specify is used for the next object drawn, or the next object pasted or pasted special (if those methods do not specify a different position). You set the units used to measure the distance using the **Units** property. The **DrawPositionX** property is read/write.

Data Type Double

Value The horizontal location for the next drawing position

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DrawPositionX_Property')}

Setting the Current Drawing Position
Creating Text Blocks
Example

DrawPositionY Property
DrawSpacingX Property
DrawSpacingY Property
Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

DrawPositionX Property, DrawPositionY Property, DrawSpacingX Property, and DrawSpacingY Property Example

This example uses the **DrawPositionX** property, **DrawPositionY** property, **DrawSpacingX** property, and **DrawSpacingY** property of the Chart object to set the position and spacing for drawing shapes.

```
Dim ABC As Object, Chart As Object
Dim Shapes

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.DrawPositionX = 1                            ' Set X coordinate for drawing
Chart.DrawPositionY = 2                            ' Set Y coordinate for drawing
Chart.DrawSpacingX = 1.5                           ' Set X coordinate spacing
Chart.DrawSpacingY = 1.5                           ' Set Y coordinate spacing

For Shapes = 1 To 4                                ' Draw shapes
   Chart.DrawShape ("Storage")
Next Shapes
```

DrawPositionY Property

Usage *ChartObject*.**DrawPositionY** = *VerticalDistance*

Description The **DrawPositionY** property lets you find or set the vertical drawing position where you want to place the next object, text, or line. The position you specify is used for the next object drawn, or the next object pasted or pasted special (if those methods do not specify a different position). You set the units used to measure the distance using the **Units** property. The **DrawPositionY** property is read/write.

Data Type Double

Value The vertical location for the next drawing position

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DrawPositionY_Property')}

Setting the Current Drawing Position
Creating Text Blocks
Example

DrawPositionX Property
DrawSpacingX Property
DrawSpacingY Property
Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

DrawSpacingX Property

Usage *ChartObject*.**DrawSpacingX** = *Spacing*

Description The **DrawSpacingX** property lets you find or set the horizontal spacing for the next shape placed. The **DrawSpacingX** property is read/write.

Data Type Double

Value The horizontal spacing for the next shape placed

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DrawSpacingX_Property')}

Drawing Shapes
Example

DrawPositionX Property
DrawPositionY Property
DrawSpacingY Property
Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

DrawSpacingY Property

Usage *ChartObject*.**DrawSpacingY** = *Spacing*

Description The **DrawSpacingY** property lets you find or set the vertical spacing for the next shape placed. The **DrawSpacingY** property is read/write.

Data Type Double

Value The vertical spacing for the next shape placed

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DrawSpacingY_Property')}

Drawing Shapes
Example 1
Example 2

DrawPositionX Property
DrawPositionY Property
DrawSpacingX Property
Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

FieldFont Property {button Flow Equivalent,JI(`FLOW.HLP>procedur','IDH_Formatting_data_fields');CW(`concfull')}

Usage *ChartObject*.**FieldFont**

Description The **FieldFont** property lets you find or set properties for the Font object for field text in a chart. All the properties of the Font object, such as bold and italic, are available through the **FieldFont** property. The **FieldFont** property is read only, but all the properties from the object it returns are read/write.

Data Type Object

Value A Font object

Flow Equivalent The **FieldFont** property is equivalent to clicking the Data Fields Options button on the Data toolbar and setting the font attributes in the field font area.

{button Related Topics,PI(`','IDH_RT_FieldFont_Property')}

FieldFont Property Example

This example uses the **FieldFont** property of the Chart object to set the style of the font used for data fields.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Field1 As Object, Field2 As Object, FieldFontStyle As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                           ' Position fields below shapes
Set Field1 = Chart.FieldTemplates.Add("Name")      ' Add a field
Field1.Format = 0                                  ' Format field as text
Field1.AccumulationMethod = 0                      ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")     ' Add a field
Field2.Format = 0                                  ' Format field as text
Field2.AccumulationMethod = 0                      ' No accumulation

Set FieldFontStyle = Chart.FieldFont               ' Set the FieldFont object
FieldFontStyle.Name = "Roman"                      ' Change the font
FieldFontStyle.Italic = True                       ' Make it italic

Obj1.FieldValues.Item("Name").Value = "Joe Smith"  ' Enter field values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"
```

FieldNamesHidden Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *ChartObject*.**FieldNamesHidden** = {True | False}

Description The **FieldNamesHidden** property lets you find or set whether field names are shown. The **FieldNamesHidden** property is read/write.

Data Type Integer (Boolean)

Value True hides field names; False shows them.

Flow Equivalent The **FieldNamesHidden** property is equivalent to clicking the Data Fields Options button on the Data toolbar and selecting the Hide Field Names option (True) or clearing the option (False).

{button Related Topics,PI(``,`IDH_RT_FieldNamesHidden_Property')}

FieldNamesHidden Property Example

This example uses the **FieldNamesHidden** property of the Chart object to hide data field names.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Field1 As Object, Field2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                           ' Position fields below shapes
Set Field1 = Chart.FieldTemplates.Add("Name")      ' Add a field
Field1.Format = 0                                  ' Format field as text
Field1.AccumulationMethod = 0                      ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")     ' Add a field
Field2.Format = 0                                  ' Format field as text
Field2.AccumulationMethod = 0                      ' No accumulation

Obj1.FieldValues.Item("Name").Value = "Joe Smith"  ' Enter field values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"

Chart.FieldNamesHidden = True                      ' Hide the field labels
```

FieldPlacement Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *ChartObject*.**FieldPlacement** = *Value*

Description The **FieldPlacement** property lets you specify the field placement in relation to shapes. The **FieldPlacement** property is read/write.

Data Type Integer

Value The values for the field placements are in the following table.

Value	Description
0	Left
1	Right
2	Above
3	Below
4	Inside Top
5	Inside Middle

Flow Equivalent The **FieldPlacement** property is equivalent to clicking the Data Fields Options button on the Data toolbar and clicking a placement location from the Field Placement box's drop-down selections.

{button Related Topics,PI(`',`IDH_RT_FieldPlacement_Property')}

FieldPlacement Property Example

This example uses the **FieldPlacement** property of the Chart object to put data fields below shapes.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Field1 As Object, Field2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                           ' Position fields below shapes

Set Field1 = Chart.FieldTemplates.Add("Name")      ' Add a field
Field1.Format = 0                                  ' Format field as text
Field1.AccumulationMethod = 0                      ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")     ' Add a field
Field2.Format = 0                                  ' Format field as text
Field2.AccumulationMethod = 0                      ' No accumulation

Obj1.FieldValues.Item("Name").Value = "Joe Smith"  ' Enter field values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"
```

FieldTemplates Property

Usage *ChartObject*.**FieldTemplates**

Description You use the **FieldTemplates** property to find the FieldTemplates collection. The **FieldTemplates** property is read only, but the properties from the collection it returns are read/write.

Data Type Collection object

Value The FieldTemplates collection

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_FieldTemplates_Property')}

FieldsHoursPerDay Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *ChartObject*.**FieldsHoursPerDay** = *HoursPerDay*

Description The **FieldsHoursPerDay** property lets you find or set the number of hours in a workday. This value is used when a field is converted between hours and days. The value can range from 1 to 24. For example, the value is used if you change the data field's format from hours to days or you link to a chart that displays data fields in a different format. The **FieldsHoursPerDay** property is read/write.

Data Type Integer

Value The number of hours in a day for fields

Flow Equivalent The **FieldsHoursPerDay** property is equivalent to clicking the Data Fields Option button on the Data toolbar and entering a number in the Hours Per Day box.

{button Related Topics,PI(`',`IDH_RT_FieldsHoursPerDay_Property')}

FieldsHoursPerDay, FieldsDaysPerWeek Properties Example

This example uses the **FieldsHoursPerDay** property and **FieldsDaysPerWeek** property of the Chart object to find and change the hours per day and days per week.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object
Dim Shape1 As Object, Shape2 As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Add a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Set Text1 = Chart.DrawTextBlock("Current Field Options")       ' Place a text block

Chart.DrawPositionX = 1                              ' Set horizontal position
Chart.DrawPositionY = 2.5                            ' Set vertical position
Set Shape1 = Chart.DrawShape("Operation")            ' Place shapes on the chart
Set Shape2 = Chart.DrawShape("Operation")
Shape1.Text = "Hours per day = " + Chart.FieldsHoursPerDay ' Display hours per day
Shape2.Text = "Days per week = " + Chart.FieldsDaysPerWeek         ' Display days per week

ABC.MsgBox "Click OK to change the number of " + Chr$(13) + "hours per day and the days per
week."                                               ' Chr$(13) is Carriage Return
Chart.FieldsHoursPerDay = 24                         ' Change hours per day
Chart.FieldsDaysPerWeek = 7                          ' Change days per week
Shape1.Text = "Hours per day = " + Chart.FieldsHoursPerDay ' Display hours per day
Shape2.Text = "Days per week = " + Chart.FieldsDaysPerWeek         ' Display days per week
```

FieldsDaysPerWeek Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *ChartObject*.**FieldsDaysPerWeek** = *DaysPerWeek*

Description The **FieldsDaysPerWeek** property lets you find or set the number of days in a workweek. This value is used when a field is converted between days and weeks. The value can range from 1 to 7. For example, the value is used if you change the data field's format from days to weeks or you link to a chart that displays data fields in a different format. The **FieldsDaysPerWeek** property is read/write.

Data Type Integer

Value The number of days in a week for fields

Flow Equivalent The **FieldsDaysPerWeek** property is equivalent to clicking the Data Fields Options button on the Data toolbar and entering a number in the Days Per Week box.

{button Related Topics,PI(`',`IDH_RT_FieldsDaysPerWeek_Property')}

FieldsOpaque Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *ChartObject*.**FieldsOpaque** = {True | False}

Description The **FieldsOpaque** property lets you find or set whether fields are opaque. The **FieldsOpaque** property is read/write.

Data Type Integer (Boolean)

Value True makes the background opaque; False makes the background transparent.

Flow Equivalent The **FieldsOpaque** property is equivalent to clicking the Data Fields Options button on the Data toolbar and selecting or clearing the Opaque Fields option.

{button Related Topics,PI(``,`IDH_RT_FieldsOpaque_Property')}

FieldsOpaque Property Example

This example uses the **FieldsOpaque** property of the Chart object to make data fields opaque.

```
'This example uses the FieldsOpaque property of the Chart object
'to make data fields opaque.

Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object, Obj3 As Object, Obj4 As Object
Dim Field1 As Object, Field2 As Object, A As Integer

Set ABC = CreateObject("ABCFlow.application")       ' Start ABC
ABC.Visible = True                                        ' Make ABC visible
ABC.New                                                   'Get a new chart
Set Chart = ABC.ActiveChart                               ' Get the active chart
Chart.FieldPlacement = 5                            ' Position fields inside shapes

Set Field1 = Chart.FieldTemplates.Add("Name")       ' Add field
Field1.Format = 0                                         ' Format field as text
Field1.AccumulationMethod = 0                             ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")      ' Add field
Field2.Format = 0                                         ' Format field as text
Field2.AccumulationMethod = 0                             ' No accumulation

Chart.DrawSpacingX = 4                                    ' Set draw spacing
Chart.DrawSpacingY = 2
Set Obj3 = Chart.DrawShape("Operation")                   ' Draw shapes
Set Obj4 = Chart.DrawShape("Operation")
Obj3.Shape.FillColor = ABC.BLUE                           ' Color the shapes
Obj4.Shape.FillColor = ABC.RED
Obj3.FieldValues.Item("Name").Value = "Joe Smith"   ' Set Field Values
Obj3.FieldValues.Item("Phone").Value = "555-1212"

Obj4.FieldValues.Item("Name").Value = "Jane Doe"
Obj4.FieldValues.Item("Phone").Value = "555-1234"
Chart.SelectShapeType ("Operation")                       ' Select shapes
Chart.ToBack                                              ' Move to back
ABC.MsgBox "Changing field type to Opaque."

Chart.FieldsOpaque = True                                 ' Make field text opaque
```

FullName Property (Chart Object)

Usage *ChartObject*.**FullName**

Description You can identify a chart's filename with or without its pathname. The **FullName** property of the Chart object returns the fully qualified pathname of the chart. (If the chart has not been saved, it returns the temporary name of the chart.) The **FullName** property is read only.

Data Type String

Value The fully qualified pathname of the chart

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_FullName_Property_Chart_Object')}

Identifying a Chart's Filename
Example

FullName Property (Application Object)
Name Property (Chart Object)

Chart Object

FullName Property (Chart Object) Example

This example uses the **FullName** property of the Chart object to display the name and path of a chart.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Chart.Save ("tst_chrt")                              ' Save the chart
Set Chart = ABC.ActiveChart                          ' Reset as active chart after save
ABC.MsgBox "Path name for this chart is " + Chart.FullName    ' Display full path of file
```

GuidelinesOn Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_GUIDELINESP');CW(`concfull')}

Usage *ChartObject*.**GuidelinesOn** = {True | False}

Description You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let you align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **GuidelinesOn** property to turn showing guidelines on and off. The **GuidelinesOn** property is read/write.

Data Type Integer (Boolean)

Value True shows guidelines; False does not.

Flow Equivalent The **GuidelinesOn** property is equivalent to clicking Guidelines on the View menu. Guidelines in the chart are displayed when the menu item is selected.

{button Related Topics,PI(``,`IDH_RT_GuidelinesOn_Property')}

Identifying a Chart's Filename
Example

AddHorizontalGuideline Method
AddVerticalGuideline Method
ClearGuidelines Method

Chart Object

HasDiskFile Property

Usage *ChartObject*.**HasDiskFile**

Description You use the **HasDiskFile** property to find if the chart has ever been saved to disk. The **HasDiskFile** property is read only.

Data Type Integer (Boolean)

Value True means the chart has been saved to disk; False means the chart is a new chart that has never been saved to disk.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_HasDiskFile_Property')}

Saving Charts
Example

FullName Property (Chart Object)
Save Method
Saved Property

Chart Object

HasDiskFile Property, Save Method Example

This example uses the **HasDiskFile** property and the **Save** method of the Chart object to check if a file has ever been saved, and save it if it has not.

```
Private Sub Command1_Click()
Dim ABC As Object, Chart As Object, Obj1 As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
Set Chart = ABC.ActiveChart                          ' Get the active chart
ABC.New
Set Obj1 = Chart.DrawShape("Operation")              ' Draw shape
Obj1.TEXT = "Unit 1"                                 ' Add text to shape

If Not Chart.HasDiskFile Then                        ' Has this file been saved?
                                                     ' If not, save the file
    Chart.Save (InputBox$("Enter the file name", "Save File",".flo"))
                                                     ' If saved, display reminder
    Else ABC.MsgBox "Save your work often!", 48, "Don't Forget."
End If
End Sub
```

LinkIndicator Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Indicators');CW(`concfull')}

Usage *ChartObject*.**LinkIndicator** = *Indicator*

Description The link indicator (up to three characters) appears on shapes with attached links. You use the **LinkIndicator** property to find or set the indicator used for linked shapes. The **LinkIndicator** property is read/write.

Data Type String

Value Text, up to three characters, that indicates that a shape in FlowCharter is linked

Flow Equivalent The **LinkIndicator** property is equivalent to clicking Chart Properties on the Format menu, clicking the Indicators tab, and entering text in the Link Indicator box.

{button Related Topics,PI(`',`IDH_RT_LinkIndicator_Property')}

LinkIndicator Property Example

This example uses the **LinkIndicator** property of the Chart object to set the link indicator for a chart.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Link1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.CloseAll                                       ' Close all charts
ABC.New                                            ' Add a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.LinkIndicator = "*L*"                        ' Set the link indicator to *L*
Set Shape1 = Chart.DrawShape("Decision")           ' Draw a shape
Set Link1 = Shape1.Shape.Link                      ' Link the shape to a new chart

ABC.MsgBox "Using the Window menu, switch to [CHART1] and notice that the shape is marked with '*L*'."
```

LinkShadow Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Indicators');CW(`concfull')}

Usage *ChartObject*.**LinkShadow** = {True | False}

Description The **LinkShadow** property lets you find or set whether shapes that have linked files show a shadow. The **LinkShadow** property is read/write.

Data Type Integer (Boolean)

Value True shows a shadow on shapes that have a linked file; False does not.

Flow Equivalent The **LinkShadow** property is equivalent to clicking Chart Properties on the Format menu, clicking the Indicators tab, and selecting or clearing the Link Shadow option.

{button Related Topics,PI(``,`IDH_RT_LinkShadow_Property')}

MasterItems Property

Usage *ChartObject*.**MasterItems**

Description The **MasterItems** property lets you find the MasterItems objects. The **MasterItems** property is read only, but all the properties from the object it returns are read/write.

Data Type Object

Value The MasterItems objects

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_MasterItems_Property')}

Displaying Master Items
Example

ChartNameShown Property

Chart Object

Name Property (Chart Object)

Usage	*ChartObject*.**Name**
Description	You use the **Name** property to return the name of the Chart object without the path. The **Name** property is read only.
Data Type	String
Value	The name of the Chart object without the path
Flow Equivalent	None

{button Related Topics,PI(``,`IDH_RT_Name_Property_Chart_Object')}

Identifying a Chart's Filename
Example

Activate Method (Chart Object)
Item Method (Charts Collection)

ActiveChart Property
Application Property
Count Property
Name Property (Application Object)
Name Property (FieldTemplate Object)
Name Property (FieldValue Object)
Name Property (Font Object)

Chart Object

Name Property (Chart Object) Example

This example uses the **Name** property of the Chart object to display the name of a chart.

```
Sub Command1_Click ()
    Dim ABC As Object, Chart As Object

    Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
    ABC.Visible = True                                     ' Make ABC visible
    ABC.New
    Set Chart = ABC.ActiveChart                            ' Get the active chart

    Chart.Save ("tst_chrt")                                ' Save the chart

    ABC.MsgBox "File name for this chart is " + Chart.Name' Display file name
End Sub
```

NextNumber Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_RENUMBER');CW(`concfull')}

Usage *ChartObject*.**NextNumber** = *NextChartNumber*

Description You use the **NextNumber** property to find or set the number for the next shape that is drawn. You can use various numbering systems, such as *1, 2, 3,* or *1.1, 1.2, 1.3,* or even text strings. The number is kept in the **NextNumber** property as a text string, because the number can contain text as well as numbers. The NextNumber property is incremented automatically each time you draw a shape. If NextNumber contains text with a number, the text remains and the number is incremented. For example; "Step 5" becomes "Step 6" when a new shape is drawn. If NextNumber contains only text, the text remains without incrementing.

The Number property of the Shape object contains the actual shape number for a particular shape. When you draw a shape, the value in the chart's NextNumber property is stored in the Number **property**, and the NextNumber property is incremented. You can change a shape's number by changing the value of the shape's Number property. The NextNumber property is read/write.

Data Type String

Value The number for the next shape drawn

Flow Equivalent The **NextNumber** property is equivalent to clicking the Set Next Shape Number command from the Shape Numbering submenu on the Format menu and entering the number for the next shape in the Set Next Number box.

{button Related Topics,PI(`',`IDH_RT_NextNumber_Property')}

NextNumber Property Example

This example uses the **NextNumber** property of the Chart object to set the number to be used by the next shape.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.NextNumber = "100"                           ' Set number for next shape drawn

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")
```

NoRepaint Property

Usage *ChartObject*.**NoRepaint** = {True | False}

Description The **NoRepaint** property lets you omit drawing each action. With the **NoRepaint** property set to True, you can have a 15% to 20% increase in speed. After the actions are complete, you update the screen using the **Repaint** method. Be sure to set the **NoRepaint** property to False when the program finishes drawing. The **NoRepaint** property is read/write.

Data Type Integer (Boolean)

Value True means to omit drawing each action; False means to draw each action.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_NoRepaint_Property')}

NoteIndicator Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Indicators');CW(`concfull')}

Usage *ChartObject*.**NoteIndicator** = *IndicatorText*

Description The **NoteIndicator** property lets you find or set the text, up to three characters, that indicates that a shape in FlowCharter has a note attached to it. The **NoteIndicator** property is read/write.

Data Type String

Value Text, up to three characters, that indicates that a shape in FlowCharter has a note attached to it

Flow Equivalent The **NoteIndicator** property is equivalent to clicking Chart Properties on the Format menu, clicking the Indicators tab, and entering text in the Note Symbol box.

{button Related Topics,PI(`',`IDH_RT_NoteIndicator_Property')}

NoteShadow Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Indicators');CW(`concfull')}

Usage *ChartObject*.**NoteShadow** = {True | False}

Description You use the NoteShadow property to find or set whether shapes that have notes have a shadow. The NoteShadow property is read/write.

Data Type Integer (Boolean)

Value True means shapes with notes have a shadow; False means they do not.

Flow Equivalent The **NoteShadow** property is equivalent to clicking Chart Properties on the Format menu, clicking the Indicators tab, and selecting the Shadow option (True) or clearing the option (False) for Note.

{button Related Topics,PI(``,`IDH_RT_NoteShadow_Property')}

NoteShadow, NoteIndicator Properties Example

This example uses the **NoteShadow** property and **NoteIndicator** property of the Chart object to add a shadow to shapes with a note and set the text indicator for shapes with a note.

```
Dim ABC As Object, Chart As Object, Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.NoteShadow = True                            ' Apply shadow to shapes with notes
Chart.NoteIndicator = "*N*"                        ' Change note indicator string

Set Shape1 = Chart.DrawShape("Operation")          ' Draw a shape
Shape1.Shape.NoteText = "Check with Production"    ' Set note text for shape
Chart.Select (0)                                   ' Select the shape
NoteViewerVisible = True                           ' Show the note viewer
```

NumberFont Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Indicators');CW(`concfull')}

Usage *ChartObject*.**NumberFont**

Description The **NumberFont** property lets you find or set properties for the Font object for shape numbers in a chart. All the properties of the Font object, such as bold and italic, are available through the **NumberFont** property. The **NumberFont** property is read only, but all the properties from the object it returns are read/write.

Data Type Object

Value A Font object

Flow Equivalent The **NumberFont** property does not have a FlowCharter equivalent. The style for numbers is determined by the style for link and note indicators.

{button Related Topics,PI(`',`IDH_RT_NumberFont_Property')}

NumberFont Property Example

This example uses the **NumberFont** property of the Chart object to set the attributes for the font used for numbers.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim ChartNumberFont As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set ChartNumberFont = Chart.NumberFont             ' Set number font object

ChartNumberFont.Name = "Roman"                     ' Set font attributes
ChartNumberFont.Bold = True
ChartNumberFont.Italic = True

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")
```

Objects Property

Usage *ChartObject*.**Objects**

Description The **Objects** property lets you find the objects included in the Objects collection. The **Objects** property is read only, but all the properties from the object it returns are read/write.

Data Type Collection object

Value The objects included in the Objects collection

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Objects_Property')}

PageCount Property

Usage *ChartObject*.**PageCount** = *Number*

Description You use the **PageCount** property to find the number of pages in the chart, including pages with no objects on them. The **PageCount** property is read only.

Data Type Integer

Value The number of pages in the chart

Flow Equivalent None

Adjusting the Page Layout
Example

ScrollLeft Property
ScrollPage Method
ScrollPosition Method
ScrollTop Property
View Property

Chart Object

PageCount Property Example

This example uses the **PageCount** property of the Chart object to show the number of used pages in a chart.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart
Chart.DrawSpacingX = 1.0                           'Set the spacing for the shapes
For X = 1 To 10                                    ' Draw shapes
   Set Shape = Chart.DrawShape("Decision")
Next X

Chart.View = 2                                     ' View used pages

ABC.MsgBox "Chart pages: " + Str$(Chart.PageCount) + "." ' Number of pages in the chart
```

PageLayout Property

Usage *ChartObject*.**PageLayout**

Description You use the **PageLayout** property to find the PageLayout object. The **PageLayout** property is read only, but the properties from the object it returns are read/write.

Data Type Collection object

Value The PageLayout object

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_PageLayout_Property')}

Protected Property

Usage *ChartObject*.**Protected**

Description You use the **Protected** property to find whether a chart is protected. The **Protected** property is read only.

Data Type Integer (Boolean)

Value True means that the chart is password protected; False means that the chart is not password protected.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ Protected_Property')}

Protecting Charts
Example

SetProtection Method

Chart Object

Protected Property Example

This example uses the **Protected** property of the Chart object to give a chart a password.

```
Dim ABC As Object, Chart As Object, Shape As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

If NOT Chart.Protected Then
   Chart.SetProtection 1,"gipper"                    ' Set password
End If
```

ReadOnly Property

Usage *ChartObject*.**ReadOnly**

Description You can determine whether a chart is read only by using the **ReadOnly** property of the Chart object. Read-only charts cannot be saved under the same filename. The **ReadOnly** property is read only.

Data Type Integer (Boolean)

Value True means that the chart is read only; False means that the chart is read/write.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ReadOnly_Property')}

ReadOnly Property Example

This example uses the **ReadOnly** property of the Chart object to determine if a chart was opened as read only.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart

Set Chart = ABC.Open(ABC.Path + "\Samples\Famltree.abc", 1)     ' Open file read only

If Chart.ReadOnly Then                                 ' Is chart read only?
   ABC.MsgBox "This file has the read-only attribute."
End If
```

Saved Property

Usage *ChartObject*.**Saved**

Description The **Saved** property determines if the Chart object in memory is the same as on disk. The **Saved** property is read only.

Data Type Integer (Boolean)

Value True means that the chart in memory is the same as the chart file on disk; False means that the chart in memory is not the same as the chart file on disk.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Saved_Property')}

ScrollLeft Property

Usage *ChartObject*.**ScrollLeft** = *Distance*

Description You use the **ScrollLeft** property to find or set the left point visible in the chart. The **ScrollLeft** property is read/write.

Data Type Double

Value The left point visible in the chart

Flow Equivalent The **ScrollLeft** property is equivalent to clicking the horizontal scroll bar.

{button Related Topics,PI(``,`IDH_RT_ScrollLeft_Property')}

Viewing a Chart
Example

PageCount Property
ScrollPage Method
ScrollPosition Method
ScrollTop Property
View Property

Chart Object

ScrollTop Property

Usage *ChartObject*.**ScrollTop** = *Distance*

Description You use the **ScrollTop** property to find or set the top point visible in the chart. The **ScrollTop** property is read/write.

Data Type Double

Value The top point visible in the chart

Flow Equivalent The **ScrollTop** property is equivalent to clicking the vertical scroll bar.

{button Related Topics,PI(``,`IDH_RT_ScrollTop_Property')}

Viewing a Chart
Example

PageCount Property
ScrollPage Method
ScrollPosition Method
ScrollLeft Property
View Property

Chart Object

SelectedLineCount Property

Usage *ChartObject*.**SelectedLineCount**

Description The **SelectedLineCount** property lets you find the number of lines in the Chart object. The **SelectedLineCount** property contains the number of selected lines, not the number of selected line segments, so the routing of the lines does not affect the count. The **SelectedLineCount** property is read only.

Data Type Integer

Value The number of lines in the Chart object

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_SelectedLineCount_Property')}

SelectedLineCount Property Example

This example uses the **SelectedLineCount** property of the Chart object to find the number of lines that are selected in a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Obj3 As Object, Line1 As Object, Line2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.DrawSpacingX = 1.5                           'Set spacing between shapes

Set Obj1 = Chart.DrawShape("Terminal")             ' Draw shapes
Set Obj2 = Chart.DrawShape("Operation")
Set Obj3 = Chart.DrawShape("Decision")

Set Line1 = Chart.DrawLine(Obj1, Obj2)             ' Draw lines
Set Line2 = Chart.DrawLine(Obj2, Obj3)
Chart.Select (1)                                   ' Select all lines

ABC.MsgBox "There are " + Str$(Chart.SelectedLineCount) + " line(s) selected in the chart."
```

SelectedObjectCount Property

Usage *ChartObject*.**SelectedObjectCount**

Description The **SelectedObjectCount** property lets you find the number of selected objects in the Chart object. It equals the sum of the values of the **SelectedShapeCount**, **SelectedLineCount**, and **SelectedOtherCount** properties. The **SelectedObjectCount** property is read only.

Data Type Integer

Value The number of selected objects in the Chart object

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_SelectedObjectCount_Property')}

SelectedObjectCount Property Example

This example uses the **SelectedObjectCount** property of the Chart object to display the number of selected objects in a chart.

```
Sub Command1_Click ()
   Dim ABC As Object, Chart As Object, Obj1 As Object
   Dim X As Integer

   Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
   ABC.Visible = True                                 ' Make ABC visible
   ABC.Activate                                       ' Bring ABC to the foregrounc
   Set Chart = ABC.New                                ' Get a new chart
   Chart.DrawSpacingX = 1.5                           ' Set draw spacing
                                                      ' between shapes

   Set Obj1 = Chart.DrawShape("FlowCharter Technical Palettes\Annotation\Question mark")   ' Draw
Operation shape
   Chart.Select (0)                                   ' Select the shape to duplicate it

   For X = 1 To 3                                     ' Duplicate shape three times
       Chart.Duplicate
   Next X

   Chart.Select (2)                                   ' Select all objects

   ABC.MsgBox "There are " + Str$(Chart.SelectedObjectCount) + " objects selected"
End Sub
```

SelectedOtherCount Property

Usage *ChartObject*.**SelectedOtherCount**

Description The **SelectedOtherCount** property lets you find the number of objects in the Chart object that are not shapes or lines. It includes master item objects such as the date and headers, OLE objects, bitmaps, and other objects pasted into FlowCharter. The **SelectedOtherCount** property is read only.

Data Type Integer

Value The number of objects in the Chart object that are not shapes or lines

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_SelectedOtherCount_Property')}

SelectedOtherCount Property Example

This example uses the **SelectedOtherCount** property of the Chart object to find the number objects other than shapes and lines that are selected in a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Obj3 As Object, Line1 As Object, Line2 As Object
Dim Text1 As Object, Text2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Terminal")             ' Draw shapes
Set Obj2 = Chart.DrawShape("Operation")
Set Obj3 = Chart.DrawShape("Decision")

Set Line1 = Chart.DrawLine(Obj1, Obj2)             ' Draw lines
Set Line2 = Chart.DrawLine(Obj2, Obj3)

Chart.DrawPositionX = 2                            ' Set draw position
Chart.DrawPositionY = 2.5

Set Text1 = Chart.DrawTextBlock("ABC FlowCharter")  ' Draw text objects
Set Text2 = Chart.DrawTextBlock("OLE2 Automation")

Chart.Select (2)                                   ' Select all objects

ABC.MsgBox "There are " + Chart.SelectedOtherCount + " items(s) selected in the chart other than lines or shapes"
```

SelectedShapeCount Property

Usage *ChartObject*.**SelectedShapeCount**

Description The **SelectedShapeCount** property lets you find the number of selected shapes in the Chart object. The **SelectedShapeCount** property is read only.

Data Type Integer

Value The number of selected shapes in the Chart object

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_SelectedShapeCount_Property')}

SelectedShapeCount Property Example

This example uses the **SelectedShapeCount** property of the Chart object to find the number of shapes that are selected in a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Obj3 As Object, Line1 As Object, Line2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Terminal")             ' Draw shapes
Set Obj2 = Chart.DrawShape("Operation")
Set Obj3 = Chart.DrawShape("Decision")

Set Line1 = Chart.DrawLine(Obj1, Obj2)             ' Draw lines
Set Line2 = Chart.DrawLine(Obj2, Obj3)
Chart.Select (2)                                   ' Select all objects

ABC.MsgBox "There are " + Chart.SelectedShapeCount + " shape(s) selected in the chart."
```

ShowLegend Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_SHOWHIDELEGEND');CW(`concfull')}

Usage *ChartObject*.**ShowLegend** = {True | False}

Description The **ShowLegend** property lets you choose to show or hide the Legend. The Legend in the shows the accumulation of the data fields in a chart. The **ShowLegend** property is read/write.

Data Type Integer (Boolean)

Value True shows the Legend; False hides the Legend.

Flow Equivalent The **ShowLegend** property is equivalent to clicking Legend on the Insert menu.

{button Related Topics,PI(`',`IDH_RT_ShowLegend_Property')}

ShowLegend Property Example

This example uses the **ShowLegend** property of the Chart object to display the Legend.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Field1 As Object, Field2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                           ' Position fields below shapes
Set Field1 = Chart.FieldTemplates.Add("Name")      ' Add a field
Field1.Format = 0                                  ' Format field as text
Field1.AccumulationMethod = False                  ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")     ' Add a field
Field2.Format = 0                                  ' Format field as text
Field2.AccumulationMethod = False                  ' No accumulation

Obj1.FieldValues.Item("Name").Value = "Joe Smith"  ' Enter field values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"

Chart.ShowLegend = True                            ' Make field legend visible
```

ShowNodesOnLines Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage *ChartObject*.**ShowNodesOnLines** = {True | False}

Description The **ShowNodesOnLines** property lets you find or set whether lines show connection nodes. Nodes appear where lines connect to each other. They help you distinguish between connected lines and lines that merely overlap. Nodes are represented by small solid circles. The **ShowNodesOnLines** property is read/write.

Data Type Integer (Boolean)

Value True means nodes are shown on lines; False means nodes are not shown on lines.

Flow Equivalent The **ShowNodesOnLines** property is equivalent to clicking Chart Properties on the Format menu, clicking the Indicators tab, and selecting the Show Nodes On Lines option (True) or clearing the option (False).

{button Related Topics,PI(``,`IDH_RT_ShowNodesOnLines_Property')}

Displaying Nodes on Connecting Lines Example

LineCrossoverSize Property
LineCrossoverStyle Property
LinkIndicator Property
LinkShadow Property
NoteIndicator Property
NoteShadow Property
NumberFont Property

Chart Object

ShowNodesOnLines Property Example

This example uses the **ShowNodesOnLines** property of the Chart object to show connection nodes on lines.

```
Dim ABC As Object, Charts As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart

Charts.AddFromTemplate ("causeff.aft")             ' Add chart from template
Set Chart = ABC.ActiveChart                        ' Get the active chart

ABC.MsgBox "Click OK to view nodes on lines."

Chart.ShowNodesOnLines = True                      ' Show connection nodes on lines
```

Type Property (Chart Object)

Usage *ChartObject*.**Type** = *ChartType*

Description The **Type** property of the Chart object lets you find or set a hidden string field up to eight characters in length indicating the chart type. This field is never used within FlowCharter, but is useful within a FlowCharter events VBX. The **Type** property is read/write.

Data Type String

Value The type of a chart. The default is "" (an empty string).

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Type_Property_Chart_Object')}

Units Property (Chart Object) {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Measure_option');CW(`concfull')}

Usage *ChartObject*.**Units** = *UnitsIndicator*

Description You use the **Units** property of the Chart object to find or set the units for measurement in the Chart object and all its child chart objects. In addition, the **Units** property specifies the size and distance values passed in the Preferences object. Default unit value is 0 (inches) for each new Preferences object.

Data Type None

Value The units used for measurements are listed in the table below.

UnitsIndicator	**Description**
0	Inches
1	Centimeters

Flow Equivalent The **Units** property is equivalent to dragging the Inches or Centimeters button from the View category in the Customize dialog box to a toolbar, and then clicking it.

{button Related Topics,PI(`',`IDH_RT_Units_Property_Chart_Object')}

Defining Measurement Units for a Chart
Example
Units Property (Preferences Object)
Chart Object

Units Property (Chart Object) Example

This example uses the **Units** property of the Chart object to set the units for a chart.

```
Dim ABC As Object, Chart As Object
Dim ChartUnits As Integer

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

ChartUnits = Val(InputBox$("Enter 0 for inches" + Chr$(13) + "Enter 1 for centimeters",   "Chart
Units"))                                             ' Get input;   Chr$(13) is Carriage Return

Chart.Units = ChartUnits                             ' Set units
```

View Property {button Flow Equivalent,JI(`FLOW.HLP>procedur','IDH_VIEWZOOMP');CW(`concfull')}

Usage *ChartObject*.**View** = *View*

Description You use the **View** property to find or set the view of the chart. The **View** property is read/write.

Data Type Integer

Value The value in the **View** property indicates the display page.

View	**Description**
0	One to one
1	Current page
2	Used pages
3	Percentage zoom

Flow Equivalent The **View** property is equivalent to clicking Normal, Page, or Full Screen on the View menu, or entering a value in the Zoom Percentage box on the standard toolbar.

{button Related Topics,PI(`','IDH_RT_View_Property')}

Viewing a Chart

PageCount Property
ScrollLeft Property
ScrollPage Method
ScrollPosition Method
ScrollTop Property
ZoomPercentage Property

Chart Object
Example

View Property Example

This example uses the **View** property of the Chart object to set the view of a chart.

```
Dim ABC As Object, Chart As Object
Dim String1 As String, String2 As String, String3 As String
Dim String4 As String, String5 As String
Dim ChartView As Integer

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Set Chart object

String1 = "Choose a View type:" + Chr$(13)             ' Create text for input
String2 = "0" + Chr$(9) + "OneToOne"                   ' Chr$(13) is Carriage Return
String3 = "1" + Chr$(9) + "CurrentPage"                ' Chr$(9) is Tab
String4 = "2" + Chr$(9) + "UsedPages"
String5 = String1 + Chr$(13) + String2 + Chr$(13) + String3 + Chr$(13) + String4

ChartView = Val(InputBox$(String5; "Chart View"))      ' Get input

Chart.View = ChartView                                 ' Set view
```

Activate Method (Chart Object)

Usage *ChartObject*.**Activate**

Description You use the **Activate** method of the Chart object to pull the chart to the front of the FlowCharter workspace. When multiple charts are open, this brings one to the front, or activates it.

Flow Equivalent The **Activate** method is equivalent to clicking the chart from the numbered list of open charts on the Window menu.

{button Related Topics,PI(`',`IDH_RT_Activate_Method_Chart_Object')}

Activating a Chart
Example

Activate Method (Application Object)
Item Method (Charts Collection)

ActiveChart Property
Application Property
Count Property
Name Property (Chart Object)
Visible Property (Application Object)

Chart Object

Activate Method (Chart Object) Example

This example uses the **Activate** method of the Chart object to bring a chart to the front of other charts.

```
Dim ABC As Object, Chart1 As Object, Chart2 As Object
Dim Obj1 As Object, Obj2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart1 = ABC.ActiveChart                       ' Get the active chart

Set Obj1 = Chart1.DrawShape("Operation")           ' Draw shapes
Set Obj2 = Chart1.DrawShape("Operation")

Set Chart2 = ABC.New                               ' Create a new chart
Set Obj1 = Chart2.DrawShape("Decision")            ' Draw shapes
Set Obj2 = Chart2.DrawShape("Decision")

Chart1.Activate                                    ' Bring Chart1 to the front
```

AddHorizontalGuideline Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_GUIDELINESP');CW(`concfull')}

Usage *ChartObject*.**AddHorizontalGuideline** *Position*
The *Position* element specifies the vertical location of the guideline.

Description The FlowCharter user can use guidelines to align objects. When dragging a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let the user align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **AddHorizontalGuideline** method to add a horizontal guideline at the vertical position passed.

Data Type Double

Value None

Flow Equivalent The **AddHorizontalGuideline** method is equivalent to dragging a guideline from the rulers.

{button Related Topics,PI(`',`IDH_RT_AddHorizontalGuideline_Method')}

Using Guidelines
Example

AddVerticalGuideline Method
ClearGuidelines Method
GuidelinesOn Property

Chart Object

AddHorizontalGuideline Method, AddVerticalGuideline Method, and GuidelinesOn Property Example

This example uses the **AddHorizontalGuideline** method, **AddVerticalGuideline** method, and **GuidelinesOn** property of the Chart object to position and show guidelines.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")   ' Start ABC
ABC.Visible = True                              ' Make ABC visible
ABC.New                                             ' Create a new chart
Set Chart = ABC.ActiveChart                     ' Get the active chart

Chart.AddHorizontalGuideline (3)                ' Place horizontal guideline at 3 units
Chart.AddVerticalGuideline (3)                  ' Place vertical guideline at 3 units
Chart.GuidelinesOn = True                       ' Show guidelines
```

AddVerticalGuideline Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_GUIDELINESP');CW(`concfull')}

Usage *ChartObject*.**AddVerticalGuideline** *Position*
The *Position* element specifies the horizontal location of the guideline.

Description You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let you align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **AddVerticalGuideline** method to add a vertical guideline at the horizontal position passed.

Data Type Double

Value None

Flow Equivalent The **AddVerticalGuideline** method is equivalent to dragging a guideline from the rulers.

{button Related Topics,PI(`',`IDH_RT_AddVerticalGuideline_Method')}

Clear_ Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_CLEAR');CW(`concfull')}

Usage *ChartObject*.**Clear_**

ObjectObject.**Clear_**

Description You use the **Clear_** method of the Chart object to clear (delete) all selected objects. You use the **Clear_** method of the Object object to delete the object object. This is useful in removing a temporary object created as part of a routine using the **SetDefaults** method.

Data Type Integer (Boolean)

Value True means the deletion was successful; False means the deletion was not successful.

Flow Equivalent The **Clear_** method is equivalent to pressing the DEL key or clicking Clear on the Edit menu.

{button Related Topics,PI(``,`IDH_RT_Clear_Method')}

Clear_ Method Example

This example uses the **Clear_** method of the Chart object to find and delete the selected objects.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

For X = 1 To 3                                     ' Draw shapes
   Set Obj1 = Chart.DrawShape("Document")
   Set Obj2 = Chart.DrawShape("Decision")
Next X

Chart.SelectShapeType "Decision"                   ' Select all Decision shapes

Chart.Clear_                                       ' Delete selected objects
```

ClearGuidelines Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_GUIDELINESP');CW(`concfull')}

Usage *ChartObject*.**ClearGuidelines**

Description You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let you align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **ClearGuidelines** property to delete all guidelines from the chart. There is currently no way to remove a single guideline.

Flow Equivalent The **ClearGuidelines** method is equivalent to dragging all guidelines from the chart back into the rulers.

{button Related Topics,PI(``,`IDH_RT_ClearGuidelines_Method')}

Using Guidelines
Example

AddHorizontalGuideline Method
AddVerticalGuideline Method
GuidelinesOn Property

Chart Object

ClearGuidelines Method Example

This example uses the **ClearGuidelines** method of the Chart object to clear guidelines from a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, xTime As Long

Set ABC = CreateObject("ABCFlow.application")   ' Start ABC
ABC.Visible = True                              ' Make ABC visible
ABC.New                                             ' Create a new chart
Set Chart = ABC.ActiveChart                     ' Get the active chart

Chart.AddHorizontalGuideline (3)                ' Place horizontal guideline at 3 units
Chart.AddVerticalGuideline (3)                  ' Place vertical guideline at 3 units
Chart.GuidelinesOn = True                       ' Show guidelines

For xTime = 1 to 100000                         ' Wait a couple of seconds
Next xTime

Chart.ClearGuidelines                           ' Clear all guidelines
```

DeselectAll Method

Usage *ChartObject*.**DeselectAll**

Description You use the **DeselectAll** method to deselect all objects. The **DeselectAll** method has the same effect as the Select method with a value of 3.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DeselectAll_Method')}

Selecting Objects in a Chart
Selecting Shapes
Example

Clear_ Method
Select Method
Selected Property
SelectShapeType Method

Chart Object

DeselectAll Method Example

This example uses the **DeselectAll** property of the Chart object to deselect all selected objects in a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

For X = 1 To 3                                     ' Draw shapes
   Set Obj1 = Chart.DrawShape("Document")
   Set Obj2 = Chart.DrawShape("Decision")
Next X

Chart.SelectShapeType "Decision"                   ' Select all Decision shapes
Chart.Cut                                          ' Cut to Clipboard
Chart.PasteSpecial 0, , .5, 2                      ' PasteSpecial with parameters

Chart.DeselectAll                                  ' Deselect all objects
```

CloseChart Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_CLOSE');CW(`concfull')}

Usage *ChartObject*.**CloseChart**

Description You use the **CloseChart** method to close the Chart object without any prompt to save the chart.

Flow Equivalent The **CloseChart** method is equivalent to clicking Close on the File menu, except that there is not a prompt to change a saved chart.

{button Related Topics,PI(`',`IDH_RT_CloseChart_Method')}

Copy Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_COPY');CW(`concfull')}

Usage *ChartObject*.**Copy**

Description You use the **Copy** method to copy selected objects to the Windows Clipboard.

Data Type Integer (Boolean)

Value True means the copy was successful; False means the copy was not successful.

Flow Equivalent The **Copy** method is equivalent to clicking Copy on the Edit menu.

{button Related Topics,PI(`',`IDH_RT_Copy_Method')}

Cutting, Copying, and Pasting Objects
Example

Cut Method
Duplicate Method (Chart Object)
Paste Method
PasteSpecial Method

Chart Object

Cut Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_CUT');CW(`concfull')}

Usage *ChartObject*.**Cut**

Description You use the **Cut** method to cut selected objects to the Windows Clipboard.

Data Type Integer (Boolean)

Value True means the cut was successful; False means the cut was not successful.

Flow Equivalent The **Cut** method is equivalent to clicking Cut on the Edit menu.

{button Related Topics,PI(`',`IDH_RT_Cut_Method')}

Cut, PasteSpecial Methods Example

This example uses the **Cut** method and **PasteSpecial** method of the Chart object to cut selected shapes to the Clipboard and then paste them back into the chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object

Set ABC = CreateObject("ABCFlow.application")   ' Start ABC
ABC.Visible = True                              ' Make ABC visible
ABC.New                                             ' Create a new chart
Set Chart = ABC.ActiveChart                     ' Get the active chart

For X = 1 To 3                                  ' Draw shapes
   Set Obj1 = Chart.DrawShape("Process")
   Set Obj2 = Chart.DrawShape("Decision")
Next X

Chart.SelectShapeType "Decision"                ' Select all Decision shapes
Chart.Cut                                       ' Cut selected shapes to the Clipboard
Chart.PasteSpecial 0, , .5, 2                   ' PasteSpecial with parameters
```

DrawFreeLine Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_CONNECTSHAPES');CW(`concfull')}

Usage *ChartObject*.**DrawFreeLine** (*HorizontalLocation*, *VerticalLocation*)
The *HorizontalLocation* element is the X location of the end point of the line.
The *VerticalLocation* element is the Y location of the end point of the line.

Description You use the **DrawFreeLine** method to draw an unconnected line from the current chart position to a specified end point. The X and Y positions are measured from the top left corner of the chart page. The line is not selected.

Data Type Object

Value The method returns the line object that is drawn. Both elements are doubles.

Flow Equivalent The **DrawFreeLine** method is equivalent to drawing a line not connected to any shapes.

{button Related Topics,PI(`',`IDH_RT_DrawFreeLine_Method')}

Setting the Current Drawing Position
Drawing Unconnected Lines
Example

DrawLine Method
DrawLineToOneObject Method

DrawPositionX Property
DrawPositionY Property
Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

DrawFreeLine Method, CurrentLineRouting Property Example

This example uses the **CurrentLineRouting** property and **DrawFreeLine** method of the Chart object to set the line routing and draw a line.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object, Obj3 As Object,
Obj4 As Object, Obj5 As Object
Dim Line1 As Object, Line2 As Object, Line3 As Object, Line4 As Object

Set ABC = CreateObject("ABCFlow.application")       ' Start ABC
ABC.Visible = True                      ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart             ' Get the active chart

Set Obj1 = Chart.DrawShape("Process")   ' Draw a shape
Chart.DrawPositionX = 1                 'Set position
Chart.DrawPositionY = 1
Chart.CurrentLineRouting = 0            ' Set routing to "direct"
Set Line1 = Chart.DrawLineToOneObject(Obj1, 0)   ' Draw line to Obj1, enter north

Chart.DrawPositionX = 3                 'Set position
Chart.DrawPositionY = 3
Set Obj2 = Chart.DrawShape("Document")        ' Draw a shape
Chart.CurrentLineRouting = 1            ' Set routing to "right angle"
Set Line2 = Chart.DrawLine(Obj1, Obj2, 2, 1)    ' Draw line to Obj2, exit south,
enter east

Chart.DrawPositionX = 5                 'Set position
Chart.DrawPositionY = 5
Set Obj3 = Chart.DrawShape("Decision")        ' Draw a shape
Chart.CurrentLineRouting = 2            ' Set routing to "curve"
Set Line3 = Chart.DrawLine(Obj2, Obj3, 2, 1)   ' Draw line to Obj3, exit south, enter
east

Chart.DrawPositionX = 5                 'Set position
Chart.DrawPositionY = 8
Set Obj4 = Chart.DrawShape("Process")         ' Draw a shape
Chart.CurrentLineRouting = 3            ' Set routing to "organization chart"
Set Line4 = Chart.DrawLine(Obj3, Obj4, 2, 1)   ' Draw line to Obj4, exit south, enter
east

Chart.DrawPositionX = 3                 'Set position
Chart.DrawPositionY = 9
Set Obj5 = Chart.DrawShape("Input/Output")   ' Draw a shape
Chart.CurrentLineRouting = 4            ' Set routing to "cause & effect"
Set Line5 = Chart.DrawLine(Obj4, Obj5, 3, 0)   ' Draw line to Obj5, exit west, enter
west
```

DrawLine Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_CONNECTSHAPES');CW(`concfull')}

Usage *ChartObject*.**DrawLine** (*ShapeObject1*, *ShapeObject2* [, *ExitDirection*] [, *EnterDirection*])
The *ShapeObject1* element is the first shape that the line is connected to.
The *ShapeObject2* element is the second shape that the line is connected to.
The *ExitDirection* element, which is optional, specifies the side where the line exits the first shape.
The *EnterDirection* element, which is optional, specifies the side where the line enters the second shape.

Description You use the **DrawLine** method to draw lines that connect two shapes. You specify the two shapes you want to connect and, optionally, the sides of the shapes that the line connects to. The line is not selected.

Data Type Object. The *ShapeObject1* element and *ShapeObject2* element are Shape objects. The *ExitDirection* element and *EnterDirection* element, which are optional, are integers.

Value The new Line_ object. The following chart describes the direction values.

Value	Direction
0	North
1	East
2	South
3	West

Flow Equivalent The **DrawLine** property is equivalent to drawing a line from one shape to another.

{button Related Topics,PI(`',`IDH_RT_DrawLine_Method')}

Drawing Lines that Connect Shapes Example

DrawFreeLine Method
DrawLineToOneObject Method

Chart Object

DrawLine Method Example

This example uses the **DrawLine** method of the Chart object to draw a line between two shapes.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object, Line1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw shape
Obj1.Text = "Unit 1"                               ' Add text to shape
Set Obj2 = Chart.DrawShape("Decision")             ' Draw shape
Obj2.Text = "Unit 2"                               ' Add text to shape

Set Line1 = Chart.DrawLine(Obj1, Obj2, 0, 2)       ' Draw a line between two shapes

Chart.Repaint                                      ' May be needed for some video modes
```

DrawLineToOneObject Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_CONNECTSHAPES');CW(`concfull')}

Usage

ChartObject.**DrawLineToOneObject** (*ShapeObject* [, *EnterDirection*])
The *ShapeObject* element is the shape that the line is connected to.
The *EnterDirection* element, which is optional, specifies the side where the line enters the shape.

Description

You use the **DrawLineToOneObject** method to draw a line from the current chart position to a specified shape. The line starts at the chart's current drawing position and ends at the shape you specify with *ShapeObject*. You can optionally specify the side of the shape that the line connects to. The line is not selected.

Data Type

Object. The *ShapeObject* element is a Shape object. The *EnterDirection* element, which is optional, is an integer.

Value

The method returns the line object that is drawn. The following table shows the values of the *EnterDirection* element and their meanings.

Value	Direction
0	North
1	East
2	South
3	West

Flow Equivalent

The **DrawLineToOneObject** property is equivalent to drawing a line to a shape.

{button Related Topics,PI(`',`IDH_RT_DrawLineToOneObject_Method')}

DrawLineToOneObject Method Example

This example uses the **DrawLineToOneObject** method of the Chart object to draw a line into a shape, entering on the east side of the shape.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object, Line1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Decision")             ' Draw a shape
Obj1.Text = "Unit 1"                               ' Add text to shape

Chart.DrawPositionX = 2                            ' Set drawing position
Chart.DrawPositionY = 0

Set Line1 = Chart.DrawLineToOneObject(Obj1, 3)     ' Draw line to Obj1, enter east

Chart.Repaint                                      ' May be needed for some video modes
```

DrawShape Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_PLACESHAPES');CW(`concfull')}

Usage *ChartObject*.**DrawShape** ([*ShapeName*])
The *ShapeName* element is the optional name of the shape.

Description You use the DrawShape method to draw shapes. The line is not selected. By default, the DrawShape **method** uses the current shape in the Shape Palette. You can optionally specify the name of the shape you want to draw. All of the shape palettes that ship with FlowCharter have names that appear in the hint line or in the bubble help when the mouse pauses over them. In FlowCharter, the shape's name is defined in the Shape Properties dialog box. (See the documentation that ships with FlowCharter for more information on the available palettes and shapes.) You can open the Properties dialog box by clicking Properties in the Options menu of the Shape Palette. Shapes are automatically placed at the chart's current drawing position.

Data Type Object. The *ShapeName* element is a string.

Value The new object or Null if the creation failed

Flow Equivalent The DrawShape method is equivalent to clicking the Shape tool in the toolbox, clicking the shape you want in the Shape Palette, and clicking in the drawing area.

{button Related Topics,PI(``,`IDH_RT_DrawShape_Method')}

DrawShape Method Example

This example uses the **CurrentShape** property and **DrawShape** method of the Chart object to set the type of shape to be created.

```
Dim ABC As Object, Chart As Object, Obj1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.CurrentShape = "Decision"                    ' Set shape to Decision diamond
Set Obj1 = Chart.DrawShape()                       ' Draw Decision shape
```

DrawTextBlock Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_TEXTBLOCKS');CW(`concfull')}

Usage *ChartObject*.**DrawTextBlock** (*TextString*)
The *TextString* element is the text you want to create.

Description You use the **DrawTextBlock** method to create a text block. The text appears at the current drawing position. The text is not selected.

Data Type Object

Value The text block that is drawn

Flow Equivalent The **DrawTextBlock** method is equivalent to clicking the Text tool in the toolbox, positioning the cursor, and typing text.

{button Related Topics,PI(``,`IDH_RT_DrawTextBlock_Method')}

Setting the Current Drawing Position
Moving Objects
Creating Text Blocks
Example

DrawPositionX Property
DrawPositionY Property

Chart Object

DrawTextBlock Method Example

This example uses the **DrawTextBlock** method of the Chart object to create text objects.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object, Text2 As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Chart.DrawPositionX = 2                              ' Set draw position
Chart.DrawPositionY = 2.5

Set Text1 = Chart.DrawTextBlock("ABC FlowCharter")   ' Draw text objects
Set Text2 = Chart.DrawTextBlock("OLE2 Automation")
```

Duplicate Method (Chart Object)

Usage *ChartObject*.**Duplicate**

Description You use the **Duplicate** method of the Chart object to create a duplicate of the selected objects. The newly created objects will be the only selected objects in the Chart after you call this method.

Data Type Integer (Boolean)

Value True means the duplication was successful; False means the duplication was not successful

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Duplicate_Method_Chart_Object')}

Duplicate Method (Chart Object) Example

This example uses the **Duplicate** method of the Chart object to create duplicates of selected shapes.

```
Dim ABC As Object, Chart As Object, Obj1 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw Operation shape
Obj1.Text = "Unit 1"                               ' Add text to shape

Chart.Select (0)                                   ' Select all shapes

For X = 1 To 3                                     ' Duplicate shape three times
   Chart.Duplicate
Next X
```

InsertObjectFromFile Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Object_command');CW(`concfull')}

Usage *ChartObject*.**InsertObjectFromFile** (*Filename* [, *AsIcon*] [, *AsLink*])
The *Filename* element lets you specify the file to insert. Quotation marks should be used whenever long filenames or long pathnames are used.
The *AsIcon* element lets you paste the file as an icon.
The *AsLink* element lets you paste the file linked.

Description You use the **InsertObjectFromFile** method to insert a new OLE client object from a file into your chart. You can optionally add the element *AsIcon* to paste the file as an icon or the element *AsLink* to paste the file linked. The method returns the file that is inserted as an object.

Data Type Object. The *Filename* element is a string. The *AsIcon* element and *AsLink* element are integers (Boolean).

Value The object that was inserted

Flow Equivalent The **InsertObjectFromFile** method is equivalent to clicking Object on the Insert menu, clicking Create from File, selecting the file you want to insert, and clicking OK. The *AsIcon* element is equivalent to selecting the Display As Icon option. The *AsLink* element is equivalent to selecting the Link to File option.

{button Related Topics,PI(``,`IDH_RT_InsertObjectFromFile_Method')}

Using OLE Client Objects
Example

DoVerb Method
ObjectType Property
PasteLink Method
UpdateFields Method

Chart Object

InsertObjectFromFile Method Example

This example uses the **InsertObjectFromFile** method of the Chart object to insert a sound from a file.

```
Dim ABC As Object, Chart As Object
Dim objOLE As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

' Insert OLE object
Set objOLE = Chart.InsertObjectFromFile("C:\WINDOWS\TADA.WAV", True, True)
```

Paste Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_PASTE');CW(`concfull')}

Usage *ChartObject*.**Paste** ([*HorizontalLocation*] [, *VerticalLocation*])
The *HorizontalLocation* element is the horizontal location of the paste.
The *VerticalLocation* element is the vertical location of the paste.

Description You use the **Paste** method to paste selected objects from the Windows Clipboard. You can optionally specify a horizontal and vertical location for the paste. You set the units used for the location using the **Units** property.

Data Type Integer (Boolean)

Value True means the paste was successful; False means the paste was not successful.

Flow Equivalent The **Paste** method is equivalent to clicking Paste on the Edit menu.

{button Related Topics,PI(`',`IDH_RT_Paste_Method')}

Cutting, Copying, and Pasting Objects
Example

Copy Method
Cut Method
Duplicate Method (Chart Object)
PasteSpecial Method

Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

PasteSpecial Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_PASTESPECIAL');CW(`concfull')}

Usage *ChartObject*.**PasteSpecial** (*Format* [, *AsIcon*] [, *HorizontalLocation*] [, *VerticalLocation*])
The *Format* element lets you specify the format to use for the paste.
The optional *AsIcon* element lets you paste the Clipboard contents as an icon.
The optional *HorizontalLocation* element is the horizontal location of the paste.
The optional *VerticalLocation* element is the vertical location of the paste.

Description You use the **PasteSpecial** method to paste selected objects from the Windows Clipboard specifying a format. You can optionally specify that the object be pasted as an icon. You can optionally specify a horizontal and vertical location for the paste. You set the units for the location using the **Units** property.

Data Type Integer (Boolean)

Value True means the paste was successful; False means the paste was not successful.

The values for the formats are in the following table.

Value	Format
0	ABC Native
1	OLE Client Embed
2	ABC Rich Text
3	Rich Text Format (RTF)
4	Unformatted Text
5	Metafile
6	Device-Independent Bitmap
7	Bitmap
8	OLE Client Link

Flow Equivalent The **PasteSpecial** method is equivalent to clicking Paste Special on the Edit menu and then specifying the format to use for the paste. Specifying that the object on the Clipboard be pasted as an icon is equivalent to selecting the Display As Icon option in the Paste Special dialog box.

{button Related Topics,PI(``,`IDH_RT_PasteSpecial_Method')}

Using Special Clipboard Formats
Example

Copy Method
Cut Method
Duplicate Method (Chart Object)
Paste Method

ClipboardFormatAvailable Property
Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

PasteLink Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_PASTELINK');CW(`concfull')}

Usage *ChartObject*.**PasteLink** [*HorizontalLocation*] [, *VerticalLocation*]
The optional *HorizontalLocation* element is the horizontal location of the paste.
The optional *VerticalLocation* element is the vertical location of the paste.

Description You use the **PasteLink** method to paste the contents of the Clipboard into the chart and link the file that is the source of the contents of the chart. You can optionally specify a horizontal and vertical location for the paste. You set the units used for the location using the **Units** property.

Data Type Integer (Boolean)

Value True means the paste link was successful; False means the paste link was not successful.

Flow Equivalent The **PasteLink** method is equivalent to clicking Paste Link on the Edit menu.

{button Related Topics,PI(`',`IDH_RT_PasteLink_Method')}

Using OLE Client Objects
Example

DoVerb Method
InsertObjectFromFile Method
UpdateFields Method

ObjectType Property
Units Property (Chart Object)
Units Property (Preferences Object)

Chart Object

PasteLink Method Example

This example uses the **PasteLink** method of the Chart object to paste link an object on the Clipboard into a chart. For the paste to work, there must something with an OLE Link format available in the Clipboard. For example, you can put an appropriate item in the Clipboard by opening a .BMP file in Paintbrush, selecting a section of it using the dotted rectangle tool, and clicking Copy in the Edit menu.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

If Chart.ClipboardFormatAvailable(8) Then          ' Is OLE Link available in Clipboard?
   Chart.PasteLink 2, 2                            ' Paste Link at chart coordinates
End If
```

PrintOut Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_PRINT');CW(`concfull')}

Usage *ChartObject*.**PrintOut** [*FromPage*] [, *ToPage*] [, *Copies*] [, *FitToPage*] [, *PrintNotes*]
The *FromPage* element, which is optional, specifies the starting page. The default is the first page.
The *ToPage* element specifies the ending page. The default is the last page.
The *Copies* element, which is optional, specifies the number of copies. The default is 1.
The *FitToPage* element, which is optional, specifies whether to fit the entire chart to one page. The default is False.
The *PrintNotes* element, which is optional, specifies whether to print notes attached to the chart. The default is False.

Description You use the **PrintOut** method to print the Chart object.

Data Type Integer (Boolean)

Value True means that the chart was printed successfully; False means that the chart did not print successfully.

The elements in the **PrintOut** method indicate the options to use when printing.

Element	Description
FromPage	Integer (default is page 1)
ToPage	Integer (default is last page)
Copies	Integer (default is 1)
FitToPage	Integer (Boolean) (default is False)
PrintNotes	Integer (Boolean) (default is False)

Flow Equivalent The **PrintOut** method is equivalent to clicking Print on the File menu and clicking printing options.

{button Related Topics,PI(`',`IDH_RT_PrintOut_Method')}

PrintOut Method Example

This example uses the **PrintOut** method of the Chart object to print a chart from page 1 through page 2, with two copies. The chart is not made to fit to the page and notes are not printed.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

Set Chart = ABC.Open(ABC.Path + "\Samples\Quality.abc")      ' Open a chart

Chart.PrintOut 1, 2, 2, 0, 0                           ' Print chart with these parameters
```

PrintSelected Method {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_PRINTDB');CW(`concfull')}

Usage *ChartObject*.**PrintSelected** [*Copies*] [, *FitToPage*] [, *PrintNotes*]
The *Copies* element, which is optional, specifies the number of copies. The default is 1.
The *FitToPage* element, which is optional, specifies whether to fit the entire chart to one page. The default is False.
The *PrintNotes* element, which is optional, specifies whether to print notes attached to the chart. The default is False.

Description You use the **PrintSelected** method to print the selected objects in the chart.

Data Type Integer (Boolean)

Value True means that the chart was printed successfully; False means that the chart did not print successfully.

The elements in the **PrintSelected** method indicate the options to use when printing.

Element	Description
Copies	Integer (default is 1)
FitToPage	Integer (Boolean) (default is False)
PrintNotes	Integer (Boolean) (default is False)

Flow Equivalent The **PrintSelected** method is equivalent to clicking Print on the File menu and clicking Print Range Selected.

{button Related Topics,PI(``,`IDH_RT_PrintSelected_Method')}

PrintSelected Method Example

This example uses the **PrintSelected** method of the Chart object to print only the selected shapes in a chart.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object
Dim Msg As String


Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")              ' Draw a shape
Set Obj1 = Chart.DrawFreeLine(5, 5)                  ' Draw a line
Set Obj1 = Chart.DrawTextBlock("Anthropology")       ' Draw a text block
Chart.Select (0)                                     ' Select only shapes in the chart

Msg = "When you click OK, only the shapes on the chart will print."
ABC.MsgBox Msg, 64, "Printing soon."
Chart.PrintSelected 1, 1, 0                          ' Print 1 copy, fit to page
```

RevertToSaved Method

Usage *ChartObject*.**RevertToSaved**

Description Use the **RevertToSaved** method to revert to the last saved copy of the document, discarding any changes.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_RevertToSaved_Method')}

RevertToSaved Method Example

This example uses the **RevertToSaved** method of the Chart object to revert to the saved version of a chart.

```
Dim ABC As Object, Chart As Object
Const MB_YesNo = 4, IDYes = 6, IDNo = 7

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

If Chart.HasDiskFile Then                              ' Has the chart been saved to disk?
   If ABC.MsgBox("Chart is saved. Revert to last saved?", MB_YesNo; "Revert to Saved") = IDYes
Then
      Chart.RevertToSaved                              ' Revert to last saved copy of chart
   End If
End If
```

Save Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_SAVE');CW(`concfull')}

Usage *ChartObject*.**Save** [*Path*] [, *FileType*]
The *Path* element, which is optional, is a full or partial pathname and filename for the save.
The *FileType* element, which is optional, specifies the type of file to save.

Description You use the **Save** method to save the current chart to disk. If the chart name ends in .AF2 and you do not specify a name, the chart is saved to a new file with an .AF3 extension. If you specify a partial pathname and filename in the optional first element, the value of DefaultFilePath determines the path. You use the second element to specify whether to save the file as a chart or a template, and whether to save as a version 3.0/4.0, 6, or 7 file. The default is to save the file as a version 7 chart.

Data Type Integer (Boolean)

Value True means that the chart was saved successfully; False means that the chart was not saved successfully.

The following table shows the possible values for the second element and their meanings.

FileType	**Save File As**
0	Chart, version 3.0/4.0
5	Template, (current version only)
6	Chart, version 6
7	Chart, version 7.0

Flow Equivalent The **Save** method is equivalent to clicking Save on the File menu and specifying a path, name, and type for the file.

{button Related Topics,PI(`',`IDH_RT_Save_Method')}

Save Method, CloseChart Method, and Saved Property Example

This example uses the **Saved** property, **Save** method, and **CloseChart** method of the Chart object to find out if a chart is saved, save it, and close it.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object, Line1 As Object
Const MB_YesNo = 4, IDYes = 6, IDNo = 7

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Open a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                ' Draw a shape
Obj1.Text = "Unit 1"                                   ' Add text to the shape
Set Obj2 = Chart.DrawShape("Decision")                 ' Draw a shape
Obj2.Text = "Unit 2"                                   ' Add text to the shape

Set Line1 = Chart.DrawLine(Obj1, Obj2, 0, 2)           ' Draw a line between two shapes

Chart.Repaint                                          ' May be needed for some video modes

If Not Chart.Saved Then                                ' Is this chart saved?
   Chart.Save "C:\Program Files\Micrografx\FlowCharter\Samples\TEST2.FLO"   ' Save the chart as
TEST2
End If

If ABC.MsgBox("Chart is saved. Ready to close?", MB_YesNo, "FlowCharter") = IDYes Then
   Chart.CloseChart                                    ' Close the chart
End If
```

{button Other Example,JI(`',`IDH_HasDiskFile_Property_Example')}

ScrollPage Method

Usage *ChartObject*.**ScrollPage** *PageNumber*
The *PageNumber* element is the page to which to scroll.

Description You use the **ScrollPage** method to scroll the chart to a particular page.

Data Type None

Value The page to which to scroll

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ScrollPage_Method')}

Viewing a Chart

PageCount Property
ScrollLeft Property
ScrollPosition Method
ScrollTop Property
View Property

Chart Object

Example

ScrollPage Method Example

This example uses the **ScrollPage** method of the Chart object to scroll to the last page of a chart.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")' Start ABC
ABC.Visible = True                           ' Make ABC visible
ABC.New                                      ' Create a new chart
Set Chart = ABC.ActiveChart                  ' Get the active chart
Chart.ZoomPercentage = 100                   ' Change the zoom percentage
Chart.DrawSpacingX = 3                       ' Horizontal spacing 3"
For X = 1 To 10 ' Draw shapes
    Set Shape = Chart.DrawShape("Decision")
    Chart.DrawSpacingX = 2                   ' Horizontal spacing 2"
    Chart.DrawSpacingY = 2                   ' Vertical spacing 2"
    Chart.NextShapeHeight = 1.5              ' Height of next shape 1.5
    Chart.NextShapeWidth = 1.5               ' Width of next shape 1.5
    Chart.DrawDirection = 1                  ' Next shape toward right

Next X

Chart.ScrollPage (Chart.PageCount)           ' Scroll to the last page of chart
```

ScrollPosition Method

Usage *ChartObject*.**ScrollPosition** *LeftDistance*, *TopDistance*
The *LeftDistance* element specifies the left part of the chart area to show in the window.
The *TopDistance* element specifies the top part of the chart area to show in the window.

Description You use the **ScrollPosition** method to scroll to a location in the chart.

Value The *LeftDistance* and *TopDistance* elements are each doubles.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ScrollPosition_Method')}

Viewing a Chart
Example

PageCount Property
ScrollLeft Property
ScrollPage Method
ScrollTop Property
View Property

Chart Object

ScrollPosition Method, ScrollLeft Property, and ScrollTop Property Example

This example uses the **ScrollPosition** method, **ScrollLeft** property, and **ScrollTop** property of the Chart object to scroll a chart and report on its position.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart                          ' Get the active chart

Chart.ScrollPosition 18, 18                          ' Scroll to 18 X 18

' Display Scroll positions
ABC.MsgBox ("Scroll Left position = " + Str(Chart.ScrollLeft) + "." + Chr$(13) + "Scroll Top position
= " + Str(Chart.ScrollTop) + ".")                    ' Chr$(13) is Carriage Return
```

Select Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_SELECT');CW(`concfull')}

Usage *ChartObject*.**Select** (*Value*)
The *Value* element specifies what to select.

Description You use the **Select** method to select specified types of objects or to deselect all objects. The **Select** method with a value of 3 has the same effect as the **DeselectAll** method.

Value The values for the types are in the following table.

Value	**Action**
0	Selects all shapes
1	Selects all lines
2	Selects everything
3	Deselects everything

Flow Equivalent The **Select** method is equivalent to clicking Select on the Edit menu and clicking Shapes (0), Lines (1), or All (2).

Note: Visual Basic Script considers the word "Select" a reserved keyword. If you want to use the Select method with Visual Basic Script (and Living FlowCharts), add an underscore to the word "Select" like this:

```
Chart.Select_ (0)      ' Select shapes
```

{button Related Topics,PI(`',`IDH_RT_Select_Method')}

Selecting Objects in a Chart
Selecting Shapes
Example

DeselectAll Method
Selected Property
SelectShapeType Method

Chart Object

Select Method, Copy Method, Paste Method, and ClipboardFormatAvailable Property Example

This example uses the **Select** method, **Copy** method, **ClipboardFormatAvailable** property, and **Paste** method of the Chart object to select a shape, copy it to the Clipboard, check the type of data in the Clipboard, and paste from the Clipboard.

```
Dim ABC As Object, Chart As Object, Obj1 As Object
Dim PasteIt

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")            ' Draw Operation shape
Obj1.Text = "Unit 1"                               ' Add text to shape

Chart.Select (0)                                   ' Select shapes
Chart.Copy                                         ' Copy shape to Clipboard

If Chart.ClipboardFormatAvailable(0) Then          ' Is Clipboard ABC Native data?
   PasteIt = Chart.Paste(2, 2)                     ' Paste shape
End If
```

SelectShapeType Method

Usage *ChartObject*.**SelectShapeType** *ShapeName*
The *ShapeName* element is a string that specifies the type of shapes to select.

Description You use the **SelectShapeType** method to select all shapes of a specific type, such as Decision (diamond) shapes.

Value The *ShapeName* element is a string that specifies the type of shapes you want to select.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_SelectShapeType_Method')}

SelectShapeType Method Example

This example uses the **SelectShapeType** method of the Chart object to select shapes of a particular type.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.DrawSpacingX = 1.5                           ' Set horizontal spacing for shapes

For X = 1 To 3                                     ' Draw shapes
   Set Obj1 = Chart.DrawShape("Document")
   Set Obj2 = Chart.DrawShape("Decision")
Next X

Chart.SelectShapeType "Decision"                   ' Select all Decision shapes
```

SetProtection Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Protecting_Charts');CW(`concfull')}

Usage *ChartObject*.**SetProtection** *Switch*, *Password*
The *Switch* element specifies whether protection is on or off.
The *Password* element specifies a password for the chart.

Description You use the **SetProtection** method to turn protection on and off by setting a protection value and a password.

Data Type None

Value The *Switch* element is an integer (Boolean). True means the chart is protected; False means the chart is not protected. The *Password* element is a string specifying the password for the chart.

Flow Equivalent The **SetProtection** method is equivalent to clicking Protect Chart on the Tools menu and entering a password.

{button Related Topics,PI(`',`IDH_RT_SetProtection_Method')}

SetProtection Method Example

This example uses the **SetProtection** method of the Chart object to set a password for a chart. After the program runs, you must choose Unprotect Chart on the File menu and type the password "turtle" to unprotect the chart.

```
Dim ABC As Object, Chart As Object, Shape As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.SetProtection 1,"turtle"                     ' Set password
```

Spelling Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Spelling_Command');CW(`concfull')}

Usage *ChartObject*.**Spelling**

Description The **Spelling** method lets you start spell checking the chart.

Flow Equivalent The **Spelling** method is equivalent to clicking the text you want to check and clicking Spelling on the Tools menu.

{button Related Topics,PI(``,`IDH_RT_Spelling_Method')}

Spelling Method Example

This example uses the **Spelling** method of the Chart object to run the Spell Checker.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object, Text2 As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart                          ' Get the active chart

Chart.DrawPositionX = 2                              ' Set draw position
Chart.DrawPositionY = 2.5
Chart.DrawSpacingX = 1.5                             ' Set X coordinate spacing
Chart.DrawSpacingY = 1.5                             ' Set Y coordinate spacing


Set Text1 = Chart.DrawTextBlock("Run the Spelll Checker")     ' Draw text objects
Set Text2 = Chart.DrawTextBlock("Is this correkt?")

Chart.Spelling                                       ' Run the Spell Checker
```

ToBack Method (Chart Object) {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Send_to_Back_Command');CW(`concfull')}

Usage *ChartObject*.**ToBack**

Description You use the **ToBack** method of the Chart object to move all selected objects in the chart to the back.

Flow Equivalent The **ToBack** method is equivalent to selecting objects and clicking the Send To Back command from the Order submenu on the Arrange menu.

{button Related Topics,PI(`',`IDH_RT_ToBack_Method_Chart_Object')}

Changing the Display Order of Objects
Example

ToBack Method (Object Object)
ToFront Method (Chart Object)
ToFront Method (Object Object)

Chart Object

ToBack, ToFront Methods (Chart Object) Example

This example uses the **ToBack** method and **ToFront** method of the Chart object to move shapes to the back.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Line1 As Object, X As Integer, A As Integer, ObjText As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.CurrentShapePalette = "BASIC - COLOR"        ' Get Shape palette

Chart.DrawPositionX = 1                            ' Set horizontal drawing position
Set ObjText = Chart.DrawTextBlock("TEXT")          ' Draw text
ObjText.Font.Size = 72                             ' Set font size
ObjText.Font.Color = ABC.RED                       ' Set font color

Chart.DrawSpacingX = 2                             ' Set horizontal draw spacing

Chart.DrawPositionX = 1                            ' Set horizontal drawing position
For A = 1 To 3
   Chart.DrawPositionY = A                         ' Set vertical drawing position
   Set Obj1 = Chart.DrawShape("Process")           ' Draw shapes
   Set Obj2 = Chart.DrawShape("Decision")
Next A

Chart.Select (0)                                   ' Select shapes

Chart.ToBack                                       ' Send to back
ABC.MsgBox "Notice your Chart has been moved to the back of the previous chart. Clicking OK will
bring it to the front."
Chart.ToFront                                      ' Send to front
```

ToFront Method (Chart Object) {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Bring_to_Front_Command');CW(`concfull')}

Usage *ChartObject*.**ToFront**

Description You use the **ToFront** method of the Chart object to move all selected objects to the front.

Flow Equivalent The **ToFront** method is equivalent to selecting objects and clicking the To Front command from the Order submenu on the Arrange menu.

{button Related Topics,PI(``,`IDH_RT_ToFront_Method_Chart_Object')}

Changing the Display Order of Objects
Example

ToBack Method (Chart Object)
ToBack Method (Object Object)
ToFront Method (Object Object)

Chart Object

UpdateFields Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Update_Data_Fields_button');CW(`concfull')}

Usage *ChartObject*.**UpdateFields**

Description The **UpdateFields** method updates all the fields for all the linked shapes in a chart so they reflect the values in the linked charts.

Flow Equivalent The **UpdateFields** method is equivalent to clicking the Update Data Fields button on the Data toolbar.

{button Related Topics,PI(`',`IDH_RT_UpdateFields_Method')}

Using OLE Client Objects
Using Linked Field Data
Example

DoVerb Method
InsertObjectFromFile Method
PasteLink Method

IsLinked Property
LinkedChartName Property
LinkFields Property
LinkIndicator Property
LinkShadow Property
ObjectType Property

LinkNOTIFY Event

Chart Object

UpdateFields Method Example

This example uses the **UpdateFields** method of the Chart object to update linked data fields.

```
Dim ABC As Object, Chart1 As Object, Chart2 As Object
Dim Obj1 As Object, Obj2 As Object, Obj3 As Object, Obj4 As Object
Dim Field1 As Object, Field2 As Object, A As Integer

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart1 = ABC.ActiveChart                         ' Get the active chart

Chart1.DrawSpacingX = 2                              ' Set horizontal draw spacing
Set Obj1 = Chart1.DrawShape("Decision")              ' Draw shapes
Set Obj2 = Chart1.DrawShape("Decision")

Chart1.FieldPlacement = 3                            ' Position fields below shapes
Set Field1 = Chart1.FieldTemplates.Add("Name")       ' Add a field
Field1.Format = 0                                    ' Format field as text

Obj1.FieldValues.Item("Name").Value = "Joe Smith"    ' Set Field Values
Obj2.FieldValues.Item("Name").Value = "Jane Doe"

Obj2.Shape.LinkedChartName = "CHART2"

ABC.New                                              ' Open a new chart
Set Chart2 = ABC.ActiveChart                         ' Get the active chart

Chart2.DrawSpacingX = 2                              ' Set horizontal draw spacing
Set Obj3 = Chart2.DrawShape("Decision")              ' Draw shapes
Set Obj4 = Chart2.DrawShape("Decision")
Set Field1 = Chart2.FieldTemplates.Add("Name")       ' Add a field
Field1.Format = 0                                    ' Format field as text
Set Field2 = Chart2.FieldTemplates.Add("Phone")      ' Add a field
Field2.Format = 0                                    ' Format field as text

Obj3.FieldValues.Item("Name").Value = "Smiley Johnson"       ' Set Field Values
Obj3.FieldValues.Item("Phone").Value = "555-6666"
Obj4.FieldValues.Item("Name").Value = "Wanda Freedman"
Obj4.FieldValues.Item("Phone").Value = "555-7777"

Chart1.UpdateFields                                  ' Update linked fields in Chart1
```

Restore Method (Chart Object)

Usage *ChartObject*.**Restore**

Description The **Restore** method of the Chart object lets you change the chart window to its previous size

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Restore_Method_Chart_Object')}

Minimizing, Maximizing, and Restoring a Window Example

Maximize Method (Chart Object)
Minimize Method (Chart Object)
Restore Method (Application Object)

Chart Object

Minimize Method (Chart Object)

Usage *ChartObject*.**Minimize**

Description The **Minimize** method of the Chart object lets you change a chart window to an icon.

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Minimize_Method_Chart_Object')}

Minimizing, Maximizing, and Restoring a Window Example

Maximize Method (Chart Object)
Minimize Method (Application Object)
Restore Method (Chart Object)

Chart Object

Maximize Method (Chart Object)

Usage *ChartObject*.**Maximize**

Description The **Maximize** method of the Chart object lets you change a chart window to its maximum size.

Flow Equivalent none

{button Related Topics,PI(``,`IDH_RT_Maximize_Method_Chart_Object')}

Minimizing, Maximizing, and Restoring a Window Example

Maximize Method (Application Object)
Minimize Method (Chart Object)
Restore Method (Chart Object)

Chart Object

Maximize Method, Minimize Method, Restore Method (Chart Object), and ActiveChart Property Example

This example uses the **Maximize** method, **Minimize** method, and **Restore** method of the Chart object and the **ActiveChart** property of the Application object to minimize, restore, and maximize a chart window.

```
Dim ABC As Object, Chart As Object
Dim x, y, z

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.CloseAll                                       ' Close all open charts

For x = 1 To 3
   ABC.New                                         ' Add a new chart
   Set Chart = ABC.ActiveChart                     ' Get the active chart
   Chart.Minimize                                  ' Minimize the new chart
Next x

ABC.MsgBox "Click OK to restore all charts to normal size."
For y = 1 To ABC.Charts.Count                      ' For all charts in the collection
   ABC.Charts.Item(y).Restore                      ' Restore each chart's window
Next y

ABC.MsgBox "Click OK to maximize all charts."
For z = 1 To ABC.Charts.Count                      ' For all charts in the collection
   ABC.Charts.Item(z).Maximize                     ' Maximize each chart's window
Next z
```

NextShapeHeight Property

Usage *ChartObject*.**NextShapeHeight** = *Height*

Description The **NextShapeHeight** property lets you find or set the height of the next shape to be drawn.

Data Type Double

Value The height of the next shape to be drawn

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_NextShapeHeight_Property')}

Drawing Shapes
Example
NextShapeWidth Property
Chart Object

NextShapeHeight, NextShapeWidth Properties Example

This example uses the **NextShapeHeight** property and **NextShapeWidth** property of the Chart object to set the height and width of the next shape drawn.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim X

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

For X = 1 To 3
   Set NewShape = Chart.DrawShape("Operation")     ' Draw a shape
   Chart.NextShapeHeight = Chart.NextShapeHeight + .25     ' Set height of next shape
   Chart.NextShapeWidth = Chart.NextShapeWidth + .25       ' Set width of next shape drawn
   Next X
```

NextShapeWidth Property

Usage *ChartObject*.**NextShapeWidth** = *Width*

Description The **NextShapeWidth** property lets you find or set the width of the next shape to be drawn.

Data Type Double

Value The width of the next shape to be drawn

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_NextShapeWidth_Property')}

Drawing Shapes
Example
NextShapeHeight Property
Chart Object

GroupAndLink Method {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_LINKDB');CW(`concfull')}

Usage *ChartObject*.**GroupAndLink** ([*NewChartPath*] [, *FieldsLinked*])
The *NewChartPath* element specifies the full pathname of the new chart. If you omit the element, FlowCharter generates a default chart pathname.
The *FieldsLinked* element specifies whether the new chart's fields are linked to the source chart. If you omit the second element, FlowCharter does not link the fields.

Description The **GroupAndLink** method lets you move a group of selected objects to another chart and replace the moved group with a shape that is linked to the chart to which the group was moved. The **GroupAndLink** method returns the shape that replaced the moved group and has two optional elements. After executing **GroupAndLink**, you can obtain the newly created chart object with the **ActiveChart** property of the Application object.

Data Type Object

Value The chart that is created. The *NewChartPath* element is a string. The *FieldsLinked* element is a Boolean.

Flow Equivalent The **GroupAndLink** method is equivalent to selecting two or more shapes, clicking the Link button on the standard toolbar, and then selecting Group And Link in the Link dialog box.

{button Related Topics,PI(``,`IDH_RT_GroupAndLink_Method')}

Creating Group Links
Example

ActiveChart Property
Link Method
LinkIndicator Property
LinkShadow Property

Chart Object

GroupAndLink Method Example

This example uses the **GroupAndLink** method of the Chart object to group and link selected shapes to a new file.

```
'This example uses the GroupAndLink method of the Chart object
'to group and link selected shapes to a new chart.
'It draws four shapes on a chart, selects the shapes,then moves them
'as a group to a new chart. The moved shapes are automatically replaced
'with one single shape that links the two charts together.
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Set ABC = CreateObject("ABCFlow.application")' Start ABC
ABC.Visible = True                            ' Make ABC visible
ABC.Activate                            ' Bring ABC to the front
ABC.New                                       ' Get a new chart
Set Chart = ABC.ActiveChart                   ' Get the active chart

Set NewShape = Chart.DrawShape("Operation")   ' Draw a shape
NewShape.Color = ABC.MakeRGB(0, 0, 255)       ' Make the shape blue
Chart.Select (0)                        ' Select the shape
For x = 1 To 3                          ' Duplicate shape three times
    Chart.Duplicate
Next x

ABC.MsgBox "Let's move these shapes to another chart."     'This just pauses
                         'the program so you can see what happens next.
Chart.Select (0)                        ' Select all shapes

NewPath = ABC.Path + "\Samples\NewChart.abc"  ' Set the chart you will be linking to

Chart.GroupAndLink (NewPath)            ' Group and link shapes to new chart
```

TypeRequiresEXE Property

Usage *ChartObject*.**TypeRequiresEXE** = {True | False}

Description You can link a compiled Visual Basic EXE program file to a chart so that the EXE program runs automatically when you open the chart. If you set the **TypeRequiresEXE** property to True, the chart requires the EXE to open. If the linked EXE cannot be run, then the chart does not open. The name of the EXE that is linked to a chart is constructed by adding .EXE to the chart Type. Note: FlowCharter only runs one instance of a linked EXE. When a second chart that is linked to an already running EXE is loaded, FlowCharter refers to the currently running EXE. It does not load a second copy of the EXE.

If you set either **TypeRequiresEXE** or **TypeUsesEXE** to True in a program, then you must also ensure that you close all charts of that Type when your program closes. You can use the **ChartTypeShutdown** method to close the charts.

Data Type Integer (Boolean)

Value True means the chart type always runs an associated program (EXE); False means it does not.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_TypeRequiresEXE_Property')}

Linking EXEs to Charts
Example

ChartTypeShutdown Method
Type Property (Chart Object)
TypeUsesEXE Property

Chart Object

TypeRequiresEXE, TypeUsesEXE Properties Example

This example uses the **TypeRequiresEXE** property and the **TypeUsesEXE** property of the Chart object to require or permit use of an executable program.

The following code is placed in the declarations section.

```
Const CHARTTYPE = "PROJECT1"
Const APPNAME = "Form1"
```

The following code is placed in the form.

```
Dim ABC As Object, Chart As Object
Dim ChartUnits As Integer

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

Set Chart = ABC.ActiveChart
Chart.DrawShape ("Decision")                         ' Draw a decision shape
Chart.DrawTextBlock "This chart requires that 'Form1' be loaded when this chart is opened."
Chart.Type = CHARTTYPE                               ' Sets the Chart Type to the constant
' Require that "PROJECT1" be loaded when this chart is opened
Chart.TypeRequiresEXE = True
' Load the CHARTTYPE when charts created with this application are opened
Chart.TypeUsesEXE = True
ABC.MsgBox "Save this chart. Close ABC. Switch to Visual Basic and stop the running application.
Make the stopped application an .EXE. Close Visual Basic and load ABC FlowCharter. Open the chart
you saved."
```

The following code is placed in the QueryUnload section of the form.

```
x = ABC.ChartTypeShutdown(CHARTTYPE, APPNAME)
```

TypeUsesEXE Property

Usage *ChartObject*.**TypeUsesEXE** = {True | False}

Description You can link a compiled Visual Basic EXE program file to a chart so that the EXE program attempts to run when you open the chart. If you set the **TypeUsesEXE** property to True, then the chart attempts to run the linked EXE when it opens. If the EXE cannot be run, the chart still opens. The name of the EXE that is linked to a chart is constructed by adding .EXE to the chart Type. Note: FlowCharter only runs one instance of a linked EXE. When a second chart that is linked to an already running EXE is loaded, FlowCharter refers to the currently running EXE. It does not load a second copy of the EXE.

If you set either **TypeRequiresEXE** or **TypeUsesEXE** to True in a program, then you must also ensure that you close all charts of that Type when your program closes. You can use the **ChartTypeShutdown** method to close the charts.

Data Type Integer (Boolean)

Value True means the chart type attempts to runs an associated program (EXE); False means it does not.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_TypeUsesEXE_Property')}

Linking EXEs to Charts
Example

ChartTypeShutdown Method
Type Property (Chart Object)
TypeRequiresEXE Property

Chart Object

LineCrossoverSize Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage *ChartObject*.**LineCrossoverSize** = *RelativeSize*

Description The **LineCrossoverSize** property lets you find or set the size of the crossover when one line crosses of another. The setting applies to bunny hops and broken lines, but has no effect when the crossover style is solid lines. (See the LineCrossoverStyle property for information on the available styles.) The **LineCrossoverSize** property is read/write.

Data Type Integer

Value The values for the relative sizes for bunny hop crossovers are in the following table. The same relative sizes apply when the style is broken lines.

RelativeSize		**Description**
0		Small
1		Medium
2		Large

Value The relative size of the crossover when one line crosses another

Flow Equivalent The **LineCrossoverSize** property is equivalent to clicking the Endson the Format menu and entering a size in the Crossovers size box to set the size of the crossover.

{button Related Topics,PI(``,`IDH_RT_LineCrossoverSize_Property')}

LineCrossoverStyle Property and LineCrossoverSize Property Example

This example uses the **LineCrossoverStyle** property and **LineCrossoverSize** property of the Chart object to set the style and size used when one line crosses another.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True                            ' Make ABC visible
Set Chart = ABC.New                           ' Make a new chart

Chart.LineCrossoverStyle = 1                  ' Set style to broken lines
Chart.LineCrossoverSize = 2                   ' Set size to large
Chart.MasterItems.HideAll                     ' Get Master Items out of the way

Chart.DrawPositionX = 1                       ' Draw 2 shapes and connect...
Chart.DrawPositionY = 2                       ' ... them with a line
Set Shape1 = Chart.DrawShape
Chart.DrawPositionX = 4
Set Shape2 = Chart.DrawShape
Chart.DrawLine Shape1, Shape2

Chart.DrawPositionX = 1                       ' Draw 2 more shapes and connect...
Chart.DrawPositionY = 1                       ' ... them with an overlapping line
Set Shape1 = Chart.DrawShape
Chart.DrawPositionX = 3
Chart.DrawPositionY = 3
Set Shape2 = Chart.DrawShape
Chart.DrawLine Shape1, Shape2
```

LineCrossoverStyle Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage *ChartObject*.**LineCrossoverStyle** = *Style*

Description The **LineCrossoverStyle** property lets you find or set the style of the crossover when one line crosses another. The **LineCrossoverStyle** property is read/write.

Data Type Integer

Value The values for the styles are in the following table.

Style	**Description**
0	 Solid lines
1	 Bunny hops
2	 Broken lines

Value The style when one line crosses another

Flow Equivalent The **LineCrossoverStyle** property is equivalent to clicking the Crossovers button on the Formatting toolbar and clicking a crossover style.

{button Related Topics,PI(``,`IDH_RT_LineCrossoverStyle_Property')}

ZoomPercentage Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_VIEWZOOMDB');CW(`concfull')}

Usage *ChartObject*.**ZoomPercentage** = *Percentage*

Description The **ZoomPercentage** property lets you find or set the view of the current chart as a percentage of actual size. The **ZoomPercentage** property is read/write.

Data Type Integer

Value The view of the current chart as a percentage of actual size. The ZoomPercentage property can be from 5 to 1600 (5% to 1600%).

Flow Equivalent The **ZoomPercentage** property is equivalent to entering a value in the Zoom Percentage box on the standard toolbar.

{button Related Topics,PI(``,`IDH_RT_ZoomPercentage_Property')}

ZoomPercentage Properties Example

This example uses the **ZoomPercentage** property of the Chart object to change the view of the current chart.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")     ' Start ABC
ABC.Visible = True                                ' Make ABC visible
ABC.New                                           ' Create a new chart
Set Chart = ABC.ActiveChart                       ' Get the active chart
Chart.ZoomPercentage = 50                         ' Set zoom to 50%
```

SendMail Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_SENDMAIL');CW(`concfull')}

Usage *ChartObject*.**SendMail**

Description The **SendMail** method creates a new e-mail message with the chart object as an attachment. The user addresses the e-mail and creates a message as he or she usually does. The **SendMail** method uses the MAPI e-mail system and is compatible with Microsoft Mail.

Data Type Integer (Boolean)

Value True means the e-mail message was created successfully; False means the creation was not successful.

Flow Equivalent The **SendMail** method is equivalent to clicking Send on the File menu.

{button Related Topics,PI(`',`IDH_RT_SendMail_Method')}

SendMail Method Example

This example uses the **SendMail** method of the Chart object to launch a new e-mail message with the chart attached. The user must address the e-mail and create a message as he or she usually does.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True                            ' Make ABC visible
ABC.New                                             ' Create a new chart
Set Chart = ABC.ActiveChart                   ' Get the active chart
Chart.SendMail                      ' Launches a new e-mail with this chart attached
```

SetDefaults Method

Usage *ChartObject*.**SetDefaults** *DefaultObject*
The *DefaultObject* element is an object that has the properties that you want to be the new defaults.

Description The **SetDefaults** method sets the defaults for subsequent objects. The *DefaultObject* is a Shape, Line_, or TextBlock object. Subsequent objects of that type have the defaults you set.

The following table lists the defaults that you set when you use the **SetDefaults** method.

Object Type	Defaults Set
Shape	Border color, border style, border width, fill color, fill pattern, shadow offset, shadow color, numbers on or off, font properties, text alignment
Line_	Color, width, style, source arrow size, source arrow style, source arrow color, destination arrow size, destination arrow style, destination arrow color
TextBlock	Font properties, text alignment

Data Type Integer (Boolean)

Value True means the defaults were created successfully; False means the defaults were not created successfully.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_SetDefaults_Method')}

SetDefaults, Clear_ Methods Example

This example uses the **SetDefaults** method of the Chart object and the **Clear_** method of the Object object to set the defaults for shapes using a dummy object and then delete the object.

```
Dim ABC As Object
Dim Chart As Object
Dim Obj As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

    Set Obj = Chart.DrawShape                      ' Create a dummy shape to hold the defaults

Obj.Shape.NumberShown = True                       ' Turn shape numbering on and set...
Obj.Shape.FillColor = ABC.GRAY                     ' ...all newly drawn shapes to gray...
Obj.Shape.ShadowColor = ABC.DK_Gray                ' ...with dark gray shadows
Obj.Shape.ShadowStyle = 1

Chart.SetDefaults Obj.Shape                        ' Set the defaults for newly draw shapes
Obj.Clear_                                         ' Clear the dummy Object
```

FullScreen Method {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Full_Screen');CW(`concfull')}

Usage *ChartObject*.**FullScreen**

Description The **FullScreen** method shows the chart on the full screen without menus or buttons. Use the **CancelFullScreen** method to return to the previous view.

Data Type Integer (Boolean)

Value True means the chart was shown successfully; False means the chart was not shown successfully.

Flow Equivalent The **FullScreen** method is equivalent to clicking Full Screen on the View menu.

{button Related Topics,PI(`',`IDH_RT_FullScreen_Method')}

FullScreen Method and CancelFullScreen Method Example

This example uses the **FullScreen** and **CancelFullScreen** methods of the Chart object to show two charts full screen without menus or buttons and then cancel the view and return to the previous view.

```
Dim ABC As Object, Chart1 As Object, Chart2 As Object
Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True                            ' Make ABC visible
ABC.Activate
Set Chart1 = ABC.New                          ' Create a new chart
Chart1.DrawPositionX = 2                ' Set draw position
Chart1.DrawPositionY = 2.5
Set NewShape = Chart1.DrawShape("Operation") ' Draw a shape
NewShape.Text = "Chart 1"
Set Chart2 = ABC.New
Chart2.DrawPositionX = 2                ' Set draw position
Chart2.DrawPositionY = 2.5
Set NewShape = Chart2.DrawShape("Operation") ' Draw a shape
NewShape.Text = "Chart 2"
Chart1.FullScreen                       ' Show the first chart full screen
ABC.MsgBox "Click here to switch to next chart"
Chart2.FullScreen                       ' Show the second chart full screen
ABC.MsgBox "Click here to switch to previous chart"
Chart1.FullScreen                       ' Show the first chart full screen
ABC.MsgBox "Click here to cancel full screen mode"
Chart1.CancelFullScreen                 ' Leave full screen mode
```

CancelFullScreen Method {button Flow Equivalent,JI(`FLOW.HLP',`IDH_Full_Screen');CW(`concfull')}

Usage *ChartObject*.**CancelFullScreen**

Description The **CancelFullScreen** method returns a chart to its previous view after you have used the **FullScreen** method to show it on the full screen without menus or buttons.

Data Type Integer (Boolean)

Value True means the chart was shown successfully; False means the chart was not shown successfully.

Flow Equivalent The **CancelFullScreen** method is equivalent to pressing **Esc** to leave the full screen view.

{button Related Topics,PI(``,`IDH_RT_CancelFullScreen_Method')}

Charts Collection

Description The Charts collection is below the Application object. Below the Charts collection are the Chart objects. You can have multiple Chart objects in the Chart collection.

Properties	Methods
Application	Add
Count	AddFromTemplate
Parent	CloseAll
	Item
	Open

{button Related Topics,PI(`',`IDH_RT_Charts_Collection')}

Add Method (Charts Collection) {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_New_Command');CW(`concfull')}

Usage	*ChartsCollection*.**Add**
Description	You use the **Add** method of the Charts collection to create a new chart with default attributes and automatically add it to the collection.
Data Type	Object
Value	The chart that is created
Flow Equivalent	The **Add** method is equivalent to choosing New on the File menu.

{button Related Topics,PI(`',`IDH_RT_Add_Method_Charts_Collection')}

Add, AddFromTemplate, CloseAll Methods (Charts Collection) Example

This example uses the **Add** method, **AddFromTemplate** method, and **CloseAll** method of the Charts collection to add charts, add charts using templates, and close all charts.

```
Dim ABC As Object, Charts As Object
Dim Shape As Object, X As Integer

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                'Open a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

For X = 1 To 3                                         ' Add charts to the collection
    ABC.Charts.Add
Next X

ABC.Charts.AddFromTemplate (ABC.Path + "\Template\Default.aft")        ' Add a chart using a template

ABC.MsgBox "There are " & ABC.Charts.Count & " charts open."       ' Count the charts

ABC.Charts.CloseAll                                    ' Close all charts in collection
```

AddFromTemplate Method

Usage *ChartsCollection*.**AddFromTemplate** (*TemplateName*)
The *TemplateName* element is the path and name of the template to use to create the chart. Quotation marks should be used whenever long filenames or long pathnames are used.

Description You use the **AddFromTemplate** method to create a new chart based on the specified chart template name. If *TemplateName* file cannot be loaded for any reason, the returned Chart.Valid is False.

Data Type Object

Value The chart that is created

Flow Equivalent The **AddFromTemplate** method is equivalent to clicking Open on the File menu, choosing file type AFT, then saving the chart as file type FLO.

{button Related Topics,PI(`',`IDH_RT_AddFromTemplate_Method')}

Creating New Charts
Example

Add Method (Charts Collection)
New Method
NewFromTemplate Method

Charts Collection

Item Method (Chart Objects Collection)

Usage *ChartsCollection*.**Item** ({*PathName* | *Number*})
The *PathName* element is a string indicating the full path and executable name of the chart**.**
Quotation marks should be used whenever long filenames or long pathnames are used.
The *Number* element is the chart's ordering position within the collection.

Description The **Item** method of the Charts collection lets you retrieve a chart from the Charts collection. The method returns a nonvalid chart object if the specified chart object does not exist.

Data Type Object

Value The Chart object

Flow Equivalent The **Item** method of the Charts collection is equivalent to opening the Window menu and choosing the chart from the numbered list of open charts.

{button Related Topics,PI(`',`IDH_RT_Item_Method_Charts_Collection')}

Activating a Chart
Example

Activate Method (Chart Object)
Item Method (FieldTemplates Collection)
Item Method (FieldValues Collection)
Item Method (Menu Collection)
Item Method (Objects Collection)

ActiveChart Property
Application Property
Count Property
Name Property (Chart Object)

Charts Collection

Item Method (Charts Collection) Example

This example uses the **Item** method of the Charts collection to retrieve all items in a chart.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object
Dim CurrentShape As Object, CurrentItem As Object
Dim X, Y

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Open a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Chart.DrawSpacingX = 1                                 ' Set the spacing for the shapes

For X = 1 To 5
    Set CurrentShape = Chart.DrawShape                 ' Draw a shape
Next X

ABC.MsgBox "Click OK to turn all items in the chart red."
Set Everything = Chart.Objects
For Y = 1 To Everything.Count                          ' For all objects in the chart
    Set CurrentItem = Everything.Item(Y)               ' Get the next item
    CurrentItem.Color = ABC.RED                        ' Make the item red
Next Y
```

{button Other Example,PI(``,`IDH_Open_Method_Example2')}

FlowCharter Events

AppQuitNOTIFY

AppQuitSUBCLASS

AppMenuSUBCLASS

AppMenuHintSUBCLASS

AppMenuPopupSUBCLASS

ChartActivateNOTIFY

ChartChangeNOTIFY

ChartCloseSUBCLASS

ChartCloseNOTIFY Event

ChartDeActivateNOTIFY

ChartNewNOTIFY

ChartOpenNOTIFY

ChartPasteNOTIFY

ChartSavedNOTIFY Event

DeleteLineNOTIFY Event

DeleteShapeNOTIFY Event

DeleteSUBCLASS

DoubleClickSUBCLASS

ExclusiveSelectionNOTIFY

FieldSetupDialogSUBCLASS Event

FieldValueChangedNOTIFY

FieldViewerHideSUBCLASS Event

FieldViewerShowSUBCLASS Event

LinkNOTIFY

NewLineNOTIFY

NewShapeNOTIFY

ObjectClickSUBCLASS

ObjectFontChangeNOTIFY

ObjectLineAttachNOTIFY

ObjectLineDeAttachNOTIFY Event

ObjectMovedNOTIFY

ObjectMoveSUBCLASS

ObjectSizedNOTIFY

ObjectSizeSUBCLASS

ObjectTextChangedNOTIFY

ReplaceShapeNOTIFY

SelectionChangeNOTIFY Event

SpecialKeySUBCLASS

{button Related Topics,PI(`',`IDH_RT_ABC_Events')}

AppQuitNOTIFY Event

Description The **AppQuitNOTIFY** event occurs when FlowCharter is closed. The **AppQuitNOTIFY** event procedure can be used for final actions that you want your program to perform before it closes. If you want the Visual Basic application to close when FlowCharter does, put a Visual Basic End statement in this procedure.

{button Related Topics,PI(``,`IDH_RT_AppQuitNOTIFY_Event')}

When FlowCharter Closes

AppQuitSUBCLASS Event

Description The **AppQuitSUBCLASS** event occurs when a request is made to close FlowCharter. The user can request that FlowCharter close by a choosing Exit oin the FlowCharter File menu, pressing **ALT+F4**, or double clicking the FlowCharter window Control box. The **AppQuitSUBCLASS** event procedure is triggered before FlowCharter closes. You can prevent FlowCharter from closing by setting the ABC1 object **Override** property to True.

{button Related Topics,PI(``,`IDH_RT_AppQuitSUBCLASS_Event')}

When FlowCharter Closes

AppMenuSUBCLASS Event

Description The **AppMenuSUBCLASS** event occurs when the user chooses an item on an add-on menu. The menu item object that was chosen is passed to the event procedure in the MenuItem variable.

{button Related Topics,PI(``,`IDH_RT_AppMenuSUBCLASS_Event')}

When MenuItems Are Chosen

AppMenuHintSUBCLASS Event

Description The **AppMenuHintSUBCLASS** event occurs when the user moves the menu cursor to an item on an add-on menu. The **AppMenuHintSUBCLASS** event procedure is triggered before FlowCharter highlights the menu item. The menu item object to be highlighted is passed to the event procedure in the MenuItem variable.

{button Related Topics,PI(`',`IDH_RT_AppMenuHintSUBCLASS_Event')}

When MenuItems Are Highlighted

AppMenuPopupSUBCLASS Event

Description

The **AppMenuPopupSUBCLASS** event occurs when the user opens an add-on menu by clicking the menu's name. Add-on menus are created with the **AddMenu** method of the Application object. The **AppMenuPopupSUBCLASS** event procedure is triggered before FlowCharter displays the add-on menu. The Menu object about to open is passed to the event procedure in the Menu variable.

Because the **AppMenuPopupSUBCLASS** event is triggered before the add-on menu opens, you can use this event procedure to determine whether any items on the add-on menu should be disabled (gray) or checked. A menu item is disabled by setting the **Enabled** property of the MenuItem object to False. A menu item is checked by setting the **Checked** property of the MenuItem object to True.

When Add-On Menus Open

ChartActivateNOTIFY Event

Description The **ChartActivateNOTIFY** event occurs when a chart is activated when a user clicks on it, chooses it from the Window menu or the Chart's System Menu Next, etc. The **ChartActivateNOTIFY** event procedure is triggered following the activation of the chart. The activated chart object is passed to the event procedure in the Chart object variable.

{button Related Topics,PI(`',`IDH_RT_ChartActivateNOTIFY_Event')}

When Charts Are Activated
ChartDeActivateNOTIFY Event

ChartDeActivateNOTIFY Event

Description The **ChartDeActivateNOTIFY** event occurs when a chart loses focus because a different chart is activated. The **ChartDeActivateNOTIFY** event procedure is triggered following the loss of focus. (It is not triggered when the chart closes.) The chart object is passed to the event procedure in the Chart object variable.

{button Related Topics,PI(``,`IDH_RT_ChartDeActivateNOTIFY_Event')}

ChartChangeNOTIFY Event

Description The **ChartChangeNOTIFY** event occurs when a chart is changed in any way. The **ChartChangeNOTIFY** event procedure is triggered following the changing of the chart. The changed chart object is passed to the event procedure in the Chart object variable.

When Charts Change

ChartCloseNOTIFY Event

Description The **ChartCloseNOTIFY** event occurs when the chart is closing. The **ChartCloseNOTIFY** event procedure is triggered following the closing of the chart. The chart object that is closed is passed to the event procedure in the Chart object variable.

{button Related Topics,PI(``,`IDH_RT_ChartCloseNOTIFY_Event')}

When Charts Close

ChartCloseSUBCLASS Event

Description The **ChartCloseSUBCLASS** event occurs when the user closes a chart by choosing Close in the FlowCharter File menu. The **ChartCloseSUBCLASS** event procedure is triggered before the closing of the chart. The chart object that is about to close is passed to the event procedure in the Chart object variable.

{button Related Topics,PI(``,`IDH_RT_ChartCloseSUBCLASS_Event')}

When Charts Close

ChartNewNOTIFY Event

Description The **ChartNewNOTIFY** event occurs when the user creates a new chart by choosing New on the File menu of FlowCharter. The **ChartNewNOTIFY** event procedure is triggered following the creation of the new chart. The new chart object is passed to the event procedure in the Chart object variable.

When New Charts Are Created

ChartOpenNOTIFY Event

Description The **ChartOpenNOTIFY** event occurs when the user opens a new chart file by choosing Open on the File menu of FlowCharter. The **ChartOpenNOTIFY** event procedure is triggered following the successful opening of the chart file. The opened chart object is passed to the event procedure in the Chart object variable.

{button Related Topics,PI(``,`IDH_RT_ChartOpenNOTIFY_Event')}

When Charts Open

ChartPasteNOTIFY Event

Description The **ChartPasteNOTIFY** event occurs when a user pastes something into a chart by choosing Paste on the Edit menu of FlowCharter. The **ChartPasteNOTIFY** event procedure is triggered following the paste. The chart object is passed to the event procedure in the Chart object variable.

{button Related Topics,PI(``,`IDH_RT_ChartPasteNOTIFY_Event')}

When Charts Are Pasted

ChartSavedNOTIFY Event

Description The **ChartSavedNOTIFY** event occurs when a user saves the chart by choosing Save on the File menu of FlowCharter. The **ChartSavedNOTIFY** event procedure is triggered following the save of a chart. The chart object is passed to the event procedure in the Chart object variable.

This event is useful for tracking chart changes.

When Charts Are Saved

DeleteLineNOTIFY Event

Description The **DeleteLineNOTIFY** event occurs when a line is deleted. The user deletes lines by selecting the lines and pressing **DEL** or choosing Clear from the Edit menu. The **DeleteLineNOTIFY** event procedure is triggered after FlowCharter performs the deletion. The line that is deleted is passed to the event procedure in the Object variable, and the chart in which the line is located is passed in the Chart variable.

When Objects Are Deleted

DeleteShapeNOTIFY Event

Description The **DeleteShapeNOTIFY** event occurs when one or more Shapes are deleted. The user deletes Shapes by selecting the Shapes and pressing **DEL** or choosing Clear from the Edit menu. The **DeleteShapeNOTIFY** event procedure is triggered after FlowCharter performs the deletion. The Shape to be deleted first is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

{button Related Topics,PI(``,`IDH_RT_DeleteShapeNOTIFY_Event')}

When Objects Are Deleted

DeleteSUBCLASS Event

Description The **DeleteSUBCLASS** event occurs when one or more Objects are deleted. The user deletes Objects by selecting the Objects and pressing **DEL** or choosing Clear from the Edit menu. The **DeleteSUBCLASS** event procedure is triggered before FlowCharter performs the deletion. The Object to be deleted first is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

{button Related Topics,PI(``,`IDH_RT_DeleteSUBCLASS_Event')}

When Objects Are Deleted

DoubleClickSUBCLASS Event

Description The **DoubleClickSUBCLASS** event occurs when the user double clicks a Shape object. The **DoubleClickSUBCLASS** event procedure is triggered before FlowCharter shows the Shape as selected. The clicked Shape is passed to the event procedure in the Object variable, and the chart in which the shape is located is passed in the Chart variable.

When Shapes Are Double Clicked

ExclusiveSelectionNOTIFY Event

Description The **ExclusiveSelectionNOTIFY** event occurs when the user selects a single Object object. The **ExclusiveSelectionNOTIFY** event procedure is triggered after FlowCharter shows the Object as selected. The selected Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable. **Note:** If the user selects more than one object, the **ExclusiveSelectionNOTIFY** event is not activated.

When Objects Are Selected

FieldSetupDialogSUBCLASS Event

Description The **FieldSetupDialogSUBCLASS** event occurs when the user tried to bring up the Setup Fields dialog box. The **FieldSetupDialogSUBCLASS** event procedure is triggered before FlowCharter brings up the Setup Fields dialog box. The Setup Fields dialog box that the user is trying to bring up is passed to the event procedure in the Setup Fields dialog variable.

FieldValueChangedNOTIFY Event

Description The **FieldValueChangedNOTIFY** event occurs when the user changes a FieldValue object. The **FieldValueChangedNOTIFY** event procedure is triggered after FlowCharter changes the FieldValue. The FieldValue that was changed is passed to the event procedure in the FieldValue variable, the Object that owns the field is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

{button Related Topics,PI(`',`IDH_RT_FieldValueChangedNOTIFY_Event')}

When Field Values Change

FieldViewerHideSUBCLASS Event

Description The **FieldViewerHideSUBCLASS** event occurs when the user is about to hide a FieldViewer object. The **FieldViewerHideSUBCLASS** event procedure is triggered before FlowCharter changes the FieldViewer. The FieldViewer that the user is trying to hide is passed to the event procedure in the FieldViewer variable.

{button Related Topics,PI(``,`IDH_RT_FieldViewerHideSUBCLASS_Event')}

Displaying the Viewers

FieldViewerShowSUBCLASS Event

Description The **FieldViewerShowNOTIFY** event occurs when the user is about to show a FieldViewer object. The **FieldViewerShowSUBCLASS** event procedure is triggered before FlowCharter shows the FieldViewer. The FieldViewer that the user is trying to show is passed to the event procedure in the FieldViewer variable.

This can be used to bring up a custom field viewer.

Displaying the Viewers

LinkNOTIFY Event

Description The **LinkNOTIFY** event occurs when a chart file is opened by double clicking the object to which it is linked. The **LinkNOTIFY** event procedure is triggered following the successful opening of the chart file. The chart object from which the linked chart was opened (the source chart) is passed to the event procedure in the Chart object variable. The linked chart object (the chart just opened) can be obtained using the **ActiveChart** property of the Application object. The Object that was double clicked in the source chart to open the linked chart is passed to the event procedure in the Object object variable.

{button Related Topics,PI(`',`IDH_RT_LinkNOTIFY_Event')}

When Linked Charts Open

IsLinked Property
Link Method

NewLineNOTIFY Event

Description The **NewLineNOTIFY** event occurs when the user draws a new Line object. The **NewLineNOTIFY** event procedure is triggered after FlowCharter draws the Line. The drawn Line is passed to the event procedure in the Object variable, and the chart in which the Line is located is passed in the Chart variable.

{button Related Topics,PI(``,`IDH_RT_NewLineNOTIFY_Event')}

When Lines Are Drawn

NewShapeNOTIFY Event

Description The **NewShapeNOTIFY** event occurs when the user draws a new Shape object. The **NewShapeNOTIFY** event procedure is triggered after FlowCharter draws the Shape. The drawn Shape is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

{button Related Topics,PI(``,`IDH_RT_NewShapeNOTIFY_Event')}

When Shapes Are Drawn

ObjectClickSUBCLASS Event

Description The **ObjectClickSUBCLASS** event occurs when the user clicks an object. The **ObjectClickSUBCLASS** event procedure is triggered before FlowCharter shows the Object as selected. The clicked Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

When Objects Are Clicked

ObjectFontChangeNOTIFY Event

Description The **ObjectFontChangeNOTIFY** event occurs when the user changes the font of one or more Text objects. The **ObjectFontChangeNOTIFY** event procedure is triggered after FlowCharter displays the Text objects in the changed font. The Text object that was changed first is passed to the event procedure in the Object variable, and the chart in which the text is located is passed in the Chart variable.

{button Related Topics,PI(`',`IDH_RT_ObjectFontChangeNOTIFY_Event')}

When Fonts Change

ObjectLineAttachNOTIFY Event

Description

The **ObjectLineAttachNOTIFY** event occurs when the user attaches a line to an Object. The **ObjectLineAttachNOTIFY** event procedure is triggered after FlowCharter attaches the Line. The Object to which the line is attached is passed to the event procedure in the Object variable, the line is passed in the Object2 variable, and the chart in which the Object is located is passed in the Chart variable.

The event is also triggered when a shape is dropped on a line creating new connections.

Object in this case only means shapes, not textblocks.

{button Related Topics,PI(``,`IDH_RT_ObjectLineDeAttachNOTIFY_Event')}

When Lines Attach

ObjectLineDeAttachNOTIFY Event

Description

The **ObjectLineDeAttachNOTIFY** event occurs when the user detaches a line from an Object. The **ObjectLineDeAttachNOTIFY** event procedure is triggered after FlowCharter detaches the Line. The Object to which the line is attached is passed to the event procedure in the Object variable, the line is passed in the Object2 variable, and the chart in which the Object is located is passed in the Chart variable.

The event is also triggered when a shape is deleted from a line connecting two other shapes.

Object in this case only means shapes, not textblocks.

When Lines Attach
ObjectLineAttachNOTIFY event

ObjectMovedNOTIFY Event

Description The **ObjectMovedNOTIFY** event occurs when an Object object is moved. The **ObjectMovedNOTIFY** event procedure is triggered after FlowCharter has moved the Object. The Object that was moved is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

{button Related Topics,PI(`',`IDH_RT_ObjectMovedNOTIFY_Event')}

When Objects Move

ObjectMoveSUBCLASS Event

Description The **ObjectMoveSUBCLASS** event occurs when the user starts to move an Object object. The **ObjectMoveSUBCLASS** event procedure is triggered before FlowCharter initiates any move behavior. The Object about to move is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

{button Related Topics,PI(``,`IDH_RT_ObjectMoveSUBCLASS_Event')}

When Objects Move

ObjectSizedNOTIFY Event

Description The **ObjectSizedNOTIFY** event occurs when an Object object is resized. The **ObjectSizedNOTIFY** event procedure is triggered after FlowCharter has resized the Object. The Object that was resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

{button Related Topics,PI(`',`IDH_RT_ObjectSizedNOTIFY_Event')}

When Objects Are Resized

ObjectSizeSUBCLASS Event

Description The **ObjectSizeSUBCLASS** event occurs when the user starts to resize an Object object. The **ObjectSizeSUBCLASS** event procedure is triggered before FlowCharter initiates any resizing behavior. The Object to be resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

{button Related Topics,PI(`',`IDH_RT_ObjectSizeSUBCLASS_Event')}

When Objects Are Resized

ReplaceShapeNOTIFY Event

Description The **ReplaceShapeNOTIFY** event occurs when the user replaces one or more Shape objects. The **ReplaceShapeNOTIFY** event procedure is triggered after FlowCharter replaces the Shape objects. The Shape to be replaced first is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

When Shapes Are Replaced

SelectionChangeNOTIFY Event

Description The **SelectionChangeNOTIFY** event occurs when the user selects or deselects one or more objects. The **SelectionChangeNOTIFY** event procedure is triggered after FlowCharter selects the objects. The chart in which the selections are being made is passed in the Chart variable, the number of objects selected is passed in the Count variable, and if one object is selected, it will be passed in the Object variable.

{button Related Topics,PI(``,`IDH_RT_SelectionChangeNOTIFY_Event')}

When Objects Are Selected

SpecialKeySUBCLASS Event

Description

The **SpecialKeySUBCLASS** event occurs when the user presses one of the special keys. The **SpecialKeySUBCLASS** event procedure is triggered before FlowCharter responds to the key press. A code representing the key is passed to the event procedure in the **WParam** variable. These codes are defined in the table below.

Key	Code
F1	1
F2	2
F3	3
F4	4
F5	5
F6	6
F7	7
F8	8
F9	9
F10	10
F11	11
F12	12
Tab	13
Esc	27
PgUp	33
PgDn	34
End	35
Home	36
Left Arrow	37
Up Arrow	38
Right Arrow	39
Down Arrow	40
Ins	45
Del	46

{button Related Topics,PI(``,`IDH_RT_SpecialKeySUBCLASS_Event')}

When Special Keys Are Pressed

ObjectTextChangedNOTIFY Event

Description The **ObjectTextChangedNOTIFY** event occurs when the user changes a text block. The **ObjectTextChangedNOTIFY** event procedure is triggered after FlowCharter changes the TextBlock. The Object that owns the text is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

When Text Changes

FieldTemplate Object

Description The FieldTemplate object is below the FieldTemplates collection. You can have multiple FieldTemplate objects.

Properties	Methods
Accumulation	There are no methods for the
AccumulationMethod	FieldTemplate object.
Application	
Format	
Hidden	
Name	
Parent	
Type	

{button Related Topics,PI(`',`IDH_RT_FieldTemplate_Object')}

Accumulation Property

Usage *FieldTemplateObject*.**Accumulation**

Description The **Accumulation** property lets you find the accumulated value of data fields, as the value will appear in the Legend. You set the type of accumulation using the **AccumulationMethod** property. The **Accumulation** property is read only.

Data Type Double

Value The value of the **Accumulation** property is the accumulated value of a FieldTemplate object. The Field Template object can be any data field that is added to a chart.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Accumulation_Property')}

Working with Data Field Values
Example

AccumulationMethod Property
Item Method (FieldTemplates Collection)
ShowLegend Property
Value Property

FieldTemplate Object

Accumulation Property Example

This example uses the **Accumulation** property of the FieldTemplate object to find a data field's accumulation.

```
Dim ABC As Object, MasterItems As Object, Chart As Object
Dim Field_Template As Object
Dim Field_Accumulation As Double

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory")    ' Add a data field

Field_Template.AccumulationMethod = 1                  ' Set accumulation method to sum

Field_Accumulation = Field_Template.Accumulation       ' Get field's accumulation
ABC.MsgBox CStr(Field_Accumulation)
```

AccumulationMethod Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *FieldTemplateObject*.**AccumulationMethod** = *Integer*

Description The **AccumulationMethod** property lets you find or specify the type of accumulation for a data field. The accumulation is calculated for the Legend. The **AccumulationMethod** property is read/write.

Data Type Integer

Value The values for the accumulation methods are in the following table.

Value	Accumulation Method
0	No Accumulation: Do not include this field in the Legend.
1	Sum: The total of all field values added together.
2	Mean: The average of all the values.
3	Median: The middle value in the entire range of values.
4	Min: The smallest value in the entire range of values.
5	Max: The largest value in the entire range of values.
6	Range: The difference between the largest and smallest values.
7	Object Count: The number of values.
8	Filled Count: The number of fields that are not empty.

Flow Equivalent The **AccumulationMethod** property is equivalent to clicking Data Field on the Insert menu, and then choosing an option in the Accumulation Method list box in the Setup Fields dialog box.

{button Related Topics,PI(`',`IDH_RT_AccumulationMethod_Property')}

Changing Data Field Attributes
Example

Add Method (FieldTemplates Collection)

Accumulation property
Format Property
Hidden Property
Name Property (FieldValue Object)
ShowLegend Property
Type Property (FieldValue Object)

FlowCharter Object

AccumulationMethod Property Example

This example uses the **AccumulationMethod** property of the FieldTemplate object to set and find a data field's accumulation method.

```
Dim ABC As Object
Dim MasterItems As Object
Dim Chart As Object
Dim Field_Template As Object
Dim Field_Accumulation_Method As Single

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory")    ' Add a data field

Field_Template.AccumulationMethod = 1                  ' Set accumulation method to sum

Field_Accumulation_Method = Field_Template.AccumulationMethod     ' Find method

Select Case Field_Accumulation_Method                  ' Report accumulation method
   Case 0
      ABC.MsgBox "No Accumulation"
   Case 1
      ABC.MsgBox "Sum"
   Case 2
      ABC.MsgBox "Mean"
   Case 3
      ABC.MsgBox "Median"
   Case 4
      ABC.MsgBox "Min"
   Case 5
      ABC.MsgBox "Max"
   Case 6
      ABC.MsgBox "Range"
   Case 7
      ABC.MsgBox "Total"
   Case 8
      ABC.MsgBox "Non Null Total"
End Select
```

Format Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *FieldTemplateObject*.**Format** = *Format*

Description The **Format** property lets you find or set the format for a data field. The Format property is read/write.

Data Type Integer

Value The values for the formats of data fields are in the following table.

Value	Duration Format
100	# w.
101	# weeks
102	# d.
103	# days
104	# h.
105	# hrs.
106	# hours
107	# m.
108	# mins.
109	# minutes
110	# s.
111	# secs.
112	# seconds
113	# TMU
114	h:m
115	m:s
116	h:m:s

Value	Percent Format
400	##%
401	#0.00%

Value	Date Format
200	M/D/YY
201	MMMM-D-YY
202	MMMM DD, YYYY
203	MMM-YY
204	MMMM YYYY

Value	Currency Format
300	$###0.00($###0.00)
301	$#,##0.00($#,##0.00)
302	$###0($###0)
303	$#,##0($#,##0)

Value	Number Format
500	###0
501	###0.00
502	###0.0000
503	#,##0
504	#,##0.00
505	#,##0.0000

Flow Equivalent The **Format** property is equivalent to clicking Data Field on the Insert menu, selecting a data field, and then choosing a format for the field in the Setup Fields dialog box.

{button Related Topics,PI(``,`IDH_RT_Format_Property')}

Changing Data Field Attributes
Example

AccumulationMethod Property
Add Method (FieldTemplates Collection)
Hidden Property
Name Property (FieldValue Object)
Type Property (FieldValue Object)

FieldTemplate Object

Format Property Example

This example uses the **Format** property of the FieldTemplate object to find a data field's format.

```
Dim ABC As Object
Dim MasterItems As Object, Chart As Object
Dim Field_Template As Object
Dim Field_Accumulation As Double
Dim Field_Accumulation_Method As Single
Dim Field_Format As Single

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory")    ' Add a data field

Field_Template.AccumulationMethod = 1                  ' Set accumulation method to sum

Field_Format = Field_Template.Format                   ' Get field format

Select Case Field_Format                               ' Report field format
   Case 100
      ABC.MsgBox "# w."
   Case 101
      ABC.MsgBox "# weeks"
   Case 102
      ABC.MsgBox "# d."
   Case 103
      ABC.MsgBox "# days"
   Case 104
      ABC.MsgBox "# h."
   Case 105
      ABC.MsgBox "# hrs."
   Case 106
      ABC.MsgBox "# hours"
   Case 107
      ABC.MsgBox "# m."
   Case 108
      ABC.MsgBox "# mins."
   Case 109
      ABC.MsgBox "# minutes"
   Case 110
      ABC.MsgBox "# s."
   Case 111
      ABC.MsgBox "# secs."
   Case 112
      ABC.MsgBox "# seconds"
   Case 113
      ABC.MsgBox "# TMU"
```

```
    Case 114
        ABC.MsgBox "h:m"
    Case 115
        ABC.MsgBox "m:s"
    Case 116
        ABC.MsgBox "h:m:s"
    Case 200
        ABC.MsgBox "M/d/yy"
    Case 201
        ABC.MsgBox "mmm-d-yy"
    Case 202
        ABC.MsgBox "MMMMM dd, yyyy"
    Case 203
        ABC.MsgBox "mmm-yy"
    Case 204
        ABC.MsgBox "MMMMM yyy"
    Case 300
        ABC.MsgBox "$###0.00($###0.00)"
    Case 301
        ABC.MsgBox "$#,##0.00($#,##0.00)"
    Case 302
        ABC.MsgBox "$###0($#,##0.00)"
    Case 303
        ABC.MsgBox "$#,##0($#,##0)"
    Case 400
        ABC.MsgBox "##%"
    Case 401
        ABC.MsgBox "#0.00%"
    Case 500
        ABC.MsgBox "###0"
    Case 501
        ABC.MsgBox "###0.00"
    Case 502
        ABC.MsgBox "###0.0000"
    Case 503
        ABC.MsgBox "#,##0"
    Case 504
        ABC.MsgBox "#,##0.00"
    Case 505
        ABC.MsgBox "#,##0.0000"
End Select
```

Hidden Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *FieldTemplateObject*.**Hidden** = {True | False}

Description The **Hidden** property lets you find or set whether a data field and its value are displayed in the chart. The **Hidden** property is read/write.

Data Type Integer (Boolean)

Value True hides a field and its value; False displays a field and its value.

Flow Equivalent The **Hidden** property is equivalent to clicking Data Field on the Insert menu, selecting a data field, and then selecting or deselecting the Hidden Field option in the Setup Fields dialog box.

{button Related Topics,PI(``,`IDH_RT_Hidden_Property')}

Changing Data Field Attributes
Example

Add Method (FieldTemplates Collection)

Accumulation Property
AccumulationMethod Property
FieldNamesHidden Property
Format Property
Name Property (FieldTemplate Object)
Type Property (FieldTemplate Object)

FieldTemplate Object

Hidden Property Example

This example uses the **Hidden** property of the FieldTemplate object to find a data field's format.

```
Dim ABC As Object
Dim MasterItems As Object
Dim Chart As Object
Dim Field_Template As Object
Dim Field_Hidden As Integer

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory")    ' Add a data field

Field_Hidden = Field_Template.Hidden                   ' Find value of hidden attribute
Select Case Field_Hidden                               ' Report value of hidden attribute
   Case True
        ABC.MsgBox "Field is Hidden."
   Case Else
        ABC.MsgBox "Field is Visible."
End Select
```

Name Property (FieldTemplate Object)

{button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *FieldTemplateObject*.**Name** = *FieldName*

Description The **Name** property lets you rename a data field or find the name of a data field. A data field's name appears in the chart next to the field's value. You name a field when you create it with the **Add** method of the FieldTemplates Collection. The **Name** property is read/write.

Data Type String

Value The name of the data field

Flow Equivalent The **Name** property is equivalent to clicking Data Field on the Insert menu, selecting a data field, and then changing the name for the field in the Setup Fields dialog box.

{button Related Topics,PI(`',`IDH_RT_Name_Property_FieldTemplate_Object')}

Changing Data Field Attributes
Example

Add Method (FieldTemplates Collection)

Accumulation Property
Format Property
Hidden Property
Name Property (Application object)
Name Property (Chart object)
Name Property (FieldValue object)
Name Property (Font object)
Type Property (FieldTemplate Object)

FieldTemplate Object

Name Property (FieldTemplate Object) and FieldTemplates Property Example

This example uses the **Name** property of the FieldTemplate object and the **FieldTemplates** property of the Chart object to get the name of a data field.

```
Dim ABC As Object, MasterItems As Object, Chart As Object
Dim Field_Template As Object
Dim Field_Name As String

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory")    ' Add a data field

Field_Name = Field_Template.Name                       ' Get the field name
ABC.MsgBox Field_Name
```

Type Property (FieldTemplate Object)

{button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *FieldTemplateObject*.**Type** = *FieldType*

Description The **Type** property of the FieldTemplate object lets you find or set the type of a data field. It is identical with the **Type** property of the FieldValue object. The **Type** property is read/write.

Data Type Integer

Value The **Type** property uses the values shown in the following table.

Value	Description
0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number

Flow Equivalent The **Type** property is equivalent to clicking Data Field on the Insert menu, clicking the arrow to the right of the Field Type text box, and then clicking the type you want for the field.

{button Related Topics,PI(`',`IDH_RT_Type_Property_FieldTemplate_Object')}

Type Property (FieldTemplate Object) Example

This example uses the **Type** property of the FieldTemplate object to get and display the type of a data field.

```
Dim ABC As Object, MasterItems As Object, Chart As Object
Dim Field_Template As Object
Dim Field_Name As String
Dim Field_Type As Single

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory")    ' Add a data field

Field_Type = Field_Template.Type                       ' Get type

Select Case Field_Type                                 ' Display type
   Case 0
      ABC.MsgBox "Field Type = Text"
   Case 1
      ABC.MsgBox "Field Type = Duration"
   Case 2
      ABC.MsgBox "Field Type = Date"
   Case 3
      ABC.MsgBox "Field Type = Currency"
   Case 4
      ABC.MsgBox "Field Type = Percent"
   Case 5
      ABC.MsgBox "Field Type = Number"
End Select
```

FieldTemplates Collection

Description The FieldTemplates collection is below the Chart object. Below the FieldTemplates collection are the FieldTemplate objects. You can have multiple FieldTemplate objects in the collection.

Properties	Methods
Application	Add
Count	DeleteField
Parent	Item

{button Related Topics,PI(``,`IDH_RT_FieldTemplates_Collection')}

Add Method (FieldTemplates Collection) {button Flow Equivalent,JI(`FLOW.HLP>procedur',`IDH_To_set_up_a_data_field');CW(`concfull')}

Usage *FieldTemplatesCollection*.**Add** (*FieldName* [, *FieldType*])
The *FieldName* element is the name of the data field you want to create.
The *FieldType* element is optional and defines the type of the data field.

Description The **Add** method of the FieldTemplates collection lets you create a data field. The field created is added to the FieldTemplates collection. You provide the name for the field and, optionally, the type of field to create. The method returns the data field created.

Data Type Object

Value The **Add** method returns the newly created FieldTemplate object.

The values for the *FieldType* element (the types of data field) are in the following table.

Value	Type
0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number (default if the element is omitted)

Flow Equivalent The **Add** method is equivalent to clicking Data Field on the Insert menu, entering a name for the field in the Setup Fields dialog box, and then selecting a type for the field.

{button Related Topics,PI(`',`IDH_RT_Add_Method_FieldTemplates_Collection')}

Adding Data Fields to a Chart
Example

Add Method (Charts Collection)

AccumulationMethod Property
Format Property
Hidden Property
Name Property (FieldTemplate Object)
Type Property (FieldTemplate Object)

FieldTemplates Collection

Add Method and Count Property (FieldTemplates Collection) Example

This example uses the **Add** method and **Count** property of the FieldTemplates collection to create and count data fields in a chart. The Count properties of other objects and collections work the same way.

```
Dim ABC As Object, Chart As Object
Dim Field_One As Object, Field_Two As Object, Field_Three As Object
Dim Field_Templates_Collection As Object
Dim ABC_Field_Count As Long

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart
Set Field_Templates_Collection = Chart.FieldTemplates

Set Field_One = Field_Templates_Collection.Add("Fred", 1)       ' Create fields with
Set Field_Two = Field_Templates_Collection.Add("Wilma", 2)     ' ("Name", FieldType)
Set Field_Three = Field_Templates_Collection.Add("Barney", 3)

ABC_Field_Count = Field_Templates_Collection.Count   ' Get count of fields in chart

ABC.MsgBox "There are " + CStr(ABC_Field_Count) + " data fields in the chart."
```

DeleteField Method {button Flow Equivalent,JI(`FLOW.HLP>procedure',`IDH_Deleting_Fields');CW(`concfull')}

Usage *FieldTemplatesCollection*.**DeleteField** *FieldTemplateObject*
The *FieldTemplateObject* element is the data field that you want to delete.

Description The **DeleteField** method lets you delete a data field. This removes the data field from every shape in the chart. Any values that were in the field are deleted.

Flow Equivalent The **DeleteField** method is equivalent to clicking Data Field on the Insert menu, selecting a data field in the Field list box, and clicking Delete.

{button Related Topics,PI(`',`IDH_RT_DeleteField_Method')}

DeleteField, Item Methods Example

This example uses the **DeleteField** method and the **Item** method of the FieldTemplates collection to identify and delete a field.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object
Dim Field1 As Object, Field2 As Object, Field3 As Object
Dim UserInput As String, Msg1 As String

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Add a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Set Shape1 = Chart.DrawShape("Operation")            ' Draw a shape
Shape1.Selected = True                               ' Select the shape

' Add fields to the field template ("Name", FieldType)
Set Field1 = Chart.FieldTemplates.Add("1: Fred", 5)
Set Field2 = Chart.FieldTemplates.Add("2: Wilma", 4)
Set Field3 = Chart.FieldTemplates.Add("3: Barney", 3)
ABC.FieldViewerVisible = True                        ' Show the field viewer

' Ask user to input the number of the field to be deleted
Msg1 = "Please Enter the number of the field to be deleted (1, 2, or 3)."
UserInput = InputBox$(Msg1, "Delete Field Box")

' Use Item(index) method to delete the correct field
Select Case UserInput
   Case "1"
      Chart.FieldTemplates.DeleteField Chart.FieldTemplates.Item(1)
      ABC.MsgBox "Field 1 deleted."
   Case "2"
      Chart.FieldTemplates.DeleteField Chart.FieldTemplates.Item(2)
      ABC.MsgBox "Field 2 deleted."
   Case "3"
      Chart.FieldTemplates.DeleteField Chart.FieldTemplates.Item(3)
      ABC.MsgBox "Field 3 deleted."
End Select
```

Item Method (FieldTemplates Collection)

Usage *FieldTemplatesCollection*.**Item** ({*Count* | *FieldName*} [, *FieldType*])
The first element is either a *Count* or a *FieldName*.
The *Count* element is the number of the item within the collection.
The *FieldName* element is the name of the field.
The second element, which is optional, is the type. The *FieldType* element lets you specify the type of element to be returned.

Description Use the **Item** method to access FieldTemplate objects, or data fields, within the FieldTemplates collection.

Data Type Object

Value Returns the next valid FieldTemplate object (data field), in the collection. If that object does not exist, the method returns Null.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Item_Method_FieldTemplates_Collection')}

Working with Data Field Values
Example

Item Method (Charts Collection)
Item Method (FieldValues Collection)
Item Method (Menu Collection)
Item Method (Objects Collection)

Accumulation Property
Value Property

FieldTemplates Collection

FieldValue Object

Description The FieldValue object is below the FieldValues collection. You can have multiple FieldValue objects.

Properties	Methods
Application	Empty
Day	
FieldTemplate	
FormattedValue	
IsEmpty	
Month	
Name	
Parent	
Type	
Value	
Year	

{button Related Topics,PI(`',`IDH_RT_FieldValue_Object')}

FieldTemplate Property

Usage *FieldValueObject*.**FieldTemplate**

Description You use the **FieldTemplate** property to find the **FieldTemplate** object that corresponds to the field value. The **FieldTemplate** property is read only, but the properties from the object it returns are read/write.

Data Type Object

Value The FieldTemplate object

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_FieldTemplate_Property')}

FieldTemplate Property Example

This example uses the **FieldTemplate** property of the FieldValue object to make data fields opaque.

```
Sub Command1_Click ()

    Dim ABC As Object, Chart As Object
    Dim Field1 As Object
    Dim Shape1 As Object
    Dim Field_Template As String

    Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
    ABC.Visible = True                                     ' Make ABC visible
    ABC.New                                                ' Create a new chart
    Set Chart = ABC.ActiveChart                            ' Get the active chart

    Set Field1 = Chart.FieldTemplates.Add("Inventory", 5)  ' Create a field

    Set Shape1 = Chart.DrawShape("Delay")                  ' Draw a Delay shape

    Shape1.FieldValues.Item("Inventory").Value = "300,000"  ' Enter text in the field

' Get the FieldTemplate Object corresponding to this field.
    Field_Template = Shape1.FieldValues.Item("Inventory").FieldTemplate
    ABC.MsgBox Field_Template
End Sub
```

FormattedValue Property

Usage *FieldValueObject*.**FormattedValue**

Description The **FormattedValue** property lets you find the text string that represents the contents of the field. The **FormattedValue** property is read only.

Data Type String

Value The text string that represents the contents of the field

Flow Equivalent None

FormattedValue, FieldTemplates Properties Example

This example uses the **FormattedValue** property of the FieldValue object and the FieldTemplates property of the Chart object to find the formatted value contained in a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object
Dim Field_Formatted_Value As String

Set ABC = CreateObject("ABCFlow.application")              ' Start ABC
ABC.Visible = True                                         ' Make ABC visible
ABC.New                                                    ' Create a new chart
Set Chart = ABC.ActiveChart                                ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory", 5)      ' Create a field

Set Shape1 = Chart.DrawShape("Delay")                      ' Draw a Delay shape

Shape1.FieldValues.Item("Inventory").Value = "300,000"   ' Enter text in the field

Field_Formatted_Value = Shape1.FieldValues.Item("Inventory").FormattedValue
ABC.MsgBox Field_Formatted_Value
```

IsEmpty Property

Usage *FieldValueObject*.**IsEmpty**

Description The **IsEmpty** property lets you find whether a data field contains any values. The **IsEmpty** property is read only.

Data Type Integer (Boolean)

Value True means the data field is empty; False means it contains a value.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_IsEmpty_Property')}

Name Property (FieldValue Object) {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetupDB');CW(`concfull')}

Usage *FieldValueObject*.**Name** = *FieldName*

Description The **Name** property of the FieldValue object lets you find the name of a data field. The field was named when you created it with the **Add** method of the FieldTemplates Collection. The **Name** property is read only.

Data Type String

Value The name of the data field

Flow Equivalent The **Name** property is equivalent to clicking Data Field on the Insert menu, selecting a data field, and then changing the name for the field in the Setup Fields dialog box.

{button Related Topics,PI(``,`IDH_RT_Name_Property_FieldValue_Object')}

Changing Data Field Attributes
Example

Add Method (FieldTemplates Collection)

AccumulationMethod Property
Format Property
Hidden Property
Name Property (Application Object)
Name Property (Chart Object)
Name Property (FieldTemplate Object)
Name Property (FieldValue Object)
Name Property (Font Object)
Type Property (FieldValue Object)

FieldValue Object

Name Property (FieldValue Object) Example

This example uses the **Name** property of the FieldValue object to find the name of a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object
Dim Field_Name As String
Dim Field_Type

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory") ' Create a field
Field1.Type = 0                                    ' Make the field's type text

Set Shape1 = Chart.DrawShape("Delay")              ' Draw a Delay shape

Field_Name = Shape1.FieldValues.Item(1).Name       ' Get the name of the field

ABC.MsgBox Field_Name
```

Type Property (FieldValue Object) {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_SetUpDB');CW(`concfull')}

Usage *FieldValueObject*.**Type** = *FieldType*

Description The **Type** property of the FieldValue object lets you find or set the type of a data field. It is identical with the **Type** property of the FieldTemplate object. The **Type** property is read/write.

Data Type Integer

Value The **Type** property uses the values shown in the following table.

Value	Description
0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number

Flow Equivalent The **Type** method is equivalent to clicking Data Field on the Insert menu, clicking the arrow to the right of the Field Type text box, and clicking the type you want for the field.

{button Related Topics,PI(`',`IDH_RT_Type_Property_FieldValue_Object')}

Changing Data Field Attributes
Example

Type Property (Chart Object)
Type Property (FieldTemplate Object)
Type Property (Line Object)
Type Property (Object Object)

FieldValue Object

Type Property (FieldValue Object) Example

This example uses the **Type** property of the FieldValue object to find the type of a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object
Dim Field_Type As Single

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory", 4)  ' Create a field
ABC.MsgBox "Field created with format 4."

Field1.Type = 0                                        ' Change the field's type to text

Set Shape1 = Chart.DrawShape("Delay")                  ' Draw a Delay shape

Field_Type = Shape1.FieldValues.Item(1).Type           ' Get type of field just created
ABC.MsgBox "The field has changed to format " + CStr(Field_Type) + "."
```

Value Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_ViewerDB');CW(`concfull')}

Usage *FieldValueObject*.**Value** = *Value*

Description The **Value** property lets you find or set the value of a data field item of a shape. The **Value** property is read/write.

Data Type Variant

Value The value of the data field item

Flow Equivalent The **Value** property is equivalent to selecting a shape, opening the Field Viewer, and entering a value for a data field.

{button Related Topics,PI(`',`IDH_RT_Value_Property')}

Working with Data Field Values
Example

AccumulationMethod Property
Item Method (FieldValues Collection)

FieldValue Object

Empty Method {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_Field_ViewerDB');CW(`concfull')}

Usage *FieldValueObject*.**Empty**

Description The **Empty** method lets you remove all values from a data field. After you use the method, the **IsEmpty** property of the FieldValue object is True.

Flow Equivalent The **Empty** method is equivalent to removing the value from a data field using the field viewer.

{button Related Topics,PI(`',`IDH_RT_Empty_Method')}

Empty Method Example

This example uses the **Empty** method of the FieldValue object to remove the value from a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object
Dim Field_Formatted_Value As String

Set ABC = CreateObject("ABCFlow.application")              ' Start ABC
ABC.Visible = True                                         ' Make ABC visible
ABC.New                                                    ' Create a new chart
Set Chart = ABC.ActiveChart                                ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory", 5)      ' Create a field

Set Shape1 = Chart.DrawShape("Delay")                      ' Draw a Delay shape

Shape1.FieldValues.Item("Inventory").Value = "300,000"   ' Enter text in the field
ABC.MsgBox "Formatted field value is " + Shape1.FieldValues("Inventory").FormattedValue + "."

Shape1.FieldValues.Item("Inventory").Empty                 ' Empty the field
ABC.MsgBox "Empty field value is " + Shape1.FieldValues.Item("Inventory").FormattedValue
```

Day Property

Usage *FieldValueObject*.**Day** = *Number*

Description You use the **Day** property to find or set the day of the month in a data field. If the data field does not contain a valid date (for example, if it is not a Date field), then the **Day** property is equal to 0. The **Day** property is read/write.

Data Type Integer

Value A number from 1 to 31

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Day_Property')}

Day, Month, Year Properties Example

This example uses the **Day**, **Month**, and **Year** properties of the FieldValue object to find and display the dates in a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object

Dim Field_Day
Dim Field_Month
Dim Field_Year

Set ABC = CreateObject("ABCFlow.application")                ' Start ABC
ABC.Visible = True                                           ' Make ABC visible
ABC.New                                                       ' Create a new chart
Set Chart = ABC.ActiveChart                                   ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Closing Date", 2)      ' Create a date field

Set Shape1 = Chart.DrawShape("Delay")                        ' Draw a Delay shape

Shape1.FieldValues.Item("Closing Date").Value = "6/28/95"    ' Enter text in the field

Field_Day = Shape1.FieldValues.Item("Closing Date").Day       ' Get the date values
Field_Month = Shape1.FieldValues.Item("Closing Date").Month
Field_Year = Shape1.FieldValues.Item("Closing Date").Year

ABC.MsgBox "The day listed in the Closing Date is " + Field_Day + " ."
ABC.MsgBox "The month listed in the Closing Date is " + Field_Month + " ."
ABC.MsgBox "The year listed in the Closing Date is " + Field_Year + " ."
```

Month Property

Usage *FieldValueObject*.**Month** = *Number*

Description You use the **Month** property to find or set the month in a data field. If the data field does not contain a valid date (for example, if it is not a Date field), then the **Month** property is equal to 0. The **Month** property is read/write.

Data Type Integer

Value A number from 1 to 31

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Month_Property')}

Year Property

Usage *FieldValueObject*.**Year** = *Number*

Description You use the **Year** property to find or set the month in a data field. If the data field does not contain a valid date (for example, if it is not a Date field), then the **Year** property is equal to 0. The **Year** property is read/write.

Data Type Integer

Value A number 1900 or larger

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Year_Property')}

FieldValues Collection

Description The FieldValues collection is below the Object object. Below the FieldValues collection are the FieldValue objects. You can have multiple FieldValue objects in the FieldValues collection.

Properties	Methods
Application	Item
Count	
Parent	

{button Related Topics,PI(`',`IDH_RT_FieldValues_Collection')}

Item Method (FieldValues Collection)

Usage	*FieldValuesCollection*.**Item** ({*Count* | *FieldName*} [, *FieldType*]) The first element is either a *Count* or a *FieldName*. The *Count* element is the number of the item within the collection. The *FieldName* element is the name of the field. The *FieldType* element, which is optional, lets you specify the type of element to be returned.
Description	Use the **Item** method of the FieldValues collection to access FieldValue objects, or data fields, within the FieldValues collection.
Data Type	Object
Value	The next valid FieldValue object, or data field, in the collection. If that object does not exist, the method returns Null.
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Item_Method_FieldValues_Collection')}

Item Method (FieldValues Collection), Value Property, and IsEmpty Property Example

This example uses the **Item** method of the FieldValues collection and the **Value** property and **IsEmpty** property of the FieldValue object to find whether a data field contains a value.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object
Dim Shape1 As Object
Dim Field_Empty As Integer

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory")     ' Create a field

Field1.Type = 0                                        ' Make the field's type text

Set Shape1 = Chart.DrawShape("Delay")                  ' Draw a Delay shape

Shape1.FieldValues.Item("Inventory").Value = "300,000"  ' Enter text in the field

Field_Empty = Shape1.FieldValues.Item("Inventory").IsEmpty
Select Case Field_Empty
    Case True
        ABC.MsgBox "Field is empty."
    Case Else
        ABC.MsgBox "Field is not empty."
End Select
```

Font Object

Description The Font object is below the Object object. You can have only one Font object for each Object object.

Properties	Methods
Application	There are no methods for the
Bold	Font object.
Color	
Italic	
Name	
Opaque	
Parent	
Size	
Strikethrough	
Underline	

{button Related Topics,PI(``,`IDH_RT_Font_Object')}

Bold Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Bold** = {True | False}

Description You use the **Bold** property to change text to be bold, including text in shape numbers using the **NumberFont** property of the Chart object and note text using the **NoteFont** property of the Shape object. The **Bold** property is read/write.

Data Type Integer (Boolean)

Value True means the text is bold; False means the text is not bold.

Flow Equivalent The **Bold** property is equivalent to selecting the text and clicking the Bold button on the Formatting toolbar.

{button Related Topics,PI(``,`IDH_RT_Bold_Property')}

Color Property (Font Object) {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Color** = *Color*

Description You use the **Color** property of the Font object to find or set the color of selected text or the text inside a shape object. The **Color** property affects only the foreground color of the text, not the background color. The **Color** property is read/write.

Data Type Long

Value The color for the text

Flow Equivalent The **Color** property of the Font object is equivalent to selecting text, clicking the Text Color button on the Formatting toolbar, and clicking the color you want.

{button Related Topics,PI(`',`IDH_RT_Color_Property_Font_Object')}

Italic Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Italic** = {True | False}

Description You use the **Italic** property to change text to be italic, including text in shape numbers using the **NumberFont** property of the Chart object and note text using the **NoteFont** property of the Shape object. The **Italic** property is read/write.

Data Type Integer (Boolean)

Value True means the text is *italic*; False means the text is not italic.

Flow Equivalent The **Italic** property is equivalent to selecting the text and clicking the Italic button on the Formatting toolbar.

{button Related Topics,PI(``,`IDH_RT_Italic_Property')}

Name Property (Font Object) {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Name** = *FontName*

Description You use the **Name** property of the Font object to set the typeface name for the font, such as "Arial" or "Roman." The **Name** property is read/write.

Data Type String

Value The typeface name

Flow Equivalent The **Name** property is equivalent to selecting the text, clicking the down arrow to the right of the Font box on the Formatting toolbar, and clicking the font you want.

{button Related Topics,PI(``,`IDH_RT_Name_Property_Font_Object')}

Font Properties (Font Object) Example

This example uses the **Name** property, **Strikethrough** property, **Color** property, **Italic** property, **Bold** property, **Underline** property, and **Size** property of the Font object to set spacing. It uses the **DrawSpacingY** property of the Chart object to set the attributes of text objects.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object, Text2 As Object, Text3 As Object
Dim Text4 As Object, Text5 As Object, Text6 As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the active chart

Chart.DrawDirection = 2                              ' Place new items downward
Chart.DrawSpacingY = .3                              ' Place new items 0.3 apart

Set Text1 = Chart.DrawTextBlock("Imagine a chocolate elephant.")
Text1.Font.Name = "Roman"                            ' Make the font Roman
Set Text2 = Chart.DrawTextBlock("Imagine a chocolate mousse.")
Text2.Font.Strikethrough = True                      ' Strike through this text
Set Text3 = Chart.DrawTextBlock("Four thousand pounds of solid chocolate --")
Text3.Font.Color = ABC.MakeRGB(0, 127, 0)            ' Make the next phrase dark green
Set Text4 = Chart.DrawTextBlock("ten feet high at the shoulder, fifteen feet long,")
Text4.Font.Italic = True                             ' Make the text italic
Set Text5 = Chart.DrawTextBlock("eight feet across -- and it is your job to eat it.")
Text5.Font.Bold = True                               ' Make the text bold
Set Text6 = Chart.DrawTextBlock("Eating the Chocolate Elephant")
Text6.Font.Underline = True                          ' Underline this text item
Text6.Font.Size = 8                                  ' and set the point size to 8
```

Opaque Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Opaque** = {True | False}

Description The Opaque property lets you set or find whether text background is opaque. The Opaque property is read/write.

Data Type Integer (Boolean)

Value True means the text background is opaque; False means it is transparent.

Flow Equivalent The **Opaque** property is equivalent to selecting the text, clicking Font on the Format menu, and then clicking the Opaque box.

{button Related Topics,PI(``,`IDH_RT_Opaque_Property')}

Opaque Property and AttachText Method Example

This example uses the **Opaque** property of the Font object and the **AttachText** method of the Line_ object to make text opaque and attach text to a line.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object, Text1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Shape1 = Chart.DrawShape("Decision")           ' Draw a Decision shape
Set Shape2 = Chart.DrawShape("Operation")          ' Draw an Operation shape
Set Line1 = Chart.DrawLine(Shape1, Shape2)         ' Draw a line connecting the shapes

Set Text1 = Chart.DrawTextBlock("This way!")       ' Create a freeform text object
Text1.Font.Opaque = True                           ' Make the text's background opaque
Line1.Line_.AttachText Text1                       ' Attach the text object to the line
```

Size Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Size** = *FontSize*

Description You use the **Size** property of the Font object to set the typeface size in points. The **Size** property is read/write.

Data Type Long

Value The point size

Flow Equivalent The **Size** property is equivalent to selecting the text and entering the size on the Font size box on the Formatting toolbar.

{button Related Topics,PI(``,`IDH_RT_Size_Property')}

Strikethrough Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Strikethrough** = {True | False}

Description You use the **Strikethrough** property to change text to have a line through it, including text in shape numbers using the **NumberFont** property of the Chart object and note text using the **NoteFont** property of the Shape object. The **Strikethrough** property is read/write.

Data Type Integer (Boolean)

Value True means the text is ~~strikethrough~~; False means the text is not strikethrough.

Flow Equivalent The **Strikethrough** property is equivalent to selecting the text you want to affect, clicking Font on the Format menu, and then clicking the Strikethrough box.

{button Related Topics,PI(``,`IDH_RT_Strikethrough_Property')}

Underline Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Text_Looks');CW(`concfull')}

Usage *FontObject*.**Underline** = {True | False}

Description You use the **Underline** property to underline selected text, including text in shape numbers using the **NumberFont** property of the Chart object and note text using the **NoteFont** property of the Shape object. The **Underline** property is read/write.

Data Type Integer (Boolean)

Value True means the text is underlined; False means the text is not underlined.

Flow Equivalent The **Underline** property is equivalent to selecting the text, clicking Font on the Format menu, and then clicking the Underline box.

{button Related Topics,PI(``,`IDH_RT_Underline_Property')}

Line_ Object

Description The Line_ object is below the Object object. You can have only one Line_ object for each Object object. If the Object object is a shape, this object is a meaningless placeholder.

Properties	Methods
Application	AttachText
Color	ReconnectDest
CrossoverSize	ReconnectSource
CrossoverStyle	Routing
DestArrowColor	
DestArrowSize	
DestArrowStyle	
Destination	
DestinationDirection	
Parent	
Source	
SourceArrowColor	
SourceArrowSize	
SourceArrowStyle	
SourceDirection	
StemColor	
StemStyle	
StemWidth	
Type	

{button Related Topics,PI(`',`IDH_RT_Line_Object')}

Color Property (Line_ Object) {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**Color** = *Color*
Description	You use the **Color** property of the Line_ object to set the color for lines, including the line ends and the stem, or find the stem color of lines. The **Color** property colors the entire line, including the ends. The **Color** property is read/write.
Data Type	Long
Value	The color for a line object
Flow Equivalent	The **Color** property of the Line_ object is equivalent to selecting a line, clicking the Line Color button on the formatting toolbar, and then clicking the color you want.

{button Related Topics,PI(`',`IDH_RT_Color_Property_Line_Object')}

Setting Line Colors
Line Color
Example

BasicColor Method
MakeRGB Method

Color Property (Font Object)
Color Property (Object Object)
DestArrowColor Property
SourceArrowColor Property
StemColor Property

Line_Object

Color, DestArrowStyle, SourceArrowStyle Properties (Line_ Object) Example

This example uses the **Color** property, the **DestArrowStyle** property, and the **SourceArrowStyle** property of the Line_ object to set the color, destination arrow style, and source arrow style for a line.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Shape1 = Chart.DrawShape("Decision")           ' Draw a Decision shape
Set Shape2 = Chart.DrawShape("Operation")          ' Draw an Operation shape
Set Line1 = Chart.DrawLine(Shape1, Shape2)         ' Draw a line connecting the shapes

Line1.Color = ABC.MakeRGB(255, 0, 0)               ' Make the line red
Line1.Line_.DestArrowStyle = 12                    ' Apply a double arrowhead
Line1.Line_.SourceArrowStyle = 4                   ' Apply a source arrow style
```

DestArrowColor Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**DestArrowColor** = *Color*
Description	You use the **DestArrowColor** property to find or set the color of the destination arrow of a line. The **DestArrowColor** property is read/write.
Data Type	Long
Value	The color of the destination arrow of a line
Flow Equivalent	The **DestArrowColor** property is equivalent to selecting a line, clicking Ends on the Format menu, and then selecting the End Color you want.

{button Related Topics,PI(`',`IDH_RT_DestArrowColor_Property')}

Setting Line Colors
Line Color
Example

BasicColor Method
MakeRGB Method

Color Property (Line_Object)
DestArrowSize Property
DestArrowStyle Property
SourceArrowColor Property
StemColor Property

Line_Object

DestArrowSize Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**DestArrowSize** = *Size*
Description	You use the **DestArrowSize** property to find or set the destination arrow size. The **DestArrowSize** property is read/write.
Data Type	Integer
Value	Arrow size can vary from 1 (smallest) to 5 (largest).
Flow Equivalent	The **DestArrowSize** property is equivalent to selecting a line, clicking Ends on the Format menu, and then selecting the End Size you want.

{button Related Topics,PI(`',`IDH_RT_DestArrowSize_Property')}

Line Width
Example

DestArrowColor Property
DestArrowStyle Property
SourceArrowSize Property
StemWidth Property

Line_Object

DestArrowStyle Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**DestArrowStyle** = *StyleNumber*
Description	You use the **DestArrowStyle** property to find or set styles or patterns for line ends, including arrows, circles, and lines. The **DestArrowStyle** property is read/write.
Data Type	Integer
Value	You set the **DestArrowStyle** property to 0 for no arrow. The following illustration shows the values of the **DestArrowStyle** property for each available style. 

Flow Equivalent The **DestArrowStyle** property is equivalent to selecting a line, clicking Ends on the Format menu, and then selecting the End Type you want.

{button Related Topics,PI(`',`IDH_RT_DestArrowStyle_Property')}

End Styles
Example 1
Example 2

DestArrowColor Property
DestArrowStyle Property
SourceArrowStyle Property
StemStyle Property

Line_Object

Destination Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_CONNECTSHAPES');CW(`concfull')}

Usage	*Line_Object*.**Destination** = *Shape*
Description	You use the **Destination** property to connect lines to shapes. The shapes that lines connect to are stored in the **Source** property and **Destination** property. When lines are unconnected, those properties are empty. The **Destination** property is read/write.
Data Type	Object
Value	A shape object that a line is connected to
Flow Equivalent	The **Destination** property is equivalent to dragging the end of a line into a shape.

{button Related Topics,PI(`',`IDH_RT_Destination_Property')}

Connecting Existing Lines to Shapes
Example

ReconnectDest Method
Source Property

Line_Object

Source Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_CONNECTSHAPES');CW(`concfull')}

Usage	*Line_Object*.**Source** = *Shape*
Description	You use the **Source** property to connect lines to shapes. The shapes that lines connect to are stored in the **Source** property and **Destination** property. When lines are unconnected, those properties are empty. The **Source** property is read/write.
Data Type	Object
Value	A shape object that a line is connected to
Flow Equivalent	The **Source** property is equivalent to dragging the end of a line into a shape.

{button Related Topics,PI(`',`IDH_RT_Source_Property')}

Connecting Existing Lines to Shapes Example

ReconnectDest Method
Destination Property

Line_Object

SourceArrowColor Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**SourceArrowColor** = *Color*
Description	You use the **SourceArrowColor** property to find or set the color of the source arrow of a line. The **SourceArrowColor** property is read/write.
Data Type	Long
Value	The color of the source arrow of a line
Flow Equivalent	The **SourceArrowColor** property is equivalent to selecting a line, clicking Ends on the Format menu, and then selecting the Start Color you want.

{button Related Topics,PI(`',`IDH_RT_SourceArrowColor_Property')}

Setting Line Colors
Line Color
Example

BasicColor Method
MakeRGB Method

Color Property (Line_Object)
DestArrowColor Property
SourceArrowSize Property
SourceArrowStyle Property
StemColor Property

Line_Object

SourceArrowSize Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**SourceArrowSize** = *Size*
Description	You use the **SourceArrowSize** property to find or set the line width. The **SourceArrowSize** property is read/write.
Data Type	Integer
Value	Arrow size can vary from 1 (smallest) to 5 (largest).
Flow Equivalent	The **SourceArrowSize** property is equivalent to selecting a line, clicking Ends on the Format menu, and then selecting the Start Size you want.

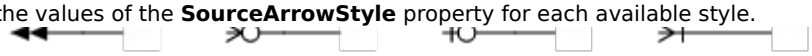
{button Related Topics,PI(`',`IDH_RT_SourceArrowSize_Property')}

Line Width
Example

DestArrowSize Property
SourceArrowColor Property
SourceArrowStyle Property
StemWidth Property

Line_Object

SourceArrowStyle Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**SourceArrowStyle** = *StyleNumber*
Description	You use the **SourceArrowStyle** property to find or set styles or patterns for line ends, including arrows, circles, and lines. The **SourceArrowStyle** property is read/write.
Data Type	Integer
Value	You set the **SourceArrowStyle** property to 0 for no arrow. The following illustration shows the values of the **SourceArrowStyle** property for each available style. 

Flow Equivalent The **SourceArrowStyle** property is equivalent to selecting a line, clicking Ends on the Format menu, and then selecting the Start Type you want.

{button Related Topics,PI(`',`IDH_RT_SourceArrowStyle_Property')}

End Styles
Example 1
Example 2

DestArrowStyle Property
SourceArrowColor Property
SourceArrowSize Property
StemStyle Property

Line_Object

StemColor Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**StemColor** = *Color*
Description	The **StemColor** property lets you find or set the color for the stem of a line (see the MakeRGB method). The stem is the part of the line between the source and destination arrows. The **StemColor** property is read/write.
Data Type	Long
Value	The color for the stem of a line
Flow Equivalent	The **StemColor** property is equivalent to selecting a line, clicking the Line Color button on the formatting toolbar, and then clicking the color you want.

{button Related Topics,PI(`',`IDH_RT_StemColor_Property')}

Setting Line Colors
Line Color
Example

BasicColor Method
MakeRGB Method

Color Property (Line Object)
DestArrowColor Property
SourceArrowColor Property
StemStyle Property
StemWidth Property

Line_Object

Line_ Object Properties Example

This example uses the properties of the Line_ object to set the color, size, and style of the destination arrow, source arrow, and stem of a line.

```
Dim ABC As Object, Chart As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawPositionX = 1                                ' The line's beginning X position
Chart.DrawPositionY = 2                                ' The line's beginning Y position
Set Line1 = Chart.DrawFreeLine(4, 4)                   ' Draw an unconnected line to X=4,Y=4

Line1.Line_.StemColor = ABC.MakeRGB(0, 0, 255)         ' Make line's stem blue
Line1.Line_.StemStyle = 3                              ' Set stem style
Line1.Line_.SourceArrowStyle = 13                      ' Set source arrow style
Line1.Line_.SourceArrowColor = ABC.MakeRGB(255, 0, 0)  ' Make source arrow red
Line1.Line_.SourceArrowSize = 3                        ' Make source arrow medium in size
Line1.Line_.DestArrowColor = ABC.MakeRGB(0, 255, 0)    ' Make destination arrow green
Line1.Line_.DestArrowSize = 5                          ' Make destination arrow large
Line1.Line_.DestArrowStyle = 5                         ' Set destination arrow style
```

StemStyle Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**StemStyle** = *StyleNumber*
Description	You use the **StemStyle** property to find or set styles or patterns for line stems. The **StemStyle** property is read/write.
Data Type	Integer
Value	Set the **StemStyle** property to 0 for a solid line, 1 for an evenly broken line, and 2 for a dashed line. The following illustration shows the values of the **StemStyle** property for each available style.

Flow Equivalent The **StemStyle** property is equivalent to selecting a line, clicking the Line Thickness button on the formatting toolbar, and then clicking the line style you want.

{button Related Topics,PI(`',`IDH_RT_StemStyle_Property')}

Line Style
Example

DestArrowStyle Property
SourceArrowStyle Property
StemColor Property
StemWidth Property

Line_Object

StemWidth Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage	*Line_Object*.**StemWidth** = *Width*
Description	You use the **StemWidth** property to find or set the width of the line stem, excluding the ends. The **StemWidth** property is read/write.
Data Type	Integer
Value	Line width can vary from 1 (hairline) to 5 (thickest).
Flow Equivalent	The **StemWidth** property is equivalent to selecting a line, clicking the Line Thickness button on the formatting toolbar, and then choosing a number in the Width box.

{button Related Topics,PI(`',`IDH_RT_StemWidth_Property')}

Line Width
Example

DestArrowStyle Property
SourceArrowStyle Property
StemColor Property
StemStyle Property

Line_Object

StemWidth, Type Properties Example

This example uses the **StemWidth** property and **Type** property of the Line_ object to make a line wide and determine the type of line drawn.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object
Dim RandomLine As Integer

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape("Decision")               ' Draw a Decision shape
Set Shape2 = Chart.DrawShape("Operation")              ' Draw an Operation shape

RandomNum = Int(10 * Rnd)                              ' Generate a random integer
If RandomNum > 4 Then RandomNum = RandomNum - 5        ' Eliminate numbers > 4
Chart.CurrentLineRouting = RandomNum                   ' Randomly set the line type

Set Line1 = Chart.DrawLine(Shape1, Shape2, 0, 1)       ' Draw a line connecting the shapes
Line1.Line_.StemWidth = 5                              ' Make the line's stem very wide

Select Case Line1.Line_.Type                           ' Display the type of line used
   Case 0
      ABC.MsgBox "This is a direct line."
   Case 1
      ABC.MsgBox "This is a right angle line."
   Case 2
      ABC.MsgBox "This is a curved line."
   Case 3
      ABC.MsgBox "This is an org-chart line."
   Case 4
      ABC.MsgBox "This is a cause/effect line."
End Select
```

Type Property (Line Object)

Usage *Line_Object*.**Type**

Description You use the **Type** property of the Line_ object to find or set which line routing was used to draw a line. The **Type** property is read/write.

Data Type Integer

Value The following table describes the values for the Type property.

Value	Type of Line
0	Direct
1	Right angle
2	Curved
3	Organization chart
4	Cause-and-effect

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_Type_Property_Line_Object')}

AttachText Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage

Line_Object.**AttachText** *TextObject* [, *LineSegment*]
The *TextObject* element is a text block object that was created using the **DrawTextBlock** method. It is the text to attach to the line.
The *LineSegment* element optionally specifies the segment of the line to which to attach the text.

Description

You use the **AttachText** method to attach text to a line. You specify the text object to attach and optionally indicate the segment to which the text should be attached.

Data Type

Integer (Boolean)

Value

The following table describes each possible value for *LineSegment*.

LineSegment	**Description**
-2	End
-3	Start (default)
-1	First
0	Last
1 through *n*	The sequential value of the line segment, where *n* is the number of segments in the line. For example, 1 is the first segment and 2 is the second segment.

Flow Equivalent

The **AttachText** method is equivalent to selecting a text block, dragging it to a line, and snapping it to that line.

{button Related Topics,PI(`',`IDH_RT_AttachText_Method')}

ReconnectDest Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_CONNECTSHAPES');CW(`concfull')}

Usage

Line_Object.**ReconnectDest** *ShapeObject* [, *EnterDirection*]
The *ShapeObject* element is the shape that the line is to connect to.
The *EnterDirection* element, which is optional, specifies the side where the line should enter the shape.

Description

You use the **ReconnectDest** method to connect an existing line to a shape or to change the side where a line enters a shape. You specify the shape that the line enters and, optionally, the side of the shape where the line enters.

Data Type

Integer (Boolean)

Value

True means the reconnection was successful; False means it was not successful. The following table shows the values of the *EnterDirection* element and their meanings.

Value	Direction
0	North
1	East
2	South
3	West

Flow Equivalent

The **ReconnectDest** method is equivalent to dragging a line end into a shape.

{button Related Topics,PI(`',`IDH_RT_ReconnectDest_Method')}

ReconnectDest, ReconnectSource Methods Example

This example uses the **ReconnectDest** method and **ReconnectSource** method of the Line_ object to connect the beginning and end of a line to objects.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawPositionX = 1                                ' The line's beginning X position
Chart.DrawPositionY = 2                                ' The line's beginning Y position
Set Line1 = Chart.DrawFreeLine(4, 4)                   ' Draw an unconnected line to X=4,Y=4

Set Shape1 = Chart.DrawShape("Terminal")               ' Draw a Terminal shape as destination
Set Shape2 = Chart.DrawShape("Connector")              ' Draw a Connector shape as the source
Line1.Line_.ReconnectDest Shape2, 1                    ' Connect end of line to bottom of Terminal
Line1.Line_.ReconnectSource Shape1, 0                  ' Connect beginning of line to top of Connector
```

ReconnectSource Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_CONNECTSHAPES');CW(`concfull')}

Usage

Line_Object.**ReconnectSource** *ShapeObject* [, *ExitDirection*]
The *ShapeObject* element is the shape that the line is to connect to.
The *ExitDirection* element, which is optional, specifies the side where the line leaves the shape.

Description

You use the **ReconnectSource** method to connect an existing line to a shape or to change the side where a line leaves a shape. You specify the shape that the line leaves and, optionally, the side of the shape where the line leaves.

Data Type

Integer (Boolean)

Value

True means the reconnection was successful; False means it was not successful. The following table shows the values of the *ExitDirection* element and their meanings.

Value	Direction
0	North
1	East
2	South
3	West

Flow Equivalent

The **ReconnectSource** method is equivalent to dragging a line end into a shape.

{button Related Topics,PI(`',`IDH_RT_ReconnectSource_Method')}

DestinationDirection Property

Usage *Line_Object*.**DestinationDirection** = *EnterDirection*

Description You use the **DestinationDirection** property of the Line_ object to set or find the side at which a line drawn between two shapes will enter the ending shape. The line enters at the center of the side. The **DestinationDirection** property is read/write.

Data Type Integer

Value The side of the ending shape into which a connecting line will enter. The following table shows the values of the *EnterDirection* element and their meanings.

Value	Direction
0	North
1	East
2	South
3	West

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_DestinationDirection_Property')}

Drawing Lines that Connect Shapes
Example
SourceDirection Property
Line_Object

DestinationDirection, SourceDirection Properties Example

This example uses the **DestinationDirection** method and **SourceDirection** method of the Line_ object to display the destination and source directions of a line connecting two shapes.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim NewLine As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.New                                    ' Make a new chart

Chart.MasterItems.HideAll                              ' Get Master Items out of the way

Chart.DrawPositionX = 1                                ' Draw 2 shapes and connect...
Chart.DrawPositionY = 2                                ' ...them with a line
Set Shape1 = Chart.DrawShape
Chart.DrawPositionX = 4
Set Shape2 = Chart.DrawShape
Set NewLine = Chart.DrawLine(Shape1, Shape2)

ABC.MsgBox "Source Direction: " + NewLine.Line_.SourceDirection
ABC.MsgBox "Destination Direction: " + NewLine.Line_.DestinationDirection
```

SourceDirection Property

Usage *Line_Object*.**SourceDirection** = *ExitDirection*

Description You use the **SourceDirection** property of the Line_ object to set or find the side at which a line drawn between two shapes will leave the starting shape. The line leaves at the center of the side. The **SourceDirection** property is read/write.

Data Type Integer

Value The side of the starting shape from which a connecting line will exit.The following table shows the values of the *ExitDirection* element and their meanings.

Value	Direction
0	North
1	East
2	South
3	West

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_SourceDirection_Property')}

MasterItems Object

Description The MasterItems object is below the Chart object. You can have only one MasterItems object.

Properties	Methods
Application	ShowAll
ChartName	UpdateDateAndTime
ChartNameShown	
Date	
DateShown	
DateStyle	
HideAll	
Logo	
LogoPathname	
LogoShown	
PageNumber	
PageNumberShown	
Parent	
Range	
Text1	
Text1Shown	
Text2	
Text2Shown	
Time	
TimeShown	

{button Related Topics,PI(`',`IDH_RT_MasterItems_Object')}

ChartName Property

Usage	*MasterItemsObject*.**ChartName**
Description	You use the **ChartName** property to find the Chart Name master item object for the chart. The **ChartName** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	The chart object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_ChartName_Property')}

Date Property

Usage	*MasterItemsObject*.**Date**
Description	You use the **Date** property to find or set the Date master item properties. The **Date** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	The Date object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Date_Property')}

Logo Property

Usage	*MasterItemsObject*.**Logo**
Description	You use the **Logo** property to find the Logo master item. You use the **LogoPathname** property to make the logo appear. The **Logo** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	The logo object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Logo_Property')}

PageNumber Property

Usage	*MasterItemsObject*.**PageNumber**
Description	The **PageNumber** property lets you find the page number included in the MasterItems object. The **PageNumber** property is read only, but all the properties from the object it returns are read/write.
Data Type	Object
Value	The page number included in the MasterItems object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_PageNumber_Property')}

Displaying Master Items
Example

MasterItems Property
PageNumberShown Property

MasterItems Object

Text1 Property

Usage	*MasterItemsObject*.**Text1**
Description	You use the **Text1** property to find the Text1 master item. The **Text1** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	The Text1 object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Text1_Property')}

Text1, Text2 Properties Example

This example uses the **Text1** method and **Text2** method of the MasterItems object to put text into the text 1 and text 2 master items.

```
Dim ABC As Object, Chart As Object, MasterItems As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems

MasterItems.Text1.Text = "This is the Text1 field"     ' Put text in text fields
MasterItems.Text2.Text = "This is the Text2 field"
```

Text2 Property

Usage	*MasterItemsObject*.**Text2**
Description	You use the **Text2** property to find the Text2 master item. The **Text2** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	The Text2 object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Text2_Property')}

Time Property

Usage	*MasterItemsObject***.Time**
Description	You use the **Time** property to find the Time master item. The **Time** property is read only, but the properties from the object it returns are read/write.
Data Type	Object
Value	The time object
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Time_Property')}

DateStyle Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**DateStyle** = *Value*

Description You use the **DateStyle** property to find or set the style of the Date master item. The **DateStyle** property is read/write.

Data Type Integer

Value The values for the **DateStyle** property are in the following table.

Value	**Style**
0	MM/DD/YY
1	Short text (Jan. 1, 1995)
2	Long text (January 1, 1995)

Flow Equivalent The **DateStyle** property is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, clicking Date in the list box, and then choosing a style for the Date master item.

{button Related Topics,PI(`',`IDH_RT_DateStyle_Property')}

DateStyle Property Example

This example uses the **DateStyle** property of the MasterItems object to find and report the date style for the Date master item.

```
Dim ABC As Object, Chart As Object, MasterItems As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems

Dim Date_Style As Single
Date_Style = MasterItems.DateStyle                     ' Find date style

Select Case Date_Style                                 ' Report date style
   Case 0
      ABC.MsgBox "Date Style is MM/DD/YY."
   Case 1
      ABC.MsgBox "Date Style is <short text>."
   Case 2
      ABC.MsgBox "Date Style is <long text>."
End Select
```

LogoPathname Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**LogoPathname** = *PathName*

Description You use the **LogoPathname** property to find or set the pathname of the Logo master item. The **LogoPathname** property is read/write. Quotation marks should be used whenever long filenames or long pathnames are used.

Data Type String

Value The pathname of the Logo master item

Flow Equivalent The **LogoPathname** property is equivalent to Chart Properties on the Format menu, clicking the Master Items tab, clicking the Logo item in the Master Items list box, and choosing a file to serve as the logo.

{button Related Topics,PI(`',`IDH_RT_LogoPathname_Property')}

LogoPathname, Range Properties Example

This example uses the **LogoPathname** property and **Range** property of the MasterItems object to find and report the date style for the master item date. The example assumes that there is a logo selection in the chart.

```
Dim ABC As Object, Chart As Object, MasterItems As Object
Dim Logo_Path_Name As String
Dim Range As Single

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Logo_Path_Name = MasterItems.LogoPathname              ' Get path to master items logo

ABC.MsgBox Logo_Path_Name                              ' Display path to master items logo

Range = MasterItems.Range                              ' Get master item page range
Select Case Range
   Case 0
      ABC.MsgBox "Master items are only on first page."
   Case 1
      ABC.MsgBox "Master items are on all pages."
End Select
```

Range Property

{button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**Range** = *RangeIndicator*

Description You use the **Range** property to find or set the range of pages that display the master items. The **Range** property is read/write.

Data Type Integer

Value The range of pages on which master items are shown, using the values in the following table.

RangeIndicator	**Pages**
0	First page only
1	All pages

Flow Equivalent The **Range** method is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then clicking All Pages or First Page Only.

{button Related Topics,PI(`',`IDH_RT_Range_Property')}

ChartNameShown Property

Usage	*MasterItemsObject*.**ChartNameShown** = {True | False}
Description	You use the **ChartNameShown** property to find or set whether the ChartName master item is displayed. The **ChartNameShown** property is read/write.
Data Type	Integer (Boolean)
Value	True shows the chart name master item; False does not.
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_ChartNameShown_Property')}

ChartNameShown, ChartName, MasterItems Properties Example

This example uses the **ChartNameShown** property and **ChartName** property of the MasterItems object and the **MasterItems** property of the Chart object to determine if the chart name master item is shown.

```
Dim ABC As Object, Chart As Object
Dim Master_Items As Object
Dim Chart_Name_Visible As Integer                    ' For ChartNameShown property value

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart
Set Master_Items = Chart.MasterItems

Chart_Name_Visible = Master_Items.ChartNameShown     ' Get ChartNameShown property value

Select Case Chart_Name_Visible                       ' Display return results
   Case True
      ABC.MsgBox "Chart name is visible. It is " + Master_Items.ChartName.Text + "."
   Case Else
      ABC.MsgBox "Chart name is not visible."
End Select
```

DateShown Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**DateShown** = {True | False}

Description You use the **DateShown** property to find or set whether the Date master item is displayed. The **DateShown** property is read/write.

Data Type Integer (Boolean)

Value True shows the date master item; False does not.

Flow Equivalent The **DateShown** property is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then selecting or deselecting the Date option in the list box.

{button Related Topics,PI(`',`IDH_RT_DateShown_Property')}

DateShown, Date Properties Example

This example uses the **DateShown** property and the **Date** property of the MasterItems object to determine if the Date master item is shown and display it if it is.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Date_Visible As Integer                    ' For DateShown property return value

Set ABC = CreateObject("ABCFlow.application")  ' Start ABC
ABC.Visible = True                             ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart
Set Master_Items = Chart.MasterItems

Date_Visible = Master_Items.DateShown          ' Get DateShown property

Select Case Date_Visible                       ' Display return results
   Case True
      ABC.MsgBox "Date is visible. It is " + Master_Items.Date.Text + "."
   Case Else
      ABC.MsgBox "Date is not visible."
End Select
```

LogoShown Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**LogoShown** = {True | False}

Description You use the **LogoShown** property to find or set whether the Logo master item is displayed. The **LogoShown** property is read/write.

Data Type Integer (Boolean)

Value True shows the logo; False does not.

Flow Equivalent The **LogoShown** property is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then selecting or deselecting the Show Logo option in the Master Items list box.

{button Related Topics,PI(`',`IDH_RT_LogoShown_Property')}

LogoShown, Logo Properties Example

This example uses the **LogoShown** property and the **Logo** property of the MasterItems object to determine if the logo master item is shown and display its width. For the width of the logo to be shown, there must be a logo in the chart.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Logo_Visible As Integer                    ' For LogoShown property return value

Set ABC = CreateObject("ABCFlow.application")  ' Start ABC
ABC.Visible = True                             ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart
Set Master_Items = Chart.MasterItems

Logo_Visible = Master_Items.LogoShown          ' Get LogoShown property

Select Case Logo_Visible                       ' Display return results
   Case True
      ABC.MsgBox "Logo is visible. Its width is " + Master_Items.Logo.Width + "."
   Case Else
      ABC.MsgBox "Logo is not visible."
End Select
```

PageNumberShown Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage	*MasterItemsObject*.**PageNumberShown** = {True | False}
Description	You use the **PageNumberShown** property to find or set whether the PageNumber master item is displayed. The **PageNumberShown** property is read/write.
Data Type	Integer (Boolean)
Value	True shows the page number master item; False does not.
Flow Equivalent	The **PageNumberShown** property is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then selecting or deselecting the Page Number option.

{button Related Topics,PI(`',`IDH_RT_PageNumberShown_Property')}

PageNumberShown, PageNumber Properties Example

This example uses the **PageNumberShown** property and the **PageNumber** property of the MasterItems object to determine if the Page number master item is shown. It then shows the text of the page number.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Page_Number_Visible As Integer                    ' For PageNumberShown property value

Set ABC = CreateObject("ABCFlow.application")         ' Start ABC
ABC.Visible = True                                    ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart
Set Master_Items = Chart.MasterItems

Page_Number_Visible = Master_Items.PageNumberShown    ' Get PageNumberShown property

Select Case Page_Number_Visible                       ' Display return results
   Case True
      ABC.MsgBox "Page Number visible. Format is " + Master_Items.PageNumber.Text + "."
   Case Else
      ABC.MsgBox "Page Number is not visible."
End Select
```

TimeShown Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage	*MasterItemsObject*.**TimeShown** = {True | False}
Description	You use the **TimeShown** property to find or set whether the Time master item is shown. The **TimeShown** property is read/write.
Data Type	Integer (Boolean)
Value	True shows the Time master item; False does not show it.
Flow Equivalent	The **TimeShown** property is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then selecting or deselecting the Time option in the list box.

{button Related Topics,PI(`',`IDH_RT_TimeShown_Property')}

TimeShown, Time Properties Example

This example uses the **TimeShown** property and the **Time** property of the MasterItems object to determine if the time master item is shown. If it is shown, the program gives its value.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Time_Visible As Integer                    ' For TimeShown property return value

Set ABC = CreateObject("ABCFlow.application")  ' Start ABC
ABC.Visible = True                             ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart
Set Master_Items = Chart.MasterItems

Time_Visible = Master_Items.TimeShown          ' Get TimeShown property

Select Case Time_Visible                       ' Display result
    Case True
        ABC.MsgBox "Time is visible. It is " + Master_Items.Time.Text + "."
    Case Else
        ABC.MsgBox "Time is not visible."
End Select
```

Text1Shown Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage	*MasterItemsObject*.**Text1Shown** = {True | False}
Description	You use the **Text1Shown** property to find or set whether the Text1 master item is displayed. The **Text1Shown** property is read/write.
Data Type	Integer (Boolean)
Value	True shows the Text1 master item; False does not show it.
Flow Equivalent	The **Text1Shown** property is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then selecting or deselecting the Text 1 option in the list box.

{button Related Topics,PI(`',`IDH_RT_Text1Shown_Property')}

Text1Shown, Text2Shown Properties Example

This example uses the **Text1Shown** property and **Text2Shown** property of the MasterItems object to determine whether the text 1 and text 2 master items are shown.

```
Dim ABC As Object, Chart As Object
Dim Text1_Visible, Text2_Visible

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Text1_Visible = Chart.MasterItems.Text1Shown           ' Determine whether Text1 is shown
Text2_Visible = Chart.MasterItems.Text2Shown           ' Determine whether Text2 is shown

Select Case Text1_Visible                              ' Display results for Text1 field
    Case True
        ABC.MsgBox "Text1 field is visible."
    Case Else
        ABC.MsgBox "Text1 field is not visible."
End Select

Select Case Text2_Visible                              ' Display results for Text2 field
    Case True
        ABC.MsgBox "Text2 field is visible."
    Case Else
        ABC.MsgBox "Text2 field is not visible."
End Select
```

Text2Shown Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage	*MasterItemsObject*.**Text2Shown** = {True | False}
Description	You use the **Text2Shown** property to find or set whether the Text2 master item is displayed. The **Text2Shown** property is read/write.
Data Type	Integer (Boolean)
Value	True shows the Text2 master item; False does not show it.
Flow Equivalent	The **Text2Shown** property is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then selecting or deselecting the Text 2 option in the list box.

{button Related Topics,PI(`',`IDH_RT_Text2Shown_Property')}

HideAll Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**HideAll**

Description You use the **HideAll** method to hide the master items in the chart.

Flow Equivalent The **HideAll** method is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then deselecting all the master items options.

{button Related Topics,PI(`',`IDH_RT_HideAll_Method')}

HideAll Method Example

This example uses the **HideAll** method of the MasterItems object to hide all master items.

```
Dim ABC As Object, MasterItems As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems

MasterItems.HideAll                                    ' Hide all master items in the chart
```

ShowAll Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**ShowAll**

Description You use the **ShowAll** method to display the master items in the chart.

Flow Equivalent The **ShowAll** method is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, and then selecting all the master items options.

{button Related Topics,PI(`',`IDH_RT_ShowAll_Method')}

ShowAll Method Example

This example uses the **ShowAll** method of the MasterItems object to show all master items.

```
Dim ABC As Object, MasterItems As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems

MasterItems.ShowAll                                    ' Show all master items in the chart
```

UpdateDateAndTime Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Master_Items');CW(`concfull')}

Usage *MasterItemsObject*.**UpdateDateAndTime** [*Date*] [, *Time*]
The *Date* element, which is optional, specifies a specific date.
The *Time* element, which is optional, specifies a specific time.

Description You use the **UpdateDateAndTime** method to update the master item time and date. If you omit the elements, the data and time are changed to the system date and time. You can optionally supply a date and a time.

Data Type The *Date* element and *Time* element are strings.

Value None

Flow Equivalent The **UpdateDateAndTime** method is equivalent to clicking Chart Properties on the Format menu, clicking the Master Items tab, selecting the Date or Time option, and then clicking the Update Date and Time option.

{button Related Topics,PI(`',`IDH_RT_UpdateDateAndTime_Method')}

UpdateDateAndTime Method Example

This example uses the **UpdateDateAndTime** method of the MasterItems object to update the date and time master items to the current system date and time.

```
Dim ABC As Object
Dim Chart As Object
Dim MasterItems As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set MasterItems = Chart.MasterItems

MasterItems.UpdateDateAndTime                          ' Update current Master Items date and time
settings

ABC.MsgBox "You've just updated your Master Item time and date settings."
```

Menu Collection

Description The Menu collection is below the Application object. Below the Menu collection are the MenuItem objects. You can have multiple MenuItem objects in the Menu collection.

Properties	Methods
Application	AppendItem
Count	DeleteItem
Parent	DeleteAll
Text	InsertItem
Visible	Item

{button Related Topics,PI(`',`IDH_RT_Menu_Collection')}

Text Property (Menu Collection)

Usage	*MenuCollection*.**Text** = *MenuName*
Description	The **Text** property of the Menu collection lets you change the name of a menu after you have created it. You may include the "&" character for keyboard shortcuts. The **Text** property is read/write.
Data Type	String
Value	The text of the menu
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Text_Property_Menu_Collection')}

Text Property (Menu Collection) Example

This example uses the **Text** property of the Menu collection to change the name of a menu after it is created.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")                    ' Start ABC
ABC.Visible = True   ' Make ABC visible
ABC.New
Set Menu = ABC.AddMenu("Statistics", ABC1, Form1.Caption)            ' Add a new menu item

ABC.MsgBox "Click OK to see the menu text change."

Menu.Text = "Organization"                                       ' Change the new menu's text
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

Visible Property (Menu Collection)

Usage	*MenuCollection*.**Visible** = {True | False}
Description	The **Visible** property of the Menu collection lets you show or hide a menu. The **Visible** property is read/write.
Data Type	Integer (Boolean)
Value	True makes the menu visible; False hides it.
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Visible_Property_Menu_Collection')}

Visible Property (Menu Collection) Example

This example uses the **Visible** property of the Menu collection to hide and reshow a menu.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")       ' Start ABC
ABC.Visible = True                                  ' Make ABC visible
ABC.New

Set Menu = ABC.AddMenu("Statistics", ABC1, Form1.Caption)  ' Create a new menu

ABC.MsgBox "Click OK to hide the new menu."

Menu.Visible = False                          ' Hide the new menu

ABC.MsgBox "Click OK to see the new menu reappear"

Menu.Visible = True                           ' Unhide the new menu
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

AppendItem Method

Usage	*MenuCollection*.**AppendItem** (*ItemName*) The *ItemName* element is the name of the item you wish to add to the menu.
Description	The **AppendItem** method lets you add a menu item to the next position, below any existing items, in a menu you created. It is customary to list items within groups in alphabetical order. If you use the name of an existing menu item, the method returns the existing MenuItem object. Otherwise it returns the new MenuItem object.
Data Type	Object
Value	The menu item you created
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_AppendItem_Method')}

DeleteItem Method

Usage *MenuCollection*.**DeleteItem** *MenuItem*
The *MenuItem* element is the MenuItem object to remove.

Description The **DeleteItem** method lets you delete a menu item from a menu.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_DeleteItem_Method')}

DeleteItem Method Example

This example uses the **DeleteItem** method of the Menu collection to delete a menu item.

```
Dim ABC As Object, Menu As Object, MenuItem As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart

Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)    ' Create the main menu

Set MenuItem = Menu.AppendItem("First Item")           ' Add items to the new menu
Menu.AppendItem ("Second Item")

ABC.MsgBox "Click on the ABC application to see the new menu items."

ABC.MsgBox "Click OK to delete a menu item."

Menu.DeleteItem MenuItem                               ' Delete a menu item
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

DeleteAll Method

Usage *MenuCollection*.**DeleteAll**

Description The **DeleteAll** method lets you remove all items from a menu.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_DeleteAll_Method')}

DeleteAll Method Example

This example uses the **DeleteAll** method of the Menu collection to remove all menu items in a menu.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart

Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)    ' Create new menu

Menu.AppendItem ("First Item")                         ' Create first menu item
Menu.AppendItem ("Second Item")                        ' Create second menu item

ABC.MsgBox "Click OK to delete all of the new menu items."

Menu.DeleteAll                                 ' Delete all menu items under new menu
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

InsertItem Method

Usage	*MenuCollection*.**InsertItem** *ItemName*, ({*PreviousItem* | *Position*}) The *ItemName* element is the name of the item you wish to add to the menu. The *PreviousItem* element is the item to position the new item after. The *Position* element is the numeric position of the new item.
Description	The **InsertItem** method lets you insert a menu item in a specified position in a menu you created. It is customary to list items within groups in alphabetical order. You provide the title of the item you wish to create, followed by the position of the item, specified either by giving the name of the existing item that the new item should be placed after or by specifying the numerical position of the item.
Data Type	Object
Value	The menu item you created
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_InsertItem_Method')}

InsertItem Method Example

This example uses the **InsertItem** method of the Menu collection to insert a menu item between two existing items.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart

Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)    ' Create a new menu

Menu.AppendItem ("First Item")                         ' Create first menu item
Menu.AppendItem ("Second Item")                        ' Create second menu item

Menu.InsertItem ("Third Item", 2)                ' Insert third item between first two
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

Item Method (Menu Collection)

Usage	*MenuCollection*.**Item** ({*ItemText* | *Position*}) The *ItemText* element is the text of the item you want to find. The *Position* element is the numeric position of the item in the menu.
Description	The **Item** method of the Menu collection lets you find a menu item either by its text or by its location in a menu.
Data Type	Object
Value	A menu item
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Item_Method_Menu_Collection')}

Item Method (Menu Collection) Example

This example uses the **Item** method of the Menu collection to display the names of the items in a menu

```
Dim ABC As Object, Menu As Object, z As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True   ' Make ABC visible
ABC.New
Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)    ' Add a new menu

Menu.AppendItem ("First Item")                         ' Append items to the new menu
Menu.AppendItem ("Second Item")

If Menu.Count <> 0 Then                                ' Start a loop
    For x = 1 To Menu.Count                            ' Get the Menu count
        Set z = Menu.Item(x)                           ' Get the current menu item

ABC.MsgBox z                                           ' Display the name of the menu item
     Next x
End If   ' End the loop
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

MenuItem Object

Description The MenuItem object is below the Menu collection. You can have multiple MenuItem objects.

Properties	Methods
Application	There are no methods for the
Checked	MenuItem object.
Enabled	
Parent	
Text	

{button Related Topics,PI(``,`IDH_RT_MenuItem_Object')}

Checked Property

Usage *MenuItemObject*.**Checked** = {True | False}

Description The **Checked** property lets you show or hide a check mark beside a menu item. The **Checked** property is read/write.

Data Type Integer (Boolean)

Value True shows a check mark beside the item; False hides it.

Flow Equivalent None

{button Related Topics,PI(`',`IDH_RT_Checked_Property')}

Adding Menus
Example

AddMenu Method
DeleteAll Method
DeleteItem Method
RemoveMenu Method

Enabled Property
Text Property (MenuItem Object)

MenuItem Object

Checked Property Example

This example uses the **Checked** method of the MenuItem object to put a check mark beside a menu item.

```
Dim ABC As Object, Menu As Object, MenuItem As Object, First As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)        ' Create a new menu

Set MenuItem = Menu.AppendItem("First Item")         ' Create menu items
Menu.AppendItem ("Second Item")

MenuItem.Checked = True                              ' Place a check on the first menu item
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

Enabled Property

Usage *MenuItemObject*.**Enabled** = {True | False}

Description The **Enabled** property lets you show a menu item or make it gray. The **Enabled** property is read/write.

Data Type Integer (Boolean)

Value True enables the item; False grays it.

ABC Equivalent None

{button Related Topics,PI(``,`IDH_RT_Enabled_Property')}

Enabled Property Example

This example uses the **Enabled** property of the MenuItem object to gray a menu item.

```
Dim ABC As Object, Menu As Object, MenuItem As Object, First As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

Set Menu = ABC.AddMenu("Test", ABC1, Form1.Caption)        ' Create a new menu

Set MenuItem = Menu.AppendItem("First Item")         ' Create first menu item
Menu.AppendItem ("Second Item")                      ' Create second menu item

MenuItem.Enabled = False                             ' Gray the first menu item
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

Text Property (MenuItem Object)

Usage *MenuItemObject*.**Text** = *ItemName*

Description The **Text** property of the MenuItem object lets you change the name of a menu item after you have added it to a menu. You may include the "&" character for keyboard shortcuts. The **Text** property is read/write.

Data Type String

Value The text of the menu item

Flow Equivalent None

Text Property (MenuItem Object) Example

This example uses the **Text** property of the MenuItem object to change the text in an item in a menu.

```
Dim ABC As Object, menu As Object, MenuItem As Object, First As Object

Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart

Set menu = ABC.AddMenu("Test", ABC1, Form1.Caption)        ' Create a new menu

Set MenuItem = menu.AppendItem("1st Item")           ' Create menu items
menu.AppendItem ("Second Item")

ABC.MsgBox "Press ALT+TAB to switch to ABC. Click the Test menu. Notice that the first menu item is named '1st Item'. ALT+TAB back to this dialog box and click OK. The first menu item will change to 'First Item.'"

MenuItem.Text = "First Item"                         ' Change "1st Item" to "First Item"
```

Note

The AddMenu method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

Introducing OLE Automation

OLE Automation is a powerful tool you can use to customize FlowCharter to meet your own specific needs. The extensive power and flexibility of Automation give you endless control over FlowCharter.

OLE Automation can provide seamless integration with outside applications. You can write automation programs that use FlowCharter information to perform tasks in other applications or use data from other applications to create and manipulate FlowCharter charts.

OLE Automation is one of the programs in the FlowCharter package. Together, they provide you with easy, efficient, and powerful Office-compatible tools.

The Help system is designed to let you move back and forth between OLE Automation Help and FlowCharter Help.

{button Related Topics,PI(`',`IDH_RT_Introduction_Chapter')}

Using the Help System

There are several ways you can access the OLE Automation help file.

- You can run the file from FlowCharter Help.
- You can run it from the Explorer.
- You can run it from Excel (which also lets you browse through the language elements and see quick descriptions of them).

{button Related Topics,PI(``,`IDH_RT_Using_the_Help_System')}

Accessing OLE Automation Help from FlowCharter Help

{button OLE Automation,}	This help for OLE Automation is linked to the FlowCharter help. To the right of selected buttons, commands, dialog boxes, and areas of dialog boxes are buttons labeled "Automation." If you click one of those buttons, you go to the OLE Automation help for a related property or method. Be sure to check the Related Topics in the OLE Automation help to find other properties or methods that are related to the FlowCharter area you are working in.
{button FLow Equivalent,}	When a property or method has an equivalent FlowCharter command, a button labeled "Flow Equivalent" appears to the right of the topic title. You can click on the button to go to the FlowCharter topic that includes a description of the related command, button, option, or other FlowCharter element.

{button Related Topics,PI(`',`IDH_RT_Accessing_from_this_Help')}

Accessing OLE Automation Help from Visual Basic
Jumping to Visual Basic Help
Using the Help System

To access OLE Automation help from FlowCharter help
To access OLE Automation Help from Excel
To access OLE Automation Help from Explorer

To access OLE Automation Help from FlowCharter help

1 Click the OLE Automation button immediately to the right of the command, option, or area that you want to know the OLE Automation equivalent for.

For example, if you want to know the equivalent for the changing the color of a shape border, go to the topic Determining How Shapes Look. Find the section about Border Color and click the OLE Automation button. The OLE Automation help file runs, showing the topic on the equivalent OLE Automation property or method.

OLE Automation button

2 To return to the FlowCharter help from the OLE Automation help, click the Flow Equivalent button that appears at the right of the topic title.

Flow Equivalent button

{button Related Topics,PI(`',`IDH_RT_To_access_ABC_OLE_Automation')}

To access OLE Automation Help from Excel
To access OLE Automation Help from Explorer

To access OLE Automation Help from Explorer

1 Open Explorer.

2 Locate the folder where FlowCharter is installed.

3 Double click the file AUTOMATE.HLP. The OLE Automation help file runs.

{button Related Topics,PI(`',`IDH_RT_Accessing_Using_File_Manager')}

Accessing OLE Automation Help from Visual Basic
Jumping to Visual Basic Help
Using the Help System

To access OLE Automation help from FlowCharter help
To access OLE Automation Help from Excel
To access OLE Automation Help from Explorer

To access OLE Automation Help from Excel

1. Run Excel.
2. Open the Insert menu and choose the Macro command. A submenu opens.
3. Choose Module. The module interface displays.
4. Open the Tools menu and choose the References command. The References dialog box opens.
5. Click OLE Automation 2.0 until an X appears in the box in front of it.
6. Click OK to close the dialog box.
7. Open the View menu and choose the Object Browser command. The Object Browser dialog box opens.
8. Click the down arrow to the right of the Libraries/Workbooks text box. A list of available libraries and workbooks appears.
9. Click FlowCharter to select it. The Objects/Modules list box shows the objects available in FlowCharter OLE Automation.
10. Click the object you want information about in the Objects/Modules list box. A short explanation appears at the bottom of the dialog box.
11. Click the button with a question mark in it, if you wish. OLE Automation help appears showing the topic about that object.
12. Click the button with a question mark in it. OLE Automation help appears showing the topic about that method or property.

Note

- If the button with a question mark in it is gray, then the help file is not installed. You must install the help file using the Installation program.

{button Related Topics,PI(`',`IDH_RT_Accessing_Using_Excel_5_0')}

Jumping to Visual Basic Help {button Visual Basic Help,JumpContents(`VB.HLP')}

You can jump to the Visual Basic help by clicking the button above. However, the jump will succeed only if Visual Basic Help is where Windows can find it. You also can access Visual Basic help by clicking on VB.HLP.

{button Related Topics,PI(``,`IDH_RT_Jumping_to_Visual_Basic')}

Running and Viewing the Sample Files

FlowCharter provides automation samples for both Visual Basic and C++. You can use these samples to get creative ideas for uses of automation, and then examine the code to see how it was done. Each program contains explanatory comments and can be copied or edited to suit your own needs.

OLE Automation includes the following samples:

- Deployment Wizard
- Double Click and Delete Events Demo
- Double Click Line Draw
- Excel Data Sample
- Field Change Notify
- Menus Sampler
- Move Event Demo
- Organizational Chart Generator
- VC++/MFC Events Sample
- Network Database
- Text on Lines

To use OLE Automation and to view the code in the samples, you must install Visual Basic 3.0 or later or Visual C++ 1.5 or later. To view the code for the Visual Basic examples, you must install the FlowCharter VB event handler in Visual Basic.

To install the FlowCharter VB event handler

1 If necessary, install Visual Basic® 3.0.

2 Run Visual Basic.

3 From the Visual Basic menu, on the File menu, click Add File. The Add File dialog box opens.

4 Switch to the Windows\System directory in the dialog box and select the file ABCAUTO.VBX to install the FlowCharter VB event handler.

To install the OCX event handler

1 If necessary, install Visual Basic® 4.0 or later.

2 Run Visual Basic.

3 From the Visual Basic menu, on the Tools menu, click Custom Controls. The Custom Controls dialog box opens.

4 Select Micrografx FlowCharter Control and click OK. The FlowCharter Control icon is added to the controls toolbar.

5 To make the control available to your program, double click on it.

To run an automation sample from Explorer

1 Click Run on the Start menu.

2 Type Program Files\Micrografx\FlowCharter\Autosamp and the name of one of the Automation samples: Deploy.exe, Events.exe, Excel.exe, Field.exe, Linedraw.exe, Menu.exe, Move.exe, Orgchart.exe, Network.exe, or T_Online.exe.

Notes

- OLE_VBX, the C++ sample is in Program Files\Micrografx\FlowCharter\Autosamp\Ole_vbx.
- You also can double click the icon in Explorer in the Program Files\Micrografx\FlowCharter\ Autosamp folder.

To run an automation sample from Visual Basic

1 Run Visual Basic.

2 From the File menu, choose Open Project.

3 In the Open Project dialog box, switch to the Program Files\Micrografx\FlowCharter\Autosamp folder and choose one of the OLE Automation samples: DEPLOY.MAK, EVENTS.MAK, EXCEL.MAK, FIELD.MAK, LINEDRAW.MAK, MENU.MAK, MOVE.MAK, ORGCHART.MAK, NETWORK.MAK, or T_ONLINE.MAK. The project window will open.

4 From the Visual Basic Run menu, choose Start.

To view the code in one of the Visual Basic sample files

1 Run Visual Basic.

2 From the File menu, choose Open Project.

3 In the Open Project dialog box, switch to the Program Files\Micrografx\FlowCharter\Autosamp directory and choose one of the OLE Automation samples: DEPLOY.MAK, EVENTS.MAK, EXCEL.MAK, FIELD.MAK, LINEDRAW.MAK, MENU.MAK, MOVE.MAK, ORGCHART.MAK, NETWORK.MAK, or T_ONLINE.MAK. The project window will open.

4 From the Visual Basic View menu, choose Code.

or

Click the View Code button in the project window.

or

Double click on the .FRM name of project in the project window, and then double click on the button in the form that opens.

Note

- If FlowCharter is not running when you run one of the samples from Program Manager or the FlowCharter window, FlowCharter will be started automatically. When you exit FlowCharter, any samples that are not running will be closed automatically.

{button Related Topics,PI(``,`IDH_RT_Running_and_Viewing_the_Sample_Files')}

Introducing OLE Automation
Jumping to Visual Basic Help

Deployment Wizard
Double Click and Delete Events Demo
Double Click Line Draw Sample
Excel Data Sample
Field Change Notify Sample

Menus Sampler
Move Event Demo
Organizational Chart Generator
VC++/MFC Events Sample
Network Database Sample
Text on Lines Sample

Deployment Wizard

The Deployment Wizard is a Visual Basic program that helps make deployment charts. Use the mouse to select the departments and phases you want on the chart. You can add and delete items from the Departments and Phases lists. Any settings you make are saved in DEPLOY.INI and restored when the program is run again.

After the chart is generated, try moving and resizing the boxes that list the Departments and Phases. They have special snapping behavior that is driven by event handling in the Deployment Wizard. (The events **ObjectSizedNOTIFY**, **ObjectSizeSUBCLASS**, and so forth, trigger the Visual Basic snapping code.)

{button Related Topics,PI(``,`IDH_RT_Deployment_Wizard')}

Double Click and Delete Events Demo

The Double Click and Delete Events demo alters the double click and **DEL** key behavior. When you double click a shape, it turns red and the text "You double-clicked on me!" appears in the shape. When you select a shape and press the **DEL** key, it remains on the page and its fill color changes to gray, instead of being deleted.

{button Related Topics,PI(`',`IDH_RT_Double_Click_and_Delete_Events_Demo')}

Running and Viewing the Sample Files

Double Click Line Draw Sample

The Double Click Line Draw sample draws a line between two shapes after you double click each shape.

{button Related Topics,PI(``,`IDH_RT_Double_Click_Line_Draw')}

Running and Viewing the Sample Files

Excel Data Sample

The Excel Data sample reads an Excel data file and uses the data to generate field values in a flowchart.

{button Related Topics,PI(`',`IDH_RT_Excel_5_0_Data_Sample')}

Running and Viewing the Sample Files

Field Change Notify Sample

The Field Change Notify sample displays a message box when a field is changed in the field viewer. The message box displays the name of the changed field and its contents.

{button Related Topics,PI(``,`IDH_RT_Field_Change_Notify')}

Running and Viewing the Sample Files

Menus Sample

The Menus sample adds a menu to FlowCharter called "Stats." This menu has two items that count the objects in the chart. When MENU.EXE shuts down, FlowCharter automatically removes the "Stats" menu.

{button Related Topics,PI(`',`IDH_RT_Menus_Sampler')}

Running and Viewing the Sample Files

Move Event Demo

The Move Event demo causes a single shape to turn yellow if it is moved. If you move more than one object, the moved objects turn green. Additionally, fields for the X and Y positions are maintained below each moved shape.

{button Related Topics,PI(`',`IDH_RT_Move_Event_Demo')}

Running and Viewing the Sample Files

Organizational Chart Generator

The Organizational Chart Generator is a Visual Basic program that makes an ORG chart from a text file. The text file uses tabs to indicate the levels in the organization. Two TXT files (ORGCHRT1.TXT and ORGCHRT2.TXT) are installed in the Program Files\Micrografx\FlowCharter\Autosamp directory that you can edit and use to generate organizational charts.

{button Related Topics,PI(`',`IDH_RT_Organizational_Chart_Generator')}

Running and Viewing the Sample Files

VC++/MFC Events Sample

The VC++/MFC Events sample is written in C++. The sample alters the double click and **DEL** key behavior. It turns an object green when it is double clicked. If the object is a shape, the shape's text changes to "C++ is easy!" When you select objects and press the **DEL** key, they remain on the page and turn gray, instead of being deleted.

{button Related Topics,PI(`',`IDH_RT_VC_MFC_Events_Sample')}

Running and Viewing the Sample Files

Network Database Sample

Network Database is a Visual Basic program that creates a visual image of the connections of a computer network. After the chart is created, you can double click on a shape for information on that node.

{button Related Topics,PI(`',`IDH_RT_Network_Database')}

Running and Viewing the Sample Files

Text on Lines Sample

Text on Lines is a Visual Basic program that demonstrates how to work with text on lines. The program opens a chart that has three text objects on a line. The user can specify which of the text objects he or she wants to turn blue.

{button Related Topics,PI(``,`IDH_RT_Text_on_Lines')}

Running and Viewing the Sample Files

How to Use this Help

This Help gives you information about how to write OLE Automation programs for FlowCharter. Use this Help to learn the highlights of Visual Basic, as well as how to automate the features of the FlowCharter product.

We recommend you take a few minutes to become familiar with this Help and its contents before using OLE Automation. You will find it provides information to help you understand the basic concepts of Visual Basic, detailed information for automating each feature of FlowCharter, and a multitude of examples for each programming property and method.

Refer to Visual Basic Concepts and Writing a Program for information to familiarize yourself with Visual Basic. They describe the basic concepts that are used throughout the remainder of the Help and provide useful information on using Visual Basic to write an automation program. Detailed information about Visual Basic is provided in the Visual Basic manuals.

Refer to Running and Setting Up FlowCharter, Handling FlowCharter Events, Working with Chart Files, Working with Objects, Working with Shapes, Working with Lines, Working with Text, Working with Data Fields, and Using Color to see how to automate the features of FlowCharter. Each topic provides detailed information and examples on how to perform each task using automation commands. Use these topics to learn the details associated with automation. These topics assume you are familiar with Visual Basic and FlowCharter. To learn about FlowCharter, see the Help for FlowCharter.

The Language Reference is a complete reference for every property, method, and event used with OLE Automation. After you know the basic concepts behind a property or method, use this as a quick reference for information on syntax and parameters, as well as a description of possible values. Topics below this one include ways to access the properties, methods, and events. Objects, alphabetical lists all the objects alphabetically. Objects, graphical provides a visual reference to the FlowCharter objects and their relationships. Properties, alphabetical lists the properties in alphabetical order. Methods, alphabetical lists the methods in alphabetical order. Events, alphabetical lists the events in alphabetical order. All Properties, Methods, Objects, and Events, alphabetical lists the properties, methods, and events in alphabetical order.

FlowCharter Menu Command equivalents provides a listing of the OLE Automation command that is equivalent to each FlowCharter menu command.

In the Contents, FlowCharter Features Not Automated describes the FlowCharter features that cannot be automated. Using C++ with OLE Automation contains information on accessing and using OLE Automation with C++.

{button Related Topics,PI(`',`IDH_RT_Introduction_Chapter')}

Conventions

This Help provides visual keys to special information with Notes and Tips.

Notes

Notes inform you of exceptions or special cases.

Tips

Tips offer ways to help you work more efficiently, and suggest shortcuts.

This manual also uses the conventions listed in the following table.

Example	Description
Height, **DrawLine**	Words in bold or green with an underline indicate properties, methods, and events. You can click on any item that is green with an underline to go to a related topic.
ShapeObject	Words in italic indicate placeholders for information you supply.
[*FieldType*]	Elements within square brackets are optional.
{*Index* | *Filename*}	Elements within braces separated by a vertical bar represent a mandatory choice between the two elements. You may choose one element or the other.
`' This is a comment`	An apostrophe (') in code introduces a comment. Comments are ignored by the system when the program runs, but provide helpful information to a person reading the code.
`ABCObject.Height = 2.5`	Text in this font represents actual Visual Basic or C++ code.
CONSTANT.FLO	Words in all capital letters represent file names or constants.
ENTER, **DEL**	Words in bold with an initial capital letter represent keys you can press on your keyboard.

{button Related Topics,PI(``,`IDH_RT_Conventions')}

Introducing OLE Automation

Accessing OLE Automation Help from Visual Basic

There are several ways you can access this help from Visual Basic. Each way brings up an appropriate topic.

- Select the FlowCharter Events custom control and press **F1**.
- Highlight a procedure name in the Procedure box ("Proc:" combo box) and press **F1**.
- Highlight a property from the Properties Window and press **F1**.

{button Related Topics,PI(`',`IDH_RT_Accessing_from_Visual_Basic')}

Introduction to Programming

Visual Basic is a variant of the BASIC programming language designed specifically for creating applications for Microsoft Windows. Visual Basic differs from earlier versions of BASIC in two important respects.

- You program Visual Basic in a graphic environment in which many aspects of program development are accomplished by drawing on the screen using a mouse. It is this distinctive characteristic of Visual Basic programming that gives the language the "visual" part of its name and greatly simplifies the process of creating applications.
- Visual Basic is an object-oriented language. Object-oriented programming (OOP) is a relatively new approach to software development.

This topic and the other topics referred to in the Objects in Visual Basic topic explain the terms and concepts behind object-oriented programming as implemented in Visual Basic. The terms defined in these topics are used throughout this manual.

{button Related Topics,PI(`',`IDH_RT_Introduction_to_Programming')}

Objects in Visual Basic

Objects in Visual Basic

In an object-oriented language such as Visual Basic, the emphasis of program development is on the definition and use of specialized software units called objects. This is in contrast to other developmental methodologies in which the major emphasis is on the flow of program execution and the actions performed by the parts of a program.

An *object* is a software structure that combines both data and the capability to act upon or process that data.

A Visual Basic object is defined by five characteristics.

- It has a unique name.
- It has data stored within it that defines its state at any given time. The data items of an object are called *properties* in Visual Basic.
- It can perform operations. These operations are called *methods*.
- It can recognize actions such as keystrokes or mouse clicks. The user or system actions that an object recognizes are called *events*. You determine the response of an object to an event through the program instructions that you place in the *event procedure* of the object.
- It has relationships to other objects. The relationships between objects divide the objects into groups called *classes*, which are arranged in a *hierarchy*.

In some object-oriented languages, you can create new types of objects. In Visual Basic, you are limited to the objects provided by Visual Basic plus any custom objects supplied by add-on products such as FlowCharter.

FlowCharter Objects

FlowCharter exemplifies the power and capabilities of the object-oriented approach to programming. The objects provided by FlowCharter make it easy for a Visual Basic application to create diagram and flowchart shapes, connect the shapes with lines illustrating relationships, and label components.

Object Hierarchy

For information about the hierarchy of objects in FlowCharter, see Objects, graphical.

{button Related Topics,PI(`',`IDH_RT_Objects_in_Visual_Basic')}

Objects, graphical

Properties in Visual Basic

The data items stored within an object are called *properties*. Each object has specific properties that you can set and control. Some examples of properties are the **BorderColor**, **BorderStyle**, **BorderWidth**, **FillColor**, and **FillPattern** properties of the FlowCharter Shape object.

You can set some properties to any value, while other properties are restricted to specific values. An example of a property with a restricted value is the **BorderWidth** property of the FlowCharter Shape object. The **BorderWidth** property must be set to a whole number from 1 (thinnest) to 5 (thickest).

FlowCharter Shape object

Methods in Visual Basic

A *method* is an operation that an object can perform. Some examples of methods are the **Activate**, **CloseChart**, **Copy**, **Cut**, **DrawLine**, **DrawShape**, and **PrintOut** methods of the FlowCharter Shape object.
The methods available to an object are predefined and depend upon the object.

{button Related Topics,PI(`',`IDH_RT_Methods_in_Visual_Basic')}

FlowCharter Shape object

Events in Visual Basic

An *event* is an action recognized by an object. Events can be triggered by the user (such as clicking an object with the mouse), by the computer system (such as a timer event), or by program code (such as an instruction to move an object).

An event is always specific to a particular object, which means that a given event can be detected by only one object. For example, a mouse click triggers an event only for the object actually clicked. Other objects beside or beneath the clicked object do not recognize that click event.

The events recognized by an object are predefined and depend upon the object. For example, the events that can be recognized by the Application object are listed below.

AppQuitNOTIFY
AppQuitSUBCLASS
AppMenuHintSUBCLASS
AppMenuPopupSUBCLASS
AppMenuSUBCLASS
ChartActivateNOTIFY
ChartDeActivateNOTIFY
ChartChangeNOTIFY
ChartCloseSUBCLASS
ChartNewNOTIFY
ChartOpenNOTIFY
ChartPasteNOTIFY
ChartSavedNOTIFY
DeleteSUBCLASS
DoubleClickSUBCLASS
ExclusiveSelectionNOTIFY
FieldValueChangedNOTIFY
LinkNOTIFY
NewLineNOTIFY
NewShapeNOTIFY
ObjectClickSUBCLASS
ObjectFontChangeNOTIFY
ObjectLineAttachNOTIFY
ObjectLineDeAttachNOTIFY
ObjectMovedNOTIFY
ObjectMoveSUBCLASS
ObjectSizedNOTIFY
ObjectSizeSUBCLASS
ObjectTextChangedNOTIFY
ReplaceShapeNOTIFY
SpecialKeySUBCLASS

If you want an object to perform a task when an event occurs, you add program instructions to the *event procedure* for that event. When an object detects an event for which it has a defined event procedure, it executes the instructions in the procedure. At the conclusion of the event procedure, the object returns to a state in which it waits for another event.

An example of a FlowCharter event procedure is shown below. This procedure saves the left and top locations of the object before it is moved in the global variables GLeft and GTop.

Use this code in an external Visual Basic application:

```
Sub ABC1_ObjectMoveSUBCLASS ( )
    GLeft = ABC1.Object.Left          ' Save left edge
    GTop = ABC1.Object.Top            ' Save top edge
End Sub
```

Note

- To do this internally, in Living FlowChart script, change the "ABC1" to "Application":

```
Sub ABC1_ObjectMoveSUBCLASS ( )
    GLeft = Application.Object.Left       ' Save left edge
    GTop = Application.Object.Top         ' Save top edge
End Sub
```

Note

- In external Visual Basic programs, FlowCharter requires that you identify the event procedures you want it to execute by registering them using the **RegisterEvent method** of the FlowCharter Application object. Unregistered events are ignored by FlowCharter, even though you may have written procedures for the events. For more information on registering events, see Registering Event Procedures.

Overriding Standard Behavior of an Application Object

Many events have *standard* actions that they perform when the event is triggered. For example, the standard action of the **ObjectSizeSUBCLASS event** is to resize the selected object or objects.

You can cancel the standard action of a SUBCLASS event of an Application object by setting the **Override** property of ABC1 to True in the object's event procedure.

The following statements provide an example of overriding an of an Application object event's standard behavior. The standard action of the **DeleteSUBCLASS event** is to delete the selected object or objects. If **Override** is set to True in the **DeleteSUBCLASS event** procedure, then the delete action is not performed.

```
Sub ABC1_DeleteSUBCLASS ( )
    Dim ABCObj As Object
    Set ABCObj = ABC1.Object
    ABC1.Override = True              ' Override standard action
End Sub
```

If you want to do this internally (in a Living FlowChart script), change "ABC1" to "Application":

```
Sub ABC1_DeleteSUBCLASS ( )
    Dim ABCObj As Object
    Set ABCObj = ABC1.Object
    Application.Override = True       ' Override standard action
End Sub
```

For information on overriding standard behavior of Chart object events and Object object events, see What Are FlowCharter Events?

{button Related Topics,PI(`',`IDH_RT_Events_in_Visual_Basic')}

Registering Event Procedures
What Are FlowCharter Events?

DeleteSUBCLASS event
ObjectSizeSUBCLASS event
Override property
RegisterEvent method

AddItem Method

See your Visual Basic manual for information on the **AddItem** method.

AutoSize Property

See your Visual Basic manual for information on the **AutoSize** property.

BackColor Property

See your Visual Basic manual for information on the **BackColor** method.

Cls Method

See your Visual Basic manual for information on the **Cls** method.

Cols Property

See your Visual Basic manual for information on the **Cols** property.

CreateObject Statement

See your Visual Basic manual for information on the **CreateObject** statement.

DatabaseName Property

See your Visual Basic manual for information on the **DatabaseName** property.

Default Property

See your Visual Basic manual for information on the **Default** property.

Dim Statement

See your Visual Basic manual for information on the **Dim** statement.

ForeColor Property

See your Visual Basic manual for information on the **ForeColor** method.

MultiLine Property

See your Visual Basic manual for information on the **MultiLine** property.

Name Property

See your Visual Basic manual for information on the **Name** property.

Override Property

See your Visual Basic manual for information on the **Override** property.

Pattern Property

See your Visual Basic manual for information on the **Pattern** property.

RemoveItem Method

See your Visual Basic manual for information on the **RemoveItem** method.

Rows Property

See your Visual Basic manual for information on the **Rows** property.

Set Statement

See your Visual Basic manual for information on the **Set** statement.

SetFocus Method

See your Visual Basic manual for information on the **SetFocus** method.

Stretch Property

See your Visual Basic manual for information on the **Stretch** property.

Visible Property

See your Visual Basic manual for information on the **Visible** property.

WordWrap Property

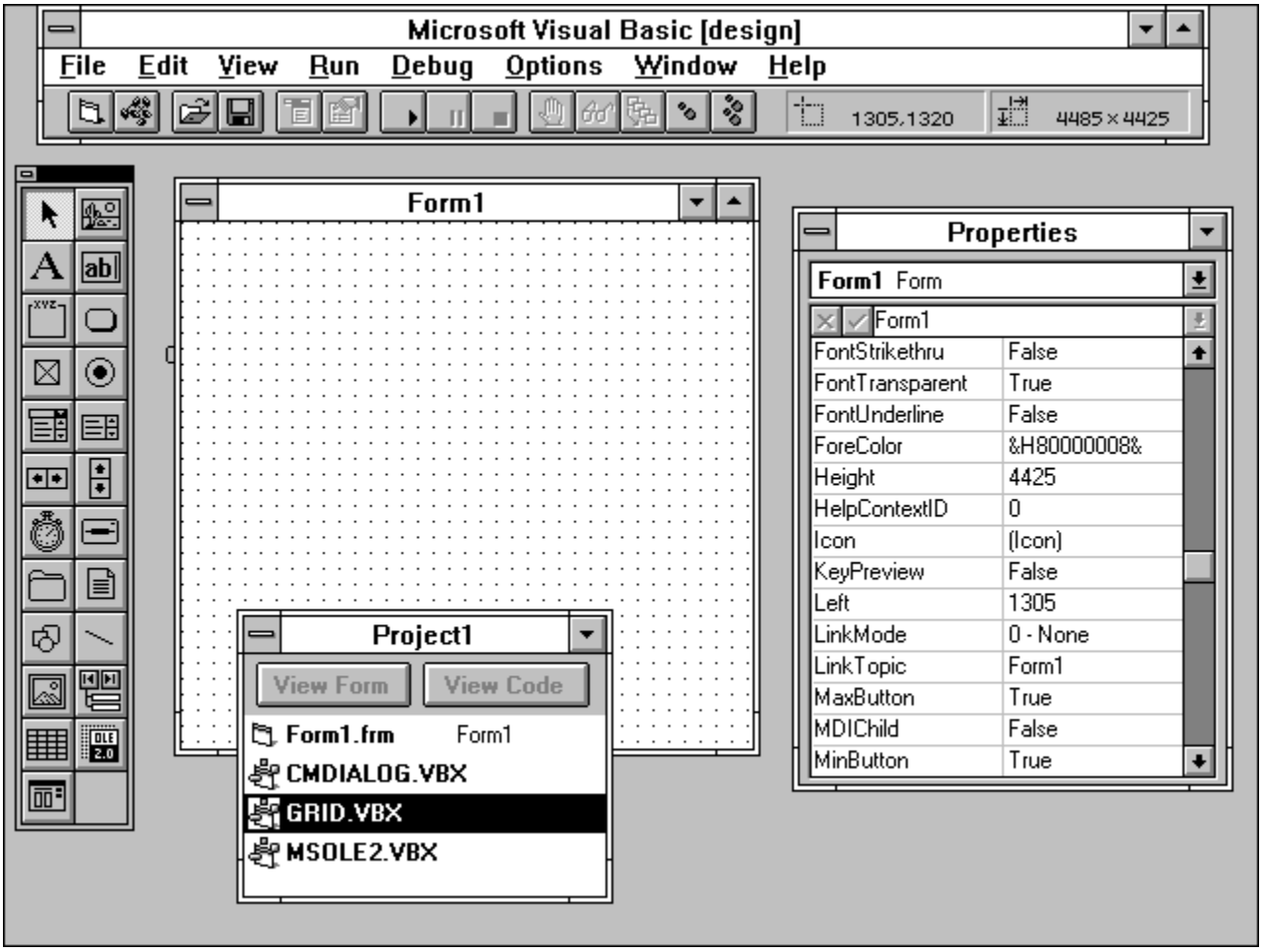
See your Visual Basic manual for information on the **WordWrap** property.

The Visual Basic Programming Environment

Starting Visual Basic displays the Visual Basic programming environment. As you would expect in a programming language designed to create Windows applications, the Visual Basic programming environment is organized into windows. It consists of five specialized windows. (Not all windows are visible when you start Visual Basic.)

- Main window
- Project window
- Form window
- Toolbox
- Properties window

You can resize and reposition these windows as desired. A typical arrangement in which all of the windows are visible is shown below.



{button Related Topics,PI(`',`IDH_RT_The_Visual_Basic_Programming_Environment')}

The Visual Basic Main Window

In Visual Basic, the Main window contains the Visual Basic Control box, title bar, Minimize button, Maximize button, menu bar, and toolbar. The Main window is generally positioned at the top of the screen, but can be moved if desired.

{button Related Topics,PI(`',`IDH_RT_The_Visual_Basic_Main_Window')}

The Visual Basic Programming Environment

The Visual Basic Tool Bar

The toolbar in the Main window contains buttons providing quick access to common Visual Basic commands. The toolbar also contains two fields giving information on the screen location and size of the selected form or object.

{button Related Topics,PI(`',`IDH_RT_The_Visual_Basic_Tool_Bar')}

The Visual Basic Programming Environment

The Visual Basic Project Window

A *project* is the collection of files used to build a Visual Basic application. Visual Basic uses specialized files to store different types of program data. To make it clear that program development in Visual Basic involves a series of files, Visual Basic programs are called projects.

The Project window lists all the files associated with the current project.

The files in a project include the following.

- A file for each form in the application. Form files have the FRM extension.
- A file for each code module in the application. Code files have the BAS extension.
- A file for each custom control used by the application. Custom control files have the VBX or OCX extension. For example, the controls provided by FlowCharter are stored in the file ABCAUTO.VBX or FLOW.OCX.

The files and various settings used by a project are saved in the project file. Project files have the MAK extension. The default project file that is loaded automatically when you start Visual Basic is called AUTOLOAD.MAK.

Commands are available in the File menu to add and remove files from the Project window.

To add the custom control file to a Visual Basic 3.0 project

1. Open the File menu.
2. Click the Add File command. The Add File dialog box opens.
3. Change to the directory where you installed FlowCharter.
4. Double click ABCAUTO.VBX or FLOW.OCX. The dialog box closes and the file is added to the project.

Tip

If you installed all options of the Visual Basic 3.0 Professional Edition, there will be many VBX files in your default project. You can shorten the time it takes Visual Basic and your programs to load by removing any VBX files you are not using regularly from the default project file. To remove a VBX file from a project, select the file in the Project window, open the File menu, and choose Remove File. When finished, save the project as AUTOLOAD.MAK using the Save Project As command of the File menu.

Besides the files in the project, the Project window contains the View Form and View Code buttons.

The View Form button lets you display the form saved in a form (FRM) file.

The View Code button lets you display the Visual Basic code saved in a form or code (BAS) file.

To display the form saved in a form file, select the form file and click View Form. To display the code saved in a form or code file, select the form or code file and click View Code.

Tip

Double click a form file to display its form, or double click a code file to display its code.

To add the custom control file to a Visual Basic 4.0 project

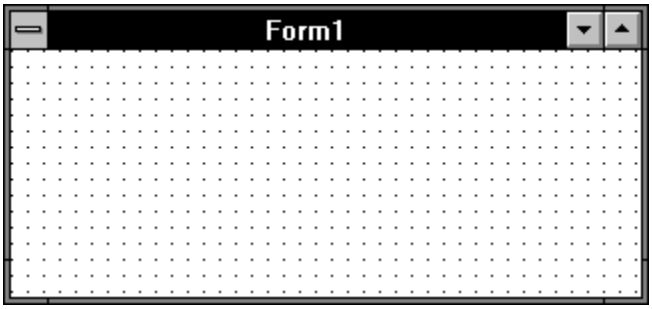
1. On the Tools menu, click Custom Controls.
2. Click the Micrografx FlowCharter Control check box.
3. Click OK.

The Visual Basic Programming Environment

The Visual Basic Form Window

The first step in creating a Visual Basic application is designing the application's windows. In Visual Basic, during the development stage, windows are referred to as *forms* and are created using the Form window.

When you start a new project, Visual Basic creates an empty form that it titles Form1. Every Visual Basic project must have at least one form, although it is not necessary that the form contain anything or actually be displayed by the completed application.



Notice that even a blank Form window looks like a standard Windows window. It can be resized, and contains a Control box, a title bar, and Minimize and Maximize buttons. The grid within the window is used to align the objects placed on the form. The grid does not appear in the completed application. The procedures for adding objects to the form are discussed in Drawing a Control.

The Form1 title given to the initial form is a default name that Visual Basic supplies automatically for a new project. You can change this name easily to suit the application you are developing.

{button Related Topics,PI(`',`IDH_RT_The_Visual_Basic_Form_Window')}

Drawing a Control
The Visual Basic Programming Environment

The Visual Basic Toolbox

The objects placed on a form are called *controls*.

To create a control, you click the appropriate button on the Toolbox and then drag the mouse over the area in the Form window where you want the control. The creation of controls using graphic methods is perhaps the most spectacular aspect of Visual Basic programming.

After you create a control, you define its properties using the control's Properties window. You define the control's operation or action in the completed application by writing Visual Basic code for the control.

Visual Basic comes with a set of standard controls that include labels, picture boxes, text boxes, check boxes, list boxes, command buttons, and timers. The Toolbox also has an item for each custom control (VBX or OCX) that is added. If you have installed the Professional Edition of Visual Basic, the Toolbox contains many more controls than what is illustrated below. (See the Tip in The Visual Basic Project Window for instructions on removing these extra controls.) When you add a new control to a project, such as that supplied by FlowCharter, the new control appears in the Toolbox. Therefore, if you are using more than the standard controls, your Toolbox will show the additional controls.

A description of the function of these controls is provided in Standard Controls.

{button Related Topics,PI(`',`IDH_RT_The_Visual_Basic_Toolbox')}

The Visual Basic Properties Window

The Properties window lets you view and set the properties of objects. An object's properties determine its appearance and behavior. The properties of an object include its name, size, screen location, color, and visibility in the completed application.

You can activate the Properties window in several ways.

- Click an object such as a form or control.

or

- Open the Windows menu in the Main window and choose Properties.

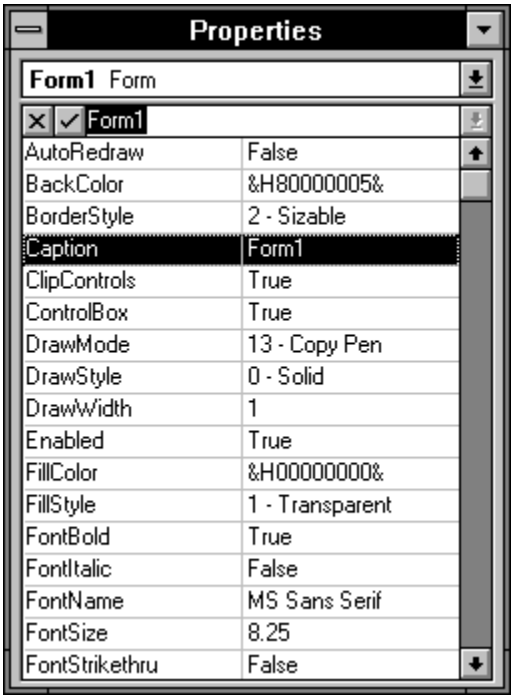
or

- Press **F4**.

or

- Click the Properties Window button in the Main window toolbar.

The Properties window contains an Object box, a Settings box, and a two-column list of the properties and settings.



The Object box displays the name and type of the object whose properties are listed. The object's name is bold and listed first. In the example above, the object is named Form1 and is a Form object. To the right of the Object box is a down arrow. Click this arrow to display a list of all of the objects associated with the currently selected form, including the Form object itself. Click the appropriate entry in the Object list to display the properties of that object.

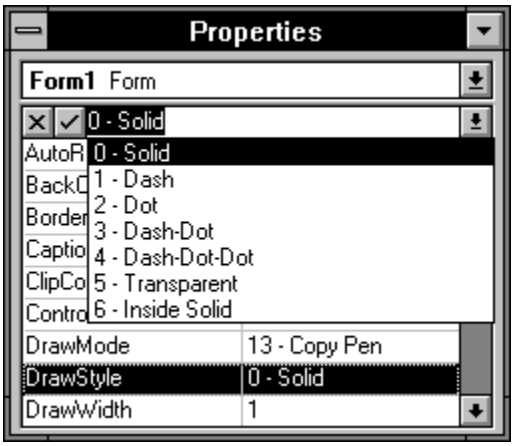
You use the Settings box to change the value of a property. The way you change a value depends on whether the property is an enumerated or nonenumerated property.

- An *enumerated* property is limited to a set of predefined values. An example of an enumerated property is the **BorderStyle** property of a Form object. **BorderStyle** must be set to None, Fixed Single, Sizable, or Fixed Double. When a selected property is enumerated, you can click the down arrow (or ellipsis for color properties) to the right of the Settings box to display the list of predefined values.
- A *nonenumerated* property is not limited to a set of predefined values. An example of a nonenumerated property is the **Name** property of an object, which can be set to any text string. When a selected property is nonenumerated, the down arrow to the right of the Settings box is gray.

To change the value of an enumerated property

1 Select the property by clicking it. The selected property is highlighted and the current value of the property appears in the Settings box.

2 Click the down arrow to the right of the Settings box and select the desired value from the list that appears. (If the property is a color rather than a setting, click the ellipsis to the right of the Settings box and select the desired color from the color palette that appears.)



Tip

You can cycle through the predefined values of an enumerated property by double clicking the property in the Properties list. For example, to change the **Visible** property of a Form object from True to False, double click Visible in the Properties box.

To change the value of a nonenumerated property

1 Double click the property you want to change. The selected property is highlighted and the current value of the property is highlighted in the Settings box.

2 To replace the current property value, just type the new value. The characters that you type appear in the property list and the Settings box.

or

To edit the current property value, click the highlighted value in the Settings box and edit the value as you would any field.

3 To accept the new property value, click the checkmark box in the Settings box or press **ENTER**.

or

To reject the new property value, click the X box in the Settings box or press **ESC**.

{button Related Topics,PI(`',`IDH_RT_The_Visual_Basic_Properties_Window')}

The Visual Basic Programming Environment

Creating a Visual Basic Application

The general steps to writing a Visual Basic application are:

1 Create a new project.

2 Design the application's forms (windows).

3 Add code to determine the application's response to desired events.

4 Run and test the application.

5 Create an executable (EXE) file.

These steps are discussed below.

Creating a New Project

The first step in creating a Visual Basic application is to create a new project. To perform this simple but important step, open the File menu in the Visual Basic Main window and choose New Project. When you create a new project, you ensure that your application begins with a "clean slate."

If you want to use controls in your application that are not part of the standard Visual Basic set, you must add the custom control (VBX or OCX) files that define the controls to your project. For example, to use the control provided by FlowCharter, you must add ABCAUTO.VBX or FLOW.OCX to your project. See The Visual Basic Project Window for instructions on adding ABCAUTO.VBX to a project.

Designing the Application's Forms

The actual programming of an application begins with the design of the forms for the application. This is the visual user interface of the application. The process of designing a form is explained in Designing a Form.

Adding Code

After you create the visual interface, you add code to specify how the program should respond to actions by a user. For example, you add code to a Command button control to determine what the program does when a user clicks the Command button. The process of adding code to an application is discussed in Adding Code to Controls, Adding Code to Forms, Writing Code in Event Procedures, Writing Code in General Procedures, and Writing Startup Code.

Running the Application

Running your application in the Visual Basic programming environment lets you test the application as you develop it.

To run an application

- Click the Start button on the Main window toolbar.

or

- Open the Run menu in the Main window and choose Start.

or

- Press **F5**.

To stop a running application

- Click the End button in the Main window toolbar.

or

- Open the Run menu and choose End.

Creating an EXE Program

The final step in creating a Visual Basic application is to make an executable file. An executable file is a version of the application that does not require the Visual Basic programming environment to execute (it runs as a standard Windows application). To make an executable file for the application, open the File menu in the Visual Basic Main window and choose Make EXE File.

After you make an executable file for your application, you can create a button for it in FlowCharter. For more information, see Adding Buttons.

{button Related Topics,PI(`',`IDH_RT_Creating_a_Visual_Basic_Application')}

Designing a Form

Visual Basic programming begins with the creation of forms. The forms you create become windows in the completed application.

The process of designing a form involves setting the properties of the form, drawing the controls that appear on the form, and setting the properties of the controls.

If your application is a simple one, then you may not have more than one form. If you need additional forms, add them with the New Form button in the toolbar or with the New Form command in the File menu.

{button Related Topics,PI(`',`IDH_RT_Designing_a_Form')}

Setting a Form's Properties

When you start a new project, you are presented with a blank form titled Form1. The properties of this form (or any new form) are set to default values.

To change the form's appearance and behavior, you must customize its properties. You can customize the properties of a form by manually setting the properties using the Properties window, by setting the properties using code, or by using a combination of manual settings and code (the usual approach).

Generally, you set the properties controlling a form's overall appearance manually. These properties include the form's **Caption**, **BackColor**, **ForeColor**, and **Borderstyle**. If you intend to change the **Name** of a form, you should also set this property manually. (The **Name** of a form is used to reference the form in code, so changing a form's **Name** after you have already written code requires replacing references to the old form **Name** in the code with the new form **Name**.)

The process of setting a form's properties manually is discussed in The Visual Basic Properties Window.

The process of setting a form's properties with code is discussed in Setting and Retrieving Properties with Code.

{button Related Topics,PI(`',`IDH_RT_Setting_a_Form_s_Properties')}

Standard Controls in Visual Basic

The Visual Basic Toolbox provides standard controls that can be placed on a form. This section gives a brief description of the functions of these controls.

 The picture box control displays graphic images from bitmaps, icons, or Windows metafiles. The size of the picture box determines how much of the graphic image is displayed. To make a picture box automatically resize itself so that it can display all of a graphic image, set the control's **AutoSize** property to True.

 The label control displays text on a form that the user cannot change. To set a label control to display multiple lines, set the control's **AutoSize** and **WordWrap** properties to True.

 The text box control provides an area on a form that can display text and accept user input. To set a text box so it can display and accept multiple lines, set the text box's **MultiLine** property to True.

 The frame control provides a way to group controls. The grouping capability provided by a frame is visual and logical (functional).

- A frame groups controls *visually* by surrounding the controls with a box and providing a title.
- A frame groups controls *logically* by permitting an application to treat the controls in the Frame as a unit.

For example, setting a frame's **Visible** property to False makes all of the controls in the frame invisible.

To place a control in a frame, you must either draw the control in the frame or Paste it into the frame. You cannot move a control into a frame.

 The command button control displays a button that can begin, interrupt, or end a process. The user executes a command button by clicking it. To enable a command button to be executed by pressing enter, set the control's **Default** property to True. Examples of command buttons are the OK and Cancel buttons you see in Windows applications.

 The check box control displays a box that can be selected or deselected, like an on/off switch. A check box control shows an X when selected. How the states of a check box are interpreted by an application depend on the code. Check boxes are commonly used for Yes/No and True/False options.

 The option button control displays a button that can be selected or deselected. Option buttons, like check boxes, are on/off switches. Unlike check boxes, however, only one option button on a form can be selected at a time. Clicking one option button automatically deselects all other option buttons on the form.

If you want more than one set of option buttons on a form, then you must group them with a frame control.

 The combo box control displays a box that combines the characteristics of a text box and a list box. A combo box lets a user select an item from a list by typing the desired item or by choosing the item from a drop-down list.

 The list box control displays items that can be selected by clicking the item or by moving the cursor to the item and pressing enter. Scroll bars are automatically added to a list box if it contains more items than can be listed at once. The control's **Additem** and **Removeitem** methods are used to add and delete list items.

 The horizontal scroll bar control provides a method of scrolling horizontally through a list or graphically indicating settings such as quantity or volume.

 The vertical scroll bar control provides a method of scrolling vertically through a list or graphically indicating settings.

 The timer control executes events at specified time intervals. A timer control can be seen when you are designing a form, but is invisible when the application executes.

 The drive list box control provides a method of selecting a disk drive when an application executes. This control automatically lists all the valid drives in the user's system.

 The directory list box control displays directories and paths when an application executes and provides a method of selecting a directory.

 The file list box control displays files in a directory when an application executes, and provides a method of selecting a file. The control's **Path** property determines the directory displayed. The control's **Pattern** property determines the files displayed.

 The shape control displays a rectangle, square, oval, circle, rounded rectangle, or rounded square. The control's **Shape** property determines its shape.

 The line control displays a horizontal, vertical, or diagonal line. The line control's **BorderStyle** property determines the type of line (solid, dotted, dashed).

 The image control displays bitmaps, icons, or Windows metafiles. An image control uses fewer system resources and displays faster than a picture control. It also offers fewer properties and methods than a picture control. Set the control's **Stretch** property to True if you want the image resized to fit the control's size and shape.

 The data control provides access to data stored in databases. The data control's **DatabaseName** property determines the database source file.

 The grid control displays a grid of cells that can contain text or pictures. The control's **Cols** and **Rows** properties determine the number of rows and columns.

Note

The grid control is available only if you add the GRID.VBX control file to a project.

 The OLE control provides a method of linking and embedding an OLE object into an application.

Note

The OLE control is available only if you add the MSOLE2.VBX control file to a project.

 The common dialog control provides standard dialog boxes for operations such as opening, saving, and printing a file. A common dialog control is shown only as an icon when you are designing a form. It appears as a full-size dialog box when the application executes.

Note

The common dialog control is available only if you add the CMDIALOG.VBX control file to a project.

 The FlowCharter control lets you add OLE Automation capabilities to your Visual Basic applications.

Note

The FlowCharter control is available only if you add the ABCAUTO.VBX or FLOW.OCX control file to a project. (See To install the FlowChart VB event handler or To install the OCX event handler.)

{button Related Topics,PI(`',`IDH_RT_Standard_Controls')}

Designing a Form

Drawing a Control

There are two ways to draw a control on a form: the normal drawing method and a shortcut method.

To draw a control using the normal method

1 Click the appropriate control tool in the Toolbox. For example, to draw a command button, click the Command Button tool.

2 Position the mouse pointer on the form where you want the control to begin. The mouse pointer shows as a cross hair.

3 Press and hold the mouse button, and drag the pointer until the control's outline is the size you want. Release the mouse button. The control appears on the form in the size and shape you drew.

To draw a control using the shortcut method

1 Click the form to which you want to add the control. (If there is only one form in your project, you can skip this step.)

2 Double click the appropriate control tool in the Toolbox. The control appears in the center of the selected form.

3 Resize (and move) the control as desired.

{button Related Topics,PI(`',`IDH_RT_Drawing_a_Control')}

Resizing a Control

Resizing a control is easy and can be performed at any time during the design process.

To resize a control

1 Select the control you want to resize by clicking it. Small, solid *handles* appear on the control.

2 Position the pointer over a handle. The pointer changes to a two-headed arrow.

3 Press and hold the mouse button, and drag the pointer in the appropriate direction to change the control's size. Release the mouse button.

Designing a Form

Moving a Control

You can move controls one at a time or in groups.

To move a single control

1 Select the control you want to move by clicking it. The control's handles appear.

2 Position the mouse pointer anywhere on the control other than on a handle.

3 Press and hold the mouse button, and drag the control to its new position. Release the mouse button.

By moving controls in a group, you can keep the relative position of the controls constant.

To move multiple controls

1 Select the first control by clicking it. The control's handles appear.

2 Select the additional controls by pressing **CTRL** while you click the control.

3 Position the mouse pointer on one of the selected controls.

4 Press and hold the mouse button, and drag the controls to their new positions. Release the mouse button.

{button Related Topics,PI(`',`IDH_RT_Moving_a_Control')}

Designing a Form

Deleting a Control

To delete a control, select the control and press **DEL**. You also can delete a control by selecting the control and choosing the Delete command in the Edit menu.

{button Related Topics,PI(`',`IDH_RT_Deleting_a_Control')}

Designing a Form

Copying or Cutting a Control

You can copy or cut a control, and then paste it back into your form or into other forms. Select the control and choose the Copy or Cut command in the Edit menu, or select the control and press the shortcut keys **CTRL+C** (Copy) or **CTRL+X** (Cut).

{button Related Topics,PI(`',`IDH_RT_Copying_or_Cutting_a_Control')}

Designing a Form

Aligning Controls

The grid that displays on forms during the development process makes it easy to align controls. As you draw, position, and resize a control, the edges of the control snap to the nearest grid position.

The precision of the grid depends on Visual Basic's Grid Width and Grid Height settings. The default settings for Grid Width and Grid Height is 120 twips, which translates into a distance between grid positions of 1/12 inch. A twip is equivalent to 1/1440 inch (1 inch = 1440 twips).

To increase the alignment precision of the grid, use a larger Grid Width or Grid Height setting. To decrease the alignment precision, use a smaller Grid Width or Grid Height setting.

To change Visual Basic's grid settings

1. Open the Options menu in the Main window and choose Environment.
2. Click the Grid Width option (scroll through the list if necessary) and type a value in the Setting box.
3. Click the Grid Height option (scroll through the list if necessary) and type a value in the Setting box.
4. Click OK to accept the new settings and close the dialog box.

If you don't want to use the grid alignment feature, you can turn it off by opening the Environment Options dialog box and setting Align To Grid to No. If you want to hide the grid, open the Environment Options dialog box and set Show Grid to No.

Designing a Form

Setting Control Properties

After you draw a control, you can customize its appearance and behavior by setting its properties.

As with forms, you probably want to set some control properties manually and some control properties with code. Generally, you set the appearance properties manually. For details on setting control properties with code, see Setting and Retrieving Properties with Code.

To change a control's properties manually

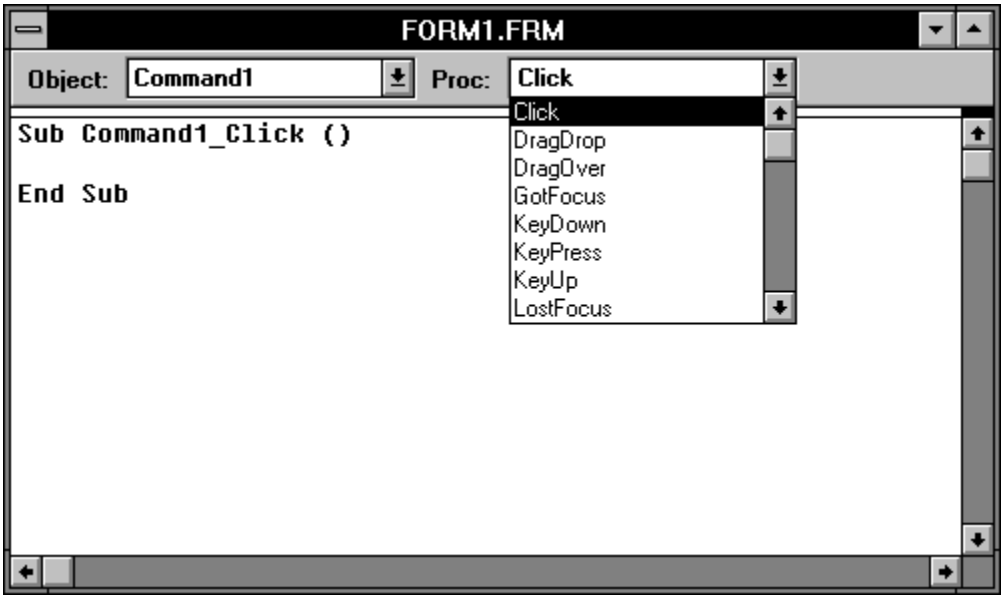
1. Select the control as the current object by clicking it. The control's handles appear.
2. The Object box of the Properties window should show the name of the control. If the Properties window is not visible, press **F4** to display it.
3. Set the properties you want to change.

{button Related Topics,PI(`',`IDH_RT_Setting_Control_Properties')}

Adding Code to Controls

Drawing a control and setting its properties let you determine how the control appears in an application, but not what it does. To prescribe what the control does, you must add Visual Basic instructions to the control. These instructions are referred to as *code*.

You add code to a control using the Code window. To display the Code window for a control, double click the control. An example of the Code window for a Command Button appears below.



The title bar at the top of the Code window identifies the control's form. Below the title bar are the Object and Procedure boxes, and below these boxes is the code area.

The Object box gives the name of the object to which the code applies; such as Command1. To select other objects on the form, including the Form object, click the down arrow to the right of the box.

Note

Besides listing the objects associated with the form, the Object box list also shows a (general) entry. This section of the form is used to write general procedures for the form, as explained in Writing Code in General Procedures.

The Procedure box gives the name of the event to which the code applies, such as the **Click** event. To select other events defined for this object, click the down arrow to the right of the box.

The two statements shown in the code area are required at the beginning and end of the event code, and are supplied automatically by Visual Basic. Any code that you want to add to an event must be typed between these two statements.

{button Related Topics,PI(`',`IDH_RT_Adding_Code_to_Controls')}

Adding Code to Forms

Forms, like controls, require code to determine how they respond to events. To add code to a form, double click the form to display the form's Code window (or select the form's name from the Object box list if the Code window is already open). The events recognized by the form are displayed in the Procedure box list.

{button Related Topics,PI(`',`IDH_RT_Adding_Code_to_Forms')}

Writing Code in Event Procedures

Visual Basic forms and controls are event-driven objects. This means that forms and controls respond to specific user actions such as being clicked, double clicked, dragged, or scrolled. The events to which a form or control responds are predefined and depend on the type of object. For example, a Command button can be clicked, but not scrolled, while a Scroll bar can be scrolled, but not clicked.

Because controls and forms are event-driven, they can execute code only when an event recognized by the object occurs. Until such an event occurs, an object is inactive.

The code that is executed when an event occurs is called an *event procedure*. For details on the syntax required for procedures, see Using Procedures.

Events can occur only when an application is running, not during program development. Thus, clicking a command button that you have just drawn on a form does not cause a Click event for the control. You must click the command button when the program is executing to trigger the Click event.

Tip

You can quickly determine which event procedures have code added to them by checking the drop down list in the Procedure box. Event procedures with added code appear in bold in the list.

{button Related Topics,PI(`',`IDH_RT_Writing_Code_in_Event_Procedures')}

Writing Code in General Procedures

In addition to the event procedures that are triggered by a form or control event, a Visual Basic application can have *general procedures*. A general procedure is any procedure that is not an event procedure.

General procedures let you write Visual Basic code that is not attached to specific events. Because general procedures are not attached to specific events, they can contain common code that is needed by several event procedures. Without a general procedure capability, you would have to write duplicate code for each event procedure that needed to perform a common action.

As an example of the benefits of general procedures, consider an application that has command buttons labeled First Record, Next Record, Previous Record, and Last Record. Rather than duplicating the code required to read a record in each command button's Click event procedure, you could write a general procedure that gets a record based on a parameter passed to the procedure. Then, the Click event procedure for each command button calls the general procedure with the appropriate parameter to read a record.

To ensure that a general procedure is not attached to a specific event, you must write the procedure either in the (general) section of a form or in a code module.

- If you want the general procedure to be available only to controls on a particular form, write the procedure in the (general) section of that form.
- If you want the general procedure to be available to any control on any form in the application, write the procedure in a code module.

To access the (general) section of a form

1. Double click the form to display the Code window for the form. The Object box shows the name of the form.
2. Click the down arrow to the right of the Object box. The object list opens.
3. Click the (general) entry in the object list.

To access a code module

- To create a code module use the New Module command in the File menu.
- To add code to a code module or to view code already saved in a code module, double click the module's name in the Project window.

{button Related Topics,PI(`',`IDH_RT_Writing_Code_in_General_Procedures')}

Writing Startup Code

A Visual Basic application must begin with some initial event or general procedure. The default setting for an application is to begin with the Load event of the first form of the application. The Load event of a form displays the form. The first form of an application is the form named Form1. Changing the name of this form does not change its status as the first form of the application.

If you want an application to begin with the Load event of another form or with a general procedure, you can change the startup setting. If it is necessary for an application to perform initialization code when it begins, the initialization code must be executed by the specified startup event or procedure.

To change the startup setting

1. Open the Options menu and choose Project. The Project Options dialog box displays.
2. Double click the Start Up Form option to open the list of startup settings.
3. Select the form that you want as the startup form.

   or

   Select Sub Main if you want the application to begin with a general procedure.
4. Click OK to accept your changes and close the dialog box.

Selecting Sub Main tells Visual Basic to start execution with a general procedure named Sub Main. This procedure must be located in one of the application's code modules.

For information on the startup code required for FlowCharter, see Starting FlowCharter.

Using Procedures

All Visual Basic programming instructions must be written as procedures. *Procedures* are organizational structures designed to promote clear, orderly, and logical programs.

Visual Basic recognizes two types of procedures.

- Sub procedures
- Function procedures

Sub procedures are further divided into two categories, event procedures and general procedures.

Sub Procedures

Sub procedures begin with a Sub statement and end with an End Sub statement. These statements are located on separate lines. The instructions to be executed by the procedure are located between the Sub and End Sub statements.

```
Sub ProcedureName (ArgumentList)
    Statements
End Sub
```

The name of the procedure follows the Sub statement. The argument list for the procedure follows the procedure name. If the procedure has no argument list, it still includes an empty set of parentheses.

When a procedure is executed, or *called*, the statements in the procedure are executed.

The rules for naming a Sub procedure depend upon whether the procedure is an event procedure or a general procedure. The name of an event procedure follows the format *object_event*, where *object* is the name of a form or control and *event* is the name of the event that triggers the procedure. For example, the name of the procedure for the Click event of a command button control named Command1 is Command1_Click. A Sub statement with the appropriate name is automatically provided by Visual Basic when you open the Code window for an event.

The names of general procedures begin with a letter; can contain only letters, numbers, and underscore (_) characters; and can be no longer than 40 characters. General procedure names have one other restriction: they cannot be words already used by Visual Basic, such as Beep, Loop, For, If, and Line.

The argument list of a procedure passes values to the procedure that can be used by the statements in the procedure. The syntax for each argument in the argument list is shown below.

```
[By Val] ArgumentName [( )] [As Type]
```

The By Val option determines whether the argument is passed by value or reference. If By Val is used, then the argument is passed *by value*, which means that a copy of the argument is passed to the procedure. Consequently, any change to the argument by the procedure does not change the original. If By Val is omitted, then the argument is passed *by reference*. When an argument is passed by reference, any change to the argument by the procedure changes the original.

The ( ) option specifies that the argument is an array.

The As *Type* option specifies the data type of the argument. If no data type is specified, the argument defaults to the Variant type. See Understanding Data Types for details on data types.

An example of a general Sub procedure is shown below. This example reads a record from a file. The file is identified by the value passed to the procedure as FileNum. The record is identified by the value passed as Rec, and the data obtained from the record is stored in Rdata.

```
Sub GetRec (FileNum as Integer, Rec as Long, Rdata as String)
    Get FileNum, Rec, Rdata
Sub End
```

An example of how this procedure can be called is shown below. It calls the GetRec procedure to read record 123 of file 2. The contents of the record is returned in TestResult.

```
GetRec 2,123,TestResult
```

Function Procedures

Function procedures begin with a Function statement and end with an End Function statement. The instructions executed by the function are located between these two statements.

Function *ProcedureName* (*ArgumentList*) [As *Type*]
Statements
ProcedureName = *FunctionResult*
End Function

The rules for naming a Function procedure and defining its argument list are the same as for Sub procedures.

The essential difference between a Function and a Sub procedure is that a Function procedure assigns a value to its name. This action must be performed somewhere in the function, as indicated by the *ProcedureName* = *FunctionResult* line in the syntax format.

Look at the following example of a Function procedure, which converts a Fahrenheit temperature to Celsius.

```
Function Celsius (Fahrenheit as Double)
    Celsius = (Fahrenheit - 32)   5/9
End Function

T = Celsius (22) + 100 / 2
```

{button Related Topics,PI(`',`IDH_RT_Function_Procedures')}

Using Procedures

Understanding Data Types

Visual Basic supports seven data types. A *data type* is a standardized method of representing data. The following table provides the names, a brief description, and the range of the Visual Basic data types.

Type	Description	Range
Integer	2-byte integer	-32,768 to 32,767
Long	4-byte integer	-2,147,483,648 to 2,147,483,647
Single	4-byte floating-point number	-3.402823E38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E38 for positive values
Double	8-byte floating-point number	-1.79769313486232E308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232E308 for positive values
Currency	8-byte fixed-decimal number	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
String	String of characters	0 to approximately 65,500 characters
Variant	Variable	Depends upon value stored

The default data type is Variant, so if no data type is declared, then Variant is assumed. If you want a variable or data value to be a different data type, then you must explicitly declare it as that data type.

You can declare a variable's data type in the argument list of a procedure or by using the As *Type* option with variable definition statements such Dim, Static, or Global.

Some examples of these date type declaration methods are shown below.

```
Dim Count As Integer
Dim Employee As String, State As String, JobNumber As Long
Global Total As Currency
```

Numeric Data Types

Although Integer, Long, Single, Double, and Currency are all numeric data types, they differ in important characteristics such as storage requirements, range, and accuracy.

If you know that a variable will always be a whole number, then consider defining the variable as an Integer or Long data type. Using these data types when possible reduces a program's memory requirements and speeds up its execution.

If a variable can have a decimal value, then define the variable as a Single, Double, or Currency data type. The Single and Double data types have larger ranges than the Currency type, but may have small rounding errors.

{button Related Topics,PI(`',`IDH_RT_Numeric_Data_Types')}

Understanding Data Types

String Data Type

Nonnumeric data, such as text, is referred to as *string* data. To enable a variable to store string data, declare it a string variable using the String data type.

Unless you specify otherwise, a string variable can store string data of any size up to approximately 65,500 characters. If you want to limit a string variable to a predefined, fixed size, then use the following syntax to declare the string variable.

```
As String   Size
```

The example below uses this format to declare a string that is fixed at 35 characters in size. If the data stored in Company is less than 35 characters, it is padded with trailing spaces up to the 35-character length. If the data stored in Company is more than 35 characters, it is truncated from the end to the defined length.

```
Dim Company As String   35
```

{button Related Topics,PI(`',`IDH_RT_String_Data_Type')}

Understanding Data Types

Variant Data Type

The default data type is Variant. Variables defined as Variant can store data in Integer, Long, Single, Double, Currency, and String formats, plus a Date/Time format.

When a Variant variable is assigned a value, Visual Basic stores the assigned data in the most efficient data format. For example, if the value assigned to a Variant variable is a whole number between -32,768 and 32,767, then the value is stored as a two-byte integer. If a decimal value is added to this variable, then Visual Basic automatically performs the conversion necessary to store the new value as a floating-point number. If the variable is later treated as a string variable, then Visual Basic converts and stores the variable's value as a string.

You can determine the current storage format of a Variant variable with the **VarType** function. VarType returns a value that indicates the current data format of a variable. The values returned by **VarType** are defined below.

Return Value	**Meaning**
0	Empty
1	Null
2	Integer
3	Long
4	Single
5	Double
6	Currency
7	Date/Time
8	String

The Empty and Null return values indicate special states. The Empty value indicates that a Variant variable has never been assigned a value. The Null value indicates that a Variant variable does not contain a valid value. For a variable to return a Null value, it must have been explicitly assigned a Null value.

{button Related Topics,PI(`',`IDH_RT_Variant_Data_Type')}

Understanding Data Types

Declaring Object Variables

A powerful feature of Visual Basic is its capability to assign objects to variables. Object variables let a Visual Basic application manipulate objects as easily as it manipulates string or numeric data.

Before you can use an object variable in your code, you must declare it using the Dim, Static, or Global statement. The general syntax for declaring an object variable is shown below.

Dim *VariableName* As [New] *ObjectType*
Static *VariableName* As [New] *ObjectType*
Global *VariableName* As [New] *ObjectType*

VariableName gives the name of the object variable. *ObjectType* determines whether the object variable is declared for a specific or generic object. Specific object variables are declared by giving the **Name** property of the object as *ObjectType*. Generic object variables are declared by giving the class (such as Form or Control) of the object as *ObjectType*.

Adding the New parameter to a declaration statement creates a new object that is identical to the object specified as *ObjectType*. When New is omitted, the declaration statement refers to an existing object.

Some examples of object variable declarations are shown below.

```
Dim InputForm As Form3
Dim NameBox As New Text1
Global AppForms As Form
Dim AppControls As Control
```

The first two examples declare specific object variables. The last two examples declare generic object variables.

An example of declaring an object variable as a FlowCharter object is shown below.

```
Dim ObjectIn As Object
```

Setting and Retrieving Properties with Code

The properties of forms and controls can be set and retrieved in code using assignment statements. This enables a program to change properties based upon calculations or data input from users, files, and other sources.

{button Related Topics,PI(`',`IDH_RT_Setting_and_Retrieving_Properties_with_Code')}

Setting Form Properties

The general syntax for setting the property of a form with code is

Form.Property = *Value*

where *Form* is the name of the form and *Property* is the name of the property. Some examples of setting a form's properties are shown below.

```
Form1.BorderStyle = 1
Form3.Width = 5000
```

If the code setting the property of a form is located on the form, then the form name can be omitted from the assignment statement, as illustrated below.

```
BorderStyle = 1
Width = 5000
```

Setting and Retrieving Properties with Code

Setting Control Properties

The general syntax for setting the property of a control with code is

Form!Control.Property = Value

where *Form* is the name of the form, *Control* is the name of the control, and *Property* is the name of the property. Some examples of setting a control's properties are shown below.

```
Form1!Command.Caption = "OK"
Form3!Label2.Visible = 1
```

If the code setting the property of a control is located on the same form as the control, then the name of the form can be omitted from the assignment statement, as illustrated below.

```
Command.Caption = "OK"
Label2.Visible = 1
```

{button Related Topics,PI(`',`IDH_RT_Setting_Control_Properties2')}

Setting and Retrieving Properties with Code

Setting FlowCharter Object Properties

The general syntax for setting the property of a FlowCharter object with code is

Object.Property = *Value*

where *Object* is the name of the FlowCharter object and *Property* is the name of the property. Some examples of setting FlowCharter object properties are shown below.

```
ABC.Visible = True
Object.Shape.FillColor = ABC.BLUE
```

{button Related Topics,PI(`',`IDH_RT_Setting_ABC_Object_Properties')}

Setting and Retrieving Properties with Code

Retrieving Properties

You can retrieve a property setting by naming the property on the right side of an assignment statement. You can use this feature to save the property setting in a variable or to assign it to another property.

The rules for naming a property to be retrieved are the same as for setting a property.

The examples below illustrate various methods of retrieving properties. In the first three examples, the property settings are retrieved and saved in variables. In the last two examples, the setting retrieved from one property is used to set another property.

```
Temp = Form2.Width
Tstate3 = Form3!Timer1.Enabled
Command2.Visible = Command1.Visible
Form5.BackColor = Form1.BackColor
```

Setting and Retrieving Properties with Code

Executing Methods

Visual Basic objects can perform various actions such as moving and printing data. The actions that an object can perform are called *methods*. The methods available to an object are predefined and depend upon the object.

The general syntax for executing a method with code is shown below. *Object* is the name of a form or control and *Method* is the name of the method to perform. If a method requires parameters, they must follow the *Object.Method* name.

Object.Method [*Parameters*]

The following example executes a form method. The **Cls** method clears a form of all graphics and text.
Form2.Cls

The following example executes a control method. The **SetFocus** method selects the specified text box as the current object and places the cursor in the text box.

Text1.SetFocus

The following example executes a FlowCharter object method. The **PrintOut** method prints the Chart object to the current printer.

Chart.PrintOut

{button Related Topics,PI(`',`IDH_RT_Executing_Methods')}

Cls method
PrintOut method
SetFocus method

Converting to the 32-bit OCX

You can use the utility VB3to4.exe to convert files written using ABCAUTO.VBX so that they will use FLOW.OCX.

Make a backup copy of your files before beginning any conversion process.

To convert

1. Open the project in Visual Basic 3.0.
2. Load the form (.FRM file).
3. On the File menu, click Save File As.
4. Enter a filename with a txt extension.
5. Check the Save As Text box.
6. Repeat for all forms.
7. From Start Run, enter VB3to4.exe *infile* [*outfile*]. (For example, VB3to4.exe deploy.txt deploy95.txt)
8. Follow the on-screen instructions.

Note

- Visual Basic 4.0 automatically recognizes and loads the .txt file as a text file.

Changes in the NOTIFY events

The NOTIFY and SUBCLASS events take parameters with OCX. The example below shows a typical change. The parameter list changes are done automatically by VB3to4.exe. It is recommended that you use the parameters of the NOTIFY and SUBCLASS events instead of the corresponding properties of the OCX.

```
Sub ABC1_LinkNOTIFY ( )                    ' ABCAUTO.VBX syntax
    Set SourceChart = ABC1.Chart           ' Save source chart
    Set SourceObject = ABC1.Object         ' Save source object
    Set CurrentChart = ABC1.Object2        ' Save linked chart
End Sub

Sub ABC1_LinkNOTIFY (ByVal LinkedToChart As Object, ByVal Object As Object, ByVal Chart   As Object)     '
ABCFLOW.OCX syntax
    Set SourceChart = Chart                ' Save source chart
    Set SourceObject = Object              ' Save source object
    Set CurrentChart = LinkedToChart       ' Save linked chart
End Sub
```

Starting FlowCharter

Starting FlowCharter is the first step in writing an automation program. You define FlowCharter as an object using the Visual Basic **Dim** statement. In Visual Basic, first, enter the following statement in the (declarations) Proc of the (general) Object to make FlowCharter a global variable.

```
Dim ABC As Object
```

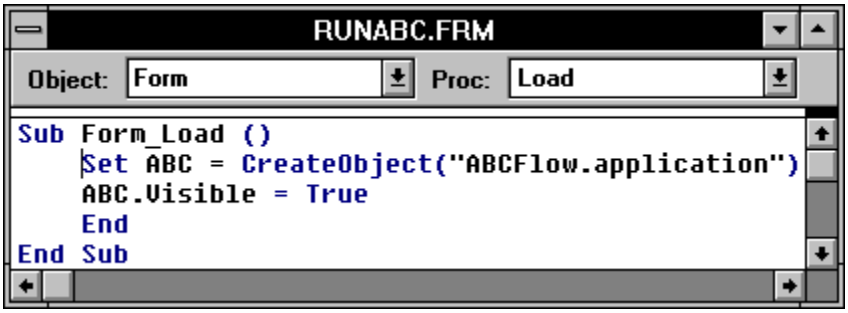
You run the application using the Visual Basic **Set** and **CreateObject** instructions. To make the application visible, you set the **Visible** property of the Application object to True.

Note

- You use these statements, along with the Dim, at the beginning of all OLE Automation programs.

Enter the following statements in the Load Proc of the Form Object.

```
Sub Form_Load ()
   Set ABC = CreateObject("ABCFlow.application")
   ABC.Visible = True
End
End Sub
```



Note

- If you do not want the program to end after FlowCharter runs, omit the End statement.

The statement `Set ABC = CreateObject("ABCFlow.application")` runs FlowCharter. When you run the program, FlowCharter runs invisibly, if it is not yet running. If it is already running, the statement has no effect, other than to establish a valid FlowCharter application object for use.

The statement `ABC.Visible = True` makes ABC visible.

The statement `End` ends the program.

Tip

You might want to save this program as RUNABC so you can open it as the beginning of your other programs. Be sure to save it with a different name before you make changes to it.

Run the program you have written so far from within Visual Basic. You see FlowCharter run. Then the Basic program comes back to the front. If you make the program into an EXE file and then run it, the program runs and stays in front.

Note

- If FlowCharter is already running, CreateObject does not run a new copy of FlowCharter. Instead, it returns a valid FlowCharter application object for the FlowCharter application that is already running.

Showing and Hiding FlowCharter

To show FlowCharter, you use this statement.

```
ABC.Visible = True
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

To hide FlowCharter, you use this statement.

```
ABC.Visible = False
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

If you set **Visible** to False, the application is still running, but it is not visible. You cannot switch to it using **ALT+TAB**, and it is not shown in the Task List dialog box that appears when you press **CTRL+ESC**. The default for **Visible** is False, so you should always set it to True immediately after you run FlowCharter.

You can use **Visible** on the right side of a statement and in conditions. The following statement sets the Boolean variable ABCShown to True or False according to the value of ABC.Visible.

```
ABCShown = ABC.Visible
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

The following statements beep your computer's speaker if FlowCharter is running but not visible.

```
If Not ABC.Visible Then
   Beep
End If
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

{button Related Topics,PI(`',`IDH_RT_Starting_ABC')}

Visible property

Controlling the FlowCharter Window

You can control the appearance of the FlowCharter window using many different properties and methods. Here are the items you can control.

- Display or hide the FlowCharter window (Visible.
- Make the FlowCharter window the active window (Activate.
- Change FlowCharter to an icon, maximize the window, and restore it to its previous state (Minimize, Maximize, Restore.
- Arrange minimized chart windows (ArrangeIcons).
- Cascade or tile the chart windows (CascadeCharts and TileCharts).
- Set the position and size of the FlowCharter window (Top, Left, Right, Bottom, Height, and Width).
- Set the title bar text of the FlowCharter window (Caption).
- Set the status bar visibility and text of the FlowCharter window (StatusBar, StatusBarVisible Property).
- Control whether the Field Viewer, Notes viewer, and Shape palette are visible in the FlowCharter window (FieldViewerVisible, NoteViewerVisible, and ShapePaletteVisible).
- Get information about FlowCharter (FullName, Path, Name, Version, and OperatingSystem).
- Set FlowCharter preferences (SmartShapeSpacing, SSSHorizontal, SSSVertical, ChannelAlignment, TouchAlignment, AlignToRulers, ShowRulers, LineSpacingX, LineSpacingY, LinkIndicator, LinkShadow, NoteIndicator, NoteShadow, NumberFont, ShowNodesOnLines, FieldPlacement, FieldNamesHidden, FieldsOpaque, FieldFont, FieldsHoursPerDay, and FieldsDaysPerWeek).
- Display Windows Help files (Help).
- Undo actions (UndoAvailable and Undo).
- Close FlowCharter (Quit).

Bringing FlowCharter or a Chart to the Front

You bring FlowCharter to the front using the **Activate** method of the Application object. You usually have to do this only after the user has done something that moves FlowCharter to the back, such as clicking another application that is visible on the screen or switching to another application using **ALT+TAB** or **CTRL+ESC**.

ABC.Activate

Note

- To do this in Living FlowChart script, change ABC. to Application.

The **Activate** method of the Application object is equivalent to clicking the FlowCharter window to bring it to the front, or using **ALT+TAB** or **CTRL+ESC** to bring it to the front.

You bring a particular chart to the front using the **Activate** method of the Chart object.

Chart.Activate

The **Activate** method of the chart object is equivalent to clicking the chart to bring it to the front or choosing the name of the chart from the Window menu.

{button Related Topics,PI(``,`IDH_RT_Bringing_ABC_or_a_Chart_to_the_Front')}

Controlling the FlowCharter Window
Activate method (Application object)
Activate method (Chart object)

Minimizing, Maximizing, and Restoring a Window

You can minimize the FlowCharter window and the chart windows to icons using the **Minimize** method of the Application object and Chart object. For example, the following statement makes FlowCharter into an icon at the bottom of the Windows screen.

ABC.Minimize

Note

- To do this in Living FlowChart script, change ABC. to Application.

The following statement makes the chart into an icon at the bottom of the FlowCharter window.

Chart.Minimize

After you minimize charts to icons, you can arrange them as described in Arranging FlowCharter Icons.

You can use the **Maximize** method of the Application object and Chart object to maximize the FlowCharter and chart windows. For example, the following statements maximize FlowCharter and then maximize the chart in it.

ABC.Maximize
Chart.Maximize

Note

- To do this in Living FlowChart script, change ABC. to Application.

After you change the size of the FlowCharter window or a chart window, you can change it to its previous size using the **Restore** method of Application object and Chart object. For example, the following statements restore the chart and then the FlowCharter windows to their previous sizes.

Chart.Restore
ABC.Restore

Note

- To do this in Living FlowChart script, change ABC. to Application.

You can resize the FlowCharter window as described in Positioning and Resizing the FlowCharter Window. You can cascade and tile charts as described in Arranging FlowCharter Charts.

{button Related Topics,PI(`',`IDH_RT_Minimizing_Maximizing_and_Restoring')}

Arranging FlowCharter Charts
Arranging FlowCharter Icons
Controlling the FlowCharter Window
Positioning and Resizing the FlowCharter Window

Maximize method
Minimize method
Restore method

Arranging FlowCharter Icons

When you have several FlowCharter chart windows minimized to icons, you can arrange them at the bottom of the FlowCharter window using the **ArrangeIcons** method of the Application object.

ABC.ArrangeIcons

Note

- To do this in Living FlowChart script, change ABC. to Application.

{button Related Topics,PI(`',`IDH_RT_Arranging_ABC_Icons')}

Controlling the FlowCharter Window
ArrangeIcons method

Arranging FlowCharter Charts

You can cascade and tile FlowCharter charts when more than one is open. You use the **CascadeCharts** method of the Application object to cascade the chart windows and the **TileCharts** method of the Application object to tile the chart windows.

ABC.CascadeCharts
ABC.TileCharts

Note

- To do this in Living FlowChart script, change ABC. to Application.

{button Related Topics,PI(`',`IDH_RT_Arranging_ABC_Charts')}

Positioning and Resizing the FlowCharter Window

You can specify the position and size of the FlowCharter window at any time. You use the **Top** property to set the top edge of the FlowCharter window, the **Left** property to set its left edge, the **Right** property to set its right edge, and the **Bottom** property to set its bottom edge. You can specify the height of the window using the **Height** property and the width of the window using the **Width** property. All these properties are in the Application object. See Minimizing, Maximizing, and Restoring Windows for more information on controlling the appearance of the FlowCharter window.

When you apply these properties to the FlowCharter window, you specify the position in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high.

You also can specify the height and width of the FlowCharter window. The following example places the FlowCharter window flush with the top and left edges of the screen, and extends it 400 pixels to the right and 500 pixels down from the top.

```
ABC.Top = 0
ABC.Left = 0
ABC.Width = 400
ABC.Height = 500
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

You can use the current values of **Top** , **Bottom** , **Left** , **Right** , **Height** , and **Width** in statements. The following example places the FlowCharter window in the top half of the screen.

```
Sub Main()
   Dim ABC As Object
   Dim MaxHeight As Integer, MaxWidth As Integer

   Set ABC = CreateObject("ABCFlow.application")

   ABC.Visible = False

   ABC.Maximize                ' Maximize ABC window
   MaxHeight = ABC.Height      ' Find height when maximized
   MaxWidth = ABC.Width        ' Find width when maximized
   ABC.Restore

   ABC.Top = -4                ' The -4 is for the window sizing border
   ABC.Left = -4               ' and can be different with other
   ABC.Width = MaxWidth + 4    '   resolutions and Windows setups
   ABC.Height = MaxHeight / 2

   ABC.Visible = True
End Sub
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Controlling the FlowCharter Window
Minimizing, Maximizing, and Restoring Windows

Bottom property
Height property
Left property
Right property
Top property
Width property

Changing the FlowCharter Title Bar

You can customize FlowCharter by changing what it says in the title bar. You set the test of the title bar using the **Caption** property of the Application object.

The following statement changes the FlowCharter title bar to *Custom Application* followed by a hyphen and the name of the active chart. FlowCharter appends a hyphen and then the name of the active chart to the caption.

```
ABC.Caption = "Custom Application"
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

You can use the current value of **Caption** in statements. For example, the following statements set the text in the title bar to *Second Window* if its value is *First Window*.

```
If ABC.Caption = "First Window" Then
   ABC.Caption = "Second Window"
End If
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

{button Related Topics,PI(`',`IDH_RT_Changing_the_ABC_Title_Bar')}

Controlling the FlowCharter Window
Caption property

Changing the FlowCharter Status Bar

You can use the text in the FlowCharter status bar to give hints and feedback to your users. You set the text in the status bar using the **StatusBar** property of the Application object. You determine and set the status bar's visibility using the **StatusBarVisible** property of the Application object.

Tip

- The status bar can hold approximately 90 characters, including spaces. The exact number it can hold depends on the characters. (For example, "m" takes up more room than "i" does.)

After running the following statements, if you change the pointer to the selection pointer and double click on a shape, the FlowCharter status bar text changes to "You double clicked!"

Note

- You must put the FlowCharter Events custom control in the Form window and put the following statement in the subroutine for double clicking.

```
Sub ABC1_DoubleClickSUBCLASS ()
    ABC1.App.StatusBar = "You double clicked!"
End Sub
```

- To do this in Living FlowChart script, add this line to the Object object OnDoubleClick event or Chart object OnObjectDoubleClick event script:

```
Application.StatusBar = "You double clicked!"
```

If you are writing an external Visual Basic program you also must add one of the statements below in the startup or initialization code of your program. See Registering Event Procedures. (If you are writing Living FlowChart script, you do not need to register events.)

If you are using FLOW.OCX, note that there is no extension for the OCXName:

```
ABC.RegisterEvent ABC1, Caption, "DoubleClickSUBCLASS"
```

If you are using ABCAUTO.VBX:

```
ABC.RegisterEvent ABC1.VBX, Caption, "DoubleClickSUBCLASS"
```

Note

The RegisterEvent method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.

If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

Note

- You can restore the normal status bar hints by setting the **StatusBar** property to "".

Enter the following statement after the **StatusBar** statement to specify that the action of double clicking will not also cause the normal response, such opening the Link dialog box or opening a linked chart. For more information on overriding events, see What Are FlowCharter Events? and Events in Visual Basic.

```
ABC1.Override = True            ' use this line if you want to override the
                                ' application event from an externally written
                                ' Visual Basic program
```

or

```
Application.Override = True     ' use this line if you want to override the
                                ' application event from within Living FlowChart
```

```
                                        ' script
```

or

```
Allow.Value = False                     ' use this line if you want to override a Chart
                                        ' object event or an Object object event
```

You can use the current value of the **StatusBar** property in statements. For example, the following statements in an external Visual Basic program add text to the status bar if its value is First Window.

```
If ABC.StatusBar = "First Window" Then
   ABC.StatusBar = "First Window" + " View"
End If
```

To do this in Living FlowChart script, the following statements add text to the status bar if its value is First Window

```
If Application.StatusBar = "First Window" Then
   Application.StatusBar = "First Window" + " View"
End If
```

{button Related Topics,PI(``,`IDH_RT_Changing_the_ABC_Status_Bar')}

Displaying the Field Viewer, Notes Viewer, and Shape Palette

You can control whether the Field Viewer, Notes viewer, QuickZoom window, and Shape palette are visible in the FlowCharter window. The properties you use are **FieldViewerVisible**, **NoteViewerVisible**, **ZoomWindowVisible**, and **ShapePaletteVisible**. You set them to True or False to show or hide the windows.

Enter the following statement to an external Visual Baisc program to make the Field Viewer visible if fields are defined.

```
ABC.FieldViewerVisible = True
```

To do this in Living FlowChart script, the following statement shows the Field Viewer:

```
Application.FieldViewerVisible = True
```

Enter the following statement to an external Visual Baisc program to make the Field Viewer invisible.

```
ABC.FieldViewerVisible = False
```

To do this in Living FlowChart script, the following statement hides the Field Viewer:

```
Application.FieldViewerVisible = False
```

To make the other windows visible or invisible, substitute **NoteViewerVisible**, **ZoomWindowVisible**, or **ShapePaletteVisible** for **FieldViewerVisible**.

The **FieldViewerVisible** property is equivalent to Field Viewer on the View menu.

The **NoteViewerVisible** property is equivalent to clicking Note on the View menu.

The **ZoomWindowVisible** property is equivalent to clicking QuickZoom on the View menu.

The **ShapePaletteVisible** property is equivalent to clicking Shape Palettes on the View menu.

You can use the current value of the window properties in statements. For example, the following statements in an external Visual Baisc program make the Shape palette invisible if the Note viewer is visible.

```
If ABC.NoteViewerVisible = True Then
   ABC.ShapePaletteVisible = False
End If
```

To do this in Living FlowChart script:

```
If Application.NoteViewerVisible = True Then
   Application.ShapePaletteVisible = False
End If
```

Getting FlowCharter System Information

You can get information about the FlowCharter program that is running using the properties **FullName**, **Path**, **Name**, **Version**, and **OperatingSystem** of the Application object.

Note

You cannot change the values of the **FullName**, **Path**, **Name**, **Version**, or **OperatingSystem** properties.

The **FullName** property returns the fully qualified path of the FlowCharter program that is running, including the name of the executable file.

For example, the following statement puts the fully qualified path of the running FlowCharter in the status bar. The result when you run the program might be a status bar entry of "Full path is C:\Program Files\Micrografx\FlowCharter\FLOW70.EXE."

```
ABC.StatusBar = "Full path is " + ABC.FullName + "."
```

To do this in Living FlowChart script:

```
ABC.StatusBar = "Full path is " + ABC.FullName + "."
```

The **Path** property returns the fully qualified path of the FlowCharter application that is running, excluding the name of the executable file. For example, the following statement puts the fully qualified path of the running FlowCharter in the status bar. The result when you run the program might be a status bar entry of "Path is C:\Program Files\Micrografx\FlowCharter."

```
ABC.StatusBar = "Path is " + ABC.Path + "."
```

To do this in Living FlowChart script:

```
Application.StatusBar = "Path is " + Application.Path + "."
```

The **Name** property returns the name of the application running. It always equals "FlowCharter" for compatibility with all FlowCharter products. For example, the following statement puts "FlowCharter" in the status bar.

```
ABC.StatusBar = ABC.Name
```

To do this in Living FlowChart script:

```
Application.StatusBar = Application.Name
```

The **Version** property returns the version of the OLE Automation application object that is running. For example, it equals "7.0" if you are running FlowCharter. The following statement puts the version number in the status bar.

```
ABC.StatusBar = "ABC OLE Automation Version " + ABC.Version + "."
```

To do this in Living FlowChart script:

```
Application.StatusBar = "OLE Automation Version " + Application.Version + "."
```

The **OperatingSystem** property returns a value according to the DOS and Windows version under which FlowCharter is running. For example, it equals "DOS 6.21;Windows 3.11" if you are running those versions. The following statement puts the operating system description in the status bar.

```
ABC.StatusBar = "OS: " + ABC.OperatingSystem + "."
```

To do this in Living FlowChart script:

```
Application.StatusBar = "OS: " + Application.OperatingSystem + "."
```

{button Related Topics,PI(`',`IDH_RT_Getting_ABC_System_Information')}

Controlling the FlowCharter Window

FullName property
Path property
Name property
Version property
OperatingSystem property

Customizing Preferences

You can use OLE Automation to set many of the FlowCharter preferences that you can set from the application. The object that contains the preferences is Preferences. You use all the preference properties in the same way. Using the preference properties is described at the end of this section.

{button Related Topics,PI(``,`IDH_RT_Customizing_Preferences')}

Alignment Options

The alignment preference properties are **SmartShapeSpacing**, **SSSHorizontal**, **SSSVertical**, **ChannelAlignment**, **TouchAlignment**, **AlignToRulers**, and **ShowRulers** of the Preferences object.

The following table shows the possible values of the alignment preference properties.

Property	Values
SmartShapeSpacing	True = Selected; False = Not Selected
SSSHorizontal	Number
SSSVertical	Number
ChannelAlignment	True = Selected; False = Not Selected
TouchAlignment	True = Selected; False = Not Selected
AlignToRulers	0 = Not Selected; 1 = Coarse; 2 = Fine
ShowRulers	True = Selected; False = Not Selected

{button Related Topics,PI(``,`IDH_RT_Alignment_Options')}

Customizing Preferences
Setting Preferences

AlignToRulers property
ChannelAlignment property
ShowRulers property
SmartShapeSpacing property
SSSHorizontal property
SSSVertical property
TouchAlignment property

Line Options

The line spacing options in the preference properties are **LineSpacingX** and **LineSpacingY** of the Preferences object, and **LineCrossoverStyle** and **LineCrossoverSize** of the Chart object.

The following table shows the possible values of the line spacing options preference properties.

Property	**Values**
LineSpacingX	Value
LineSpacingY	Value

The following table shows the possible values of the **LineCrossoverStyle** property.

Style	**Description**
0	▪ Solid lines
1	▪ Bunny hops
2	▪ Broken lines

The following table shows the possible values of the **LineCrossoverSize** property.

RelativeSize	**Description**
0	▪ Small
1	▪ Medium
2	▪ Large

{button Related Topics,PI(`',`IDH_RT_Line_Options')}

Customizing Preferences
Setting Preferences

LineCrossoverSize property
LineCrossoverStyle property
LineSpacingX property
LineSpacingY property

Indicator Options

The indicator options are **LinkIndicator**, **LinkShadow**, **NoteIndicator**, **NoteShadow**, **NumberFont**, and **ShowNodesOnLines**. These options are part of the Shape object, not the Preferences object.

The following table shows the possible values of the indicator properties.

Property	**Values**
LinkIndicator	String ("" by default)
LinkShadow	True = Selected; False = Not Selected
NoteIndicator	String ("-N" by default)
NoteShadow	True = Selected; False = Not Selected
NumberFont	Font object
ShowNodesOnLines	True = Selected; False = Not Selected

{button Related Topics,PI(``,`IDH_RT_Indicator_Options')}

Customizing Preferences
Setting Preferences

LinkIndicator property
LinkShadow property
NoteIndicator property
NoteShadow property
NumberFont property
ShowNodesOnLines property

Field Options

The field options are **FieldPlacement**, **FieldNamesHidden**, **FieldsOpaque**, **FieldFont**, **FieldsHoursPerDay**, and **FieldsDaysPerWeek**. These options are part of the Chart object, not the Preferences object.

The following table shows the possible values of the indicator properties.

Property	Values
FieldPlacement	0 = Left, 1 = Right, 2 = Above, 3 = Below, 4 = Inside Top, 5 = Inside Middle
FieldFont	Object
FieldNamesHidden	True = Selected; False = Not Selected
FieldsOpaque	True = Selected; False = Not Selected
FieldsHoursPerDay	Number
FieldsDaysPerWeek	Number

For more information on the data field preferences, see Setting Data Field Preferences.

{button Related Topics,PI(``,`IDH_RT_Field_Options')}

Customizing Preferences
Setting Data Field Preferences
Setting Preferences

FieldFont property
FieldNamesHidden property
FieldPlacement property
FieldsOpaque property
FieldsHoursPerDay property
FieldsDaysPerWeek property

Setting Preferences

You set all preferences in approximately the same way. You access the information about preferences using the **Preferences** property of the Application object. For example, you can set **AlignToRulers** to fine.

```
ABC.Preferences.AlignToRulers = 2
```

To turn aligning to rulers off, set the value to 0. To set the alignment to coarse, set the value to 1. You set the other preferences in the same way by changing **AlignToRulers** to the appropriate property and changing the value to what you want.

You set preferences that are part of the Chart object instead of the Preferences objects slightly differently. For example, you can turn on **FieldNamesHidden**.

```
ABC.Chart.FieldNamesHidden = True
```

To do this in Living FlowChart script:

```
Application.Chart.FieldNamesHidden = True
```

You can use the current value of preferences in statements. For example, the following statements put the current setting of the alignment preference in the status bar.

```
If ABC.Preferences.AlignToRulers = 0 Then
   ABC.StatusBar = "Alignment: Off"
ElseIf ABC.Preferences.AlignToRulers = 1 Then
   ABC.StatusBar = "Alignment: Coarse"
Else
   ABC.StatusBar = "Alignment: Fine"
End If
```

To do this in Living FlowChart script:

```
If Application.Preferences.AlignToRulers = 0 Then
   Application.StatusBar = "Alignment: Off"
ElseIf Application.Preferences.AlignToRulers = 1 Then
   Application.StatusBar = "Alignment: Coarse"
Else
   Application.StatusBar = "Alignment: Fine"
End If
```

You use the other preferences in the same way by changing **AlignToRulers** to the appropriate property and changing the values as appropriate.

{button Related Topics,PI(`',`IDH_RT_Setting_Preferences')}

Customizing Preferences

Alignment Options
Field Options
Indicator Options
Line Options

AlignToRulers property
FieldNamesHidden property
Preferences property

Setting Defaults

You can set the defaults for objects that are in the Shape, Line_, and TextBlock objects. You use the **SetDefaults** method of the Chart object. You set the defaults by passing the method an object that has the defaults you want to set.

The following table lists the defaults that you set when you use the **SetDefaults** method.

Object Type	Defaults Set
Shape	Border color, border style, border width, fill color, fill pattern, shadow offset, shadow color, numbers on or off, font properties, text alignment
Line_	Color, width, style, source arrow size, source arrow style, source arrow color, destination arrow size, destination arrow style, destination arrow color
TextBlock	Font properties, text alignment

For example, the following statements first create an object that has shape numbering turned on, has a dark gray fill, and has a dark gray shadow to its lower right. Then the statements set the Shape defaults using that statement. Finally the statements delete the object using the **Clear_** method of the Object object.

```
Set DefaultObj = Chart.DrawShape          ' Create an object to have defaults
DefaultObj.NumberShown = True             ' Assign defaults
DefaultObj.FillColor = ABC.GRAY
DefaultObj.ShadowColor = ABC.DK_GRAY
DefaultObj.ShadowStyle = 1
Chart.SetDefaults DefaultObj              ' Set defaults
DefaultObj.Clear_                         ' Delete the object
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

You set the defaults for Line_ and TextBlock objects similarly.

{button Related Topics,PI(``,`IDH_RT_Setting_Defaults')}

Customizing FlowCharter

You can customize FlowCharter in several ways. You can add and remove menus using methods and properties of the Application object, the Menu collection, and the MenuItem object. To make it easy to run other programs, you can add and remove buttons in the toolbox using the methods of the Application object. To provide feedback to the user, you can display messages in dialog boxes, show a percent-complete gauge, and change the pointer to an hourglass using methods and properties of the Application object.

None of these methods and properties have equivalents in FlowCharter.

{button Related Topics,PI(`',`IDH_RT_Customizing_ABC')}

Adding Menus

You can add as many new menus to FlowCharter as you want. If you add more than fit on the menu bar, it wraps so all the menus fit. It is best not to have too many menus so the user can easily find the menu he or she wants to use. Each menu should contain commands related to the same type of activity.

You add a menu to FlowCharter using the **AddMenu** method of the Application object. The menu is added to FlowCharter at the left of the Window menu, so you set the order of the menus by the order in which you create them. The menu is added to the Menu collection. The **AddMenu** method requires three parameters and allows a fourth parameter.

The first parameter is the title of the menu. For example, you might give it the title "&Costs" (the & underlines the next character and makes it the mnemonic).

The second parameter identifies the OCX or VBX that receives notification events when the menu is used. Normally you use ABC1.VBX if you are using ABCAUTO.VBX and ABC1 if you are using FLOW.OCX. This parameter registers menus for the **AppMenuSUBCLASS** event. When the OCX or VBX shuts down (when the program ends), the menu is removed from FlowCharter.

The third parameter is the name of the program adding the menu to FlowCharter. The easiest way to identify the program is using Form1.Caption.

The fourth parameter, which is optional, lets you specify a chart type for the menu. A chart type is a hidden string field up to eight characters in length indicating the chart type. This field is never used within FlowCharter, but it is useful within FlowCharter events OCX/VBX. For example, if two OLE Automation programs are running, you could change the fourth parameter to avoid conflicts.

For example, the following statements create a menu named Costs with a mnemonic of C, register it with the **AppMenuSUBCLASS** event by specifying ABC1.VBX if you are using ABCAUTO.VBX or ABC1 if you are using FLOW.OCX. Then specify the program using Form1.Caption. The menu object is placed in the variable Menu.

```
Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
ABC.New                                          ' Create a new chart

Dim CostMenu As Object

Set CostMenu = ABC.AddMenu("&Costs", ABC1, Form1.Caption)
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

It is generally best to add any menus that will be necessary at the beginning of a session and leave them until FlowCharter closes. All added menus are automatically removed when FlowCharter closes. Changing menus during a session can be disconcerting to a user.

If you wish, you can change the name of the menu after you create it using the **Text** property of the Menu collection. For example, the following statement changes the name of a menu to Cost with a mnemonic of C.

```
CostMenu.Text = "&Cost"
```

You can temporarily make a menu invisible and then show it again using the **Visible** property of the Menu collection. For example, the following statement hides a menu.

```
CostMenu.Visible = False
```

If you need to remove a menu while FlowCharter and your program are running, you can use the **RemoveMenu** method of the Application object. For example, the following statement removes the menu that was created with the title "&Costs" from FlowCharter and from the Menu collection.

```
ABC.RemoveMenu "&Costs"
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

After you create a menu, you can add items to it. You add items below any existing items using the **AppendItem** method and add items in specific places using the **InsertItem** method, both of the Menu collection.

If you use the name of an existing menu, the methods return the existing MenuItem object. Otherwise they return the new MenuItem object.

With the **AppendItem** method, you provide the title of the item you wish to create. For example, the following statements create the menu item "Overruns" with a mnemonic of O and place it in the OverrunItem object.

```
Dim OverrunItem As Object
Set OverrunItem = CostMenu.AppendItem("&Overruns")
```

With the **InsertItem** method, you provide the title of the item you wish to create, as with the **AppendItem** method, followed by the position of the item. You can specify the position by giving the name of the existing item that the new item should be placed before or by specifying the numerical position of the item. All other items are shifted down. For example, the following statements create the menu item "Explanation" with a mnemonic of E after the existing menu item "&Overruns."

```
Dim ExplanationItem As Object
Set ExplanationItem = CostMenu.InsertItem("&Explanation", "&Overruns")
```

The following statements create the menu item "Overtime" with a mnemonic of V in the second position in the menu.

```
Dim OvertimeItem As Object
Set OvertimeItem = CostMenu.InsertItem("O&vertime", 2)
```

With both methods, you can use the title "-" to create a separator (a solid horizontal line) to divide items into logical groups.

You indicate the status of some types of menu items with a check mark in front of the item. You use the **Checked** property of the MenuItem object to set and remove check marks. For example, the following statement puts a check mark to the left of a menu item.

```
CostItem.OvertimeItem.Checked = True
```

When a menu item is not available because choosing it is inappropriate in the current situation, you make the item gray. You use the **Enabled** property of the MenuItem object to gray a menu item. The

following statement turns an item gray.

```
OvertimeItem.Enabled = False
```

If you wish, you can change the name of a menu item after you create it using the **Text** property of the MenuItem object. For example, the following statement changes the name of a menu item to Overtime with a mnemonic of T.

```
OvertimeItem.Text = "Over&time"
```

You can remove an item from a menu using the **DeleteItem** method of the Menu collection. For example, the following statement removes an item from a menu.

```
CostMenu.DeleteItem OvertimeItem
```

If you wish, you can delete all the items from a menu using the **DeleteAll** method of the Menu collection.

```
CostMenu.DeleteAll
```

You can find the menu item that is in a specific position using the **Item** method of the Menu collection. You specify either the position of the item in the menu or the text in the item. For example, the second statement puts into CurrentItem the Menu object that is in the third position in the menu. The third statement puts into NextItem the Menu object that has the text Over&time.

```
Dim CurrentItem As Object, NextItem As Object
Set CurrentItem = CostMenu.Item (3)
Set NextItem = CostMenu.Item("Over&time")
```

{button Related Topics,PI(`',`IDH_RT_Adding_Menus')}

Customizing FlowCharter

AppMenuSUBCLASS event

AddMenu method
AppendItem method
DeleteItem method
InsertItem method
Item method
RemoveMenu method

Checked property
Enabled property
Text property
Visible property

Adding Other Applications to the Menu

You can the names of other applications to the Add On submenu of the Tools menu so you can run other programs easily from FlowCharter, both programs you have written and commercial programs, such as Microsoft Excel.

You create the menu items using the **CreateAddOn** method of the Application object.

The first parameter is the position of the menu item. Use -1 for the first available position.

The second parameter is the hintline text. For example, if you enter a second parameter of "Run Excel" (with no punctuation), the hint line is "Click to Run Excel." (with "Click to" before it and a period after it).

The third parameter is the name of the program to run, including the fully qualified path. If the path contains a long filename, the string must be contained within quote marks.

The fourth parameter is no longer used.

A fifth parameter is the name you want for the menu item. If no title is specified for the menu item, the hintline text is used. If there is no hint name, the name of the executable file is used (including extension.)

When you use the **CreateAddOn** method, an item is added to the submenu accessed from the Add Ons item on the Tools menu. Add-ons are stored in the Registry automatically and loaded automatically when the application is closed and opened. Add-ons can be created only through OLE automation. You can specify the text of the menu item. There is no limit to the number of add-ons. For example, the following command adds Excel to the Add Ons submenu.

```
ABC.CreateAddOn(3,"Run Excel", "C:\OFFICE95\Excel\Excel.exe", "","Excel")
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

The menu items you create using the **CreateAddOn** method appear each time FlowCharter is run until you remove them using the **RemoveAddOn** method of the Application object.

With the **RemoveAddOn** method, you specify either the position on the menu or the name of the menu item. For example, the line below removes the menu item for Excel regardless of its position in the toolbox.

```
ABC.RemoveAddOn "Excel"
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

{button Related Topics,PI(`',`IDH_RT_Adding_Buttons')}

Providing Feedback

You can provide feedback to your users in several ways, from as simple a thing as changing the cursor to indicate that the user should wait for a moment to more complex things such as posting a gauge that shows how far an operation has progressed, showing a hint line, and posting a dialog box that the user must respond to.

When you have an operation that will take a long time (anything approaching a second) and the user cannot usefully click somewhere (such as on Cancel), it is customary to change the cursor. You can change the cursor to the wait cursor using the **Hourglass** property of the Application object.

```
ABC.Hourglass = True
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Posting a gauge that shows the progress of an operation uses the **PercentGauge** method, the **PercentGaugeValue** property, the **HidePercentGauge** method, and the **PercentGaugeCancelled** method, all of the Application object.

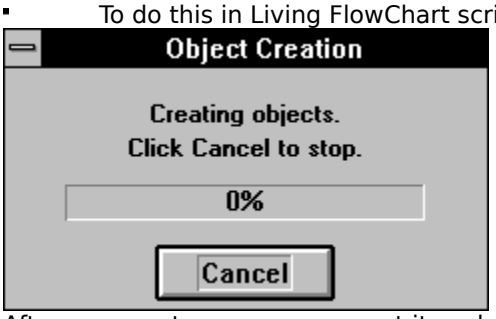
You create a percent gauge, with its value set to 0, using the **PercentGauge** method. It takes three optional parameters. The first is the name that goes in the title bar. The second is the first line of text above the gauge. The third is the second line of text above the title bar.

For example, the following statement creates the gauge shown.

```
ABC.PercentGauge "Object Creation", "Creating objects.", "Click Cancel to stop."
```

Note

- To do this in Living FlowChart script, change ABC. to Application.



After you create a gauge, you set its value using the **PercentGaugeValue** property. Set it equal to a number from 0 to 100 to have the gauge show the appropriate position. For example, if the operation is 53% complete, the following statement makes the gauge show that value.

```
ABC.PercentGaugeValue = 53
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

You check to see if the user has chosen the Cancel button in the gauge using the **PercentGaugeCancelled** method. For example, the following statement sets the value of CancelCreation to True or False depending on whether the user has chosen the Cancel button.

```
CancelCreation = ABC.PercentGaugeCancelled
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

After the operation is complete, or if the user clicks Cancel, you need to remove the gauge. You do that using the **HidePercentGauge** method.

```
ABC.HidePercentGauge
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Most often, the value for the **PercentGaugeValue** property is a variable that you compute immediately before changing the value of the gauge. For example, the following statements determine the completion value and change the gauge only if the percentage is different. To avoid slowing the loop, it only redraws the gauge when the percentage has changed by at least 1. This example also shows the use of the **PercentGaugeCancelled** method and the **HidePercentGauge** method.

```
ABC.PercentGauge "Object Creation", "Creating objects.", "Click Cancel to stop."

OldPercentDone = 0
CreateCount = 1000

For Creation = 1 to CreateCount

   CancelCreation = ABC.PercentGaugeCancelled      ' Cancelled?
   If CancelCreation Then
      ABC.HidePercentGauge                         ' Get rid of gauge
      Exit For                                     ' Leave creation loop
   End If

   Chart.DrawShape                                 ' Create the shape

   PercentDone = Int(Creation / CreateCount * 100) ' Find percentage done
   If PercentDone <> OldPercentDone Then           ' Has percentage changed?
      ABC.PercentGaugeValue = PercentDone          ' Set gauge
      OldPercentDone = PercentDone                 ' Reset comparison value
   End If

Next Creation

ABC.HidePercentGauge
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

You can show a hint line using the **Hint** method of the Application object. You usually use a hint line to describe a menu, command, or button you have added, or to give information about a percentage gauge or dialog box you have posted. You most often use the events **AppMenuHintSUBCLASS**, **AppMenuSUBCLASS**, and **AppMenuPopupSUBCLASS**. For example, the following line puts the line "Click Cancel to stop creation." in the hint line.

```
ABC.Hint "Click Cancel to stop creation."
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Note that the hint line you place only stays until the user moves the mouse so that a different hint line appears. If you want to make the hint line stay until you change it to "" you should use the **StatusBar** method.

You can post a dialog box using the **MsgBox** method of the Application object. The method is similar to the MsgBox function used in the Basic programming language, with three parameters. The first parameter is the message that goes in the dialog box.

The second parameter, which is optional, defines the type of dialog box. If you omit the second parameter, the value is 0.

The third parameter, which is optional, sets the title bar text of the dialog box. If you omit the third parameter, the title of the dialog box is "Micrografx FlowCharter 7."

The value of the second parameter can be the sum of values from the table. For example, if you want to show a Stop icon along with Yes and No buttons, the value of the second parameter is 19. The easiest way to set the values is to use the constants and sum them into a variable. The following table shows the values available.

Note

- In Visual Basic, these values have constants associated with them, such as MB_OK. Those constants are not available for OLE Automation.

Value	Effect
0	Display the OK button only.
1	Display the OK and Cancel buttons.
2	Display the Abort, Retry, and Ignore buttons.
3	Display the Yes, No, and Cancel buttons.
4	Display the Yes and No buttons.
5	Display the Retry and Cancel buttons.
16	Display the stop icon.
32	Display the question mark icon.
48	Display the exclamation point icon.
64	Display the information icon.
0	The first button is the default.
256	The second button is the default.
512	The third button is the default.
0	The dialog box is application modal, so FlowCharter is suspended until the user responds to the dialog box.
4096	The dialog box is system modal, so all applications are suspended until the user responds to the dialog box.

For example, the following statements set the type and then create the dialog box with a title of "Cancel Creation."

```
MessageBoxType = 4 + 16                                    ' Show Yes, No, and Stop icon
Response = ABC.MsgBox "You cancelled the creation. Are you sure?", MessageBoxType, "Cancel Creation"
```

Note

- To do this in Living FlowChart script, remove ABC. from MsgBox.

The value put into the variable Response depends on the button that the user clicked. The following table shows the value of the **MsgBox** method according to the button that the user selected.

Note

- In Visual Basic, these values have constants associated with them, such as IDOK. Those constants are not available for OLE Automation.

Button Selected	Value
OK	1
Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

{button Related Topics,PI(``,`IDH_RT_Providing_Feedback')}

Customizing FlowCharter

AppMenuHintSUBCLASS
AppMenuPopupSUBCLASS
AppMenuSUBCLASS

HidePercentGauge method
MsgBox method
PercentGaugeCancelled method
PercentGauge method
StatusBar method

Hourglass property
PercentGaugeValue property

Displaying Help

You can display help at any time that you wish, based on the actions of your user. You use the **Help** method of the Application object to display the help topic of your choice.

There are two optional parameters you can use with the **Help** method. If you use the **Help** method with no parameters, FlowCharter help appears showing the default topic.

The first parameter, a text string, specifies a Windows help file. You can use it to specify a help file other than the one shipped with FlowCharter, so that you can direct your users to a help file that you prepared for your particular application.

The second parameter, a long or a text string, is a context ID or help context string to call a particular topic in the help file.

The following statement opens the Help window with the Shape Tool topic displayed because its context ID is 71681.

```
ABC.Help, 71681
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

If you write your own help, then the statement is something like this.

```
ABC.Help "C:\Program Files\Micrografx\FlowCharter
   \FLOW70.EXE\MYHELP.HLP", "Getting_Started"
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Using the **Help** method is equivalent to positioning the pointer or opening a dialog box, and then pressing **F1**.

{button Related Topics,PI(`',`IDH_RT_Displaying_Help')}

Help method

Closing FlowCharter

When you wish, you can close FlowCharter using the **Quit** method of the Application object.

When you use the **Quit** method, FlowCharter closes. It does not prompt the user to save changes on open files. Before you close FlowCharter, you should save the files you want to be saved.

The following statement closes FlowCharter.

```
ABC.Quit
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

Using the **Quit** method is equivalent to opening the File menu and choosing Exit, or using any of the other methods to close FlowCharter, except that it does not prompt for saving changes.

{button Related Topics,PI(``,`IDH_RT_Closing_ABC')}

Quit method

Window Handles

Using window handles is an advanced feature of OLE automation. Window handles are useful for calling the Windows API calls directly from OLE Automation. For more information, open the Visual Basic 3.0 Help and search for "hWnd."

You can get the handle of the windows within FlowCharter, including the handle for the main window, the Field Viewer, the Note viewer, and the Shape palette. The properties associated with the windows are **WindowHandle** (of the Application object for FlowCharter and of the Chart object for a specific chart), **FieldViewerWindowHandle**, **NoteViewerWindowHandle**, and **ShapePaletteWindowHandle** of the Application object. These properties provide the handle to each of the windows. If the window is not visible, they are Null.

{button Related Topics,PI(`',`IDH_RT_Closing_ABC')}

FieldViewerWindowHandle property
NoteViewerWindowHandle property
ShapePaletteWindowHandle property
WindowHandle property

Creating New Charts

You can create a new chart using default attributes or attributes that were saved in a template.

In FlowCharter, you create a new chart with default attributes by clicking New on the File menu. A new chart window opens.

You can create a new chart using the attributes in a specific template by clicking Open on the File menu. The Open File dialog box opens. In the Files of type list box, select Micrografx FlowCharter Template (*.aft), choose the drive, directory, and template file you want to open, and click Open. Click Save As on the File menu, enter a new name in the File name box, in the Files of type list box, select Micrografx FlowCharter (*.flo), and then click OK.

To create a new chart using OLE Automation, you use the **New** method or the **NewFromTemplate** method of the Application object or the **Add** method or the **AddFromTemplate** method of the Chart collection.

Use the **New** method or the **Add** method to create a new chart with default attributes. This opens a new chart window. For example, the following statements each create a new chart, resulting in two new charts.

```
ABC.New
Charts.Add
```

Use the **NewFromTemplate** method or **AddFromTemplate** method to create a new chart with attributes based on the chart template's name. For example, the following statements each open a new chart based on the template PYRAMIDT.AFT.

```
ABC.NewFromTemplate "C:\Program Files\Micrografx\
    FlowCharter\Template\Cool Sheets\PyramidT.aft"
Chart.AddFromTemplate "C:\Program Files\Micrografx\
    FlowCharter\Template\Cool Sheets\PyramidT.aft"
```

There is no practical difference in the effect of **New** and **Add** or in the effect of **NewFromTemplate** and **AddFromTemplate**.

Add method

AddFromTemplate method

New method

NewFromTemplate method

Opening Charts

Each chart is stored in a separate file, which contains the shapes, lines, and text in your chart. Chart filenames end with an AF3, AF2, or FLO extension.

In FlowCharter, you open a chart by opening the File menu and choosing Open. The Open dialog box appears, and you can choose the drive and directory that contain the file you want to open.

Using OLE Automation, you open a chart using the **Open** method of the Charts collection or the **Open** method of the Application object. With each, you specify a fully qualified pathname or partial pathname. If you specify a partial pathname (just the name of the file, for example), the path is the current value of the **DefaultFilePath** property. If the chart is already open, the **Open** method moves the chart to the front. You can optionally specify that the chart is to be opened read only.

For example, the following statement opens the file MYCHART.FLO, located in the path specified in the **DefaultFilePath** property. The file is opened as read only.

```
ABC.Open "MYCHART.ABC", True
```

Identifying a Chart's Filename

Setting a Default Path for Charts

DefaultFilePath property

Open method

Setting a Default Path for Charts

You use the **DefaultFilePath** property of the Application object to set the default path for all files that are opened or saved. For example, the following statement sets the default path for all files, and then opens a read-only chart without specifying a pathname.

```
ABC.DefaultFilePath = "C:\Program Files\Micrografx\FlowCharter\Samples"
ABC.Open MYCHART.ABC, True
```

Opening Charts

DefaultFilePath property

Identifying a Chart's Filename

You can find a chart's filename with or without its pathname. The **FullName** property of the Chart object returns the fully qualified pathname of the chart. (If the chart has not been saved, it returns the temporary name of the chart.) The **Name** property of the Chart object returns the name of the chart without the path. You access the information about charts using the **Charts** property of the Application object.

For example, the following statements return the fully qualified pathname and the name without the path of a chart.

```
FullyQualifiedPathname = Chart.FullName
NameOnly = Chart.Name
```

{button Related Topics,PI(``,`IDH_RT_Identifying_a_Chart_s_Filename')}

Saving Charts

In FlowCharter, when you save a chart, FlowCharter stores the chart in a file on disk. Each chart is saved in a separate file. When you save a chart, you can assign it a name and choose where you want to store it on a disk.

In OLE Automation, you can find if the chart has been saved to disk and if the file on disk is the same as the file in memory. You can save the current version of the chart to a specified or default pathname.

You use the **HasDiskFile** property of the Chart object to find out if the chart has ever been saved to disk. For example, the following statement puts into the variable EverSaved whether the current chart has a file on disk.

```
EverSaved = Chart.HasDiskFile
```

You can use the **Saved** property of the Chart object to find if the file saved on disk is the same chart as currently resides in memory. If the value of the **Saved** property is True, there is no need to save the chart. For example, the following statement puts into the variable FileCurrent whether the current chart has been changed since it was last changed.

```
FileCurrent = Chart.Saved
```

You use the **Save** method of the Chart object to save a file. You can optionally specify a path and filename. If you only specify a filename, the pathname is the value of the **DefaultFilePath** property.

You also can optionally specify the type of file to save it as. The following table shows the types of files possible.

File Type	Save File As
0	Chart, version 3.0, 4.0, and Toolkit
1	Template, version 3.0
2	Chart, version 2.0
3	Template, version 2.0

For example, the following saves a file as a template with the name MYTEMPL.AFT.

```
Chart.Save "MYTEMPL.AFT", 1
```

It is not necessary to specify a file type when saving FlowCharter files. Just specify the file name with the extension .FLO (for example, MYFILE.FLO) and the file will automatically be saved as a FlowCharter file.

You can use the **HasDiskFile** property, **Saved** property, and **Save** method together to save a file only when necessary. For example, the following statements save a file with a new name if it has never been saved, or save it with its current name if it has been changed since it was last saved.

```
EverSaved = Chart.HasDiskFile                    ' Is chart on disk?
If Not EverSaved Then                            ' If not, save it
   NextFile = "File" + ChartFileCount + ".ABC"   ' Create filename
   ABC.Hint "Saving chart as " + NextFile + "."
   Chart.Save NextFile
```

```
   ChartFileCount = ChartFileCount + 1                ' Increment file counter
Else                                                  ' Else, check if changed
   FileCurrent = Chart.Saved
   If Not FileCurrent Then                            ' If changed, save it
      ABC.Hint "Saving chart."
      Chart.Save
   End If
End If
```

You can use the **Export** method of the Chart object to export the chart to a graphics file. The file extension you provide determines the type of file created. This is the equivalent of clicking Export on the File menu in FlowCharter. Quotation marks should be used whenever long filenames or long pathnames are used.

ChartObject.**Export** (*FileName*)

{button Related Topics,PI(`',`IDH_RT_Saving_Charts')}

Read-Only Charts

Reverting to the Last Saved Version

DefaultFilePath property

Export Method

HasDiskFile property

Saved property

Save method

Reverting to the Last Saved Version

You use the **RevertToSaved** method of the Chart object to revert to the last saved copy of the document, discarding any changes. For example, the following statement removes the current version of the chart from memory and opens the version on the hard disk.

Chart.RevertToSaved

Saving Charts

RevertToSaved method

Read-Only Charts

Some charts are opened as read-only charts, either because they were opened that way using the **Open** method or because the person does not have rights to save the chart under the same filename. You can determine whether a chart is read only using the **ReadOnly** property of the Chart object. For example, the following statement puts into the variable ReadOnlyFile whether the user can save the file under its current path and filename.

```
ReadOnlyFile = Chart.ReadOnly
```

The **ReadOnly** property is read only. You cannot change its value. To open a file as read only, use the optional *AsReadOnly* parameter of the **Open** method of the Charts collection or of the Application object.

{button Related Topics,PI(``,`IDH_RT_Read_Only_Charts')}

Closing Charts

You can close just the active chart or close all of your charts at once. In FlowCharter, you close the open charts in the order that they are arranged on the screen. If the open chart contains changes that you have not saved, FlowCharter displays a message asking if you want to save the changes.

In OLE Automation, you use the **CloseChart** method of the Chart object to close the chart.

Note

- When you use the **CloseChart** method, the user does not get a prompt to save the chart. For example, the following statement closes a chart without any prompt to the user.

```
Chart.CloseChart
```

Closing All Charts at Once

In FlowCharter, you close all the open charts by opening the File menu and choosing Close All. If any of the charts contain unsaved changes, FlowCharter asks if you want to save the changes.

In OLE Automation, you use the **CloseAll** method of the Application object or the **CloseAll** method of the Charts collection to close all charts in the FlowCharter workspace.

Note

- If any of the charts contain unsaved changes, FlowCharter does not ask if you want to save the changes.

For example, the following two statements each close all the open charts.

```
ABC.CloseAll
ChartCollection.CloseAll
```

{button Related Topics,PI(`',`IDH_RT_Closing_Charts')}

CloseChart method

Activating a Chart

Activating a chart lets you return a previously created chart. In FlowCharter, you activate a chart by clicking on it or by opening the Window menu and choosing the chart from the numbered list of open charts.

In OLE Automation, you can activate a chart using the **Activate** method of the Application object and the **Item** method of the Charts collection. You can find the current active chart using the **ActiveChart** property of the Application object.

The **Item** method lets you identify the chart you want to bring to the front. It takes one parameter, which is either a string indicating the full path and executable name of the chart or a number that is the chart's ordering position within the collection. For example, the following statement brings the chart C:\Program Files\Micrografx\FlowCharter\Samples\MYCHART.ABC to the front and places the chart object in ActiveChart.

```
ActiveChart = ABC.Charts.Item("C:\Program Files\Micrografx\FlowCharter\Samples\
MYCHART.ABC").Activate
```

If the chart is not open, the method returns a nonvalid chart object.

The **Count** property of the Charts collection contains the number of charts in the collection. (The **Count** property exists in several collections. All of them work approximately the same way.) You can use the **Count** property to loop through the open charts. For example, the following statements search through the chart collection looking for the chart MYCHART.FLO and bring it to the front when it is found.

```
For ChartCount = 1 to ChartsCollection.Count          ' Search collection
   Set CurrentChart = ABC.Item(ChartCount)
   If CurrentChart.Name = "MYCHART.ABC" Then
        Exit For                                      ' Exit when chart is found
   End If
Next ChartCount
CurrentChart.Activate                                 ' Activate chart
```

{button Related Topics,PI(`',`IDH_RT_Activating_a_Chart')}

Activate method

ActiveChart property

Count property

Item method

Protecting Charts

At times, you may want to prevent other people from editing your chart. After you assign a password to protect the chart, no one is able to move, edit, add, or delete objects in the chart until they enter the password correctly.

In FlowCharter, you can use password protection to manage linked files. By assigning each person in a work group a different password, you can ensure that each person has access to make changes only to his or her own charts.

In OLE Automation, you use the **SetProtection** method of the Chart object to turn the protection on and off. The **SetProtection** method has two parameters. The first is a Boolean that turns protection on and off. The second is the password. For example, the following statements turn protection on, and then turn it back off.

```
Chart.SetProtection True, "Fred"
Chart.SetProtection False, "Fred"
```

You use the **Protected** property of the Chart object to identify whether a chart is protected. The **Protected** property is read only, so you cannot use it to change the protection of a chart.

For example, the following statements turn off password protection for a chart if the chart is protected.

```
If CurrentChart.Protected Then
    CurrentChart.SetProtection False, CurrentChartPassword
End If
```

Protected property

SetProtection method

Linking Charts

You can link charts together. After the charts are linked, you can double click a designated shape in one chart to open the linked chart automatically.

{button Related Topics,PI(``,`IDH_RT_Linking_Charts')}

Linking Shapes to Other Charts

In FlowCharter, you link an object, such as a shape, to an active chart by clicking the Selector tool in the toolbox, selecting the shape you want to link to another chart, then clicking the Link button on the Standard toolbar. In the Link dialog box you identify the chart to which you want to link and choose an action.

In OLE Automation, you can link shapes to other charts, determine if a shape is linked to another chart, and link to field data from another chart.

You use the **LinkedChartName** property of the Shape object to provide the full pathname of a chart linked to an object and link the shape to the file. For example, the following statement links a shape to a chart.

```
MyShape.Shape.LinkedChartName = "C:\Program Files\Micrografx\FlowCharter\Samples\
LINKCHT.ABC"
```

You use the **IsLinked** property of the Shape object to find if a shape is linked to another chart. The **IsLinked** property is read only. The property returns True if the object contains a link to another chart.

```
ShapeLinked = MyShape.Shape.IsLinked
```

You use the **LinkFields** property of the Shape object to accumulate the linked chart's data into the shape's field information, if the shape is linked to another chart with field information. For example, the following statement turns on putting field data from the linked chart into the shape's field data.

```
MyShape.Shape.LinkFields = True
```

Creating Group Links

The group and link function lets you move a group of selected objects to another chart, and replace the moved group with a shape that is linked to the chart to which the group was moved.

In FlowCharter, you group and link by selecting the objects you want to move to another chart, and then clicking the Link button on the Standard toolbar. In the Link dialog box you identify the chart to which you want to link, and choose Group and Link. The moved group is replaced by the currently selected shape in the Shape Palette.

In OLE Automation, you group and link with the **GroupAndLink** method of the Chart object.

The **GroupAndLink** method returns the shape that replaced the moved group and has two optional parameters. The first parameter specifies the full pathname of the new chart. If the first parameter is omitted, FlowCharter generates a default chart pathname. The second parameter specifies whether the new chart's fields are linked to the source chart. If the second parameter is omitted, FlowCharter does not link the fields. Use a variable with a True value for the second parameter to link the fields.

After executing **GroupAndLink**, you can obtain the newly created chart object with the **ActiveChart** property of the Application object.

The example shown below moves the selected objects to a chart named LINKCHT.FLO. The LINKCHT.FLO fields are not linked to the source chart.

```
Set ShapeGroupLink = GroupAndLink("C:\Program Files\Micrografx\FlowCharter\Samples\
LINKCHT.ABC")
```

Linking Charts

ActiveChart property

GroupAndLink method

Opening a Linked Chart

You can use a linked shape to open a linked chart. In FlowCharter, you open a linked chart by double clicking the linked shape. The linked chart opens and becomes the active chart.

In OLE Automation, you use the **Link** method of the Shape object to open the linked chart. For example, the following statement opens the chart attached to a shape and puts the linked chart into LinkedChart.

```
Set LinkedChart = MyShape.Shape.Link
```

Note

- If there is no value in the **LinkedChartName** property, using the **Link** method creates a new chart with an automatically generated filename.

{button Related Topics,PI(``,`IDH_RT_Opening_a_Linked_Chart')}

Linking Charts

Link method

LinkedChartName property

Choosing Link Indicators

The link indicators appear in linked shapes. You can specify indicators for the linked shapes and place a shadow on objects with linked files.

In OLE Automation, use the LinkIndicator property of the Chart object to specify the indicator, up to three characters, used for linked shapes. Use the LinkShadow property of the Chart object to show a shadow on shape objects that have linked files. For example, the following statements specify a link indicator of LNK and show a shadow.

```
Chart.LinkIndicator = "LNK"
Chart.LinkShadow = True
```

{button Related Topics,PI(`',`IDH_RT_Choosing_Link_Indicators')}

Linking Charts

Linking EXEs to Charts

You can link a compiled Visual Basic EXE program file to a chart so that the EXE program runs automatically when you open the chart. This feature is illustrated by the Deployment Wizard sample program shipped with OLE Automation. The Deployment Wizard automatically links DEPLOY.EXE to every new chart that you create with the Deployment Wizard.

You use the **TypeRequiresEXE** and **TypeUsesEXE** properties of the Chart object to link an EXE to a chart.

- If you set the **TypeRequiresEXE** property to True, the chart requires the EXE to open. If the linked EXE cannot be run, then the chart does not open.
- If you set the **TypeUsesEXE** property to True, then the chart attempts to run the linked EXE when it opens. If the EXE cannot be run, the chart still opens, after FlowCharter displays a warning.

The name of the EXE that is linked to a chart by these properties is determined by the **Type** property of the chart. The EXE name is constructed by adding .EXE to the chart **Type**. In the case of the Deployment Wizard sample program, for example, the chart **Type** is DEPLOY. Therefore, the EXE linked to a new chart created by the Deployment Wizard is DEPLOY.EXE.

The following sample code specifies the chart **Type** and links an EXE to the chart. If the CHARTTYPE variable is set to "DEPLOY," then this code sample links the chart to DEPLOY.EXE.

```
Chart.Type = CHARTTYPE
Chart.TypeRequiresEXE = True
```

If you set either **TypeRequiresEXE** or **TypeUsesEXE** to True in a program, then you also must ensure that you close all charts of that **Type** when your program closes. You use the **ChartTypeShutdown** method of the Application object to close the charts. The following code sample, located in the Form.QueryUnload procedure of your program, closes all charts of the **Type** CHARTTYPE.

```
ABC.ChartTypeShutdown CHARTTYPE, APPNAME
```

Note

- FlowCharter only runs one instance of a linked EXE. When a second chart that is linked to an already running EXE is loaded, FlowCharter refers to the currently running EXE. It does not load a second copy of the EXE.

ChartTypeShutdown method

Type property

TypeRequiresEXE property

TypeUsesEXE property

Launching Applications

You can launch other Windows applications from within FlowCharter. Launching lets you open other applications without using the Program Manager. It also lets you easily send information about a chart and shape to an application, such as a database, that is prepared to receive it.

{button Related Topics,PI(``,`IDH_RT_Launching_Applications1')}

Setting Shapes to Launch Applications

To set a shape to have a launch, you enter the program it is to launch. To launch an application, you use shapes in the Chart object with attached launches.

In FlowCharter, you set a shape to launch an application by selecting the shape you want to use and clicking the Link button on the Standard toolbar. In the Link dialog box, you specify the command line, directory, and flags.

In OLE Automation, you use the **LaunchCommand** property of the Shape object to set a command to launch for the object.

For example, the following statement sets a shape to launch Excel.

```
CurrentChart.Shape.LaunchCommand = "C:\EXCEL\EXCEL.EXE"
```

Launching Applications

LaunchCommand property

Launching Applications

You use the shape you set for launching to launch the application. An indicator appears on shapes set for linking or launching.

In FlowCharter, to launch an application, you double click the shape you set for launching.

In OLE Automation, you use the **Launch** method of the Shape object to execute the shape's launch. You identify whether a shape is set to launch an application using the **IsLaunched** property of the Shape object. For example, the following statements check to see if a shape has a launch and, if it does, they launch the application and put a Boolean value in LaunchSuccessful to indicate whether the launch succeeded.

```
If CurrentChart.Shape.IsLaunched Then
   LaunchSuccessful = CurrentChart.Shape.Launch
End If
```

Printing Charts

In FlowCharter, when you open the File menu and choose the Print command, the Print dialog box opens. You can print all the pages, a range of pages, or the selected objects in the chart.

In OLE Automation, you use the **PrintOut** method of the Chart object to print the chart object. The parameters for the **PrintOut** method specify the options to use when printing.

Parameter	**Description**
FromPage	Integer (default is page 1)
ToPage	Integer (default is last page)
NumberOfCopies	Integer (default is 1)
FitToPage	Integer (Boolean) (default is False)
PrintNotes	Integer (Boolean) (default is False)

For example, the following statement prints a chart using the default parameters.

```
Chart.PrintOut
```

You use the **PrintSelected** method of the Chart object to print the selected objects in the chart. The parameters for the **PrintSelected** method indicate the options to use when printing.

Parameter	**Description**
NumberOfCopies	Integer (default is 1)
FitToPage	Integer (Boolean) (default is False)
PrintNotes	Integer (Boolean) (default is False)

For example, the following statement prints the selected objects in a chart using the default parameters.

```
Chart.PrintSelected
```

You use the **Printer** property of the Application Object to specify the current printer, the one to use when printing. When you read the value of the **Printer** property, it returns the current printer and port. For example, it might return "HP LaserJet III on LPT2:."

When you set the value, the program uses a "loose matching" routine so, for example, setting the **Printer** property to "HP Laser" or "LPT2" chooses "HP LaserJet III on LPT2:" if that is the printer on LPT2:. If more than one printer matches the value you set, the exact match is used first. If there is not exact match, the first one alphabetically is used. For example, the following statement sets the printer to the first printer available on LPT1:.

```
ABC.Printer = "LPT1:"
```

You use the **PrintBlankPages** property of the PageLayout object to specify whether a blank page should be printed if there are no objects on the page. For example, the following statement will print pages even if they are blank when you use the **PrintOut** method.

```
Chart.Layout.PrintBlankPages = True
```

In FlowCharter, you can click Print Preview on the View menu to see a preview of the page. In OLE Automation, you use the **PrintPreview** method of the Chart object to do this.

```
Chart.PrintPreview
```

{button Related Topics,PI(`',`IDH_RT_Printing_Charts')}

PrintBlankPages property

Printer property

PrintOut method

PrintPreview Method

PrintSelected method

Adjusting the Page Layout

Page layout options affect the orientation and dimensions of the pages in a chart. To adjust the page layout, you specify the object, then specify the properties of the drawing area and page.

In FlowCharter, you open the File menu and choose the Page Layout command. The Page Layout dialog box opens. You use this dialog box to choose paper size, orientation, page margins, units of measure, print order, and whether to print blank pages.

In OLE Automation, you can specify the object and various options. The following table shows the properties of the PageLayout object and their meanings. You use the **PageLayout** property of the Chart object to access the PageLayout object.

Property	**Meaning**
Height	Height of the drawing area
Width	Width of the drawing area
MarginBottom	Bottom margin of the page
MarginLeft	Left margin of the page
MarginRight	Right margin of the page
MarginTop	Top margin of the page
Orientation	Portrait (0) or landscape (1)
PageHeight	Height of the page
PageWidth	Width of the page
PageOrder	Across, then down  (0) *or* down, then across  (1)
PaperSize	Size of the paper to be printed

The **PaperSize** property uses a "loose matching" routine when you are setting the value. For example, setting the **PaperSize** property to "let" chooses "Letter 8 1/2 x 11 in."

You use the **PageCount** property of the Chart object to set the number of pages in a chart.

For example, the following statements set the drawing height to 8.5 inches, the drawing width to 11 inches, the margins to 0.5 inches on all sides, the orientation to landscape, the page height to 8.5 inches, the page width to 11 inches, the page order to across-then-down, and the paper size to "Letter 8 1/2 x 11 in" and sets the chart to have four pages.

```
Chart.Layout.Height = 8.5
Chart.Layout.Width = 11
Chart.Layout.MarginBottom = .5
Chart.Layout.MarginLeft = .5
Chart.Layout.MarginRight = .5
Chart.Layout.MarginTop = .5
Chart.Layout.Orientation = 1
Chart.Layout.PageHeight = 8.5
Chart.Layout.PageWidth = 11
Chart.Layout.PageOrder = 0
Chart.Layout.PaperSize = "Letter"
Chart.PageCount = 4
```

PageLayout property

PaperSize property

Height property

Width property

MarginBottom property

MarginLeft property

MarginRight property

MarginTop property

Orientation property

PageHeight property

PageWidth property

PageOrder property

Displaying Master Items

You can define and display useful pieces of information in a chart by displaying Master Items.

In FlowCharter, you click Chart Properties on the Format menu and then click the Master Items tab. In this dialog box you choose whether to display the chart name, page numbers, the date and time, and a logo. You also can enter one or two text lines, possibly for use as header and footer text, can choose the format for the date, and can choose whether master items appear on the first page or on all pages.

In OLE Automation, you can specify the same information using properties of the MasterItems object. You access the information about master items using the **MasterItems** property of the Chart object. The following table shows the properties of the MasterItems object and their meanings.

Property	Meaning
ChartName	Chart name master item object
ChartNameShown	Whether the chart name master item is shown (Boolean)
Date	Date master item object
DateShown	Whether the date master item is shown (Boolean)
DateStyle	MM/DD/YY (0), short text (1), long text (2)
Logo	Logo master item (the Logo property is read only, but the properties from the object it returns are read/write)
LogoPathname	Fully qualified pathname of the logo
LogoShown	Whether the logo master item is shown (Boolean)
PageNumber	Page number master item object
PageNumberShown	Whether the page number master item is shown (Boolean)
Range	First page only (0) or all pages (1)
Text1	Text1 master item (the Text1 property is read only, but the properties from the object it returns are read/write)
Text1Shown	Whether the text1 number master item is shown (Boolean)
Text2	Text1 master item (the Text2 property is read only, but the properties from the object it returns are read/write)
Text2Shown	Whether the text2 number master item is shown (Boolean)
Time	Time master item object
TimeShown	Whether the time master item is shown (Boolean)

The following table shows the methods of the MasterItems object and their meanings.

Method	Meaning
HideAll	Hide all master items
ShowAll	Show all master items
UpdateDateAndTime	Update the date and time to the system date and time or to a specified date and time

The following statements show the date and time, with the date in long text format, and show a first line of text in bold.

```
Chart.MasterItems.DateShown = True
Chart.MasterItems.DateStyle = 2
Chart.MasterItems.TimeShown = True
Chart.MasterItems.Text1.Text = "First line of text."
Chart.MasterItems.Text1.Font.Bold = True
Chart.MasterItems.Text2Shown = True
```

Viewing a Chart

Scrolling through a chart lets you display all areas of the chart. In OLE Automation, you can scroll through a chart by specifying the left and top points in the chart. You can scroll to a specific page or location.

You use the **ScrollLeft** property of the Chart object to set the left point visible in the chart and the **ScrollTop** property of the Chart object to set the top point visible in the chart.

You use the **ScrollPage** method of the Chart object to scroll the chart to a particular page and the **ScrollPosition** method of the Chart object to scroll to a location in the chart by specifying a vertical and horizontal position.

You use the **View** property of the Chart object to view a particular page of the document. The following table shows the parameters for the **View** property and their meanings.

Value	Description
0	One to one
1	Current page
2	Used pages
3	Percentage zoom

For example, the following statements change the view to show the current page and then go to the second page in the chart.

```
Chart.View = 1
Chart.ScrollPage 2
```

You use the **ZoomPercentage** property of the Chart object to change the magnification of the current document. You can set the view to any value from 5% to 1600% of the actual size of the objects in the chart.

For example, the following statement changes the view to show the chart at 200% of its actual size

```
Chart.ZoomPercentage = 200
```

ScrollLeft property

ScrollPage method

ScrollPosition method

ScrollTop property

View property

ZoomPercentage property

Giving a Presentation

The Full Screen feature in FlowCharter lets you show charts as "slides" in a presentation easily, without the distracting menus and buttons. The Full Screen command is on the View menu. With OLE Automation, you use the **FullScreen** method of the chart object. Use the **CancelFullScreen** method of the chart object to return the chart to the previous view.

In FlowCharter, you can use linked charts to move from one "slide" (chart) to another by double clicking the linked shape. With OLE Automation, you can give a slide show by showing successive charts at the full screen view, delaying each one for a few seconds.

For example, the following statements show two charts on the full screen, with an appropriate delay routine between them, and then return to the previous view.

```
Chart1.FullScreen
[Delay routine]
Chart2.FullScreen
Chart2.CancelFullScreen
```

CancelFullScreen method

FullScreen method

Using Guidelines

You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline. Guidelines let you align shapes of different sizes for an attractive, organized look.

In FlowCharter, you drag guidelines from the rulers. If the Snap to Grid option is selected in the Tools Options Alignment dialog box, guidelines snap to ruler position. The guidelines do not appear in the printed chart.

In OLE Automation, you can toggle the guidelines, add horizontal and vertical guidelines, and clear all guidelines.

You use the **GuidelinesOn** property of the Chart object to turn showing guidelines on and off. You use the **AddHorizontalGuideline** method of the Chart object to add a horizontal guideline at the vertical position passed. Use the **AddVerticalGuideline** method of the Chart object to add a vertical guideline at the horizontal position passed. Use the **ClearGuidelines** method of the Chart object to remove all guidelines from the chart.

For example, the following statements create a horizontal guideline four inches down from the top, create a vertical guideline three inches over from the left, turn showing guidelines on, and then remove all guidelines from the chart.

```
Chart.AddHorizontalGuideline 4
Chart.AddVerticalGuideline 3
Chart.GuidelinesOn = True
Chart.ClearGuidelines
```

AddHorizontalGuideline method

AddVerticalGuideline method

ClearGuidelines method

GuidelinesOn property

Defining Measurement Units for a Chart

The measurement units for a chart specify the size and distance values. In FlowCharter, the units are determined by the paper size you select. However, you can explicitly set the measurement units in FlowCharter by dragging the Inches or Centimeters button from the View category in the Customize dialog box to a toolbar, and then clicking it.

In OLE Automation, you use the **Units** property of the Chart object to specify the units for measurement in the chart object and all its child chart objects. In addition, the **Units** property specifies the size and distance values passed in the Preferences object. The default unit value is 0 (inches) for each new Preferences object.

Value	Description
0	Inches
1	Centimeters

For example, the following statement sets the measurement unit to centimeters.

```
Chart.Units = 1
```

Units property

Sending Electronic Mail

In FlowCharter, you can attach the current chart to an e-mail message so you can send it using a MAPI e-mail system such as Microsoft Mail. You bring up the e-mail system by choosing Send on the FlowCharter File menu.

In OLE Automation, you use the **SendMail** method of the Chart object to create a new e-mail message with a Chart object attached. The user must then address the mail to the appropriate person and can add a message.

For example, the following statement creates a new e-mail message with the chart object as an attachment.

Chart.SendMail

{button Related Topics,PI(``,`IDH_RT_Sending_Electronic_Mail')}

SendMail method

Identifying an Object

You can identify objects using the **Type** and **UniqueID** properties of the Object object and the **ShapeName** property of the Shape object. You access the information about objects using the **Objects** property of the Chart object.

The Type property contains a value that specifies the type of the object. You cannot change the value of the Type property. The following table shows the values and their meanings.

Value	Meaning
0	Shape
1	Line
2	Text
3	Bitmap
4	OLE client object
5	Master item

By determining the type of object, you can limit the operations you perform on it. For example, if the value of the **Type** property is 1 (line), then you would not set the font size.

The **UniqueID** property returns a unique identifier that you can use to choose an object in the Objects collection using, for example, the **ItemFromUniqueID** method. The identifier is unique for each object in each chart. If you wish, you could create a database containing the **UniqueID** property values for all the objects in a chart to make it easy to identify and act on each of them.

The **ShapeName** property contains the name of the shape, such as "Process" or "Decision." You cannot change the value of the **ShapeName** property.

After you find the type of shape for the shape you want, you could, for example, use the information to create another shape of the same type. The following statements find the name of the current shape and then create another shape of the same type using the **DrawShape** method.

```
CurrentShape = Shape.ShapeName
Set SameShape = Chart.DrawShape(CurrentShape)
```

DrawShape method
ItemFromUniqueID method

Objects property
ShapeName property
Type property
UniqueID property

Finding the Number of Items

You can use OLE Automation to find how many objects there are and how many objects of different types are selected.

{button Related Topics,PI(``,`IDH_RT_Finding_the_Number_of_Items')}

Finding the Total Number of Objects

You can find how many objects there are using the **Count** property of the Objects collection. For example, you can use this property to post a message in the hint line telling how many objects are in a chart. You cannot change the value of the **Count** property.

The following statement sets ObjectCount to the number of objects in a chart.

```
ObjectCount = ABC.ActiveChart.Objects.Count
```

When you know how many objects are in a chart, you can specify them by number using the **Item** method in the Objects collection. For example, the following statements turn all the objects in a chart green.

```
Dim ChartObjects As object
Set ABC = CreateObject("ABCFlow.application")
ABC.New                                          ' Create a new chart
Set ChartObjects = ABC.ActiveChart.Objects
For ItemCount = 1 to ChartObjects.Count
   ChartObjects.Item(ItemCount).Color = ABC.GREEN
Next ItemCount
```

Note

- The statements above turn the objects, including TextBlocks, but not the text inside a shape, green. To turn the text inside a shape green, use the **Color** property described in **Setting Text Colors**.

{button Related Topics,PI(``,`IDH_RT_Finding_the_Total_Number_of_Objects')}

Finding the Number of Objects Selected

Sometimes it is useful to know the number of objects that are selected in a chart. You can use the **SelectedObjectCount**, **SelectedShapeCount**, **SelectedLineCount**, and **SelectedOtherCount** properties in the Chart object to find out how many objects in a chart are selected. You cannot change the values of any of these properties except by selecting or deselecting objects.

The **SelectedObjectCount** property contains the total number of selected objects in the chart. It equals the sum of the values of the **SelectedShapeCount**, **SelectedLineCount**, and **SelectedOtherCount** properties.

For example, the following statement sets TotalSelected to the number of objects in the chart.

```
TotalSelected = Chart.SelectedObjectCount
```

The **SelectedLineCount** property contains the number of selected lines, not the number of selected line segments, so the routing of the lines does not affect the count.

The **SelectedShapeCount** property contains the number of selected shapes.

The **SelectedOtherCount** property contains the number of objects selected that are not shapes or lines. It includes TextBlock objects, master item objects such as the date and headers, OLE objects, bitmaps, and other objects pasted into FlowCharter.

Finding the Number of Items

SelectedLineCount property
SelectedObjectCount property
SelectedOtherCount property
SelectedShapeCount property

Finding Objects in a Chart

You can use the ItemFrom methods to find one or more objects in a chart that meet a criterion.

You use the **ItemFromAll**, **ItemFromShapes**, **ItemFromLines**, **ItemFromSelection**, **ItemFromText**, **ItemFromFieldValue**, **ItemFromAttachments**, **ItemFromNumber**, and **ItemFromUniqueID** methods in the same way. These methods are all in the Object Collections object. The following table shows the parameters you specify with each of the methods, the object that it returns, and an example of the method.

Method	Parameters	Return Object	Example
ItemFromAll	None	Object	ItemFromAll()
ItemFromShapes	None	Shape object	ItemFromShapes()
ItemFromLines	None	Line object	ItemFromLines()
ItemFromSelection	None	Selected object	ItemFromSelection()
ItemFromText	Text	Object containing text	ItemFromText("Buy")
ItemFromFieldValue	Field template object and value	Object with value in the field	ItemFromFieldValue (Field1,1200)
ItemFromAttachments		One or two objects or line object	Attached shape, text, ItemFromAttachments (NewObj1,NewObj2)
ItemFromNumber	Shape number	Shape with the number	ItemFromNumber(3)
ItemFromUniqueID	Unique ID	Object with that ID	ItemFromUniqueID (7)

The **ItemFromAll**, **ItemFromShapes**, **ItemFromLines**, and **ItemFromSelection** methods do not take any parameters. They return all objects, shape objects, line objects, and selected objects, respectively.

The **ItemFromText** method returns objects that contain the text you specify.

The **ItemFromFieldValue** method returns objects with the value in the specified field.

The **ItemFromAttachments** method returns the objects that are attached to the one or two objects you specify. For example, if you specify two shapes, this method would return the line connecting them.

The **ItemFromNumber** method returns the shape with the number you specify.

The **ItemFromUniqueID** method returns the object with the unique identifier you specify. You can find the identifier using the **UniqueID** property of the Object object.

For example, the following statement sets the unique identifier of an object into the variable CurrentID.

```
CurrentID = NewObj1.UniqueID
```

The **Valid** property, found in the Chart object, contains a Boolean value based on whether the current object is valid or not. You normally use the **Valid** property in the While portion of a Do While loop to ensure that only valid objects are used.

Note

- By default, the current object is valid unless set otherwise.

To use the ItemFrom methods, you use them in a loop, most often a Do While loop. Each time the loop executes, the method returns the next object, so you can test the objects for a property value and act on the objects that meet that value. For example, the following changes to red all shapes that have the word "Buy" in them.

```
Sub Form_Load ()
    Dim ABC As object
    Dim ChartObjects As object
    Dim TestObject As object
    Set ABC = CreateObject("ABCFlow.application")
    ABC.New                                          ' Create a new chart
    Set ChartObjects = ABC.ActiveChart.Objects
    Do
        Set TestObject = ChartObjects.ItemFromText("Buy")
        TestObject.Shape.FillColor = ABC.RED
    Loop While TestObject.Valid
    End
End Sub
```

Note

- If you change the line `TestObject.Shape.FillColor = ABC.RED` to `TestObject.Color = ABC.RED`, this example also turns master item text to red if it contains the word "Buy."

If you wish, you can reset all searches to start at the beginning of the items in the chart using the **ResetSearch** method, found in the Objects collection.

```
ChartObjects.ResetSearch
```

{button Related Topics,PI(`',`IDH_RT_Finding_Objects_in_a_Chart')}

ItemFromAll method
ItemFromAttachments method
ItemFromFieldValue method
ItemFromLines method
ItemFromNumber method
ItemFromSelection method
ItemFromShapes method
ItemFromText method
ItemFromUniqueID method

ResetSearch method

Valid property

Selecting Objects in a Chart

The ItemFrom methods described in the previous section of this chapter let you identify objects in a chart so you can make changes to them, but they do not select them, and only the **ItemFromSelection** method makes any note of whether objects are selected.

You can use the **Selected** property in the Object object to determine if an object is selected. For example, the following statements turn an object black if it is selected.

```
If NewObj1.Selected Then
   NewObj1.Color = ABC.BLACK
End If
```

You also can use the **Selected** property to select an object. For example, the following statement selects the specified object.

```
NewObj1.Selected = True
```

You can use the **Select** method of the Chart object to select and deselect a group of objects. The following table shows the action of the values.

Value	Action
0	Selects all shapes in addition to anything already selected
1	Selects all lines in addition to anything already selected
2	Selects everything
3	Deselects everything

Values of 0, 1, and 2 in the **Select** method are the equivalent of opening the Select dialog box from the Select command on the FlowCharter Edit menu and choosing Shapes, Lines, or All.

You can use the **SelectShapeType** method in the Chart object to select all shapes of a specific type. For example, the following statement selects all Decision (diamond) shapes in addition to any objects already selected.

```
Chart.SelectShapeType("Decision")
```

You can deselect all objects using the **DeselectAll** method of the Chart object. The **DeslectAll** method has the same effect as the **Select** method with a value of 3.

```
Chart.DeselectAll
```

DeselectAll method
ItemFromSelection method
Select method
SelectShapeType method

Selected property

Moving Objects

You move objects using the **Top**, **Bottom**, **Left**, **Right**, **CenterX**, and **CenterY** properties of the Object object. You also can use those properties to find the location of an object.

Note

- These properties move objects, but do not resize them. Resize objects with **Width** and **Height**.

The **Top**, **Bottom**, **Left**, and **Right** properties describe the location of the specified side of the object. The **CenterX** and **CenterY** properties describe the horizontal and vertical positions of the center of the object. For example, the following statements set the top of the object to two inches from the top of the page and set the center of the object three inches from the left side of the page.

```
Object.Top = 2
Object.CenterX = 3
```

The following statements check to see if the center of the object is within one inch of the upper left corner of the page. If it is, the object is moved so its bottom and left sides are two inches from the top and left side of the page.

```
If Object.CenterX <= 1 and Object.CenterY <= 1 Then
   Object.Bottom = 2
   Object.Left = 2
End If
```

You can use the **FlippedVertical** and **FlippedHorizontal** properties of the Object object to specify that selected objects are flipped.

*ObjectObject***.FlippedVertical** = {True | False}

You can use the **Rotation** property of the Object object to specify that the selected objects are rotated clockwise in a 90 degree increment.

*ObjectObject***.Rotation** = *Value*

Rotation values are defined in the following table.

Value	Amount of Rotation
0	0
1	90
2	180
3	270

Bottom property
CenterX property
CenterY property
FlippedHorizontal Property
FlippedVertical Property

Height property
Left property
Right property
Rotation Property
Top property

Width property

Arranging Objects

You can arrange objects using the **SpaceEvenly** method or the **Align** method of the Chart object. These are the equivalent of using the Space Evenly or Align commands on the Arrange menu or buttons on the Arrange toolbar.

The **SpaceEvenly** method lets you space objects either across or down, based on their centers or edges.

ChartObject.**SpaceEvenly** (*Direction*)

Value	Action
0	Space evenly across, centers
1	Space evenly down, centers
2	Space evenly across, edges
3	Space evenly down, edges

The **Align** method lets you align selected objects based on any edge, their vertical centers or their horizontal centers.

ChartObject.**Align** (*By*)

Value	Action
0	Align, left
1	Align, centers
2	Align, right
3	Align, top
4	Align, middle
5	Align, bottom

{button Related Topics,PI(``,`IDH_RT_Arranging_Objects')}

Align Method
SpaceEvenly Method

Resizing Objects

You can resize objects using the **StretchType**, **Height**, and **Width** properties in the Object object. You use the StretchType property to specify the type of stretching behavior assigned to an object. Set the value to 0 for normal behavior (the anchor is the center of the object and opposite sides both move, as when you stretch normally in FlowCharter). Set the value to 1 for fixed side behavior in FlowCharter (the anchor is the opposite side from the handle grabbed, as when you hold down the **Ctrl** key and stretch in FlowCharter). If you set the value to 1 and then resize the object with OLE Automation statements, the top and left sides are fixed (as if you were stretching from the right or bottom center handle).

For example, the following statements set the **StretchType** property to fix the top and left sides and then set the height to 1.25 inches and the width to 2.5 inches.

```
Object.StretchType = 1
Object.Height = 1.25
Object.Width = 2.5
```

You also can use the **MakeSameSize** method of the Chart object to make all selected objects the same height according to width, height, both, or text. It is equivalent to choosing one of the MakeSameSize options on the Arrange menu or toolbar in FlowCharter.

ChartObject.**MakeSameSize** (*AccordingTo*)

The following table describes the values for the **MakeSameSize** method.

Value	According To
0	Width
1	Height
2	Both
3	Fit to Text

Height property
MakeSameSize Method
StretchType property
Width property

Changing the Display Order of Objects

You can change the order in which objects display using the **ToBack** and **ToFront** methods. The methods are in both the Chart object and the Object object.

The **ToBack** method with an object is equivalent to clicking the Send To Back button on the Arrange toolbar with one or more objects selected.

The **ToFront** method with an object is equivalent to clicking the Bring To Front button on the Arrange toolbar with one or more objects selected.

The first statement below moves the selected objects to the back. The second statement moves the object to the back.

Chart.ToBack
Object.ToBack

The first statement below moves the selected objects to the front. The second statement moves the object to the front.

Chart.ToFront
Object.ToFront

{button Related Topics,PI(``,`IDH_RT_Changing_the_Display_Order_of_Objects')}

ToBack method
ToFront method

Setting the Current Drawing Position

When you draw using FlowCharter, you click where you want to place the next object, text, or line. With OLE Automation, the position is determined by the most recent draw position (plus **DrawSpacingX** and **DrawSpacingY** for shapes; see **Drawing Shapes** for more information). You can specify a location using the **DrawPositionX** and **DrawPositionY** properties in the Chart object. The position you specify is used for the next object drawn or the next object pasted or pasted special (if those methods do not specify a different position).

For example, the following statements specify that the center of the next object is to be four inches from the left side of the page and five inches from the top of the page.

```
Chart.DrawPositionX = 4
Chart.DrawPositionY = 5
```

Drawing Shapes

DrawPositionX property
DrawPositionY property
DrawSpacingX property
DrawSpacingY property

Cutting, Copying, and Pasting Objects

You use the **Cut**, **Copy**, and **Paste** methods in the Chart object just as you would open the FlowCharter Edit menu and choose the Cut, Copy, and Paste command. All three methods return a Boolean value equal to True if the operation was successful or False if the operation failed.

The **Paste** method has the additional ability to specify where to place the pasted object.

Paste([*HorizontalLocation*] [, *VerticalLocation*])

For example, the following statements cut whatever is currently selected and paste it so its upper left corner is two inches from the left margin and three inches from the top margin of the page.

```
Successful = Chart.Cut
If Successful Then
   Successful = Chart.Paste(2,3)          ' Paste if cut is successful
End If
If Not Successful Then                    ' Deal with cut or paste failure
   ABC.MsgBox "Cut or Paste Failed", 48    ' 48 is the exclamation point
End If
```

If you omit the location for the **Paste** method, it places the object as described in Setting the Current Drawing Position.

Setting the Current Drawing Position

Copy method
Cut method
Paste method

Using Special Clipboard Formats

The **PasteSpecial** method in the Chart object lets you paste from the Clipboard specifying a format. It is the same as choosing the Paste Special command in the FlowCharter Edit menu and then specifying the format to use for the paste. The method returns a Boolean value equal to True if the operation was successful or False if the operation failed.

PasteSpecial (*Format* [, *AsIcon*] [, *HorizontalLocation*] [, *VerticalLocation*])

You also can specify that the object on the Clipboard be pasted as an icon using the second parameter. This is equivalent to selecting the Display As Icon option in the FlowCharter Paste Special dialog box. You can specify the location of the paste. If you omit the location for the **PasteSpecial** method, it places the object as described in Setting the Current Drawing Position.

The following table shows the formats for the **PasteSpecial** method and for the **ClipboardFormatAvailable** property, which is explained below.

Value	Format
0	ABC Native
1	OLE Client Embed
2	ABC Rich Text
3	Rich Text Format (RTF)
4	Unformatted text
5	Metafile
6	Device Independent Bitmap
7	Bitmap
8	OLE Client Link

For example, the following statement pastes the Clipboard object as an OLE client link icon with its upper left corner two inches from the left margin and three inches from the top margin of the page.

```
Successful = Chart.PasteSpecial(8,True,2,3)
```

You use the **ClipboardFormatAvailable** property in the Chart object to find out whether the object in the Clipboard is in a format that you want. The property returns a Boolean value equal to True if the format is available or False if the format is not available.

The **ClipboardFormatAvailable** property uses the same values and formats as the **PasteSpecial** method described above.

For example, the following puts the Boolean value True or False in CanPaste depending on whether the object currently in the Clipboard can be pasted as a DIB.

```
CanPaste = Chart.ClipboardFormatAvailable(6)
```

Setting the Current Drawing Position

ClipboardFormatAvailable property
PasteSpecial method

Duplicating Objects

The **Duplicate** method is in both the Chart object and the Object object. The method in the Chart object duplicates whatever is currently selected and returns a Boolean value equal to True if the operation was successful or False if the operation failed.

In the Object object, the **Duplicate** method makes a duplicate of that object and returns the duplicate object.

For example, the following statements duplicate the selected chart objects, then make a duplicate of the object stored in the variable Object.

```
Successful = Chart.Duplicate
If Successful Then
   DuplicatedObject = Object.Duplicate
End If
```

Duplicate method

Clearing Selected Objects

The **Clear_** method of the Chart object deletes the selected objects from the chart. It is the equivalent of pressing the **DEL** key or opening the FlowCharter Edit menu and choosing Clear. The method deletes whatever is currently selected and returns a Boolean value equal to True if the operation was successful or False if the operation failed. For example, the following statement deletes the selected objects.

```
Successful = Chart.Clear_
```

The **Clear_** method of the Object object deletes the object object. You usually use it as part of a routine using the **SetDefaults** method. The **Clear_** method deletes the indicated object and returns a Boolean value equal to True if the operation was successful or False if the operation failed. For example, the following statement deletes the indicated object.

```
Successful = Chart.DefaultObject.Clear_
```

{button Related Topics,PI(``,`IDH_RT_Clearing_Selected_Objects')}

Restoring Objects

The **RestorePicture** method of the Object object lets you restore bitmap and OLE client objects to their original size.

For example, the following statement restores an object.

PasteObject.RestorePicture

RestorePicture method

Using OLE Client Objects

You can use OLE Automation to work with objects that are linked or embedded in a chart using OLE. To work with linked or imbedded objects, you use the **InsertObjectFromFile** method, the **PasteLink** method, the **UpdateFields** method, the **OLE** property, the **ObjectType** property, and the **DoVerb** method.

You use the **InsertObjectFromFile** method of the Chart object to insert a new OLE client object from a file. Quotation marks should be used whenever long filenames or long pathnames are used. You can optionally add parameters to specify that the file be inserted as an icon or linked. The method returns the file that is inserted as an object. The **InsertObjectFromFile** method is equivalent to opening the FlowCharter Insert menu, choosing Object, choosing the Create from File option, selecting the file you want to insert, and clicking OK. The *AsIcon* element is equivalent to selecting the Display As Icon option. The *AsLink* element is equivalent to selecting the Link to File option.

InsertObjectFromFile(*FileName* [, *AsIcon*] [, *AsLink*])

For example, the following statement inserts an Excel file into the chart and sets InsertedOleObject equal to the new object.

```
InsertedOleObject = Chart.InsertObjectFromFile(C:\EXCEL\DATA.XLS)
```

You use the **PasteLink** method of the Chart object to paste the contents of the Clipboard into the chart and link the file that is the source of the contents of the chart. The **PasteLink** method is equivalent to opening the FlowCharter Edit menu and choosing Paste Link. For example, the following statement pastes and links the contents of the Clipboard 2 inches from the left side of the chart and 3 inches from the top of the chart.

```
Chart.PasteLink(2,3)
```

You use the **UpdateFields** method of the Chart object to update all the linked fields in the chart. For example, the following statement updates the linked fields in the chart.

```
Chart.UpdateFields
```

The **UpdateFields** method is equivalent to clicking the Update Data Fields button on the Data toolbar.

You use the **OLE** property of the Object object and the **ObjectType** property of the Object object to find the short object class name of an object that is embedded or linked.

For example, these statements append the name of the OLE object type to the end of the object's text for all linked objects.

```
Set ABCObjects = Chart.Objects
Do
   Set Object = ABCObjects.ItemFromAll
   If ObjectType = 4 Then
        Object.Text = Object.Text + "OLE: " + Object.OLE.ObjectType
   End If
Loop While Object.Valid
```

Note

- You cannot change the value in the **ObjectType** property.

You use the **DoVerb** method of the OLE object to specify an OLE verb to execute if the object is a

linked or embedded OLE object. If you do not specify a verb, the default verb is used.

For example, the following statements find the OLE objects in a chart and execute the default verb for each of them.

```
Set ABCObjects = Chart.Objects
Do
    Set Object = ABCObjects.ItemFromAll
    If Object.OLE.ObjectType = 4 Then
        Object.OLE.DoVerb
    End If
Loop While Object.Valid
```

{button Related Topics,PI(`',`IDH_RT_Using_OLE_Client_Objects')}

Restoring OLE Objects

DoVerb method
InsertObjectFromFile method
PasteLink method
UpdateFields method

ObjectType property
OLE property

Restoring OLE Objects

The **RestorePicture** method of the OLE object lets you restore OLE client objects to their original size. This method is nearly the same as the Object object's **RestorePicture** method. The difference is that this method only works on OLE objects, while the Object object's **RestorePicture** method handles bitmaps as well as OLE objects.

For example, the following statement restores an OLE object.

OLEObject.RestorePicture

{button Related Topics,PI(``,`IDH_RT_Restoring_OLE_Objects')}

Using OLE Client Objects

RestorePicture method

Speeding Up Actions

You can speed the actions of OLE Automation as much as 400% to 1000% (4 to 10 times faster) when you want to perform a number of actions before returning control of the screen to the user. You can achieve this efficiency by preventing the screen from repainting until all objects have been created or changed.

You stop the objects from being painted using the **NoRepaint** property in the Chart object. You then update the screen using the **Repaint** method in the Chart object after you have specified all the changes to be performed.

For example, the following statements turn off painting, create 200 shapes, and then repaint the chart.

```
Chart.NoRepaint = True
For DrawFast = 1 to 200
    Chart.DrawShape
Next DrawFast
Chart.NoRepaint = False
Chart.Repaint
```

You can also speed actions when you are creating many objects that all have the same characteristics. For example, suppose you are creating many shapes and want them all to have text that is red 17-point Futura with a green shadow. Create one object, set the style for it, and then use the **Duplicate** method of the Object object to make as many copies as you need.

You can also speed actions by setting the defaults for Line_, Shape, and TextBlock objects using the **SetDefaults** method. With that method, you create an object with the defaults you want and then pass it to the method. For example, the following statement sets the defaults for lines to the defaults of the line object named LineObject.

```
Chart.SetDefaults LineObject
```

{button Related Topics,PI(`',`IDH_RT_Speeding_Actions')}

Setting Defaults

Duplicate method
Repaint method
SetDefaults method

NoRepaint property

What's New

The following commands are new in FlowCharter.

ChartDeActivateNOTIFY event

ChartSavedNOTIFY event

ObjectLineDeAttachNOTIFY event

ObjectTextChangedNOTIFY event

StatusBarVisible property (Application object)

ZoomWindowVisible property (Application object)

Align method (Chart object)

Export method (Chart object)

ImportShape method (Chart object)

MakeSameSize method (Chart object)

PrintPreview method (Chart object)

ReplaceText method (Chart object)

SpaceEvenly method (Chart object)

FlippedHorizontal property (Object object)

FlippedVertical property (Object object)

Rotation property (Object object)

ApplyDefaults method (Object object)

CrossoverSize property(Line_ object)

CrossoverStyle property (Line_ object)

Routing property (Line_ object)

The following commands were in ABC FlowCharter 4.0, but are not appropriate in FlowCharter 7. They are not invalid in FlowCharter 7, but they are ignored.

IndexWindowHandle property (Application object)

IndexVisible property (Application object)

LaunchFlags property (Shape object)

LaunchStartDir property (Shape object)

RestorePicture method (OLE object)

ShapeSizing property (Preferences object)

The following commands were in ABC FlowCharter 4.0, but are not appropriate in FlowCharter 7.

They are handled as described.

LaunchIndicator property (Chart object) calls **LinkIndicator** property

LaunchShadow property (Chart object) calls **LinkShadow** property

The Line_ object is now equivalent to the Line object. The references may be used interchangeably.

If you are using the OCX, the NOTIFY and SUBCLASS commands now take parameters as shown below.

AppQuitNOTIFY ()
AppQuitSUBCLASS (Override As Boolean)
AppMenuHintSUBCLASS (ByVal MenuItem As Object, Override As Boolean)
AppMenuPopupSUBCLASS (ByVal Menu As Object, Override As Boolean)
AppMenuSUBCLASS (ByVal MenuItem As Object, Override As Boolean)
ChartActivateNOTIFY (ByVal Chart As Object)
ChartDeActivateNOTIFY (ByVal Chart As Object)
ChartChangeNOTIFY (ByVal Chart As Object)
ChartCloseSUBCLASS (ByVal Chart As Object, Override As Boolean)
ChartNewNOTIFY (ByVal Chart As Object)
ChartOpenNOTIFY (ByVal Chart As Object)
ChartPasteNOTIFY (ByVal Chart As Object)
ChartSavedNOTIFY (ByVal Chart As Object)
DeleteSUBCLASS (ByVal Chart As Object, Override As Boolean)
DoubleClickSUBCLASS (ByVal Object As Object, ByVal Chart As Object, Override As Boolean)
ExclusiveSelectionNOTIFY (ByVal Object As Object, ByVal Chart As Object)
FieldValueChangedNOTIFY (ByVal FieldValue As Object, ByVal Object As Object, ByVal Chart As Object)
LinkNOTIFY (ByVal LinkedToChart As Object, ByVal Object As Object, ByVal Chart As Object)
NewLineNOTIFY (ByVal Object As Object, ByVal Chart As Object)
NewShapeNOTIFY (ByVal Object As Object, ByVal Chart As Object)
ObjectClickSUBCLASS (ByVal Object As Object, ByVal Chart As Object, Override As Boolean)
ObjectFontChangeNOTIFY (ByVal Object As Object, ByVal Chart As Object)
ObjectLineAttachNOTIFY (ByVal Line As Object, ByVal Object As Object, ByVal Chart As Object)
ObjectLineDeAttachNOTIFY (ByVal Line As Object, ByVal Object As Object, ByVal Chart As Object)
ObjectMovedNOTIFY (ByVal Object As Object, ByVal Chart As Object)
ObjectMoveSUBCLASS (ByVal Object As Object, ByVal Chart As Object, Override As Boolean)
ObjectSizedNOTIFY (ByVal Object As Object, ByVal Chart As Object)
ObjectSizeSUBCLASS (ByVal Object As Object, ByVal Chart As Object, Override As Boolean)
ObjectTextChangedNOTIFY (ByVal Object As Object, ByVal Chart As Object)
ReplaceShapeNOTIFY (ByVal Object As Object, ByVal Chart As Object)
SpecialKeySUBCLASS (ByVal KeyCode As Integer, Override As Boolean)

{button Related Topics,PI(`',`IDH_RT_Whats_New')}

Undoing Actions

You can choose FlowCharter's Undo command using the **Undo** method of the Application object. You can find out if there is anything to undo using the **UndoAvailable** property of the Application object. Using the **Undo** method is equivalent to opening the FlowCharter Edit menu and choosing Undo.

The following statements undo the last action if it is available. Whether the last action was undone is put in the status bar.

```
If ABC.UndoAvailable = True Then
   ABC.Undo
   ABC.StatusBar = "Last action undone."
Else
   ABC.StatusBar = "Nothing available to undo."
End If
```

{button Related Topics,PI(``,`IDH_RT_Speeding_Actions')}

Undo method
UndoAvailable property

Formatting Objects

In FlowCharter, you can use the Format Painter to apply formats to objects. In OLE Automation, you use the **ApplyDefaults** property to apply the chart's default styling to an object.

ObjectObject.**ApplyDefaults**

You first use *ChartObject*.**SetDefaults** (*ObjectObject*) to define the default styling for shapes, lines, and textblocks. Then you use the **ApplyDefaults** property.

{button Related Topics,PI(``,`IDH_RT_Formatting_Objects')}

ApplyDefaults Method
SetDefaults Method

Drawing Shapes

You can draw any shape in the Shape Palette. Use the **DrawShape** method of the Chart object to draw shapes. By default, **DrawShape** uses the current shape in the Shape Palette. For information on specifying the shape, see the Choosing a Shape in the Palette.

```
Set ABCObject = Chart.DrawShape
```

All the shape palettes that ship with ABC have predefined names that appear in the hintline when the mouse pauses over them, and which are listed in the documentation that ships with ABC. In ABC the shape's name is defined in the Shape Palette Item Information dialog box. You can open the Shape Palette Item Information dialog box by choosing Item Information on the Options menu of the Shape Palette.

In ABC OLE Automation you can optionally specify the type of shape you want to draw by specifying the shape's name. The program uses a "loose matching" routine so, for example, setting the shape's name to "Proc" chooses "Process." If more than one shape matches the value you set, the exact match is used first. If there is not exact match, the first one alphabetically is used.

```
Set ABCObject = Chart.DrawShape("Proc")
```

Shapes are automatically placed at the chart's current drawing position. (See Setting the Current Drawing Position for more information on the current drawing position.) Alternatively, you can use the **DrawDirection** property of the Chart object to specify the direction for placing new shapes. The following table shows the values for the **DrawDirection** property.

0	North
1	East
2	South
3	West
10	Stacked

You specify the horizontal and vertical distance from a shape to the next one you create using the **DrawSpacingX** property and **DrawSpacingY** property of the Chart object. The **DrawSpacingX** property and **DrawSpacingY** property are equivalent to clicking Options on the Tool menu, clicking the Alignment tab, and entering horizontal and vertical spacing.

You can use the **NextShapeHeight** property and **NextShapeWidth** property of the Chart object to specify the height of the next shape to be drawn.

For example, the following statements set the horizontal spacing to two inches, the vertical spacing to three inches. They then specify the height for the next shape drawn and then that the next shape should be to the right of the current shape.

```
Chart.DrawSpacingX = 2          ' Horizontal spacing 2"
Chart.DrawSpacingY = 3          ' Vertical spacing 2"
Chart.NextShapeHeight = .5      ' Height of next shape .5
Chart.NextShapeWidth = .5       ' Width of next shape .5
Chart.DrawDirection = 1         ' Next shape toward right
```

You can import a graphics file into a shape using the **ImportShape** method. A shape is created and the graphics file is inserted into it. This is the equivalent of clicking Import on the Tools menu in FlowCharter. Quotation marks should be used whenever long filenames or long pathnames are used.

ChartObject.**ImportShape** (*FileName*)

See Moving Objects and Resizing Objects for information on changing the size and position of shapes.

See Drawing Lines that Connect Shapes for information on drawing lines to connect shapes.

{button Related Topics,PI(`',`IDH_RT_Drawing_Shapes')}

Choosing a Shape in the Palette
Drawing Lines that Connect Shapes
Moving Objects
Resizing Objects
Setting the Current Drawing Position

DrawShape method
ImportShape Method

DrawDirection property
DrawSpacingX property
DrawSpacingY property
NextShapeHeight property
NextShapeWidth property

Using the Shape Palette

FlowCharter provides a wide variety of shape palettes you can use in drawing charts. Each palette contains several shapes that you can choose from. With OLE Automation, you can display or hide the Shape Palette, open a different Shape Palette, and choose a shape from the palette.

{button Related Topics,PI(``,`IDH_RT_Using_the_Shape_Palette')}

Displaying and Hiding the Shape Palette

Use the **ShapePaletteVisible** property of the Application object to display or hide the Shape Palette or to determine whether the Shape Palette is visible. When this property is True, the Shape Palette is displayed; when False, it is hidden.

```
ABC.ShapePaletteVisible = True          ' Displays the Shape Palette
```

Using the Shape Palette

ShapePaletteVisible property

Opening a Different Shape Palette

You can open any of the Shape Palettes that ship with FlowCharter. Use the **CurrentShapePalette** property of the Chart object to open a different Shape Palette or determine the name of the current Shape Palette. The name of the Shape Palette appears in the title bar of the palette. The name is not related to the filename of the palette.

For example, the following statement opens the Auditing Shape Palette.

```
Chart.CurrentShapePalette = "Auditing"
```

{button Related Topics,PI(``,`IDH_RT_Opening_a_Different_Shape_Palette')}

Using the Shape Palette

CurrentShapePalette property

Choosing a Shape in the Palette

The **DrawShape** method of the Chart object draws the current shape in the Shape Palette, unless you specify a shape to draw with the method. In FlowCharter, you choose a shape to be the current shape by clicking it in the Shape Palette. The name of the shape appears in the hintline when you pass over the shape with the mouse.

With OLE Automation you use the **CurrentShape** property of the Chart object to choose a shape as the current shape. The program uses a "loose matching" routine so, for example, setting the **CurrentShape** property to "Dec" chooses "Decision." If more than one shape matches the value you set, the exact match is used first. If there is not exact match, the first one alphabetically is used.

For example, the following statements choose the Decision shape as the next shape to be drawn, and then draw the shape.

```
Chart.CurrentShape = "Decision"        ' Decision is current shape
Set ABCObject = Chart.DrawShape        ' Draw the current shape
```

{button Related Topics,PI(`',`IDH_RT_Choosing_a_Shape_in_the_Palette')}

Using the Shape Palette

CurrentShape property
DrawShape method

Adding Text to Shapes

You can use OLE Automation to add text inside any shape. The text appears inside the text area defined for the shape. Adding text to a shape is equivalent to typing while a shape is selected in FlowCharter.

To add text to a shape, use the **Text** property of the Object object. The following example draws the shape shown above.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape("External Operation")
ABCObject.Text = "Text inside a shape"
```

If you are reading the text from a shape, you can use the **TextLF** property to preserve the Returns. If you use the **Text** property, the Returns are changed to spaces.

```
ShapeText = Shape1.TextLF
```

{button Related Topics,PI(`',`IDH_RT_Adding_Text_to_Shapes')}

Fitting Shapes to Text

TextLF property
Text property

Fitting Shapes to Text

You can automatically fit shapes to the size of the text inside them. This is especially useful when the length of the text string may vary.

In FlowCharter, you do this by clicking the Text tool and then the Fit to Text button on the Arrange+ toolbar when the shape is selected. With OLE Automation, you use the **FitShapeToText** method of the Shape object.

The following example draws a shape, adds text to the shape, then fits the shape to the text.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape("Document")
ABCObject.Text = "This is a sample of fitting shapes to text."
ABCObject.Shape.FitShapeToText
```

Adding Text to Shapes
FitShapeToText method

Numbering Shapes

You can use various numbering systems for shapes, such as *1, 2, 3; 1.1, 1.2, 1.3;* or even text strings.

The number used for the next new shape you draw is stored in the **NextNumber** property of the Chart object. The number is kept as a text string, since the number can contain text as well as numbers.

The **NextNumber** property is incremented automatically each time you draw a shape. If **NextNumber** contains text with a number, the text remains and the number is incremented. For example, "Step 5" will become "Step 6" when a new shape is drawn. If **NextNumber** contains only text, the text remains without incrementing. For example, "Step Five" will stay as "Step Five" even after a new shape is drawn. This is especially useful when you want the shape number to be a placeholder for a company name or department name.

The **Number** property of the Shape object contains the shape number for a specific shape. When you draw a shape, the value in the chart's **NextNumber** is stored in **Number**, and **NextNumber** is incremented. You can change a shape's number by changing the value of the shape's **Number** property.

You also can use the **Renumber** method to change a shape's number. **Renumber** changes a shape's **Number** property to the chart's **NextNumber** value, and increments **NextNumber**.

The following example illustrates **NextNumber** and **Renumber**.

```
Sub ShapeNumbers()
   Dim ShapeOne As Object, ShapeTwo As Object, ShapeThree As Object
   Dim Chart As Object, ABC As Object
   Set ABC = CreateObject("ABCFlow.Application")
   ABC.New                              ' Create a new chart
   Set Chart = ABC.ActiveChart
                                        ' NextNumber initially defined as 1
   Set ShapeOne = Chart.DrawShape       ' ShapeOne.Number=1; NextNumber=2
   Set ShapeTwo = Chart.DrawShape       ' ShapeTwo.Number=2; NextNumber=3

   ShapeOne.Shape.Number = "Step 1"     ' ShapeOne.Number=Step 1; NextNumber=3
   Chart.NextNumber = "Step 2"          ' NextNumber=Step 2
   ShapeTwo.Shape.Renumber              ' ShapeTwo.Number=Step 2; NextNumber=Step 3
   Set ShapeThree = Chart.DrawShape     ' ShapeThree.Number=Step 3; NextNumber=Step 4
End Sub
```

{button Related Topics,PI(``,`IDH_RT_Numbering_Shapes')}

NextNumber property
Number property
Renumber method

Formatting Shape Numbers

You can format shape numbers by choosing a typeface, size, color, and text attributes.

In FlowCharter, the format defined for Link and Note indicators is also used for shape numbers. Select the Indicator Options button at the side of the dialog box, and choose number formatting options in the Number Font area of the dialog box.

Using OLE Automation, you can select the same options using the NumberFont object of the Chart object. Like other font objects, the NumberFont object has the following properties.

Bold	True if text is **bold**; False if text is not bold.
Color	The color used in shape numbers. This value can be one of the 16 color constants, such as ABC.Blue.
Italic	True if text is *italic*; False if text is not italic.
Name	The typeface name used for shape numbers, such as "Arial" or "Roman."
Size	The point size of shape numbers.
Strikethrough	True if text is ~~strikethrough~~; False if text is not strikethrough. This attribute is not available to shape numbers in FlowCharter.
Underline	True if text is underlined; False if text is not underline.

See Formatting Text for more information on these properties.

The following statements change shape number text to Helvetica 12-point bold italic.

```
Chart.NumberFont.Name = Helvetica
Chart.NumberFont.Size = 12
Chart.NumberFont.Bold = True
Chart.NumberFont.Italic = True
```

The **Opaque** property is not available in the NumberFont object. Shape numbers are opaque when the other text in the shape is opaque, and transparent when the other text is transparent. For example, the following statement makes both shape text and shape numbers opaque.

```
ABCObject.Font.Opaque = True
```

Formatting Text

Bold property
Color property
Italic property
Name property
Opaque property
Size property
Strikethrough property
Underline property

Hiding Shape Numbers

Shapes can be numbered automatically. If shapes are numbered, you can hide the shape numbers if you wish. This feature is useful when you do not want numbers to appear in certain shape types, such as documents or decisions.

In FlowCharter, you show or hide shape numbers in selected shapes by clicking Shape Numbering on the Format menu and selecting or clearing Display Shape Numbers.

With OLE Automation, you use the **NumberShown** property of the Shape object to display or hide shape numbers. Make **NumberShown** equal to True to display numbers, as on the first statement below or False to hide numbers, as in the second statement below.

```
ABCObject.Shape.NumberShown = True
ABCObject.Shape.NumberShown = False
```

{button Related Topics,PI(``,`IDH_RT_Hiding_Shape_Numbers')}

NumberShown property

Fill, Border, and Shadow Colors

You can color a shape by setting its fill color, its border color, and its shadow color.

You set the fill color for shapes using the **FillColor** property of the Shape object or the **Color** property of the Object object. Both properties produce the same effect.

For example, the following statements draw a shape and then change its fill color to blue using the **FillColor** property. They then change its fill color to red using the **Color** property.

```
Dim NewObj1 As Object
Set NewObj1 = Chart.DrawShape
NewObj1.Shape.FillColor = ABC.BLUE
NewObj1.Color = ABC.RED
```

You set the border color for shapes using the **BorderColor** property of the Shape object. For example, the following statement makes the border of a shape blue.

```
NewObj1.Shape.BorderColor = ABC.BLUE
```

You set the shadow color for shapes using the **ShadowColor** property of the Shape object. For example, the following statement makes the shadow of a shape blue.

```
NewObj1.Shape.ShadowColor = ABC.BLUE
```

BorderColor property
Color property
FillColor property
ShadowColor property

Fill Pattern

You can fill a shape with any of the patterns available in FlowCharter.

In FlowCharter, you change a selected shape's fill pattern by clicking Fill on the Format menu, selecting Pattern, and then choosing a pattern.

To set or read a shape's fill pattern with OLE Automation, use the **FillPattern** property of the Shape object. Set **FillPattern** to 0 for a transparent fill or to 1 for a solid fill. See **FillPattern** for each available pattern.

The following statements draw a shape and then change its fill pattern to vertical stripes.

```
Set ABCObject = Chart.DrawShape("Process")
ABCObject.Shape.FillPattern = 4
```

{button Related Topics,PI(`',`IDH_RT_Fill_Pattern')}

FillPattern property

Border Style and Width

You can choose different line styles for shape borders. A shape border includes not only the outside edge of a shape, but also any interior lines used in the shape (for example, the concentric circles on the inside of a 5 1/2" floppy disk shape). FlowCharter provides many useful border styles, including solid and dashed lines and an invisible border.

In FlowCharter, you change a selected shape's border style by clicking the Line Style button on the Formatting toolbar, and then choosing a style from the list. You set the width of the border by clicking the arrows next the the Line Weight box.

To set or read a shape's border style with OLE Automation, use the **BorderStyle** property of the Shape object. Set **BorderStyle** to 0 for an invisible border and 1 for a solid line border. See **BorderStyle** for each available style.

Use the **BorderWidth** property of the Shape object to change or read the width of the border. **BorderWidth** can have a value ranging from 1 (hairline) to 5 (thickest).

Note

- **BorderWidth** is applied only if **BorderStyle** is 1; it does not apply to dashed or dotted borders.

The following statements draw a Process shape and then change its border to a dotted line.

```
Set ABCObject = Chart.DrawShape("Process")
ABCObject.Shape.BorderStyle = 3
```

The following statements draw a decision shape and change its border to a very thick solid line.

```
Set ABCObject = Chart.DrawShape("Process")
ABCObject.Shape.BorderStyle = 1
ABCObject.Shape.BorderWidth = 5
```

BorderStyle property
BorderWidth property

Shadow Style and Width

You can add a drop shadow to shapes and choose the position and width of the shadow.

In FlowCharter, you add a shadow to a selected shape by clicking the Shadow button on the Formatting toolbar, and then choosing a shadow position from the list. To change the offset, click Shadow in the Format menu, and then choose a value in the Width box.

With OLE Automation you add a shadow using the **ShadowStyle** property of the Shape object. **ShadowStyle** can have a value from 0 to 4, with 0 being no shadow and 1 through 4 being the positions shown in **ShadowStyle**.

The width of a shadow (the distance the shadow appears away from the shape) is determined by the **ShadowOffset** property of the Shape object. **ShadowOffset** can have a value ranging from 1 (hairline) to 5 (thickest).

You also can use the **ShadowStyle** and **ShadowOffset** properties to read the values of the current shadow of a shape.

The following statements draw a shape, and then add a drop shadow with medium thickness.

```
Set ABCObject = Chart.DrawShape("Document")
ABCObject.Shape.ShadowStyle = 2
ABCObject.Shape.ShadowOffset = 3
```

ShadowOffset property
ShadowStyle property

Replacing Shapes

You can replace one or more shapes in a chart with a different type of shape. When you replace shapes, the new shapes connect to the lines of the old shapes.

In FlowCharter. you replace selected shapes by choosing the new shape in the Shape Palette, clicking the Shape tool in the toolbox, and clicking the Replace Shape button on the Arrange+ toolbar.

With OLE Automation, use the **ReplaceShape** method of the Shape object to replace shapes. You can replace shapes with the chart's current shape or with any shape type you specify.

ShapeObject.**ReplaceShape** [*ShapeType*]

ShapeType is an optional parameter that specifies the shape type that will be used to replace the shape referred to in *ShapeObject.*

The following example replaces all Operation shapes in a chart with External Operation shapes.

```
Set Objs = ABC.ActiveChart.Objects
Do
   Set ABCObject = Objs.ItemFromShapes
   If ABCObject.Shape.ShapeName = "Operation" Then
        ABCObject.Shape.ReplaceShape "External Operation"
   End If
While ABCObject.Valid
```

{button Related Topics,PI(`',`IDH_RT_Replacing_Shapes')}

ReplaceShape method

Selecting Shapes

OLE Automation provides several ways to select shapes. You can select a single shape, all shapes of a particular type, or all shapes in the chart.

To select a single shape, use the **Selected** property of the Object object. Set the **Selected** property to True to select the shape or False to deselect the shape. For example, the following statements draw a shape and then select it.

```
Dim ABCShape As Object
Set ABCShape = Chart.DrawShape
ABCShape.Selected = True
```

To select all shapes of a particular type, such as Process or Decision, use the **SelectShapeType** method of the Chart object. This method takes one parameter: a string indicating the type of shape. The statement below selects all Document shapes.

```
Chart.SelectShapeType "Document"
```

To select all the shapes in a chart, use the **Select** method of the Chart object. The **Select** method can select all shapes, all lines, or all objects in a chart. It can also be used to deselect all objects. The **Select** method takes one parameter, an integer indicating the selection.

0	Selects all shapes
1	Selects all lines
2	Selects all objects (shapes, lines, text blocks)
3	Deselects all objects

For example, the following statements deselect all objects in a chart and then select only the lines.

```
Chart.Select 3
Chart.Select 1
```

Deselecting Shapes

Select method
Selected property
SelectShapeType method

Deselecting Shapes

OLE Automation provides three ways to deselect shapes.

You can deselect a single object by using the selected property of the Object object. Set the **Selected** property to False to deselect the shape.

You can deselect all the objects that are currently selected by using the **DeselectAll** method of the Chart object. The **DeselectAll** method requires no extra parameters.

You can deselect all the objects that are currently selected by using the **Select** method of the Chart object, but the **Select** method must be followed by the number 3 to deselect objects.

In the statements below, the first line deselects only the selected object. The last two lines deselect all objects.

```
ABCObject.Selected = False
Chart.DeselectAll
AllChart.Select 3
```

{button Related Topics,PI(``,`IDH_RT_Deselecting_Shapes')}

Selecting Shapes

DeselectAll method
Select method
Selected property

Opening the Note Window

The Note window displays notes for the currently selected shape.

In FlowCharter. you open and close the Note window by clicking Note in the View menu.

Using OLE Automation, you open and close the Note window using the **NoteViewerVisible** property of the Application object. Set this property to True to open the Note window or False to close the Note window. You also can use this property to check whether the window is already open.

The following example checks to see if the Note window is open, and then closes the Note window.

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.Application")
If ABC.NoteViewerVisible Then              ' If Note window is open
   ABC.NoteViewerVisible = False           ' Close the Note window
End If
```

{button Related Topics,PI(``,`IDH_RT_Opening_the_Note_Window')}

NoteViewerVisible property

Attaching a Note to a Shape

You can attach notes to any shape in a chart. In FlowCharter,, notes are added in the Note window while the shape is selected. With OLE Automation, you do not need to open the Note window to attach notes to a shape.

Use the **NoteText** property of the Shape object to attach notes to shapes.

The following example draws a shape, then adds a note to the shape.

```
Dim ShapeObject As Object

Set ShapeObject = Chart.DrawShape("Document")
ShapeObject.Text = "Text inside a shape"
ShapeObject.Shape.NoteText = "This is note text attached to the shape"
```

If you are reading the note text from a shape, you can use the **NoteTextLF** property to preserve the Returns. If you use the **NoteText** property, the Returns are changed to spaces.

```
NoteText = Shape1.NoteTextLF
```

{button Related Topics,PI(``,`IDH_RT_Attaching_a_Note_to_a_Shape')}

NoteText property
NoteTextLF property

Choosing Note Indicators

You can use indicators to identify shapes that have attached notes. Indicators include shadows around the symbol and symbols next to the shape number. The default indicator for notes is **-N**. You can use up to three characters to create your own indicators.

The text settings used for indicator symbols are the same as for the shape number. See the Formatting Shape Numbers for more information on formatting shape numbers.

In FlowCharter. you choose note indicators in the Indicator tab of the Format Chart dialog box.

With OLE Automation you use the **NoteIndicator** and **NoteShadow** properties of the Chart object. The **NoteIndicator** property identifies the three character symbol as a string. The **NoteShadow** property is a Boolean value that determines whether a shadow displays around shapes with attached notes.

```
Chart.NoteIndicator = "*N*"          ' Set *N* as the note symbol
Chart.NoteShadow = 1                 ' Use shadow to indicate notes
```

{button Related Topics,PI(`',`IDH_RT_Choosing_Note_Indicators')}

Formatting Shape Numbers

NoteIndicator property
NoteShadow property

Formatting Note Text

You can format note text just as you can format other text objects. The formatting for note text appears in the Note window and in printed notes. Note text is formatted for each shape individually.

The **NoteFont** property of the Shape object returns a font object with properties that you can set. That object has the following properties.

Bold	True if text is **bold**; False if text is not bold
Italic	True if text is *italic*; False if text is not italic
Strikethrough	True if text is ~~strikethrough~~; False if text is not strikethrough
Underline	True if text is underlined; False if text is not underline
Name	The typeface name of the font
Size	The point size of the font

See Formatting Text for more information on these properties.

As with other text objects, the **NoteFont** property of the shape object returns a font object that has the **Opaque** property, but it is not useful for formatting notes. The following example formats the note text for all shapes in a chart.

```
Dim ABC As Object
Dim AllShapes As Object
Dim CurrentShape As Object

Set ABC = CreateObject("ABCFlow.Application")
ABC.New                                          ' Create a new chart
Set AllShapes = ABC.ActiveChart.Objects

Do
   Set CurrentShape = AllShapes.ItemFromShapes
   CurrentShape.Shape.NoteFont.Name = "Arial"
   CurrentShape.Shape.NoteFont.Size = 12
   CurrentShape.Shape.NoteFont.Italic = True
Loop While CurrentShape.Valid
```

{button Related Topics,PI(``,`IDH_RT_Formatting_Note_Text')}

Formatting Text

Bold property
Italic property
Name property
NoteFont property
Opaque property
Size property
Strikethrough property
Underline property

Printing Notes

You can print the notes that are attached to shapes.

In FlowCharter, there are two ways to print notes. Choose the Print command directly from the Note window, or open the File menu in the main window and choose Print, and then select the Print Notes option in the Print dialog box.

Using OLE Automation, you print notes in a way similar to using the Print dialog box. The **PrintOut** and **PrintSelected** methods print a chart, and can also print notes associated with the chart.

Chart.PrintOut [*FromPage*] [,*ToPage*] [,*Copies*] [,*FitToPage*] [,*PrintNotes*]
Chart.PrintSelected [*Copies*] [,*FitToPage*] [,*PrintNotes*]

All parameters in these methods are optional. To print notes, the *PrintNotes* parameter must be 1 (True). Look at the following examples.

```
Chart.PrintOut   , , 2, , 1          ' Print 2 copies of chart and notes
Chart.PrintSelected   , 1, 1         ' Print selected objects to fill the page with attached notes
```

{button Related Topics,PI(`',`IDH_RT_Printing_Notes')}

PrintOut method
PrintSelected method

Drawing Lines

You can draw lines by specifying the starting and ending points of a line in space, by drawing that is connected to one shape, or by connecting two shapes with a line. You access information about lines using the **Line_** property of the Object object.

{button Related Topics,PI(`',`IDH_RT_Drawing_Lines')}

Drawing Unconnected Lines
Drawing Lines to One Shape
Drawing Lines that Connect Shapes

Line_ property

Drawing Unconnected Lines

Unconnected lines are lines that are drawn from one point to another and are not connected to any shapes or lines.

Use the **DrawFreeLine** method of the Chart object to draw an unconnected line.

DrawFreeLine (*XPosition*, *YPosition*)

The line starts at the chart's current drawing position and ends at the point you specify with *XPosition* and *YPosition*. The X and Y positions are measured from the top left corner of the FlowCharter page. By default, the positions are measured in inches, but you can measure position in centimeters by changing the **Units** property of the Preferences object. See Setting the Current Drawing Position for more information.

For example, the following statement draws a line from (1,1) to (2,4).

```
Chart.DrawPositionX = 1
Chart.DrawPositionY = 1
Set NewLine = Chart.DrawFreeLine (2, 4)
```

The type of routing used for the line is determined by the chart's current line routing. See Setting Line Routing for more information.

{button Related Topics,PI(``,`IDH_RT_Drawing_Unconnected_Lines')}

Drawing Lines
Setting the Current Drawing Position
Setting Line Routing

DrawFreeLine method
Units property

Drawing Lines to One Shape

You can draw lines that are unconnected on one end, and connected to a shape on the other end. If you move the shape, the connected line follows.

Use the **DrawLineToOneObject** method of the Chart object to draw lines that connect to only one shape.

DrawLineToOneObject (*ShapeObject* [,*EnterDirection*])

The line starts at the chart's current drawing position and ends at the shape you specify with *ShapeObject*. See the Setting the Current Drawing Position for information on setting the chart's current drawing position.

You can optionally use a second parameter that specifies the direction that the line enters the shape. The following table describes each of the *EnterDirection* values.

0	North
1	East
2	South
3	West

For example, the following statement draws a line from the current position to the shape specified in ShapeObject.

```
Set NewLine = Chart.DrawLineToOneObject (ShapeObject, 0)
```

The type of routing used for the line is determined by the chart's current line routing. See Setting Line Routing for more information.

Drawing Lines
Setting the Current Drawing Position
Setting Line Routing

DrawLineToOneObject method

Drawing Lines that Connect Shapes

You can draw lines that connect two shapes. If you move either of the shapes, the connected line follows.

Use the **DrawLine** method of the Chart object to draw lines that connect two shapes.

DrawLine (*ShapeObject1*, *ShapeObject2* [,*ExitDirection*] [,*EnterDirection*])

The line starts at *ShapeObject1* and ends at *ShapeObject2*. You can optionally use third and fourth parameters that specify which direction the line exits *ShapeObject1* and enters *ShapeObject2*. The following chart describes each of the direction values.

0	North
1	East
2	South
3	West

For example, the following statement draws a line from the bottom of Shape1 to the top of Shape2.

```
Set NewLine = Chart.DrawLine (Shape1, Shape2, 2, 0)
```

The type of routing used for the line is determined by the chart's current line routing. See Setting Line Routing for more information.

You can find or set the side of the source shape that the line leaves using the **SourceDirection** property and can find or set the side of the destination shape that the line enters using the **DestinationDirection** property. The chart above describes each of the direction values.

Drawing Lines
Setting Line Routing

DestinationDirection property
DrawLine method
SourceDirection property

Connecting Existing Lines to Shapes

The shapes that lines connect to are stored in the **Source** and **Destination** properties of the Line_ object. When lines are unconnected, these properties are empty. You can use these properties to change which shapes the line connects or to connect a line that was previously unconnected.

For example, the following statements connect LineObject to ShapeObject1 and ShapeObject2.

```
LineObject.Line_.Source = ShapeObject1
LineObject.Line_.Destination = ShapeObject2
```

You can optionally specify the directions that the line enters each shape with the **ReconnectSource** and **ReconnectDest** methods of the Line_ object.

ReconnectSource (*ShapeObject* [,*ExitDirection*])
ReconnectDest (*ShapeObject* [,*EnterDirection*])

These methods connect the line to a shape specified by *ShapeObject.* You can optionally specify the direction the line enters and exits the shapes. The following chart describes each of the direction values.

0	North
1	East
2	South
3	West

For example, the following statements connect LineObject to the bottom of ShapeObject1 and the top of ShapeObject2.

```
LineObject.Line_.ReconnectSource ShapeObject1 0
LineObject.Line_.ReconnectDest ShapeObject2 2
```

ReconnectDest method
ReconnectSource method
Source property
Destination property

Setting Line Routing

FlowCharter has several types of routing available for lines.

In FlowCharter you set the routing for new lines to be drawn by clicking one of the line routing buttons with no lines selected. You also can change the type of routing for lines that have already been drawn.

With OLE Automation, you can specify the type of routing for new lines with the **CurrentLineRouting** property of the Chart object. The following table describes the values for the **CurrentLineRouting** property.

0	Direct
1	Right Angle
2	Curved
3	Org Chart
4	Cause-and-Effect

You can use the **Type** property of the Line_ object to find or set the line routing for a line. The values of the **Type** property have the same meaning as **CurrentLineRouting**, which is described in the table above.

The following statements use Direct line routing to draw a line.

```
Dim NewLine As Object
Dim LineType As Short
Chart.CurrentLineRouting = 0          ' Current line routing is Direct
NewLine = Chart.DrawFreeLine (5, 4)   ' Draw a new line
LineType = NewLine.Line_.Type         ' The new line is Direct
```

You also can use the **Routing** property of the Line_ object to set the line routing type for existing lines. The parameters are the same. There is no equivalent function in FlowCharter.

{button Related Topics,PI(``,`IDH_RT_Setting_Line_Routing')}

CurrentLineRouting property
Routing Property
Type property

Formatting Lines

You format lines by changing the color, width, and style of the two ends and the main body (stem) of a line. All line formatting is based on the Line_ object.

Lines are composed of three parts: the arrow at the start of the line (SourceArrow), the main body of the line (Stem), and the arrow at the end of the line (DestArrow). The properties used to format lines deal with these three parts.

Line Color
Line Width
Line Style
End Styles

Line Color

You can color an entire line, including the ends, with one property, or you can color each piece of the line with a separate property. You can use the FlowCharter constants to specify the color. For example, use ABC.Red for the color red.

The **Color** property colors the entire line, including the ends. The properties **SourceArrowColor**, **StemColor**, and **DestArrowColor** color individual parts of a line. These properties also can be used to find out the colors of a line.

The following example changes blue lines to red. All other lines are changed to blue stems and red ends.

```
If Line_.Color = ABC.Blue Then              ' If the line stem is blue, ...
   Line_.Color = ABC.Red                    ' ...change the entire line to red
Else                                        ' If the line stem is not blue, ...
   Line_.SourceArrowColor = ABC.Red         ' ...change both ends to red...
   Line_.DestArrowColor = ABC.Red
   Line_.StemColor = ABC.Blue               ' ...and change the stem to blue
End I
```

{button Related Topics,PI(``,`IDH_RT_Line_Color')}

Formatting Lines

Color property
DestArrowColor property
SourceArrowColor property
StemColor property

Line Width

You can vary the width of lines and the size of line ends. Varying line widths can distinguish data flows and draw attention to certain transitions or data transfers in a chart.

In FlowCharter you change a selected line's width by choosing a number in the Width box on the Line tab of the Format Object dialog box.

With OLE Automation, you use the **SourceArrowSize**, **StemWidth**, and **DestArrowSize** properties of the Line_ object to determine the line width. Line width can vary from 1 (hairline) to 5 (thickest).

```
ABCObject.Line_.SourceArrowSize = 2        ' Width of source arrow is 2
ABCObject.Line_.StemWidth = 2              ' Width of stem is 2
ABCObject.Line_.DestArrowSize = 1          ' Width of destination arrow is 1
```

{button Related Topics,PI(`',`IDH_RT_Line_Width')}

Formatting Lines

DestArrowSize
SourceArrowSize
StemWidth

Line Style

You can choose different styles or patterns for lines, including solid and dashed lines.

In FlowCharter you change a selected line's style by clicking the Line Style button on the Formatting toolbar, and then choosing a style from the list.

To set or read a line's style with OLE Automation, use the **StemStyle** property of the Line_ object. Set **StemStyle** to 0 for an invisible line and 1 for a solid line. See **StemStyle** for each available style.

The following statements draw a line and then change its style to dotted.

```
Set ABCObject = DrawFreeLine (4, 5)
ABCObject.Line_.StemStyle = 3
```

{button Related Topics,PI(``,`IDH_RT_Line_Style')}

Formatting Lines

StemStyle property

End Styles

You can choose different styles or patterns for line ends, including arrows, circles, and lines.

In FlowCharter you change a selected line's end style by clicking the Arrowheads button on the Formatting toolbar and then choosing a style from the list.

To set or read a line's source arrow style with OLE Automation, use the **SourceArrowStyle** property of the Line_ object. Set **SourceArrowStyle** to 0 for no arrow. See **SourceArrowStyle** for each available style.

To set or read a line's destination arrow style with OLE Automation, use the **DestArrowStyle** property of the Line_ object. Set **DestArrowStyle** to 0 for no arrow. See **DestArrowStyle** for each available style.

The following statements draw a line and then change its end styles.

```
Set ABCObject = Chart.DrawFreeLine (4, 5)
ABCObject.Line_.SourceArrowStyle = 2
ABCObject.Line_.DestArrowStyle = 3
```

Formatting Lines
DestArrowStyle property
SourceArrowStyle property

Displaying Nodes on Connecting Lines

You can display or hide nodes on lines. Nodes appear where lines connect to each other. They help you distinguish between connected lines and lines that merely overlap. Nodes are represented by small solid circles.

In FlowCharter you display nodes by clicking Chart Properties on the Format menu, clicking the Indicators tab, and then selecting the Show Nodes On Lines option.

With OLE Automation you display or hide nodes using the **ShowNodesOnLines** property of the Chart object. Set **ShowNodesOnLines** to True to display nodes; False to hide nodes.

```
Chart.ShowNodesOnLines = True       ' Display nodes
Chart.ShowNodesOnLines = False      ' Hide nodes
```

ShowNodesOnLines property

Setting Line Crossovers

You can set the style and size when lines cross over each other. If you do not choose a crossover style, the lines cross with no indication, which may make it difficult to tell which lines connect which shapes.

In FlowCharter you set the style and size of crossovers by clicking Ends on the Format menu, and then selecting the Type and Size you want.

You use the **LineCrossoverStyle** property of the Chart object to specify the type of crossover. You can specify values for the crossovers as shown in the following table.

0	Solid lines
1	Bunny hops
2	Broken lines

You use the **LineCrossoverSize** property of the Chart object to specify the size of the crossover when one line crosses another. The setting applies to bunny hops and broken lines, but has no effect when the crossover style is solid lines. The possible values of the property are shown in the following table.

0	Small
1	Medium
2	Large

For example, the following statements set crossovers to be medium bunny hops.

```
Chart.LineCrossoverStyle = 1        ' Bunny hops
Chart.LineCrossoverSize = 1         ' Medium
```

You also can use the **CrossoverSize** and **CrossoverStyle** properties of the Line_ object to specify the size and style of crossovers for individual lines. The parameters are the same.

{button Related Topics,PI(``,`IDH_RT_Setting_Line_Crossovers')}

CrossoverSize Property
CrossoverSize Property
LineCrossoverSize property
LineCrossoverStyle property

Attaching Text to Lines

Text on lines can describe the flow of information and relationships between connected shapes. You can choose the typeface, size, style, and color of the attached text. When you move a line, the attached text moves with it.

In FlowCharter, you attach text to a selected line by typing.

To attach text to lines using OLE Automation, you create a text block and then attach it to an existing line. See the Creating Text Blocks and Drawing Lines for more information on creating text blocks and lines.

Use the **AttachText** method of the Line_ object to attach text to a line. You can attach the text to the start or end of the line or to any segment of the line.

LineObject.**AttachText** *TextObject* [,*SegmentNumber*]

The *LineObject* specifies the line to which you are attaching the text. *TextObject* specifies the text block that you are attaching to the line. The *SegmentNumber* indicates the segment of the line to which the text is to be attached, as defined in the following table.

-3	Start
-2	End
-1	First
0	Last
1 through *n*	The sequential value of the line segment, where *n* is the number of segments in the line. For example, 1 is the first segment, and 2 is the second segment.

The following illustrations show how the text is placed on a line.

The following example places text on new lines as they are drawn.

```
Sub ABC1_NewLineNOTIFY ()
   Dim TextBlock As Object
   Set TextBlock = ABC1.Chart.DrawTextBlock ("Text on a Line")
   ABC1.Object.Line_.AttachText TextBlock -1
End Sub
```

To format text on a line, see Formatting Text.

{button Related Topics,PI(`',`IDH_RT_Attaching_Text_to_Lines')}

Deleting Lines

There are two ways to delete lines with OLE Automation: delete all lines attached to a specific shape or select a line and clear it or cut it.

Use the **DeleteLines** method of the Shape object to delete all the lines attached to a specified shape. Deleting lines with this method does not place the lines in the Windows Clipboard.

For example, the following statement deletes the lines attached to the shape referred to as ABCObject.

```
ABCObject.Shape.DeleteLines
```

{button Related Topics,PI(``,`IDH_RT_Deleting_Lines')}

DeleteLines method

Creating Text Blocks

A text block is a freeform, independent object in a chart. Text blocks are not associated with any shape or line.

In FlowCharter, you create text blocks by first clicking the Text tool in the toolbox. Then you can click anywhere in the chart and type, or you can drag the mouse to create the block and type.

To create a text block using OLE Automation, you first specify the current drawing position, then draw the text block and specify the text string.You access the information about text blocks using the **TextBlock** property of the Object object.

You use the **DrawPositionX** and **DrawPositionY** properties to specify the X and Y coordinates of the upper left corner of the text block. This defines the current drawing position for the chart. The X and Y coordinates are measured from the top left corner of the page. By default, the positions are measured in inches, but you can measure position in centimeters by changing the **Units** property of the Preferences object. See Setting the Current Drawing Position for more information.

Specifying the drawing position is not required, but helps to control the appearance of the chart. Alternatively, you can draw the text block first, and then move the text block to the desired location. See Moving Objects for more information on moving text blocks.

You use the **DrawTextBlock** method with the Chart object to draw a text block at the current position with the specified text string.

Tip

- Use Chr$(13) to add a new line to the text string in the **DrawTextBlock** method. For example, Chart.DrawTextBlock ("Line 1" + Chr$(13) + "Line 2")

The following example creates a text block two inches from the top of the page and one inch from the left of the page.

```
Dim ABCObject As Object

Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Set ABCObject = Chart.DrawTextBlock ("This is a text block")
```

{button Related Topics,PI(`',`IDH_RT_Creating_Text_Blocks')}

Moving Objects
Setting the Current Drawing Position

DrawPositionX property
DrawPositionY property
DrawTextBlock method
TextBlock property
Units property

Adding Text to a Shape

You can use OLE Automation to add text inside any shape. The text appears inside the text area defined for the shape. Adding text to a shape is equivalent to typing while a shape is selected in FlowCharter.

To add text to a shape, use the **Text** property of the Object. The following example creates a shape with text.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape ("External Operation")
ABCObject.Text = "Text inside a shape"
```

If you are reading the text from a shape, you can use the **TextLF** property to preserve the Returns. If you use the **Text** property, the Returns are changed to spaces.

```
ShapeText = Shape1.TextLF
```

{button Related Topics,PI(``,`IDH_RT_Adding_Text_to_a_Shape')}

Sizing Shapes to Text

Text property
TextLF property

Sizing Shapes to Text

You can automatically fit shapes to the size of the text inside them. This is especially useful when the length of the text string may be varied and you want to avoid hiding text that will not fit within the shape.

In FlowCharter, you do this by selecting the shape, and then clicking Fit to Text in the Arrange menu. In OLE Automation, you use the **FitShapeToText** method for the Shape object.

This example draws a shape, adds text to the shape, and then fits the shape to the text.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape ("Document")
ABCObject.Text = "This is a sample of fitting shapes to text"
ABCObject.Shape.FitShapeToText
```

{button Related Topics,PI(`',`IDH_RT_Sizing_Shapes_to_Text')}

Adding Text to a Shape

FitShapeToText method

Adding Notes to a Shape

You can attach notes to any shape in a chart. In FlowCharter, notes are added in the Note window. With OLE Automation, you do not need to open the Note window to attach notes to a shape.

You use the **NoteText** property of the Shape object to attach notes to shapes.

The following example draws a shape, then adds a note to the shape.

```
Dim ShapeObject As Object

Set ShapeObject = Chart.DrawShape ("Document")
ShapeObject.Text = "Text inside a shape"
ShapeObject.Shape.NoteText = "This is note text attached to the shape"
```

See the Formatting Text for information on formatting note text.

If you are reading the note text from a shape, you can use the **NoteTextLF** property to preserve the Returns. If you use the **NoteText** property, the Returns are changed to spaces.

```
NoteText = Shape1.NoteTextLF
```

Formatting Text

NoteTextLF property
NoteText property

Attaching Text to a Line

You can attach text to the start or end of a line or to any segment of a line. When text is attached to a line, the text is moved and positioned automatically with the line.

In FlowCharter you can attach text by dragging a text block onto a line or by selecting a line and then typing. To attach text to lines using OLE Automation, you must first create a text block, then attach it to a line.

You use the **DrawTextBlock** method to create the text block. See Creating Text Blocks for information about creating text blocks.

Use the **AttachText** method of the Line_ object to attach a text block to a line. The **AttachText** method has two parameters.

LineObject.**AttachText** *TextBlock* [,*SegmentNumber*]

The *TextBlock* parameter is the text block object you are attaching to the line in *LineObject*. The *SegmentNumber* is an optional parameter that indicates which segment of the line will contain the text.

The following table describes each value of *SegmentNumber*.

SegmentNumber	**Line Segment**
-3	Start
-2	End
-1	First
0	Last
1 through *n*	The sequential value of the line segment, where *n* is the number of segments in the line. For example, 1 is the first segment, and 2 is the second segment.

The default value of *SegmentNumber* is -1, which attaches a text block to the center of the first segment of the line.

The following illustrations show the segment on which text is placed based on the *SegmentNumber*.

The following statements draw a line, then create a text block and attach it to the line.

```
Dim LineObject As Object
Dim TextBlock As Object

DrawPositionX = 2
DrawPositionY = 1
Set LineObject = Chart.DrawFreeLine (4, 5)
Set TextBlock = Chart.DrawTextBlock ("Text on a line")
LineObject.Line_.AttachText (TextBlock, 1)
```

{button Related Topics,PI(`',`IDH_RT_Attaching_Text_to_a_Line')}

Unattaching Text from a Line

You can separate text from a line without deleting the text. This is equivalent to dragging the text away from a line in FlowCharter. Unattaching text from a line has no effect on the line.

To find if text is attached to a line, you use the **AttachedToLine** property of the TextBlock object. To unattach text, use the **UnattachFromLine** method and move the text to another position. For more information on moving text blocks, see Moving Objects.

The following example draws a line, creates a text block, then attaches the text block to the line. Then the example checks to see if the text is attached to the line, unattaches the text, and moves it to another location.

```
Dim LineObject As Object
Dim TextBlock As Object

DrawPositionX = 2
DrawPositionY = 1
Set LineObject = Chart.DrawFreeLine (4, 5)
Set TextBlock = Chart.DrawTextBlock ("Text on a line")
LineObject.Line_.AttachText (TextBlock, 1) = True

If TextBlock.AttachedToLine Then
   TextBlock.UnattachFromLine
   TextBlock.Left = 3
   TextBlock.Top = 6
End If
```

Attaching Text to a Line
Moving Objects

AttachedToLine property
UnattachFromLine method

Formatting Text

You can format a text object in FlowCharter by changing its typeface, its size, its text attributes (such as bold or italic), its color, and its alignment.

All text formatting applies to the entire object. You cannot format a single word differently than the rest of the text in the object.

{button Related Topics,PI(`',`IDH_RT_Formatting_Text')}

Text Typeface and Size

You can change the typeface and size of text, as well as determine the current typeface and size of the text.

Use the **Size** and **Name** properties of the Font object to specify the point size and typeface of the text, respectively. You access those properties using the **Font** property of the Object object or the **NoteFont** property of the Shape object.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Size = 10                ' Text block or text on a line
ABCObject.Font.Size = 10                ' Shape text
ABCObject.Shape.NoteFont.Size = 10      ' Note text
```

The **Size** property uses a Long value to specify point sizes, such as 10 or 12.

The **Name** property uses a string to determine the typeface. When changing the typeface, FlowCharter matches the highest quality typeface containing the string. For example, `Font.Name = "Roman"` sets the typeface to "Times New Roman" if both "Tms Roman" and "Times New Roman" are available, because "Times New Roman" is a TrueType typeface.

The following example checks to see if a shape's text is Arial 10 pt. If it is, the text changes to Times New Roman.

```
Dim ABCShape As Object

Set ShapeCollection = ABC.ActiveChart.Objects
Set ABCShape = ShapeCollection.ItemFromShapes

Do
   If ABCShape.Font.Name = "Arial" and ABCShape.Font.Size = 10 Then
        ABCShape.Font.Name = "Times New Roman"
   End If
   Set ABCShape = ShapeCollection.ItemFromShapes
Loop While ABCShape.Valid
```

Formatting Text

Font property
Name property
NoteFont property
Size property

Bold, Italic, Underline, and Strikethrough

You can change the attributes of text, such as bold, italic, underline, and strikethrough, as well as determine the current attributes of the text.

The **Bold, Italic, Underline,** and **Strikethrough** properties of the Font object take Boolean values that turn each attribute on or off.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Bold = True                ' Text block or text on a line
ABCObject.Font.Bold = True                ' Shape text
ABCObject.Shape.NoteFont.Bold = True      ' Note text
```

The following example creates a text block and makes the text bold and italic.

```
Dim ABCObject As Object

Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Set ABCObject = Chart.DrawTextBlock ("This is a text block")
ABCObject.Font.Bold = True
ABCObject.Font.Italic = True
```

The following example changes all strikethrough text in shapes to underline.

```
Dim ABCShape As Object

Set ShapeCollection = ABC.ActiveChart.Objects
Set ABCShape = ShapeCollection.ItemFromShapes

Do
   If ABCShape.Font.Strikethrough Then
        ABCShape.Font.Strikethrough = False
        ABCShape.Font.Underline = True
   End If
   Set ABCShape = ShapeCollection.ItemFromShapes
Loop While ABCShape.Valid
```

{button Related Topics,PI(``,`IDH_RT_Bold_Italic_Underline_and_Strikethrough')}

Formatting Text

Bold property
Italic property
Strikethrough property
Underline property

Text Color

You can change the color of text, as well as determine the current text color. The text color affects only the foreground color of the text; it does not affect the background color. See Text Background for information on the background of text.

You use the **Color** property of the Font object to specify the text color. You can use constants to specify one of the sixteen basic VGA colors, or you can specify the RGB values using the **MakeRGB** method.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Color = ABC.BLUE                 ' Text block or text on a line
ABCObject.Font.Color = ABC.BLUE                 ' Shape text
ABCObject.Shape.NoteFont.Color = ABC.BLUE       ' Note text
Chart.FieldFont.Color = ABC.BLUE                ' Field text
Chart.NumberFont.Color = ABC.BLUE               ' Shape numbers
Chart.MasterItems.Date.Font.Color = ABC.BLUE    ' Text in the Date Master Item
```

The following example changes all blue text in shapes to red. The color Blue is specified by a constant, and the color Red is specified by its RGB value.

```
Dim ABCShape As Object

Set ShapeCollection = ABC.ActiveChart.Objects
Set ABCShape = ShapeCollection.ItemFromShapes

Do
   If ABCShape.Font.Color = ABC.Blue Then
        ABCShape.Font.Color = ABC.MakeRGB (255, 0, 0)
   End If
   Set ABCShape = ShapeCollection.ItemFromShapes
Loop While ABCShape.Valid
```

Formatting Text
Text Background

Color property
MakeRGB method

Text Background

You can make the background behind text opaque or transparent. When a text background is opaque, you cannot see through the text to the objects beneath it. For example, when text is on a line, you can see the line through the text if the background is transparent; you cannot see the line if the background is opaque.

In FlowCharter you make a text background opaque by selecting the text, clicking Font in the Format menu, and selecting the Opaque option. You deselect the Text Background button to make the background transparent.

With OLE Automation, you use the **Opaque** property of the Font object to make a text background opaque or transparent.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Opaque = True                ' Text block or text on a line
ABCObject.Font.Opaque = True                ' Shape text
ABCObject.Shape.NoteFont.Opaque = True      ' Note text
```

{button Related Topics,PI(`',`IDH_RT_Text_Background')}

Formatting Text

Opaque property

Text Alignment

You can align the text inside shapes and in text blocks using the TextAlignment property of the object. The **TextAlignment** property uses the following Integer values to represent combinations of vertical and horizontal alignment.

Value	Vertical	Horizontal
0	Top	Left
1	Top	Center
2	Top	Right
3	Middle	Left
4	Middle	Center
5	Middle	Right
6	Bottom	Left
7	Bottom	Center
8	Bottom	Right

For example, the following line centers the text at the top of the shape.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape ("Document")
ABCObject.Text = "This is a sample of aligning text"
ABCObject.TextAlignment = 1
```

The following example creates a text block and left-justifies the text in the text block's vertical center.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawTextBlock ("This is a sample of aligning text")
ABCObject.TextAlignment = 3
```

{button Related Topics,PI(`',`IDH_RT_Text_Alignment')}

Formatting Text

TextAlignment property

Checking Spelling

You can check the spelling of selected text in FlowCharter by choosing the text you want to check and clicking Spelling in the Tools menu. If you do not select text, all text in the chart is checked. With OLE Automation, you use the **Spelling** method of the Chart object. For example, the following statement starts the Spelling Checker.

Chart.Spelling

{button Related Topics,PI(`',`IDH_RT_Checking_Spelling')}

Spelling method

Finding and Replacing Text

You can find text in FlowCharter by clicking Find or Replace on the Edit menu and filling in the options in the dialog box.

In OLE Automation, you can replace text using the **ReplaceText** method of the Chart object. There is no OLE equivalent to Find.

ChartObject.**ReplaceText** (*FindText*, *ReplacementText* [,*MatchCase*] [,*WholeWord*])

{button Related Topics,PI(`',`IDH_RT_Finding_Text')}

ReplaceText Method

Adding Data Fields to a Chart

In FlowCharter, you can attach a data field table to your chart's shapes. The data field table applies to all shapes in a chart, so any changes you make to the data field descriptions apply to all the shapes. You can store data field tables in templates for use in other charts. You use the **FieldTemplate** property to find the FieldTemplate object for a specified FieldValue object.

When you create a link from an existing chart to a new chart, the existing chart's field table is copied to the new chart and the values are accumulated according to the methods you specify. That means you do not have to recreate the data field descriptions in each new chart.

For example, a chart's data fields might be Assigned, Due, and Cost. A specific shape might have the value "Britt Barnes" in the Assigned field, "01/09/95" in the Due field, and "$1,200" in the Cost field.

In FlowCharter, you add data fields by choosing the Data Field command on the Insert menu to open the Setup Fields dialog box, entering a name for the field, and selecting a type for the field.

In OLE Automation, data fields are stored in the FieldTemplates collection. You use the **FieldTemplates** property to find the FieldTemplates collection of the Chart object.

When you create a field using the **Add** method, it is added to that collection. The **Add** method has two parameters. The first is the name of the field. The second parameter, which is optional, sets the type of field. The following table shows the values of the second parameter and their meanings.

0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number (default if the parameter is omitted)

You set the name, type, format, accumulation method, and hidden options of a field using properties of the FieldTemplate object as described in Changing Data Field Attributes.

The last statement in the following subroutine adds the object Field1 to the FieldTemplates collection. The field has the field name "Assigned" and is text type because of the 0 value in the second parameter. (If you omit the second parameter, the default is 5, which specifies a number.) The field is created with the default attributes for a text object.

```
Sub Form_Load ()
   Dim ABC As Object
   Set ABC = CreateObject("ABCFlow.application")
   ABC.Visible = True
   Dim Chart As Object
   Set Chart = ABC.New
   Dim Field1 As Object
   Set Field1 = Chart.FieldTemplates.Add("Assigned", 0)
   Dim Shape1 As Object
   Set Shape1 = Chart.DrawShape("Delay")                    'Draw a Delay shape
   Shape1.FieldValues.Item("Assigned").Value = "Mary Smith" 'Enter text in the field
```

```
End Sub
```

{button Related Topics,PI(`',`IDH_RT_Adding_Data_Fields_to_a_Chart')}

Changing Data Field Attributes

Add method
FieldTemplate property
FieldTemplates property

Changing Data Field Attributes

After you create a field, you can describe its attributes using the **Name**, **Type**, **Format**, **AccumulationMethod**, and **Hidden** properties of the FieldTemplate object. These properties correspond to the options in the Setup Fields dialog box.

The name of a field appears in the chart next to the field value. The **Name** property lets you rename a field, such as "Task 1" You name a field at the time you create it using the **Add** method, so you only use this property to change the name of a field.

Data fields can have one of six major types.

0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number (default if the parameter is omitted)

Most of the major types have choices within them. For example, dates can be M/D/YY (8/1/94), MMM-D-YY (Aug-1-94), MMMM DD, YYYY (August 01, 1994), and so forth. The **Format** property lets you specify the format of the field.

100	# w.	200	M/D/YY
101	# weeks	201	MMMM-D-YY
102	# d.	202	MMMM DD, YYYY
103	# days	203	MMM-YY
104	# h.	204	MMMM YYYY
105	# hrs.		
106	# hours		
107	# m.	300	$###0.00($###0.00)
108	# mins.	301	$#,##0.00($#,##0.00)
109	# minutes	302	$###0($###0)
110	# s.	303	$#,##0($#,##0)
111	# secs.		
112	# seconds		
113	# TMU	500	###0
114	h:m	501	###0.00
115	m:s	502	###0.0000
116	h:m:s	503	#,##0
		504	#,##0.00
		505	#,##0.0000
400	##%		
401	#0.00%		

The **AccumulationMethod** property lets you specify the type of accumulation used to calculate the Legend values and the linked fields' values in any linked shapes.

0	No Accumulation
1	Sum
2	Mean
3	Median
4	Min
5	Max
6	Range
7	Total
8	Non-Null Total

The **Hidden** property lets you specify whether a field and its value are displayed in the chart. Set the **Hidden** property to True to hide the field or to False to display the field.

For example, the following subroutine, which assumes that there is a field named Cost, changes the field's attributes so that it is named Expense, changes the type and format to text, sets the accumulation method to No Accumulation, and makes it hidden.

```
Sub ABC1_ChartChangeNOTIFY ()
    Dim Chart As Object
    Set Chart = ABC.ActiveChart

    Dim CurrentField As Object
    Set CurrentField = Chart.FieldTemplates.Item("Cost")
    CurrentField.Name = "Expense"
    CurrentField.Type = 0
    CurrentField.Format = 0
    CurrentField.AccumulationMethod = 0
    CurrentField.Hidden = True
End Sub
```

{button Related Topics,PI(`',`IDH_RT_Changing_Data_Field_Attributes')}

Adding Data Fields to a Chart

AccumulationMethod property
Add method
Format property
Hidden property
Name property
Type property

Deleting Data Fields from a Chart

Deleting a data field removes it from every shape in the chart and deletes its values. You delete data fields using the **DeleteField** method in the FieldTemplates collection.

FieldTemplatesCollection.**DeleteField** *FieldTemplateObject*

With the **DeleteField** method, you provide the FieldTemplate object as a parameter. For example, the following statement deletes the field object contained in CurrentField.

```
Chart.FieldTemplates.DeleteField CurrentField
```

The following subroutine, which assumes that there are data fields, uses the **Count** property, the **Item** method, and the **DeleteField** method. It searches through the data fields in the FieldTemplates collection and deletes any that have a type of 3 (currency).

```
Sub ABC1_ChartCloseSUBCLASS ()
   Dim Chart As Object
   Dim FieldTemplates As Object
   Set Chart = ABC.ActiveChart

   Dim CurrentField As Object
   Set FieldTemplates = Chart.FieldTemplates
   For FieldNumber = 1 to FieldTemplates.Count
        Set CurrentField = FieldTemplates.Item(FieldNumber)
        If CurrentField.Type = 3 Then
             FieldTemplates.DeleteField CurrentField
        End If
   Next FieldNumber
End Sub
```

See the Finding Objects in a Chart for more information on using collections.

{button Related Topics,PI(`',`IDH_RT_Deleting_Data_Fields_from_a_Chart')}

Finding Objects in a Chart

Count property
DeleteField method
Item method

Setting Data Field Options

You can set preferences for data fields, such as the font used to display data fields in a chart and the placement of data fields relative to shapes.

In FlowCharter, you set data field preferences by clicking Chart Properties on the Format menu, clicking the Data Fields tab, and then choosing options in the dialog box. All the settings in this dialog box are also available using OLE Automation.

The **FieldNamesHidden** property of the Chart object lets you choose whether you want to show the field names in a chart. Values in the field are not affected by this property. Set the **FieldNamesHidden** property to True to hide the field names or to False to display them.

The **FieldPlacement** property of the Chart object lets you position fields in relation to their associated shapes. You use the values in this table.

0	Left
1	Right
2	Above
3	Below
4	Inside Top
5	Inside Middle

The **FieldsOpaque** property of the Chart object lets you choose whether the background of fields is opaque.

The **FieldsHoursPerDay** property of the Chart object lets you set the number of hours in a workday. This value is used when a field is converted between hours and days. For example, the value is used if you change the field's format from hours to days or you link to a chart that displays fields in a different format.

The **FieldsDaysPerWeek** property of the Chart object lets you set the number of days in a work week. This value is used when a field is converted between days and weeks when totaling durations.

The **FieldFont** property of the Chart object returns the Font object used for fields. You can format field text just as you format other text objects. The formatting for fields applies to all fields in the chart and to the text in the Legend.

The **FieldFont** property contains the following properties.

Bold	True if text is **bold**; False if text is not bold
Italic	True if text is *italic*; False if text is not italic
Strikethrough	True if text is ~~strikethrough~~; False if text is not strikethrough
Underline	True if text is underline; False if text is not underline
Opaque	True if text is opaque; False if text is not opaque
Name	The typeface name of the font
Size	The point size of the font

See **Formatting Text** for more information on these properties.

The following example sets preferences for data fields in a chart.

```
Dim ABC As Object
Dim Chart As Object
Set ABC = CreateObject ("ABCFlow.application")
ABC.New
Set Chart = ABC.ActiveChart

Chart.FieldFont.Name = "Arial"
Chart.FieldFont.Size = 12
```

```
Chart.FieldFont.Bold = True
Chart.FieldFont.Italic = True
Chart.FieldFont.Strikethrough = False
Chart.FieldFont.Underline = False
Chart.FieldFont.Opaque = False
```

{button Related Topics,PI(``,`IDH_RT_Setting_Data_Field_Preferences')}

Formatting Text

Bold property
FieldFont property
FieldNamesHidden property
FieldPlacement property
FieldsDaysPerWeek property
FieldsHoursPerDay property
FieldsOpaque property

Italic property
Name property
Opaque property
Size property
Strikethrough property
Underline property

Working with Data Field Values

You can enter values into fields for any shape in a chart. You do not have to enter values for all fields or all shapes.

In FlowCharter, you enter values in fields by selecting Field Viewer on the View menu, and working in the Field Viewer dialog box.

The FieldValues collection of the Object object contains the data field values. To enter values, you specify the values using the **Value** property of the FieldValue object and the **Item** method of the FieldTemplates collection. You access the values of data fields using the **FieldValues** property of the Object object.

For example, the following statements set values for an existing shape, Shape1. The statements assume that the chart has the fields Assigned, Due, and Cost.

```
Shape1.FieldValues.Item("Assigned").Value = "Beginning"
Shape1.FieldValues.Item("Due").Value = "1/3/95"
Shape1.FieldValues.Item("Cost").Value = "200"
```

You also can read values. For example, the following subroutine reads the Profit in all shapes and turns the text red for all that are negative.

```
Sub ABC1_DoubleClickSUBCLASS ()
   Dim ABC As Object
   Dim Chart As Object
   Set Chart = ABC.ActiveChart

   Dim AllShapes As Object
   Set AllShapes = Chart.Objects

   Dim Shape As Object
   For ChartCount = 1 to AllShapes.Count
      Set Shape = AllShapes.ItemFromShapes(ChartCount)
      If Shape.FieldValues.Item("Profit").Value < 0 Then
         Shape.Font.Color = ABC.RED
      End If
   Next ChartCount
End Sub
```

You can read or set the day, month, and year of a Date field (Type = 2) using the **Day** property, **Month** property, and **Year** property of the FieldValue object. For example, the following statements change any October 15 due dates to October 17.

```
Set Field = Shape1.FieldValues.Item("Due")
If Field.Month = 10 And Field.Day = 15 Then Field.Day = 17
```

You can find out if a data field contains any value using the **IsEmpty** property of the FieldValue object. For example, the following statement puts into the variable NoCostExists whether the existing "Cost" data field for Shape1 is empty.

```
NoCostExists = Shape1.FieldValues.Item("Cost").IsEmpty
```

You can read the format of the value in a data field using the **FormattedValue** property of the FieldValue object. For example, the following statement puts the formatted value from the existing "Cost" data field for Shape 1 into the variable CostFormat.

```
CostFormat = Shape1.FieldValues.Item("Cost").FormattedValue
```

You use the **Accumulation** property in the FieldTemplate object to return the accumulated value for a specific field. For example, the following statements calculate the accumulated value of the field template created as Cost in TotalCost.

```
Dim TotalCost As Double
TotalCost = Cost.Accumulation
```

The **Accumulation** property is read only.

{button Related Topics,PI(`',`IDH_RT_Working_with_Data_Field_Values')}

Knowing When Data Fields Change
Opening the Field Viewer

Accumulation property
Day property
FieldValues property
FormattedValue property
IsEmpty property
Month property
Value property
Year property

Item method

Knowing When Data Fields Change

Sometimes you want to know when the user has changed a data field value so you can react to that change. You can use the **FieldValueChangedNOTIFY** event to be notified when a field has changed.

Note

- For custom contol notification to function, you must register events using the **RegisterEvent** method of the Application object. (See Registering Event Procedures.) For example, the following statement registers the **FieldValueChangedNOTIFY** event.

```
ABC.RegisterEvent ABC1, Form1.Caption, "FieldValueChangedNOTIFY"
```

Note

- The RegisterEvent method requires that you use FlowCharter custom controls with your project. The FlowCharter custom controls are in ABCAUTO.VBX and FLOW.OCX.
If you are using Visual Basic 3.0 or earlier, add ABCAUTO.VBX. If you are using Visual Basic 4.0, add FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX. See To install the VBX event handler for details on installing ABCAUTO.VBX.

When you are notified with the **FieldValueChangedNOTIFY** event that a field has changed, you can use the FieldValue object of the OCX or VBX (ABC1.FieldValue) to access that field. ABC1.Object tells you which shape or line had the change made in the field value. (ABC1.Chart tells you which chart the object is in.)

The following subroutine checks any changed field. If the field is the cost and if it is zero, then the object changes to red.

```
Sub ABC1_FieldValueChangedNOTIFY ()
   If ABC1.FieldValue.Name = "Cost" And ABC1.FieldValue.Value = 0 Then
      ABC1.Object.Color = ABC1.App.RED
   End If
```

{button Related Topics,PI(`',`IDH_RT_Knowing_When_Data_Fields_Change')}

FieldValueChangedNOTIFY event
RegisterEvent method

Opening the Field Viewer

The Field Viewer dialog box is used in FlowCharter to enter and display data in fields for a selected shape.

You open the Field Viewer using the **FieldViewerVisible** property in the Application object. Set the property to True to display the Field Viewer or to False to hide it.

```
ABC.FieldViewerVisible = True        ' Field Viewer is open
ABC.FieldViewerVisible = False       ' Field Viewer is closed
```

FieldViewerVisible property

Viewing the Legend

The Legend in FlowCharter shows the totals of the data fields in a chart. The totals reflect the current state of the chart and update automatically when any field changes.

The totals in the Legend have the same font and style as other data field fonts. See Setting Data Field Preferences for information about formatting field data.

Fields with the accumulation method No Accumulation do not appear in the Legend. See Changing Data Field Attributes for information on setting a data field's accumulation method.

In FlowCharter, you show or hide the Legend by clicking Legend on the Insert menu. In OLE Automation, you use the **ShowLegend** property of the Chart object to show or hide the Legend or to determine whether the Legend is already displayed. When **ShowLegend** is True, the Legend displays; when it is False, the Legend is hidden.

The following statement shows the Legend.

```
Chart.ShowLegend = True
```

Use the following statement to hide the Legend.

```
Chart.ShowLegend = False
```

Changing Data Field Attributes
Setting Data Field Preferences

ShowLegend property

Using Linked Field Data

One of the most useful features of FlowCharter is linked charts. Linking charts lets you have a top-level chart showing only summaries, and then go quickly to the linked charts to see details that would otherwise obscure the overall picture. You can find information about linking using the help system for FlowCharter.

The **LinkFields** property in the Shape object returns True if the object's field data show the accumulation of the field data in the linked chart. For example, the following statements set the value of LinkedData to the value of the Cost field if the object is linked to another chart and shows the information from that chart.

```
If Shape1.LinkFields Then
   LinkedData = Shape1.Fields.Item("Cost").Value
End If
```

The **UpdateFields** method in the Chart object updates all the fields for all the linked shapes in a chart so they reflect the values in the linked charts. It is the equivalent of clicking the Update Data Fields button on the Data toolbar. For example, the following statement updates the fields for all shapes in the chart that are linked to another chart.

```
Chart.UpdateFields
```

For information on the **LinkNOTIFY** event, the **LinkIndicator** property, the **LinkShadow** property, the **IsLaunched** property, the **IsLinked** property, the **LinkedChartName** property, and the **Link** method, see Linking Charts.

You can empty data fields using the **Empty** method of the FieldValue object. For example, the following statement empties the value from the existing "Assigned" data field for Shape1.

```
Shape1.FieldValues.Item("Assigned").Empty
```

Linking Charts

LinkNOTIFY event

Empty method
Link method
UpdateFields method

IsLaunched property
IsLinked property
LinkedChartName property
LinkFields property
LinkIndicator property
LinkShadow property

Color Constants Description

When you want to set colors quickly and want to choose only among the sixteen VGA colors, you can use the colors whose names are defined as constants in OLE Automation, such as ABC.BLUE. The following table lists the color constants.

Color	Name
White	WHITE
Black	BLACK
Red	RED
Green	GREEN
Blue	BLUE
Yellow	YELLOW
Magenta	MAGENTA
Cyan	CYAN
Gray	GRAY
Dark Red	DK_RED
Dark Green	DK_GREEN
Dark Blue	DK_BLUE
Dark Yellow	DK_YELLOW
Dark Magenta	DK_MAGENTA
Dark Cyan	DK_CYAN
Dark Gray	DK_GRAY

For example, suppose you want to set the color of shape numbers to red. The following statements do that.

```
Dim ABC As Object
Set ABC = CreateObject (ABCFlow.Application)
ABC.New                                          ' Create a new chart
ABC.ActiveChart.NumberFont.Color = ABC.RED
```

It does not matter what variable you use for the FlowCharterapplication. The "ABC" in the previous statements depends on the **Dim**. The following statements have exactly the same effect.

```
Dim ABCApplication As Object
Set ABCApplication = CreateObject (ABCFlow.Application)
ABC.New                                          ' Create a new chart
ABCApplication.ActiveChart.NumberFont.Color = ABCApplication.RED
```

Note

- You cannot change the values of the constants. For example, you cannot make the constant **DK_YELLOW** yield the color red.

{button Related Topics,PI(`',`IDH_RT_Color_Constants_Description')}

Color Equivalents

BasicColor Method Description

A quick way to set the color is using the array of colors in the **BasicColor method** of the Application object. As with the defined constants, the **BasicColor method** lets you set colors from the sixteen VGA colors. For example, the following statement sets the color of shape numbers color to blue.

```
ABC.ActiveChart.NumberFont.Color = ABC.BasicColor(4)
```

The following chart lists the **BasicColor method** values.

Color	BasicColor
White	0
Black	1
Red	2
Green	3
Blue	4
Yellow	5
Magenta	6
Cyan	7
Gray	8
Dark Red	9
Dark Green	10
Dark Blue	11
Dark Yellow	12
Dark Magenta	13
Dark Cyan	14
Dark Gray	15

Note

- You cannot change the values in the **BasicColor method**. For example, you cannot make **BasicColor**(10) yield the color purple.

{button Related Topics,PI(`',`IDH_RT_BasicColor_Method_Description')}

RGB Values

If you want to use a color that is not one of the 16 defined in OLE Automation, you can specify the color as quantities of red, green, and blue. You specify each color with a number from 0 (no color) through 255 (solid color). By specifying values for red, green, and blue, you can choose from over 16 million colors. You specify the colors using the **MakeRGB method** of the Application object.

For example, **MakeRGB**(0,0,255) is no red, no green, and solid blue, so it specifies blue. **MakeRGB**(255,255,0) is solid red, solid green, and no blue, so it specifies yellow. **MakeRGB**(127,0,255) is 50% red, no green, and solid blue, so it specifies a purple color. For example, the following statement sets the color of shape numbers to purple.

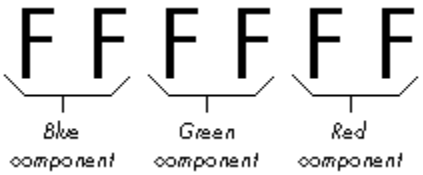
```
ABC.ActiveChart.NumberFont.Color = ABC.MakeRGB(127,0,255)
```

{button Related Topics,PI(`',`IDH_RT_RGB_Values')}

Color Double Values

You can specify a color as a double. You probably only want to use this method when you are passed a color from another application.

In this method, the color equals the decimal equivalent of six hexadecimal digits. The first two hexadecimal digits set the blue component, the second two set the green component, and the third two set the red component. (Notice that the order is reversed from when you are using **MakeRGB**.)



For example, FF0000 in hexadecimal means solid blue, no green, and no red. If you set an object to &HFF0000 (or to 16777216, the decimal equivalent of &HFF0000), the object is set to blue.

Most programmers work in hexadecimal rather than decimal for colors, using the Visual Basic language element &H to specify that the following number is in hexadecimal, but you can use whichever makes you most comfortable. For example, the following statements both set the color of shape numbers to blue.

```
ABC.ActiveChart.NumberFont.Color = &HFF000
ABC.ActiveChart.NumberFont.Color = 16777216
```

Tip

You can use the Windows Calculator to convert between hexadecimal and decimal numbers. After choosing Scientific in the Calculator's View menu, type a number and select Hex or Dec to convert it. See your Windows documentation for more information.

Color Equivalents

The following table shows the equivalents for the sixteen VGA colors for the four-color methods. You can set over sixteen million different colors using **MakeRGB** or double values.

Color	Name	BasicColor	MakeRGB	Double (Decimal)	Double (Hex)
White	WHITE	0	(255,255,255)	16777215	FFFFFF
Black	BLACK	1	(0,0,0)	0	0
Red	RED	2	(255,0,0)	255	FF
Green	GREEN	3	(0,255,0)	65280	FF00
Blue	BLUE	4	(0,0,255)	16711680	FF0000
Yellow	YELLOW	5	(255,255,0)	65535	FFFF
Magenta	MAGENTA	6	(255,0,255)	16711935	FF00FF
Cyan	CYAN	7	(0,255,255)	16776960	FFFF00
Gray	GRAY	8	(192,192,192)	12632256	C0C0C0
Dark Red	DK_RED	9	(127,0,0)	127	7F
Dark Green	DK_GREEN	10	(0,127,0)	32512	7F00
Dark Blue	DK_BLUE	11	(0,0,127)	8323072	7F0000
Dark Yellow	DK_YELLOW	12	(127,127,0)	326397	7F7F
Dark Magenta	DK_MAGENTA	13	(127,0,127)	8323199	7F007F
Dark Cyan	DK_CYAN	14	(0,127,127)	8355584	7F7F00
Dark Gray	DK_GRAY	15	(127,127,127)	8355711	7F7F7F

The following five statements all do the same thing. Each changes the shape's fill color to blue.

```
Object.Shape.FillColor = ABC.BLUE
Object.Shape.FillColor = ABC.BasicColor(4)
Object.Shape.FillColor = ABC.MakeRGB(0,0,255)
Object.Shape.FillColor = 16711680
Object.Shape.FillColor = &HFF0000
```

You can use the **MakeRGB method** of the Application object to find the double equivalent of RGB values. For example, the following statement puts the double value for blue (16711680) into the variable CurrentColor.

```
Dim CurrentColor As Long
CurrentColor = ABC.MakeRGB(0,0,255)
```

{button Related Topics,PI(`',`IDH_RT_Color_Equivalents')}

Setting Shape Colors

You can color a shape by setting its fill color, its border color, and its shadow color.

You set the fill color for shapes using the **FillColor** property of the Shape object or the **Color** property of the Object object. Both properties produce the same effect. For example, the following statements draw a shape and then change its fill color to blue using the **FillColor** property. They then change its fill color to red using the **Color** property.

```
Dim NewObj1 As Object
Set NewObj1 = Chart.DrawShape
NewObj1.Shape.FillColor = ABC.BLUE
NewObj1.Color = ABC.RED
```

You set the border color for shapes using the **BorderColor** property of the Shape object. For example, the following statement makes the border of a shape green.

```
NewObj1.Shape.BorderColor = ABC.GREEN
```

You set the shadow color for shapes using the **ShadowColor** property of the Shape object. For example, the following statement makes the shadow of a shape gray.

```
NewObj1.Shape.ShadowColor = ABC.GRAY
```

BorderColor property
FillColor property
Color property
ShadowColor property

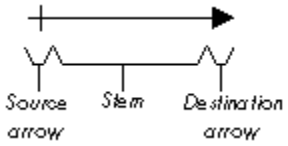
Setting Line Colors

You set the colors for lines by coloring the entire line at once or by coloring parts of the line. You use the **Color** property to set the color for the entire line, including the source arrow, the stem, and the destination arrow.

You can use the **Color** property with either the Line_ object or the Object object. Using either object produces the same effect. For example, the following statements draw a line and then change its color to blue using the Line_ object. They then change the line's color to red using the Object object.

```
Dim NewObj1 As Object
Set NewObj1 = DrawFreeLine (3,4)
NewObj1.Line_.Color = ABC.BLUE
NewObj1.Color = ABC.RED
```

You can color parts of a line by coloring the source arrow, the destination arrow, and the stem.



You color the arrow at the source of a line using the **SourceArrowColor** property of the Line_ object. For example, the following statement makes a source arrow red.

```
NewObj1.Line_.SourceArrowColor = ABC.RED
```

You color for stem of a line (the part excluding the destination arrow and source arrow) using the **StemColor** property of the Line_ object. For example, the following statement makes a stem yellow.

```
NewObj1.Line_.StemColor = ABC.YELLOW
```

You color the arrow at the destination of a line using the **DestArrowColor** property of the Line_ object. For example, the following statement makes a destination arrow blue.

```
NewObj1.Line_.DestArrowColor = ABC.BLUE
```

{button Related Topics,PI(`',`IDH_RT_Setting_Line_Colors')}

Color property
DestArrowColor property
SourceArrowColor property
StemColor property

Setting Text Colors

You can set the color of all the text objects that can occur in FlowCharter. You set the color for text using the **Color** property of the Font object.

There are variations in how you set text color depending on where the text occurs. The following examples show each type of text object.

```
Object.Font.Color = ABC.BLUE                    ' Text block or text on a line
Object.Font.Color = ABC.BLUE                    ' Shape text
Object.Shape.NoteFont.Color = ABC.BLUE          ' Note text
Chart.FieldFont.Color = ABC.BLUE                ' Field text
Chart.NumberFont.Color = ABC.BLUE               ' Shape numbers
Chart.MasterItems.Date.Font.Color = ABC.BLUE    ' Text in the Date master item
```

You set the color of all the MasterItems text objects in the same way as the **Date** object in the above example. To set the text color for the **ChartName**, **Logo**, **PageNumber**, **Text1**, **Text2**, and **Time** MasterItems objects, replace **Date** in the above example with the appropriate MasterItems object.

For more information on formatting text, see Formatting Text.

{button Related Topics,PI(`',`IDH_RT_Setting_Text_Colors')}

Formatting Text

ChartName property
Color property
Date property
Logo property
PageNumber property
Text1 property
Text2 property
Time property

What Are FlowCharter Events?

FlowCharter events are the key to running an automation program. FlowCharter sends out a signal when a FlowCharter event occurs. An event occurs when something happens in FlowCharter. For example, when the user presses the **DEL** key, an event occurs; FlowCharter sends out an event signal, or notification.

As a developer, you can write a program in Visual Basic to access FlowCharter using "Application events." Or you can control FlowCharter internally using Visual Basic script and/or Living FlowCharts. For example, suppose you want an program to turn any shape gray when the user tries to delete it. Since you don't know when the user will delete a shape, and you don't know what shape the user will delete, you want your program alerted each time the user presses the **DEL** key.

Other examples of FlowCharter events are creating a new chart, selecting a shape, moving a shape, replacing a shape, and drawing a line. FlowCharter events can be triggered by actions of the user or by program instructions.

Event Procedures

Each time an event occurs, your program or script can run a procedure specific for that event. The procedure that runs as a result of an event is called an *event procedure*.

Example 1

When a user presses the **DEL** key, your program or script can automatically run the **DeleteSUBCLASS** event procedure.

Example 2

You can write a procedure that runs whenever someone clicks on a specific shape. This example shows you how to write Visual Basic script that runs when the user clicks on a shape.

1 Click the shape for which you want to write the procedure.

2 On the Tools menu, click Edit Object Script.

3 In the Proc box, click the down arrow and click OnClick. The Script Editor inserts the procedure’s stub.

4 Write the script that you want to run when the shape is clicked.

5 When you are finished, click Close.

Three Levels of Event Procedures

FlowCharter now has three levels of event procedures: the object level, the chart level, and the application level (OCX or VBX). When an event occurs, the first event procedure that runs is the Object object’s event procedure. The second is the Chart object’s event procedure. The third is the application level event procedure.

All object events have equivalent chart events. For example, an object has an OnClick event. The chart has an event called OnObjectClick that occurs when any object in the chart is clicked.

Not all events are available for all levels. Some events do not occur to objects, but only to charts. For example, when the user closes the chart, there is no object equivalent. In this case, the chart’s OnClose event procedure runs, then the application level event procedure runs.

The Object object and Chart object events are only accessible via object script and chart script. You can only access FlowCharter events externally via the "Application events."

Each of these three event levels has its own set of events that fire to it. They are listed below:

Object Object Events

OnAbort	OnLeave
OnBeginEditFields	OnLineAttach
OnBegineditText	OnLineDeattach
OnChangeLayer	OnLink

OnChangeLayerFinished
OnClearLaunch
OnClearLine
OnClick
OnClockTick
OnContextMenuCommand
OnContextMenuHint
OnContextMenuPopup
OnCreate
OnDelete
OnDeselect
OnDoubleClick
OnExecute
OnFieldValueChanged
OnFinish
OnFontChange
OnInitiateEntity
OnLaunch
OnMove
OnMoveFinished
OnQueryAcceptEntity
OnReplace
OnRotate
OnRotateFinished
OnSelect
OnSetLaunch
OnSetLink
OnSize
OnSizeFinished
OnStart
OnStyleChange
OnTextChange
OnUserEvent

Note: Details on Object object events and Chart object events are on our web site, at http://www.micrografx.com/

Chart Object Events

OnAbort
OnActivate
OnClick
OnClockTick
OnClose
OnClosePalette
OnContextMenuPopup
OnContextMenuCommand
OnContextMenuHint
OnCreate
OnCustomSave
OnDeactivate
OnDefFontChange
OnDefStyleChange
OnDelete
OnDoubleClick
OnFieldSetupDialog
OnFieldViewerHide
OnFieldViewerShow
OnFinish
OnLayerAdd
OnLayerAddFinished
OnLayerDeleter
OnLayerDeleteFinished
OnLayerLock
OnLayerRename
OnLayerShow
OnMenuCommand
OnMenuHint
OnMenuPopup
OnModify
OnObjectBeginEditFields
OnObjectBeginEditText
OnObjectCreate
OnObjectDelete
OnObjectDeselect
OnObjectDoubleClick
OnObjectExecute
OnObjectFieldValueChanged
OnObjectFontChange
OnObjectInitiateEntity
OnObjectLaunch
OnObjectLeave
OnObjectLineAttach
OnObjectLineDeattach
OnObjectLinkClear
OnObjectMove
OnObjectMoveFinished
OnObjectQueryAcceptEntity
OnObjectReplace
OnObjectRotate
OnObjectRotateFinished
OnObjectSelect
OnObjectSetLaunch
OnObjectSetLink
OnObjectSize
OnObjectSizeFinished
OnObjectStyleChange
OnObjectTextChange
OnOpen
OnOpenPalette
OnPaste
OnQueryCanClose
OnSave
OnSelectionChange
OnSpecialKey

OnObjectChangeLayer
OnObjectChangeLayerFinished
OnObjectClearLaunch
OnObjectClick
OnObjectClockTick
OnObjectContextMenuCommand
OnObjectContextMenuHint
OnObjectContextMenuPopup
OnStart
OnUserEvent

Note: Details on Object object events and Chart object events are on our web site, at http://www.micrografx.com/

Application events (Events sent to external applications OLE Automation, via OCX or VBX)

AppQuitNOTIFY
AppQuitSUBCLASS
AppMenuHintSUBCLASS
AppMenuPopupSUBCLASS
AppMenuSUBCLASS
ChartActivateNOTIFY
ChartDeActivateNOTIFY
ChartChangeNOTIFY
ChartCloseSUBCLASS
ChartNewNOTIFY
ChartOpenNOTIFY
ChartPasteNOTIFY
DeleteSUBCLASS
DoubleClickSUBCLASS
ExclusiveSelectionNOTIFY
FieldValueChangedNOTIFY
LinkNOTIFY
NewLineNOTIFY
NewShapeNOTIFY
ObjectClickSUBCLASS
ObjectFontChangeNOTIFY
ObjectLineAttachNOTIFY
ObjectLineDeAttachNOTIFY
ObjectMovedNOTIFY
ObjectMoveSUBCLASS
ObjectSizedNOTIFY
ObjectSizeSUBCLASS
ObjectTextChangedNOTIFY
ReplaceShapeNOTIFY
SpecialKeySUBCLASS

Warning

- You should not put a method inside an event if that method generates the event. For example, you should not put a call to the **Link** method inside the **LinkNOTIFY** event. If you do so, the program will go into an endless loop, FlowCharter will crash, Visual Basic may crash, and Windows will become unstable.

Note

- OLE Automation for external applications (via OCX or VBX) requires that you identify the event procedures you want executed by registering the events. For more information on registering events, see Registering Event Procedures.

Types of Events: Subclass and Notify

There are two types of events:subclass and notify. Subclass events carry out the standard behavior **after** they run the event procedure. Notify events carry out the standard behavior **before** running the event procedure.

The most important difference between the two event types is that with Subclass events, you can perform your own behavior *instead* of the standard behavior. Notify events perform your programmed behavior *in addition* to the standard behavior.

Subclass Events

Subclass events are events that run the event procedure before carrying out the standard behavior, thus allowing you to intervene in the standard behavior. You can recognize Subclass events by the presence of a Handled parameter.

When a subclass event occurs, three steps happen, in this order:

1 FlowCharter runs the event's procedure, if any.

2 FlowCharter notifies the next level of event procedures.

3 FlowCharter carries out the event's standard behavior.

All three of these steps happen only if you do not stop the event process. You can stop the event process of a

subclass event in two ways: you can stop its standard behavior from occurring and you can stop FlowCharter from notifiying the next event level.

When an event occurs, FlowCharter calls the event procedure for the object to which the event occurred. For example, shapes have OnClick event procedures. When you click a shape, a click event occurs, and the shape's OnClick procedure runs. FlowCharter then notifies the chart that an object click event has occurred, then notifies the application interface that a click event has occurred.

If the event was a chart event, FlowCharter calls the chart's event procedure. For example, charts have an OnClose procedure. When a user closes the chart, the OnClose procedure runs. Last, FlowCharter closes the chart.

Subclass event procedures are identified either by the use of SUBCLASS in the event's name or by its Handled parameter. Some subclass events also have an Allow parameter.

Notify Events

Notify events are events that carry out their standard behaviors first, before they run the event procedure. When a notify event occurs, three things happen, in this order:

1 FlowCharter carries out the event's standard behavior.

2 FlowCharter runs the event's procedure, if any.

3 FlowCharter notifies the next level of event procedures.

For example, you can have a shape that is always the same height and width. This example shows an object event procedure that occurs after the user has rersized an object. The event is a notify event and occurs when the resizing is finished. Thee event passes the new height and width to the object event.

```
Sub Object_OnSizeFinished((Newheight, NewWidth)

Width = NewHeight

EndSub
```

Event handlers are not recursive. That is, although the above example changes the size of the object, these changes do not cause the OnSizeFinished event to be called again.

Notify event procedures are identified either by the use of NOTIFY in the event's name or by the fact that the event does NOT have a Handled parameter. Some Notify event procedures also have event names that end with something like Finished, Moved or Changed.

Stopping the standard behavior of a subclass event

Stopping the standard behavior of an Object object or Chart object event

You can stop the standard behavior of an Object object or Chart object event. You may want to prevent a behavior like a deletion when you want to protect an object from being deleted. In the event procedure, you could display a dialog box advising the user that this object cannot be deleted.

To prevent a standard behavior, use the Allow parameter and set it to FALSE. Put this line of script into the event procedure for the object. For example, to prevent a shape from being deleted, put this line of script into the shape's OnDelete procedure:

```
Allow.Value = FALSE
```

To permit the standard behavior of an event set the Allow parameter to TRUE. This line is usually not necessary because the default value is TRUE. You would only need to do this if you had set the value to FALSE elsewhere.

```
Allow.Value   = TRUE
```

Usually, you will want to use this line of script with a line that prevents further event notification. Stopping event

notification is discussed in the next section.

Preventing Further Event Notification

To prevent the further notification of events to the other levels, use the Handled parameter.

You may want to stop the event notification process because you have prevented the standard behavior, like a deletion. If you do not want a shape to be deleted, you probably don't want to pass on the deletion event to the chart and application levels.

To prevent further notification, use the Handled parameter and set it to TRUE. Put this line of script into the event procedure for the object. For example, to prevent further notificaiton of a deletion, put this line of script into the shape's OnDelete procedure:

```
Handled.Value=TRUE
```

To permit further notification of an event, set the Handled parameter to FALSE. This line is usually not necessary because the default value is FALSE. You would only need to do this if you had set the value to TRUE elsewhere.

```
Handled.Value=FALSE
```

{button Related Topics,PI(`',`IDH_RT_What_Are_ABC_Events')}

Registering Event Procedures

The purpose of registering events is to identify the event procedures that you want OLE Automation to execute. Registering an event does not determine whether the event executes when triggered, but only whether the procedure associated with the event is executed also.

OLE Automation requires that all event procedures except **AppMenuSUBCLASS** be registered before they can be executed. The **AppMenuSUBCLASS** event is registered automatically when an add-on menu is created with the **AddMenu** method of the Application object.

Normally, you register event procedures in the startup or initialization code of your program. If you want to "turn off" an event procedure that you have registered previously, you can unregister it. When your program ends, all events for all FlowCharter OCXs/VBXs on any form in your program are automatically unregistered.

Note

- If the program will be used with ABC FlowCharter 4.0, then you must unregister each registered event manually. The events will not be unregistered automatically when ABC FlowCharter 4.0 shuts down. (See below for information on the **UnregisterEvent** method.)

You use the **RegisterEvent** method of the Application object to register an event procedure. The general syntax for the **RegisterEvent** method if you have added the OCX/VBX control to your form is shown below. (See To install the FlowCharter VB event handler.)

ABC**.RegisterEvent** *VBXName*.VBX, *IdString*, "*EventName*" [,*ChartType*]

The *VBXName*.VBX parameter identifies the OLE Automation control to which the registered events apply. Unless you have changed the OLE Automation control's Name property from its default setting, *VBXName* is ABC1.

If you are using the OCX control, the general syntax is slightly different. (See To install the OCX event handler.) Note that there is no extension for the *OCXName*. Unless you have changed the OLE Automation control's Name property from its default setting, *OCXName* is also ABC1.

ABC**.RegisterEvent** *OCXName*, *IdString*, "*EventName*" [,*ChartType*]

The *IdString* parameter identifies the Visual Basic form on which the OLE Automation control is located. *IdString* is normally the **Caption** property setting of the form (Form1.Caption, by default).

The *EventName* parameter is the name of the event being registered. This name must be enclosed in quotes.

The *ChartType* parameter is optional. If *ChartType* is omitted, the registered events apply to all charts. If you wish to register the event for only a particular type of chart, then specify that chart **Type** for this parameter. You set a chart's type with the **Type** property of the Chart object.

Note

- OLE Automation does not permit two or more FlowCharter applications that are running at the same time to register the same events, unless the events are registered for different chart types. For information on this restriction, see Registering Events and Multiple FlowCharter Applications.

Some examples of registering events are shown below. The first example registers the **ChartNewNOTIFY** event for all charts in the application. The second example registers the **DeleteSUBCLASS** event for charts of the type Hourly.

```
ABC.RegisterEvent ABC1.VBX, Caption, "ChartNewNOTIFY"
ABC.RegisterEvent ABC1.VBX, Caption, "DeleteSUBCLASS", "Hourly"
```

You use the **UnRegisterEvent** method of the Application object to unregister an event. The general syntax for the **UnRegisterEvent** method is shown below.

ABC**.UnRegisterEvent** *VBXName*.VBX, "*EventName*" [, *ChartType*]

VBXName.VBX (or *OCXName* with no extension) is the name of the OLE Automation control, *EventName* is the name of the event being unregistered, and *ChartType* is the chart **Type**. *EventName* must be enclosed in quotes. *ChartType* is optional.

Some examples of unregistering events are shown below. The first example unregisters the **ChartNewNOTIFY** event for all charts in the application. The second example unregisters the **DeleteSUBCLASS** event for charts of the type Hourly.

```
ABC.UnRegisterEvent ABC1.VBX, "ChartNewNOTIFY"
ABC.UnRegisterEvent ABC1.VBX, "DeleteSUBCLASS", "Hourly"
```

Note: You do not need to register internal events (Chart object and Object object events).

Registering Events and Multiple FlowCharter Applications

AddMenu method

RegisterEvent method

UnRegisterEvent method

AppMenuSUBCLASS event

ChartNewNOTIFY event

DeleteSUBCLASS event

Caption property

Type property

Registering Events and Multiple FlowCharter Applications

OLE Automation imposes a restriction on the registration of events for multiple ABC applications. The same events cannot be registered by two or more ABC applications running at the same time, unless the events are registered for different chart types.

The **AppQuitNOTIFY**, **AppQuitSUBCLASS**, and **SpecialKeySUBCLASS** events are exempt from this limitation and can be registered by multiple ABC applications without restriction.

An example will make this restriction clear. Assume that you are running an ABC application that registers the **DeleteSUBCLASS** event using the statement shown below.

```
ABC.RegisterEvent ABC1, AutoParts, "DeleteSUBCLASS"
```

If you now attempt to run a second ABC application that registers the **DeleteSUBCLASS** event, you will be notified of the event conflict by ABC and asked to close the first application before the second application can run.

To avoid this kind of event conflict, design your programs so that they deal with specific chart types. This lets you to register events only for those chart types, thereby avoiding any event conflicts with other concurrently running ABC applications.

Note

- You do not need to register internal events (Chart object and Object object events).

{button Related Topics,PI(`',`IDH_RT_Registering_Events')}

Event Variables

OLE Automation passes information to its event procedures by setting various OLE Automation object variables. These object variables are local to the event procedure and are reset each time the event procedure is called. If you need to save the value of one of these OLE Automation object variables between executions of the event procedure, you must save the value of the LE Automation object variable in a global object variable.

Not all OLE Automation object variables apply to all events. The event definitions that appear in later sections of this chapter describe which OLE Automation object variables are valid for each event.

The OLE Automation variables passed to events are defined below.

Variable	**Definition**
App	The Application object that triggered this event.
Chart	The Chart object in which this event occurred.
Object	The Object object to which the event applies. This variable is set only if the event applies to a single Object.
Object2	The second object to which the event applies. This variable is only set using the **ObjectLineAttachNOTIFY** event.
FieldValue	The FieldValue object to which the event applies.
Menu	The Menu object to which this event applies.
MenuItem	The MenuItem object to which this event applies.
WParam	Set only for the **SpecialKeySUBCLASS** event. (This is a Long variable, not an Object variable.)
LParam	Reserved for future use.
Override	A property that lets you cancel normal FlowCharter behavior in response to an event. It is automatically reset to False at the end of every event call.
VBX	A property that is used for registering events and adding menus to let FlowCharter add a communication path to your program. (This property is not available if you are using the OCX custom control.)

The following example of an event procedure uses an OLE Automation object variable. It tests the **Type** property of the Chart object passed to the event procedure. If the **Type** property is PartTime, then the event procedure sets **Override** to True, which cancels the standard behavior for the event.

```
Sub ABC1_ChartMoveSUBCLASS ( )
    If ABC1.Chart.Type = PartTime Then
        ABC1.Override = True
    End If
End Sub
```

{button Related Topics,PI(`',`IDH_RT_Event_Variables')}

Override property
SpecialKeySUBCLASS event
Type property

When FlowCharter Closes

FlowCharter has two events relating to closing: **AppQuitSUBCLASS** and **AppQuitNOTIFY**.

The **AppQuitSUBCLASS** event occurs when a request is made to close FlowCharter. The user can request that FlowCharter close by a clicking Exit on the File menu of FlowCharter, pressing the keyboard shortcut **ALT+F4**, or double clicking the Control box of the FlowCharter window. The **AppQuitSUBCLASS** event procedure is triggered before FlowCharter closes. You can prevent FlowCharter from closing by setting the ABC1 **Override** property to True.

An **AppQuitSUBCLASS** example is shown below. This example tests the UpdateDone variable to determine if FlowCharter should be closed. If UpdateDone is False, then the procedure overrides the user's request to close FlowCharter and displays a message indicating that FlowCharter cannot be closed.

```
Sub ABC1_AppQuitSUBCLASS ( )
    If UpdateDone = False Then                  ' Test variable
        ABC1.Override = True                    ' Override close behavior
        Message = "Update incomplete. Cannot close FlowCharter."
        ABC.MsgBox Message                      ' Display message
    End If
End Sub
```

Note

- To do this in Living FlowChart script, remove ABC. from the MsgBox method, and change ABC1 to Application.

The **AppQuitNOTIFY** event occurs when FlowCharter is closed. The **AppQuitNOTIFY** event procedure can be used for final actions that you want your program to perform before it closes. If you want the Visual Basic application to close when FlowCharter does, put a Visual Basic End statement in this procedure.

An **AppQuitNOTIFY** example is shown below. This example calls the general procedure ShutDown to perform its final actions and then executes End to close the application.

```
Sub ABC1_AppQuitNOTIFY ( )
    ShutDown                                    ' Perform final actions
    End                                         ' Close application
End Sub
```

AppQuitNOTIFY event

AppQuitSUBCLASS event

Override property

When Add-On Menus Open

The **AppMenuPopupSUBCLASS** event occurs when the user opens an add-on menu by clicking the menu's name. Add-on menus are created with the **AddMenu** method of the Application object. The **AppMenuPopupSUBCLASS** event procedure is triggered before FlowCharter displays the add-on menu. The menu that is about to open is passed to the event procedure in the Menu object variable.

Because the **AppMenuPopupSUBCLASS** event is triggered before the add-on menu opens, you can use this event procedure to determine whether any items on the add-on menu should be disabled (gray) or checked. A menu item is disabled by setting the **Enabled** property of the MenuItem object to False. A menu item is checked by setting the **Checked** property of the MenuItem object to True.

An **AppMenuPopupSUBCLASS** example is shown below. This example assumes that other code has created an add-on menu with the items Shape Count and Line Count.

```
Sub ABC1_AppMenuPopupSUBCLASS ( )
    Dim ShapeCmd As Object                          ' Declare object variable
    Dim LineCmd As Object                           ' Declare object variable
    If CurrentShapeCount = 0 Then
        Set ShapeCmd = ABC1.Menu.Item("Shape Count")  ' Get MenuItem object
        ShapeCmd.Enabled = False                    ' Gray item
    End If
    If CurrentLineCount = 0 Then
        Set LineCmd = ABC1.Menu.Item("Line Count")    ' Get MenuItem object
        LineCmd.Enabled = False                     ' Gray item
    End If
End Sub
```

Note

- To do this in Living FlowChart script, remove ABC. from the MsgBox method, and change ABC1 to Application.

The procedure tests the status of the CurrentShapeCount and CurrentLineCount variables to determine whether the Shape Count and Line Count menu items should be disabled. The Shape Count menu item is disabled when the CurrentShapeCount is zero. The Line Count menu item is disabled when the CurrentLineCount is zero.

AddMenu method

AppMenuPopupSUBCLASS event

Checked property

Enabled property

When MenuItems Are Highlighted

The **AppMenuHintSUBCLASS** event occurs when the user moves the menu cursor to an item on an add-on menu. The **AppMenuHintSUBCLASS** event procedure is triggered before FlowCharter highlights the menu item. The menu item to be highlighted is passed to the event procedure in the MenuItem object variable.

An **AppMenuHintSUBCLASS** example is shown below. This example illustrates how to use the **AppMenuHintSUBCLASS** event procedure to display hint line messages describing the purpose of items on an add-on menu.

```
Sub ABC1_AppMenuHintSUBCLASS ( )
    If ABC1.MenuItem = "Shape Count" Then
        ABC1.App.Hint("Click to display shape count")          ' Show hint
    End If
    If ABC1.MenuItem = "Line Count" Then
        ABC1.App.Hint("Click to display line count")          ' Show hint
    End If
End Sub
```

This example determines which menu item is to be highlighted and displays the appropriate hint line message.

AppMenuHintSUBCLASS event

When MenuItems Are Chosen

The **AppMenuSUBCLASS** event occurs when the user chooses an item on an add-on menu. The menu item object that was chosen is passed to the event procedure in the MenuItem variable.

Note

- Do not register the **AppMenuSUBCLASS** event. The **AppMenuSUBCLASS** event is registered automatically when an add-on menu is created.

An **AppMenuSUBCLASS** example is shown below. This example determines which menu item is chosen and executes the general procedure that performs the function of the item. Because it may take some time to count the shapes or lines, the procedure displays the hourglass cursor using the **Hourglass** property of Application object before it performs the counting operation. When the counting is complete, it clears the hourglass.

```
Sub ABC1_AppMenuSUBCLASS ( )
    ABC.Hourglass = True                    ' Display hourglass
    If ABC1.MenuItem = "Shape Count" Then
        X = Shapes.Count
    End If
    If ABC1.MenuItem = "Line Count" Then
        Y = Object.Count
    End If
    ABC.Hourglass = False                   ' Clear hourglass
End Sub
```

Note

- To do this in Living FlowChart script, change ABC. to Application.

AppMenuSUBCLASS event

Hourglass property

When Charts Open

The **ChartOpenNOTIFY** event occurs when the user opens a new chart file by clicking Open on the File menu of FlowCharter. The **ChartOpenNOTIFY** event procedure is triggered following the successful opening of the chart file. The opened chart object is passed to the event procedure in the Chart object variable.

A **ChartOpenNOTIFY** example is shown below. This example stores the path and the page count of the newly opened chart in the global variables CurrentPath and CurrentPages.

```
Sub ABC1_ChartOpenNOTIFY ( )
    CurrentPath = ABC1.Chart.FullName          ' Save path of chart
    CurrentPages = ABC1.Chart.PageCount        ' Save chart page count
End Sub
```

Note

- To do this in Living FlowChart script:

```
    CurrentPath = Chart.FullName               ' Save path of chart
    CurrentPages = Chart.PageCount             ' Save chart page count
```

{button Related Topics,PI(`',`IDH_RT_When_Charts_Open')}

ChartOpenNOTIFY event

When Linked Charts Open

The **LinkNOTIFY** event occurs when a chart file is opened by double-clicking the object to which it is linked. The **LinkNOTIFY** event procedure is triggered following the successful opening of the chart file.

The chart object from which the linked chart was opened (the source chart) is passed to the event procedure in the Chart object variable. The linked chart object (the chart just opened) can be obtained using the **ActiveChart** property of the Application object. The Object that was double-clicked in the source chart to open the linked chart is passed to the event procedure in the Object object variable.

The **LinkNOTIFY** example below saves the source chart, source object, and linked chart in the global object variables SourceChart, SourceObject, and CurrentChart.

```
Sub ABC1_LinkNOTIFY ( )
    SourceChart = ABC1.Chart              ' Save source chart
    SourceObject = ABC1.Object            ' Save source object
    CurrentChart = ABC1.ActiveChart       ' Save linked chart
End Sub
```

Note

- To do this in Living FlowChart script, remove ABC1. from the script.

{button Related Topics,PI(`',`IDH_RT_When_Linked_Charts_Open')}

ActiveChart property

LinkNOTIFY event

When New Charts Are Created

The **ChartNewNOTIFY** event occurs when the user creates a new chart by clicking New on the File menu of FlowCharter. The **ChartNewNOTIFY** event procedure is triggered following the creation of the new chart. The new chart object is passed to the event procedure in the Chart object variable.

A **ChartNewNOTIFY** example is shown below. It enlarges the new chart to full size.

```
Sub ABC1_ChartNewNOTIFY ( )
    ABC1.Chart.Maximize
End Sub
```

Note

- To do this in Living FlowChart script, remove ABC1. from the script.

{button Related Topics,PI(`',`IDH_RT_When_New_Charts_Are_Created')}

ChartNewNOTIFY event

When Charts Are Activated

The **ChartActivateNOTIFY** event occurs when a chart is activated by clicking it or choosing it from a menu. The **ChartActivateNOTIFY** event procedure is triggered following the activation of the chart. The activated chart object is passed to the event procedure in the Chart object variable. The **ChartDeActivateNOTIFY** event occurs when a different chart is activated. (It does not occur when a chart closes.)

The **ChartActivateNOTIFY** example shown below tests the **Type** property of the activated chart and executes the general procedure DeployEditMode if it is a CHARTTYPE type. If the activated chart is not a CHARTTYPE type, then its **Caption** (title) property is set to the standard caption "Micrografx FlowCharter 7."

```
Sub ABC1_ChartActivateNOTIFY ( )
    If ABC1.Chart.Type = CHARTTYPE Then
        Call DeployEditMode
    Else
        ABC1.App.Caption = ""                    ' Set to standard caption
    End If
End Sub
```

Note

- To do this in Living FlowChart script, remove ABC1. from the script.

```
    If Chart.Type = CHARTTYPE Then
        MsgBox "This chart is a charttype type."
    Else
        Application.Caption = ""                 ' Set to standard caption
    End If
```

Caption property

ChartActivateNOTIFY event

ChartDeActivateNOTIFY Event

Type property

When Charts Change

The **ChartChangeNOTIFY** event occurs when a chart is changed in any way. The **ChartChangeNOTIFY** event procedure is triggered following the changing of the chart. The changed chart object is passed to the event procedure in the Chart object variable.

The **ChartChangeNOTIFY** example shown below sets the chart's Type property to MODIFIED to indicate that it has been changed.

```
Sub ABC1_ChartChangeNOTIFY ( )
    ABC1.Chart.Type = MODIFIED
End Sub
```

ChartChangeNOTIFY event

When Charts Are Saved

The **ChartSavedNOTIFY** event occurs when a user saves a chart by using the Save command on the File menu of FlowCharter. The **ChartSavedNOTIFY** event procedure is triggered after the save. The changed chart object is passed to the event procedure in the Chart object variable.

{button Related Topics,PI(`',`IDH_RT_When_Charts_Are_Saved')}

ChartSavedNOTIFY event

When Charts Are Pasted

The **ChartPasteNOTIFY** event occurs when a user pastes something into a chart by pressing the keyboard shortcut **CTRL+V** or clicking Paste on the Edit menu of FlowCharter. The **ChartPasteNOTIFY** event procedure is triggered following the paste. The chart object is passed to the event procedure in the Chart object variable.

A **ChartPasteNOTIFY** example is shown below. After a paste operation, the objects pasted into a chart are the only selected objects. This example uses this feature to color the pasted objects blue.

```
Sub ABC1_ChartPasteNOTIFY ( )
    Dim Obj As Object, Objs As Object          ' Declare variables
    ABC.Hourglass = True                       ' Display hourglass

    Set Objs = ABC1.Chart.Objects
    Do
        Set Obj = Objs.ItemFromSelection       ' Get selected object
        Obj.Color = ABC.BLUE                   ' Color it blue
    Loop While Obj                             ' Loop until done

    ABC.Hourglass = False                      ' Clear hourglass
End Sub
```

Note

- To do this in Living FlowChart script:

```
    Application.Hourglass = True               ' Display hourglass

    Set Objs = Chart.Objects
    Do
        Set Obj = Objs.ItemFromSelection       ' Get selected object
        Obj.Color = BLUE                       ' Color it blue
    Loop While Obj                             ' Loop until done

    Application.Hourglass = False              ' Clear hourglass
```

{button Related Topics,PI(`',`IDH_RT_When_Charts_Are_Pasted')}

ChartPasteNOTIFY event

When Charts Close

The **ChartCloseSUBCLASS** event occurs when the user closes a chart by clicking Close on the File menu of FlowCharter. The **ChartCloseSUBCLASS** event procedure is triggered immediately before the user is prompted to save changes and the chart is closed. The closing chart object is passed to the event procedure in the Chart object variable.

The **ChartCloseSUBCLASS** example below calls a general procedure to update an external database when a chart is closed.

```
Sub ABC1_ChartCloseSUBCLASS ( )
    UpdateDatabase
End Sub
```

ChartCloseSUBCLASS event

When Objects Are Clicked

The **ObjectClickSUBCLASS** event occurs when the user clicks an object. The **ObjectClickSUBCLASS** event procedure is triggered before FlowCharter shows the Object as selected.

The clicked Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An **ObjectClickSUBCLASS** example is shown below. This example tests the **Type** property of the active chart and sets the clicked object to Green if it is a PartTime type and to Yellow otherwise.

```
Sub ABC1_ObjectClickSUBCLASS ( )
    If ABC1.Chart.Type = PartTime Then
        ABC1.Object.Color = ABC1.App.GREEN
    Else
        ABC1.Object.Color = ABC1.App.YELLOW
    End If
End Sub
```

To do this in Living FlowChart script:

```
If Chart.Type = PartTime Then
        Color = Application.GREEN
    Else
        Color = Application.YELLOW
    End If
```

ObjectClickSUBCLASS event

Type property

When Objects Are Selected

The **ExclusiveSelectionNOTIFY** event occurs when the user selects a single Object object. The **ExclusiveSelectionNOTIFY** event procedure is triggered after FlowCharter shows the Object as selected.

The selected Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An **ExclusiveSelectionNOTIFY** example is shown below. It copies any single object selected to the Clipboard.

```
Sub ABC1_ExclusiveSelectionNOTIFY ( )
    If ABC1.Chart.Copy
End Sub
```

{button Related Topics,PI(`',`IDH_RT_When_Objects_Are_Selected')}

ExclusiveSelectionNOTIFY event

When Objects Move

OLE Automation has two events relating to moving objects: **ObjectMoveSUBCLASS** and **ObjectMovedNOTIFY**.

The **ObjectMoveSUBCLASS** event occurs when the user starts to move an Object object. The **ObjectMoveSUBCLASS** event procedure is triggered before FlowCharter initiates any move behavior.

The Object about to move is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An **ObjectMoveSUBCLASS** example is shown below. This example saves the left and top locations of the object before it is moved in the global variables GLeft and GTop.

```
Sub ABC1_ObjectMoveSUBCLASS ( )
    GLeft = ABC1.Object.Left            ' Save left edge
    GTop = ABC1.Object.Top              ' Save top edge
End Sub

GLeft = Left                            ' Save left edge
GTop = Top                              ' Save top edge
```

The **ObjectMovedNOTIFY** event occurs when an Object object is moved. The **ObjectMovedNOTIFY** event procedure is triggered after FlowCharter has moved the Object.

The Object that was moved is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An **ObjectMovedNOTIFY** example is shown below. This example uses the **Type** property of the moved Object to decide its action. For a PHASE type, the procedure sets the top edge of the moved Object to GTop. For a DEPT type, the procedure sets the left edge of the moved Object to GLeft.

```
Sub ABC1_ObjectMovedNOTIFY ( )
    If ABC1.Object.Type = PHASE Then
        ABC1.Object.Top = GTop
    End If
    If ABC1.Object.Type = DEPT Then
        ABC1.Object.Left = GLeft
    End If
End Sub
```

ObjectMoveSUBCLASS event

ObjectMovedNOTIFY event

Type property

When Objects Are Resized

OLE Automation has two events relating to resizing objects: **ObjectSizeSUBCLASS** and **ObjectSizedNOTIFY**.

The **ObjectSizeSUBCLASS** event occurs when the user starts to resize an Object object. The **ObjectSizeSUBCLASS** event procedure is triggered before FlowCharter initiates any resizing behavior.

The Object to be resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

The **ObjectSizeSUBCLASS** example shown below saves the height and width of the object before it is resized in the global variables GHeight and GWidth.

```
Sub ABC1_ObjectSizeSUBCLASS ( )
    GHeight = ABC1.Object.Height        ' Save object height
    GWidth = ABC1.Object.Width          ' Save object width
End Sub
```

The **ObjectSizedNOTIFY** event occurs when an Object object is resized. The **ObjectSizedNOTIFY** event procedure is triggered after FlowCharter has resized the Object.

The Object that was resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

The **ObjectSizedNOTIFY** example below tests the height and width of the resized object and resets these properties to GHeight and GWidth if they are less than those values.

```
Sub ABC1_ObjectSizedNOTIFY ( )
    If ABC1.Object.Height < GHeight Then
        ABC1.Object.Height = GHeight
    End If
    If ABC1.Object.Width < GWidth Then
        ABC1.Object.Width = GWidth
    End If
End Sub
```

ObjectSizeSUBCLASS event

ObjectSizedNOTIFY event

When Objects Are Deleted

The **DeleteSUBCLASS** event occurs when one or more Objects are deleted. The user deletes Objects by selecting the Objects, and then pressing **DEL** or clicking Clear on the Edit menu. The **DeleteSUBCLASS** event procedure is triggered before FlowCharter performs the deletion.

The Object to be deleted first is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable. You can use the **SelectedObjectCount** property of the Chart object to find the number of Objects selected for deletion. You can use the **ItemFromSelection** method of the Objects collection in a loop to access the Objects to be deleted.

A **DeleteSUBCLASS** example is shown below. This example tests all of the Objects selected for deletion and cancels the selection (and deletion) of any Objects of the DEPT type.

```
Sub ABC1_DeleteSUBCLASS ( )
    Dim Obj As Object                       ' Declare variable

    Do
        Set Obj = Objects.ItemFromSelection ' Get object to delete
        If Obj.Type = DEPT Then             ' If DEPT type, then
            Obj.Selected = False            ' cancel selection
        End If
    Loop While Obj.Valid                    ' Loop until done
End Sub
```

{button Related Topics,PI(`',`IDH_RT_When_Objects_Are_Deleted')}

DeleteSUBCLASS event

ItemFromSelection method

SelectedObjectCount property

When Shapes Are Double Clicked

The **DoubleClickSUBCLASS** event occurs when the user double clicks a Shape object. The **DoubleClickSUBCLASS** event procedure is triggered before FlowCharter shows the Shape as selected.

The clicked Shape is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

The **DoubleClickSUBCLASS** example below beeps to indicate that a shape was double clicked and sets the **Override** property to cancel FlowCharter's standard behavior for the double click event.

```
Sub ABC1_DoubleClickSUBCLASS ( )
    Beep
    ABC1.Override = True
End Sub
```

DoubleClickSUBCLASS event

When Shapes Are Drawn

The **NewShapeNOTIFY** event occurs when the user draws a new Shape object. The **NewShapeNOTIFY** event procedure is triggered after FlowCharter draws the Shape.

The newly drawn Shape is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

The **NewShapeNOTIFY** example below sets the color of the newly drawn shape to yellow.

```
Sub ABC1_NewShapeNOTIFY ( )
    ABC1.Object.FillColor = ABC1.App.YELLOW                    ' Set color
End Sub
```

NewShapeNOTIFY event

When Shapes Are Replaced

The **ReplaceShapeNOTIFY** event occurs when the user replaces one or more Shape objects. The **ReplaceShapeNOTIFY** event procedure is triggered after FlowCharter replaces the Shape objects.

The Shape to be replaced first is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable. You can use the **ItemFromShapes** method of the Objects collection in a loop to access the Shapes to be replaced.

The **ReplaceShapeNOTIFY** example below counts the number of Decision shapes in a chart and reports that count to the user.

```
Sub ABC1_ReplaceShapeNOTIFY ( )
    Dim Obj As Object, Objs As Object                ' Declare variables

    ABC.Hourglass = True                             ' Display hourglass
    Counter = 0                                      ' Initialize count

    Set Objs = ABC1.Chart.Objects
    Do
        Set Obj = Objs.ItemFromShapes
        If Obj.Shape.ShapeName = "Decision" Then     ' If Decision...
            Counter = Counter + 1                    ' ...bump counter
    Loop While Obj                                   ' Loop until done

    ABC.Hourglass = False                            ' Clear hourglass
    ABC.MsgBox "Decision Shape Count = " + Counter   ' Show results
End Sub
```

{button Related Topics,PI(`',`IDH_RT_When_Shapes_Are_Replaced')}

ItemFromShapes method

ReplaceShapeNOTIFY event

When Lines Are Drawn

The **NewLineNOTIFY** event occurs when the user draws a new Line object. The **NewLineNOTIFY** event procedure is triggered after FlowCharter draws the Line.

The object to which the newly drawn Line is attached is passed to the event procedure in the Object variable, and the chart in which the Line is located is passed in the Chart variable.

The **NewLineNOTIFY** example below sets the color of the object to which the line was just attached to green.

```
Sub ABC1_NewLineNOTIFY ( )
    ABC1.Object.Color = ABC1.App.GREEN          ' Set color
End Sub
```

NewLineNOTIFY event

When Lines Attach

The **ObjectLineAttachNOTIFY** event occurs when the user attaches a line to an Object. The **ObjectLineAttachNOTIFY** event procedure is triggered after FlowCharter attaches the Line. When a line is detached, the **ObjectLineDeAttachNOTIFY** event procedure is triggered

The Object to which the line is attached is passed to the event procedure in the Object variable, the line is passed in the Object2 variable, and the chart in which the Object is located is passed in the Chart variable.

An **ObjectLineAttachNOTIFY** example is shown below.

```
Sub ABC1_ObjectLineAttachNOTIFY ( )
    ABC1.Object.Color = ABC1.App.GREEN          ' Set color
End Sub
```

{button Related Topics,PI(`',`IDH_RT_When_Lines_Attach')}

ObjectLineAttachNOTIFY event

ObjectLineDeAttachNOTIFY event

When Fonts Change

The **ObjectFontChangeNOTIFY** event occurs when the user changes the font of one or more Text objects. The **ObjectFontChangeNOTIFY** event procedure is triggered after FlowCharter displays the Text objects in the changed font.

The Text object that was changed first is passed to the event procedure in the Object variable, and the chart in which the text is located is passed in the Chart variable. You can use the **ItemFromSelection** method of the Objects collection in a loop to access the changed Text objects.

The **ObjectFontChangeNOTIFY** example below searches for all TextBlock objects (a Type 2 Text object is a TextBlock object) and resizes them to the same font size when a TextBlock's font size changes.

```
Sub ABC1_ObjectFontChangeNOTIFY ( )
    Dim Text As Object, Objs As Object              ' Declare variables

    ABC.Hourglass = True                            ' Display hourglass

    Set Objs = ABC1.Chart.Objects

    Do
        Set Text = Objs.ItemFromSelection           ' Get selected object
        If Text.Type = 2 Then Exit Do               ' Exit if Type 2
    Loop While Text                                 ' Loop until done

    If Text Then   ' If TextBlock found...
        Objs.ResetSearch                            ' ...reset to first object
        Size = Text.Font.Size                       ' Get font size
        Do
            If Text.Type = 2 Then                   ' If TextBlock object
                Text.Font.Size = Size               ' Set font size
            End If
        Loop While Text                             ' Loop until done

    ABC.Hourglass = False                           ' Clear hourglass
End Sub
```

{button Related Topics,PI(`',`IDH_RT_When_Fonts_Change')}

ItemFromSelection method

ObjectFontChangeNOTIFY event

When Field Values Change

The **FieldValueChangedNOTIFY** event occurs when the user changes a FieldValue object. The **FieldValueChangedNOTIFY** event procedure is triggered after FlowCharter changes the FieldValue.

The FieldValue that was changed is passed to the event procedure in the FieldValue variable, the Object that owns the field is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

A **FieldValueChangedNOTIFY** example is shown below. The example tests the new value of the FieldValue object to ensure that it is between 0 and 1000. If the field value is outside of this range, the invalid value is cleared and the user is instructed to enter a value in the valid range.

```
Sub ABC1_FieldValueChangedNOTIFY ( )
    ChangedObject = ABC1.Object
    Message = "Data not in range. Please enter a number between 0 and 1000."
    If ABC1.FieldValue < 0 or > 1000 Then
        ABC1.FieldValue.Empty                    ' Clear field value
        ABC1.App.MsgBox Message                  ' Display instructions
    End If
End Sub
```

FieldValueChangedNOTIFY event

When Special Keys Are Pressed

The **SpecialKeySUBCLASS** event occurs when the user presses one of the special keys. The **SpecialKeySUBCLASS** event procedure is triggered before FlowCharter responds to the key press.

A code representing the key is passed to the event procedure in the **WParam** variable. These codes are defined in the table below.

Key	Code	Key	Code
F1	1	**TAB**	13
F2	2	**ESC**	27
F3	3	**PGUP**	33
F4	4	**PGDN**	34
F5	5	**END**	35
F6	6	**HOME**	36
F7	7	**LEFTARROW**	37
F8	8	**UP ARROW**	38
F9	9	**RIGHTAARROW**	39
F10	10	**DOWN ARROW**	40
F11	11	**INS**	45
F12	12	**DEL**	46

A **SpecialKeySUBCLASS** example is shown below. It checks for **F11** and **F12**. If **F11** is found, FlowCharter is maximized. If **F12** is found, FlowCharter is minimized.

```
Private Sub ABC1_SpecialKeySUBCLASS(ByVal KeyCode As Integer, Override As Boolean)
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.application")       ' Start ABC
   If ABC1.WParam = 11 Then                         ' If F11...
        ABC.Maximize                                ' ...maximize ABC and...
        ABC1.Override = True                        ' ...override standard behavior
   End If
   If ABC1.WParam = 12 Then                         ' If F12...
        ABC.Minimize                                ' ...minimize ABC and...
        ABC1.Override = True                        ' ...override standard behavior
   End If
End Sub
```

{button Related Topics,PI(`',`IDH_RT_When_Special_Keys_Are_Pressed')}

SpecialKeySUBCLASS event

When Text Changes

The **ObjectTextChangedNOTIFY** event occurs when the user changes a TextBlock object. The **ObjectTextChangedNOTIFY** event procedure is triggered after FlowCharter changes the TextBlock.

The Object that owns the text is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An **ObjectTextChangedNOTIFY** event example is shown below. The example displays the text that was changed.

```
Sub ABC1_ObjectTextChangedNOTIFY ( )
    Dim ChangedObject As Object
    Set ChangedObject = ABC1.Object
    ABC1.App.MsgBox "New Text: " + ChangedObject.Text          ' Display the new text
End Sub
```

ObjectTextChangedNOTIFY event

Importing Files in Other Formats

In FlowCharter, you use the Open command on the File menu to import charts with other formats into FlowCharter charts. This is not a common operation. To support it, there would have to be OLE Automation commands for all the dialog boxes for naming the subcharts, choosing the chart font, and so forth.

If you need to import charts with other formats, you should have your user perform the command in FlowCharter rather than using OLE Automation.

Printing to a File

In FlowCharter you print to a file by choosing the Print command on the File menu. Then, in the Print dialog box, you choose the Print to File option before you print. Printing to a file can be useful in certain instances, but often you can print directly to a printer. Printing to a file may be supported in a later version of OLE Automation.

With some printers, you can print to a file by setting an option in the driver, usually in the driver's Options dialog box. You always can print to a file by using the Windows Control Panel to set the driver to the port FILE:. If you need to print to a file and cannot do it using the printer driver, you should have your user perform the action in FlowCharter rather than using OLE Automation.

Saving a Workspace

You usually use the Save Workspace command on the File menu when you have a standard method of working. With OLE Automation, you can set up a standard appearance using the commands to open files and position their windows.

Special Shape Properties

OLE Automation does not let you set connect points, set the rectangle that contains text, or specify label alignment. These abilities are often used to make corrections due to the way that objects were originally created. With OLE Automation, there is less likely a need for these changes.

If you have a particular need that is outside what FlowCharter normally does, you can set up a template and use it as the basis for a chart using **AddFromTemplate**. Alternatively, you can have your user perform the commands in FlowCharter rather than using OLE Automation.

Editing Shapes in the Shape Palette

In FlowCharter, you can choose the ABC Media Manager to perform activities such as arranging the shapes in a palette, pasting new shapes into a palette, or deleting shapes from a palette. None of these activities are available in OLE Automation. You also cannot alter the shape properties for a particular shape in the palette. If customized palettes are required, use FlowCharter to set up the palettes and then reference the custom palette.

Setting Preferences for the Shape Palette

In FlowCharter, you can use the ABC Media Manager to perform activities such as setting the palette title bar or button face size. These activities are not available in OLE Automation. If customized palettes are required, use FlowCharter to set up the palettes and then reference the custom palette.

Printing Notes Without Shapes

In FlowCharter you can print notes directly from the Note window by choosing the Print command from the Note menu. OLE Automation does not allow printing only notes. To print notes, you must print both notes and their associated shapes (the equivalent of using the Print command in the File menu of the main window).

If you need to print notes without shapes, you should have your user perform the command in FlowCharter rather than using OLE Automation.

Moving Guidelines

Guidelines are a powerful feature of FlowCharter. With OLE Automation, you can create guidelines using the **AddVerticalGuideline** and **AddVerticalGuideline** methods. You can choose whether guidelines display in the chart using the **GuidelinesOn** property. You can delete all guidelines using the **ClearGuidelines** method. These facilities are described in Using Guidelines.

After you create guidelines, you cannot act on them except to delete all the guidelines in a chart. If you need to change the placement of guidelines, you should delete them all and recreate the ones you want in the new positions you need.

{button Related Topics,PI(`',`IDH_RT_ing_Guidelines')}

Using Guidelines

AddHorizontalGuideline method
AddVerticalGuideline method
GuidelinesOn property
ClearGuidelines method

Opening ABC 1.x Files

You cannot open ABC 1.x files using OLE Automation.

If necessary, you can have your user open the files in FlowCharter rather than using OLE Automation.

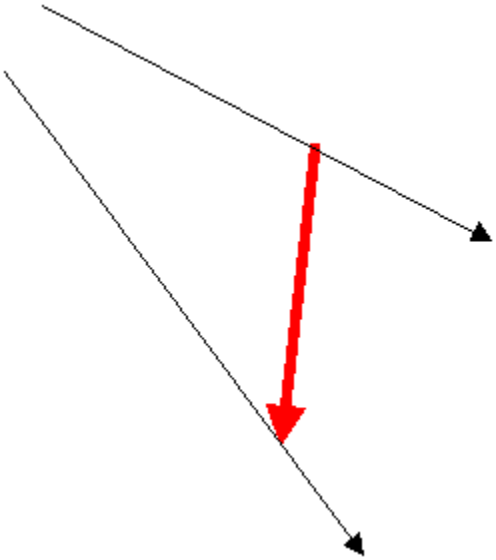
Using Mixed Fonts

You can specify the font for a particular object. There is no way to specify different fonts within a particular piece of text, however.

If necessary, you can have your user specify fonts for a particular word or phrase rather than using OLE Automation.

Drawing Certain Lines

It is not possible to draw certain lines using OLE Automation because there is no way to specify the start and finish positions of the line. For example, in the following illustration you cannot specify how to draw the red line.



If necessary, you can have your user draw the line rather than using OLE Automation.

OLE Object

Description

The OLE object is below the Object object. Note that this is an OLE client object from another application, not an OLE Automation object. You can have only one OLE object for each Object object.

Properties	Methods
Application	DoVerb
ObjectType	
Parent	

{button Related Topics,PI(`',`IDH_RT_OLE_Object')}

ObjectType Property

Usage	*OLEObject*.**ObjectType**
Description	The **ObjectType** property lets you find the short object class name of an object that is embedded or linked. The **ObjectType** property is read only.
Data Type	String
Value	The short object class name of an object that is embedded or linked. The name depends on the OLE server. For example, the name for bitmaps might be "Paintbrush Picture" and the name for Word for Windows is "Document."
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_ObjectType_Property')}

Using OLE Client Objects
Example

DoVerb Method
InsertObjectFromFile Method
PasteLink Method
UpdateFields Method

OLE Property

OLE Object

ObjectType, Objects Properties Example

This example uses the **ObjectType** property of the OLE object and the **Objects** property of the Chart object to find the type of an OLE object.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Everything = Chart.Objects                         ' Get all items in the chart
Do
    Set Current = Everything.ItemFromAll               ' Choose the next item
                                                       ' Display the OLE type
    ABC.MsgBox "Current object's OLE type is """ + Current.OLE.ObjectType + """"
Loop While Current.Valid And Current.Type <> 5         ' Skip over master items
```

DoVerb Method

Usage *OLEObject*.**DoVerb** ([*Verb*])

Description You use the **DoVerb** method to specify an OLE verb to execute if the object is a linked or embedded OLE object. If you do not specify a verb, the default verb is used.

Flow Equivalent None.

{button Related Topics,PI(`',`IDH_RT_DoVerb_Method')}

Using OLE Client Objects
Example

InsertObjectFromFile Method
PasteLink Method
UpdateFields Method

ObjectType Property
OLE Property

OLE Object

Language Reference

In the descriptions of the OLE Automation procedures, methods, and events, each element has the following entries, as appropriate.

{button Flow Equivalent,} Steps in the FlowCharter application that are equivalent to using the property, method, or event. Click the Flow Equivalent button at the right of the topic title to jump to the FlowCharter application help topic that describes the equivalent.

{button Related Topics,} Click the button at the end of the title to see elements that are related to the element, topics that describe the element, and objects that contain the element. You can click the jump terms to go to the related topics.

Example A concise programming use of the element is provided in the Example topic, which is the last item on the Related Topics list. Click the Example topic and the example appears in a separate window that is always on top of all other windows. Click the X button in the window bar to close the window. To copy all or part of the example, select the text and press **CTRL-C** to put the selected text on the Windows Clipboard so you can paste it into your program.

Note

You should maximize the example window before you copy to the Clipboard to avoid unexpected wraps in longer statements. Some long lines may still wrap, so be sure to unwrap them in your program.

Usage The syntax for using the element in a program. You must replace items in *italic* with the appropriate variable from a Dim statement.

Description How you might use the element.

Data Type The type of the element (Object, Collection, Event, String, Integer, Double, Long, Variant).

Value The values that you can set the element to or that the element returns.

{button Related Topics,PI(`',`IDH_RT_Reference_Chapter')}

Properties

A

Accumulation
AccumulationMethod
ActiveChart
AlignToRulers
Application

AttachedToLine

B

Bold
BorderColor
BorderStyle
BorderWidth
Bottom (Object object)
Bottom (Application object)

C

Caption

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Z

Methods

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

RemoveAddOn
RemoveMenu
Renumber
Repaint

ReplaceShape
ReplaceText
Restore (Application object)
Restore (Chart object)
RestorePicture (Object Object)

RevertToSaved

S

Save
ScrollPage
ScrollPosition
Select
SelectShapeType

SendMail
SetDefaults
SetProtection
ShowAll
SpaceEvenly Method

Spelling

T

TileCharts
ToBack (Object object)
ToBack (Chart object)
ToFront (Object object)
ToFront (Chart object)

U

UnattachFromLine
Undo
UnRegisterEvent
UpdateDateAndTime
UpdateFields

Objects, Alphabetical

Application
Chart
Charts collection
FieldTemplate
FieldTemplates collection

FieldValue
FieldValues collection
Font
Line_
MasterItems

Menu collection
MenuItem
Object
Objects collection
OLE

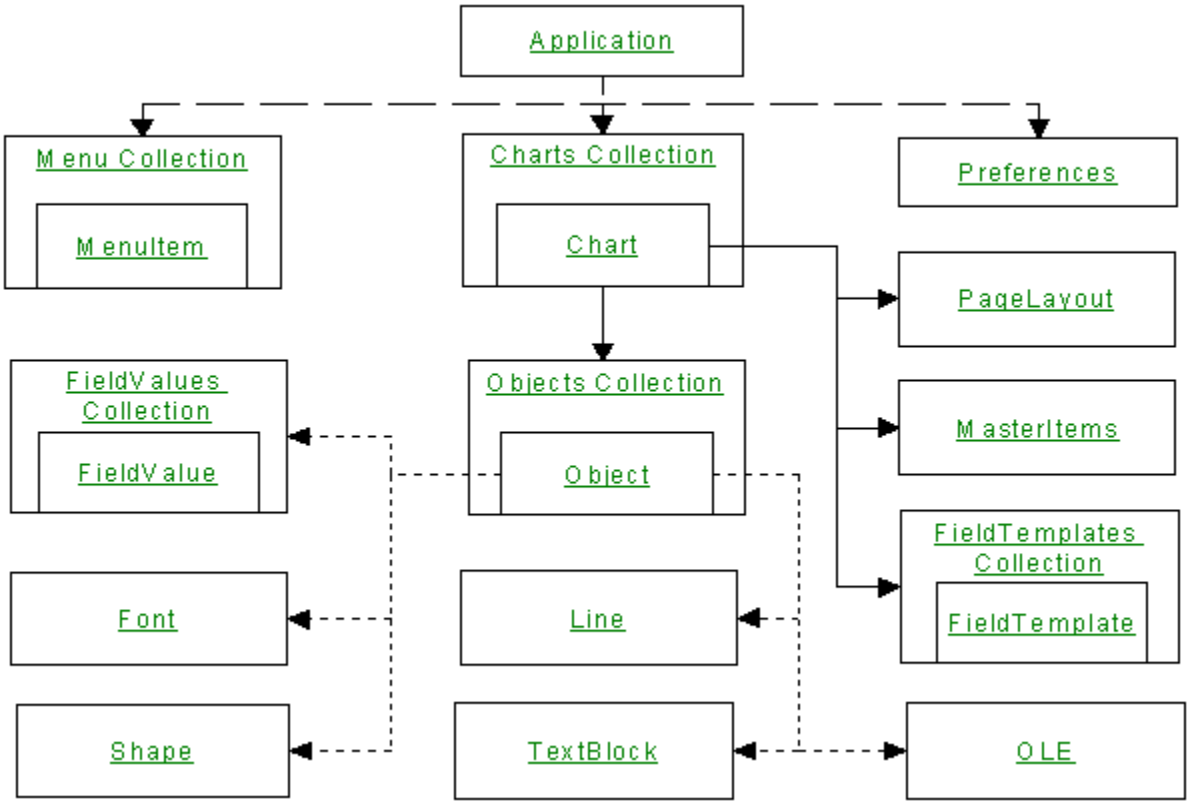
PageLayout
Preferences
Shape
TextBlock

{button Related Topics,PI(``,`IDH_RT_Objects_Alphabetical')}

Objects, graphical

Objects, Graphical

The OLE Automation object hierarchy, diagrammed below, shows how the objects and collections available in OLE Automation relate to each other. The diagram includes the collections, which can contain more than one object. You can click on an object or collection to see the properties and methods contained in it and get information about those language elements.



At the top of the OLE Automation hierarchy is the Application object, which is the interface to FlowCharter. There can be only one FlowCharter Application object at a time running on your system.

Branching from the Application object are the Charts collection, Preferences object, and Menus collection. An FlowCharter Application can have multiple Charts collections and Menus collections, but only one Preferences object.

Branching from the Charts collection are the Chart objects. Each Chart object is restricted to a single PageLayout and MasterItems object, but can have multiple FieldTemplate and Object objects.

Below the Object object are the Shape, Line_, TextBlock, OLE, and Font objects, and the FieldValues collection. Each Object object can have multiple FieldValue objects, but only one Shape, Line_, TextBlock, OLE, and Font object.

{button Related Topics,PI(`',`IDH_RT_Objects_Graphical')}

Events

A

AppQuitSUBCLASS
AppMenuSUBCLASS
AppMenuHintSUBCLASS
AppMenuPopupSUBCLASS

C

ChartActivateNOTIFY
ChartDeActivateNOTIFY Event
ChartChangeNOTIFY
ChartCloseSUBCLASS
ChartNewNOTIFY

ChartOpenNOTIFY
ChartPasteNOTIFY

D

DeleteSUBCLASS
DoubleClickSUBCLASS

E

ExclusiveSelectionNOTIFY

F

FieldValueChangedNOTIFY

L

LinkNOTIFY

N

NewLineNOTIFY
NewShapeNOTIFY

O

ObjectClickSUBCLASS
ObjectFontChangeNOTIFY
ObjectLineAttachNOTIFY
ObjectLineDeAttachNOTIFY Event
ObjectMovedNOTIFY

ObjectMoveSUBCLASS
ObjectSizedNOTIFY
ObjectSizeSUBCLASS
ObjectTextChangedNOTIFY

R

ReplaceShapeNOTIFY

S

SpecialKeySUBCLASS

All Properties, Methods, Objects, and Events

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Z

FlowCharter Menu Command Equivalents

This topic displays the menu commands available in FlowCharter. When using an OLE Automation Tool element, it is equivalent to executing the related command.

File Menu Command	OLE Automation Equivalent
New	New method
Open	Open method
Close	CloseChart method
Close All	
Save	Save method
Save As	Save method
Save Workspace	
Import	ImportShape Method
Export	Export Method
Page Setup	PageLayout object
Print Preview	PrintPreview Method
Print	PrintOut method
	PrintSelected method
Print Setup	Printer Property
Send	SendMail method
Recent File	
Exit	Quit method

Edit Menu Command	OLE Automation Equivalent
Undo	Undo method
Redo	
Cut	Cut method
Copy	Copy method
Paste	Paste method
Paste Special	PasteSpecial method
Clear	Clear method
Duplicate	Duplicate Method (Object Object)
Select All	Select method
Select	Select method
Find	
Replace	ReplaceText Method
Links	
Objects	

View Menu Command	OLE Automation Equivalent
Normal	View Property
Page	View Property
Full Screen	FullScreen Method
Toolbars	
Status Bar	StatusBarVisible Property
Rulers	ShowRulers Property
Shape Palette	ShapePaletteVisible Property
Worksheet	
Entity Manager	
Field Viewer	FieldViewerVisible property

Note	NoteViewerVisible Property
CoolSheets	
Guidelines	GuidelinesOn Property
Grid	
QuickZoom	ZoomWindowVisible Property
Zoom	ZoomPercentage Property

Insert Menu Command	**OLE Automation Equivalent**
Link	
Data Field	FieldTemplates collection
Legend	ShowLegend property
Starting Point	
CoolSheet	
Guideline	
Horizontal	
Vertical	
SPC Chart	
Object	InsertObjectFromFile method

Format Menu Command	**OLE Automation Equivalent**
Fill	FillPattern Property
	FillColor Property
Line	Line_Object
Ends	DestArrowColor Property
	DestArrowSize Property
	DestArrowStyle Property
	SourceArrowColor Property
	SourceArrowSize Property
	SourceArrowStyle Property
Effects	
Font	TextBlock Object
Text Alignment	Align Method
Object Properties	
Chart Properties	
Shape Numbering	Number Property
	NumberShown Property
Display Shape Number	
Set Next Shape Number	
Auto Renumber	
Renumber Tool	

Tools Menu Command	**OLE Automation Equivalent**
Spelling	Spelling method
Protect Chart	Protected Property
Run FlowChart	
Shape Action Wizard	
Edit Object Script	
Edit Chart Script	
Import Data	
Report	

Input	
Output	
Metrics	
Add-Ons	
Customize	
Options	Preferences Object

Arrange Menu Command	**OLE Automation Equivalent**
Order	
Send to back	ToBack Method (Object Object)
Bring to front	ToFront Method (Object Object)
Send Backward	
Bring Forward	
Layers	
Layer Manager	
Edit All Layers	
Add Layer	
Move to Layer	
Move Back One Layer	
Move Forward One Layer	
Align	Align Method
Left	
Center	
Right	
Page Center	
Top	
Middle	
Bottom	
Page Middle	
Make Same Size	MakeSameSize Method
Width	
Height	
Both	
Fit to Text	
Space Evenly	SpaceEvenly Method
Across, Center	
Down, Center	
Across, Edges	
Down, Edges	
Center	
Entire Chart	
Selected Objects	
Group	
Ungroup	
Make New Shape	
Break Apart	
Combine	
Connect Closed	
Connect Open	
Disconnect	
Join	

Outline	
Trim	
Intersect	
Fragment	
Punch	
Slice	
Rotate/Flip	
Rotate Right	Rotation Property
Rotate Left	Rotation Property
Angle	
Flip Horizontal	FlippedHorizontal Property
Flip Vertical	FlippedVertical Property
Replace Shape	
Reverse Ends	
Fit To Text	FitShapeToText Method

Window Menu Command	**OLE Automation Equivalent**
New Window	
Arrange All	ArrangeIcons method
Split	
charts	Activate method

Help Menu Command	**OLE Automation Equivalent**
FlowCharter Help	
Micrografx Home Page	
Office Compatible	
Tip of the Day	
About FlowCharter	

Using C++

Use the Microsoft C++ ClassWizard to help you access OLE Automation functionality. The Add Class… button in the ClassWizard lets you read a typelib file and create a .H and a .CPP file from the typelib file. Use this function to create the OLE Automation FLOW70.H and FLOW70.CPP files from the FlowCharter typelib file named FLOW70.TLB.

After you have generated the FLOW70.CPP and FLOW70.H files, you can start making calls to FlowCharter with a C++ program.

Note

If you want to port C++ code written for ABC FlowCharter 4.0 to FlowCharter 7, these files will have to be recreated with the ClassWizard and recompiled. C++ programs written for ABC FlowCharter 4.0 and FlowCharter 7 are not compatible.

C++: Creating an Application Object

The first step in accessing FlowCharter is to create an application object, as illustrated below.

```
#include "FLOW70.H"            // This file generated by ClassWizard

Application ABC;

ABC.CreateDispatch ("ABCFlow.application");
ABC.SetVisible (TRUE);
```

In the sample code above, type Application is a class generated by the ClassWizard defined in FLOW70.H. All of the FlowCharter Application APIs are available in the Application class. All FlowCharter properties are preceded with Set or Get, as explained in FLOW70.H. The call CreateDispatch("ABCFlow.application") performs the same function as Set ABC = CreateObject("ABCFlow.application") in Visual Basic. This call starts FlowCharter if FlowCharter is not already running, and returns a valid application object that you can use to call the APIs. The last line in this sample, ABC.SetVisible(TRUE), performs the same function as ABC.Visible = True in Visual Basic.

All other FlowCharter Automation objects use AttachDispatch instead of CreateDispatch, as illustrated by the sample below.

```
// Setup the ABCChart
Chart ABCChart;

// Get the active Chart
ABCChart.AttachDispatch(ABC.GetActiveChart());

VARIANT vEmpty;
VariantInit(&vEmpty);

ABCChart.DrawShape(vEmpty);
VariantClear(&vEmpty);
```

The sample code above assumes that FlowCharter is a valid FlowCharter application object. The call to ABCChart.AttachDispatch(ABC.GetActiveChart( )) puts the ABCChart object in a state that enables calls to all Chart Object APIs, such as the DrawShape call shown in the sample.

Chart.DrawShape takes one optional parameter, the name of the type of shape to draw. In Visual Basic, optional parameters are handled internally, so you can just omit an optional parameter and Visual Basic does all the work. However, in C++, you must declare a variable of type VARIANT, initialize it, and pass it to DrawShape. If you want to tell FlowCharter to draw a Process shape, use the code shown below instead.

```
VARIANT vName;
VariantInit(&vName);

V_VT(&vName) = VT_BSTR;
V_BSTR(&vName) = SysAllocString("Process");
ABCChart.DrawShape(vName);
VariantClear(&vName);
```

If you are using the Microsoft Foundation Classes, you can use the COleVariant wrapper class in place of variants. The COleVariant class manages many of the cumbersome tasks associated with variants. The sample below performs the same operation as the sample above.

```
COleVariant vName("Process");
ABCChart.DrawShape(vName);
```

You can also use the '=' operator in place of AttachDispatch as demonstrated below.

```
Chart ABCChart;

// Get the active Chart
ABCChart = ABC.GetActiveChart();
```

For more information on VARIANT variables, see the OLESDKV2.HLP file shipped with VC++ 1.5, and search for "Variant Manipulation Functions."

C++: Events

To use the FlowCharter Events VBX in C++, you need to use the CVBControl interface provided in MFC. The simplest way is to check "Custom VBX Controls" on the Options menu of MFC AppWizard when starting your new project.

In the OLE Automation sample Ole_vbx.EXE program, the FlowCharter Events VBX was dropped into the project's ABOUT box using AppStudio. Then the Message Maps Tab was chosen from the ClassWizard on the Resource menu. On the WM_INITDIALOG message the events are registered with FlowCharter.

```
LPDISPATCH lpVBXDisp =
(LPDISPATCH)m_pABCVBX->GetNumProperty("lVBX");

VARIANT vEmpty;

VariantInit(&vEmpty);
m_ABC.RegisterEvent(lpVBXDisp, "C++ Events Sample",
"DoubleClickSUBCLASS", vEmpty);
m_ABC.RegisterEvent(lpVBXDisp, "C++ Events Sample",
"DeleteSUBCLASS", vEmpty);
```

In the example above, note the use of *casting* on the lpVBXDisp assignment statement. MFC's VBX support does not allow the transfer of OLE IDispatch pointers to or from properties of the VBX. Therefore, a series of mirror invisible properties were added to the FlowCharter VBX.

To allow the ABCAUTO.VBX to work with MFC, some redundant properties were added that are "Num" (long) properties. You **must** cast them to (LPDISPATCH) before using them.

FlowCharter	**C++**
App	lApp
Chart	lChart
Object	lObject
Object2	lObject2
FieldValue	lFieldValue
Menu	lMenu
MenuItem	lMenuItem
VBX	lVBX

When responding to a VBX event, use the Message Maps Tab in App Studio to help link your code to the VBX. The sample below illustrates how a program can respond to the double click event from FlowCharter.

```
Void cAboutDig::OnDoubleClickSubclassAbc1(Unit, int, Cwnd*, LPVOID)
{
   // Setup the ABCObject
   Object ABCObject

   ABCObject.AttachDispatch((LPDISPATCH)m_pABCVBX->
        GetNumProperty("lObject"), FALSE);

   // Set the passed object to green
```

```
    ABCObject.SetColor(RGB(0,0xff,0));

    ABCObject.SetText("C++ is easy");

    m_pABCVBX->SetNumProperty("Override", TRUE);
}
```

C++: Notes

When using methods and properties that return a string, check to see if the return value is NULL before assigning it to a CString.

Additionally, all strings returned from OLE Automation must be freed using SysFreeString.

Using C++

Boolean Data Type

A Boolean data type can have a value of either **True** or **False**. **True** is a constant equal to -1. **False** is a constant equal to 0.

Difference Between Writing for External Applications and Internal Scripts

Writing script for Living FlowCharts is a little different from writing Visual Basic code for an external application. You have to change some of the ways you do things. Here is a list of changes:

- You refer to the ABC1 object properties in the external VB code and the Application object properties in the internal script.
- You have to register and unregister events for the external applications, but do not have to for the internal scripts.
- If you are writing an external Visual Basic application, you must install FlowCharter custom controls. The version of the control depends on the version of Visual Basic that you are using. (Visual Basic 4.0 is not backward-compatible with older versions of the controls, those with VBX file extensions.)

  If you are using Visual Basic 4.0, install FLOW.OCX. See To install the OCX event handler for details on installing FLOW.OCX.

  If you are using Visual Basic 3.0 or earlier, install ABCAUTO.VBX. See To install the VBX event handler for details on installing ABCAUTO.VBX.
- You can refer to the current object generically using the “me” method. For example, if you want to accumulate the current object, the script would look like this:

```
Current.AccumulateData me
```

- You need to use ABC.MsgBox only when you are writing an external application. In Living FlowCharter script, you can use just MsgBox.
- In the internal script, the default object is the Shape or Line Object object.
- You can use the new Chart property of the Object object in Living FlowChart script to access Chart object properties. For example, you can get the name and path to the chart within object script by using these lines:

```
CurrentPath = Chart.Fullname                ‘Save path of chart
CurrentPages = Chart.PageCount              ‘Save chart page cound
```

You can use the MsgBox method to display variables without text strings in message boxes as in this example:

```
MsgBox Chart.Name
```

PageLayout Object

Description The PageLayout object is below the Chart object. You can have only one PageLayout object.

Properties	Methods
Application	There are no methods for the
Height	PageLayout object.
MarginBottom	
MarginLeft	
MarginRight	
MarginTop	
Orientation	
PageHeight	
PageOrder	
PageWidth	
PaperSize	
Parent	
PrintBlankPages	
Width	

{button Related Topics,PI(``,`IDH_RT_PageLayout_Object')}

Height Property (PageLayout Object)

Usage *PageLayoutObject*.**Height**

Description You use the **Height** property of the PageLayout object to find the height of the drawing area. The **Height** property is read only.

Data Type Double

Value The height of the drawing area in pixels

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Height_Property_PageLayout_Object')}

Adjusting the Page Layout
Example

Height Property (Application Object)
Height Property (Object Object)
MarginBottom Property
MarginLeft Property
MarginRight Property
MarginTop Property
Orientation Property
PageHeight Property
PageOrder Property
PageWidth Property
PaperSize Property
Width Property (PageLayout Object)

PageLayout Object

MarginBottom Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Margins_option');CW(`concfull')}

Usage *PageLayoutObject*.**MarginBottom** = *Distance*

Description You use the **MarginBottom** property to find or set the bottom page margins. The **MarginBottom** property is read/write.

Data Type Double

Value The bottom page margin in the current units

Flow Equivalent The **MarginBottom** property is equivalent to clicking Page Setup in the File menu and entering a number in the Margin Bottom box.

{button Related Topics,PI(``,`IDH_RT_MarginBottom_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginLeft Property
MarginRight Property
MarginTop Property
Orientation Property
PageHeight Property
PageWidth Property
PaperSize Property
Units Property (Chart Object)
Units Property (Preferences Object)
Width Property (PageLayout Object)

PageLayout Object

MarginLeft Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Margins_option');CW(`concfull')}

Usage *PageLayoutObject*.**MarginLeft** = *Distance*

Description You use the **MarginLeft** property to find or set the left page margins. The **MarginLeft** property is read/write.

Data Type Double

Value The left page margin in the current units

Flow Equivalent The **MarginLeft** property is equivalent to clicking Page Setup in the File menu and entering a number in the Margin Left text box.

{button Related Topics,PI(``,`IDH_RT_MarginLeft_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginBottom Property
MarginRight Property
MarginTop Property
Orientation Property
PageHeight Property
PageWidth Property
PaperSize Property
Units Property (Chart Object)
Units Property (Preferences Object)
Width Property (PageLayout Object)

PageLayout Object

MarginRight Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Margins_option');CW(`concfull')}

Usage *PageLayoutObject*.**MarginRight** = *Distance*

Description You use the **MarginRight** property to find or set the right page margin. The **MarginRight** property is read/write.

Data Type Double

Value The right page margin in the current units

Flow Equivalent The **MarginRight** property is equivalent to clicking Page Setup in the File menu and entering a number in the Margin Left text box.

{button Related Topics,PI(`',`IDH_RT_MarginRight_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginBottom Property
MarginLeft Property
MarginTop Property
Orientation Property
PageHeight Property
PageWidth Property
PaperSize Property
Units Property (Chart Object)
Units Property (Preferences Object)
Width Property (PageLayout Object)

PageLayout Object

MarginTop Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Margins_option');CW(`concfull')}

Usage *PageLayoutObject*.**MarginTop** = *Distance*

Description You use the **MarginTop** property to find or set the top page margin. The **MarginTop** property is read/write.

Data Type Double

Value The top page margin in the current units

Flow Equivalent The **MarginTop** property is equivalent to clicking Page Setup in the File menu and entering a number in the Margin Top text box.

{button Related Topics,PI(`',`IDH_RT_MarginTop_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginBottom Property
MarginLeft Property
MarginRight Property
Orientation Property
PageHeight Property
PageWidth Property
PaperSize Property
Units Property (Chart Object)
Units Property (Preferences Object)
Width Property (PageLayout Object)

PageLayout Object

Orientation Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Orientation_option');CW(`concfull')}

Usage *PageLayoutObject*.**Orientation** = *Value*

Description You use the **Orientation** property to find or set portrait or landscape orientation for the page. The **Orientation** property is read/write.

Data Type Integer

Value The value in the **Orientation** property indicates the page orientation.

Value	Orientation
0	Portrait
1	Landscape

Flow Equivalent The **Orientation** property is equivalent to clicking Page Setup in the File menu and selecting the page orientation.

{button Related Topics,PI(`',`IDH_RT_Orientation_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginBottom Property
MarginLeft Property
MarginRight Property
MarginTop Property
PageHeight Property
PageWidth Property
PaperSize Property
Width Property (PageLayout Object)

PageLayout Object

PageLayout Properties Example

This example uses properties of the PageLayout object and the **PageLayout** property of the Chart object to set up the FlowCharter page.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True                            ' Make ABC visible
ABC.New                                       ' Add a new chart
Set Chart = ABC.ActiveChart                   ' Get the active chart

Chart.Units = 0                               ' Set units to inches

Chart.PageLayout.Orientation = 1              ' Set landscape page orientation

Chart.PageLayout.MarginLeft = 0               ' Set left margin
Chart.PageLayout.MarginRight = 0              ' Set right margin
Chart.PageLayout.MarginTop = 0                ' Set top margin
Chart.PageLayout.MarginBottom = 0             ' Set bottom margin

If Chart.PageLayout.Width > 7 Then            ' Check current page width
   Chart.PageLayout.PageWidth = 7             ' Make pages 7" wide
End If

If Chart.PageLayout.Height > 5 Then           ' Check current page height
   Chart.PageLayout.PageHeight = 5            ' Make pages 5" high
End If
```

PageHeight Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Page_height');CW(`concfull')}

Usage *PageLayoutObject*.**PageHeight** = *Distance*

Description You use the **PageHeight** property to find or set the height of the page. The **PageHeight** property is read/write.

Data Type Double

Value The height of the page in the current units

Flow Equivalent The **PageHeight** property is equivalent to clicking Page Setup in the File menu and entering a number in the Paper Size Height text box.

{button Related Topics,PI(`',`IDH_RT_PageHeight_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginBottom Property
MarginLeft Property
MarginRight Property
MarginTop Property
Orientation Property
PageWidth Property
PaperSize Property
Units Property (Chart Object)
Units Property (Preferences Object)
Width Property (PageLayout Object)

PageLayout Object

PageOrder Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Page_Order_option');CW(`concfull')}

Usage *PageLayoutObject*.**PageOrder** = *Order*

Description You use the **PageOrder** property to find or set the order in which to print the pages in the chart. The **PageOrder** property is read/write.

Data Type Integer

Value The value of the **PageOrder** property method determines the order to print the pages in the chart as shown in the following table.

Value	Description	Pattern
0	Across, then down	▪
1	Down, then across	▪

Flow Equivalent The **PageOrder** property is equivalent to clicking Page Setup in the File menu and clicking one of the Page Order options.

{button Related Topics,PI(``,`IDH_RT_PageOrder_Property')}

Adjusting the Page Layout Example

PrintBlankPages Property

PageLayout Object

PageWidth Property {button Flow Equivalent,JI(`FLOW.HLP>large>large',`IDH_Page_height');CW(`concfull')}

Usage *PageLayoutObject*.**PageWidth** = *Distance*

Description You use the **PageWidth** property to find or set the width of the page. The **PageWidth** property is read/write.

Data Type Double

Value The width of the page in the current units

Flow Equivalent The **PageWidth** property is equivalent to clicking Page Setup in the File menu and entering a value in the Paper Size Width text box.

{button Related Topics,PI(`',`IDH_RT_PageWidth_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginBottom Property
MarginLeft Property
MarginRight Property
MarginTop Property
Orientation Property
PageHeight Property
PaperSize Property
PrintBlankPages Property
Units Property (Chart Object)
Units Property (Preferences Object)
Width Property (Application Object)
Width Property (Object Object)
Width Property (PageLayout Object)

PageLayout Object

PaperSize Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Page_height');CW(`concfull')}

Usage *PageLayoutObject*.**PaperSize** = *Size*

Description You use the **PaperSize** property to find or set the size of paper to be printed. The program uses a "loose matching" routine when you are setting the value so, for example, setting the **PaperSize** property to "let" chooses the size "Letter 8 1/2 x 11 in." The **PaperSize** property is read/write.

Data Type Double

Value The size of the paper

Flow Equivalent The **PaperSize** property is equivalent to clicking Page Setup in the File menu and selecting a value in the Paper Size list box.

{button Related Topics,PI(`',`IDH_RT_PaperSize_Property')}

Adjusting the Page Layout
Example

Height Property (PageLayout Object)
MarginBottom Property
MarginLeft Property
MarginRight Property
MarginTop Property
Orientation Property
PageHeight Property
PageWidth Property
Width Property (PageLayout Object)

PageLayout Object

PaperSize, PageOrder, PrintBlankPages Properties Example

This example uses the **PaperSize** property, the **PageOrder** property, and the **PrintBlankPages** property of the PageLayout object and the **PageLayout** property of the Chart object to prepare a chart for printing.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Printed

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Add a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.PageLayout.PaperSize = "Letter"              ' Use a Letter 8.5 x 11 size page
Chart.PageLayout.PageOrder = 1                     ' Print pages down then across
Chart.PageLayout.PrintBlankPages = False           ' Omit printing pages with no objects

Chart.DrawPositionX = 3.5                          ' Set X location for the first shape
Chart.DrawPositionY = 4.75                         ' Set Y location for the first shape
Set NewShape = Chart.DrawShape                     ' Place the first shape on page 1
NewShape.Text = "Page one"                         ' Enter text in the shape

Chart.DrawPositionX = 10.5                         ' Set X location for the next shape
Set NewShape = Chart.DrawShape                     ' Place the second shape on page 3
NewShape.Text = "Page three. Page two is blank."   ' Enter text in the shape
NewShape.Shape.FitShapeToText                      ' Enlarge the shape if necessary

Chart.DrawPositionX = 10.5                         ' Set X location for the next shape
Chart.DrawPositionY = 14.25                        ' Set Y location for the next shape
Set NewShape = Chart.DrawShape                     ' Place the third shape on page 4
NewShape.Text = "Page four"                        ' Enter text in the shape

Chart.View = 2                                     ' Display the used pages

ABC.Printer = "LPT1"                               ' Select the first printer on LPT1
Printed = Chart.PrintOut                           ' Print the chart
```

PrintBlankPages Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Print_Blank_Pages_option');CW(`concfull')}

Usage *PageLayoutObject*.**PrintBlankPages** = {True | False}

Description You use the **PrintBlankPages** property to specify whether a blank page should be printed if there are no objects on the page.The **PrintBlankPages** property is read/write.

Data Type Integer (Boolean)

Value True prints a blank page when there is a blank page in the chart; False prints only pages that have objects on them.

Flow Equivalent The **PrintBlankPages** property is equivalent to clicking Page Setup in the File menu and selecting or deselecting the Print Blank Pages option.

{button Related Topics,PI(``,`IDH_RT_PrintBlankPages_Property')}

Printing Charts
Example

PageOrder Property

PageLayout Object

Width Property (PageLayout Object)

Usage	*PageLayoutObject*.**Width**
Description	You use the **Width** property to find or set the width of the drawing area. The **Width** property is read only.
Data Type	Double
Value	The width of the drawing area
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_Width_Property_PageLayout_Object')}

Adjusting the Page Layout
Example

MarginBottom Property
MarginLeft Property
MarginRight Property
MarginTop Property
Orientation Property
PageHeight Property
PageWidth Property
PaperSize Property
Width Property (Application Object)
Width Property (Object Object)

PageLayout Object

Preferences Object

Description The Preferences object is below the Application object. You can have only one Preferences object.

Properties	Methods
AlignToRulers	There are no methods for the
Application	Preferences object.
ChannelAlignment	
LineSpacingX	
LineSpacingY	
Parent	
ShowRulers	
SmartShapeSpacing	
SSSHorizontal	
SSSVertical	
TouchAlignment	
Units	

{button Related Topics,PI(``,`IDH_RT_Preferences_Object')}

AlignToRulers Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_ALIGN');CW(`concfull')}

Usage *PreferencesObject*.**AlignToRulers** = *Value*

Description The **AlignToRulers** property lets you find or set the choices for aligning to rulers. The **AlignToRulers** property is read/write.

Data Type Integer

Value The values for the **AlignToRulers** property are in the following table.

Value	Description
0	Off (not selected)
1	Coarse
2	Fine

Flow Equivalent The **AlignToRulers** property is equivalent to clicking Options on the Tools menu, clicking the Alignment tab, and selecting or clearing the Snap to Grid option, and selecting the Coarse or the Fine option.

{button Related Topics,PI(`',`IDH_RT_AlignToRulers_Property')}

Alignment Options
Example

ChannelAlignment Property
ShowRulers Property
SmartShapeSpacing Property
SSSHorizontal Property
SSSVertical Property
TouchAlignment Property
Units Property (Chart Object)
Units Property (Preferences Object)

Preferences Object

Preferences Object Properties Example

This example uses properties of the Preferences object to set chart preferences.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")    ' Start ABC
ABC.Visible = True                               ' Make ABC visible

ABC.New                                          ' Create a new chart
ABC.Preferences.AlignToRulers = 2                ' Set fine ruler alignment
ABC.Preferences.ChannelAlignment = True          ' Enable channel alignment
ABC.Preferences.LineSpacingX = 2                 ' Horizontal spacing for new lines
ABC.Preferences.LineSpacingY = 2                 ' Vertical spacing for new lines
ABC.Preferences.ShowRulers = True                ' Display the rulers
ABC.Preferences.SmartShapeSpacing = True         ' Enable shape spacing
ABC.Preferences.SSSHorizontal = 1                ' Shape spacing horizontal value
ABC.Preferences.SSSVertical = 1                  ' Shape spacing vertical value
ABC.Preferences.TouchAlignment = True            ' Enable touch alignment
ABC.Preferences.Units = 0                        ' Set units to inches

ABC.MsgBox "On the Tools menu, click Options. Then click the Alignment tab to verify the settings."
```

ChannelAlignment Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_ALIGN');CW(`concfull')}

Usage *PreferencesObject*.**ChannelAlignment** = {True | False}

Description The **ChannelAlignment** property lets you find or set whether channel alignment is turned on. The **ChannelAlignment** property is read/write.

Data Type Integer (Boolean)

Value True turns channel alignment on; False turns it off.

Flow Equivalent The **ChannelAlignment** property is equivalent to clicking Options on the Tools menu, clicking the Alignment tab, and selecting the Channel Alignment option (True) or clearing the option (False).

{button Related Topics,PI(``,`IDH_RT_ChannelAlignment_Property')}

LineSpacingX Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_LINESPACE')}

Usage *PreferencesObject*.**LineSpacingX** = *HorizontalLineSpacing*

Description The **LineSpacingX** property lets you find or set the horizontal value used by line spacing. The **LineSpacingX** property is read/write.

Data Type Double

Value The number of inches or centimeters that line spacing is to use horizontally

Flow Equivalent The **LineSpacingX** property is equivalent to clicking Options on the Tools menu, clicking the Line spacing tab, and entering a value in the Horizontal Spacing box.

{button Related Topics,PI(`',`IDH_RT_LineSpacingX_Property')}

Line Options
Example

LineSpacingY Property
Units Property (Chart Object)
Units Property (Preferences Object)

Preferences Object

LineSpacingY Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_LINESPACE')}

Usage *PreferencesObject*.**LineSpacingY** = *VerticalLineSpacing*

Description The **LineSpacingY** property lets you find or set the vertical value used by line spacing. The **LineSpacingY** property is read/write.

Data Type Double

Value The number of inches or centimeters that line spacing is to use vertically

Flow Equivalent The **LineSpacingY** property is equivalent to clicking Options on the Tools menu, clicking the Line Spacing tab, and entering a value in the Vertical Spacing box.

{button Related Topics,PI(``,`IDH_RT_LineSpacingY_Property')}

Line Options
Example

LineSpacingX Property
Units Property (Chart Object)
Units Property (Preferences Object)

Preferences Object

ShowRulers Property {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Rulers_Command');CW(`concfull')}

Usage *PreferencesObject*.**ShowRulers** = {True | False}

Description The **ShowRulers** property lets you find or set whether the rulers are shown. The **ShowRulers** property is read/write.

Data Type Integer (Boolean)

Value True turns the rulers on; False turns them off.

Flow Equivalent The **ShowRulers** property is equivalent to clicking Rulers on the View menu.

{button Related Topics,PI(`',`IDH_RT_ShowRulers_Property')}

SmartShapeSpacing Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_ALIGN');CW(`concfull')}

Usage *PreferencesObject*.**SmartShapeSpacing** = {True | False}

Description The **SmartShapeSpacing** property lets you find or set whether shape spacing is turned on. The **SmartShapeSpacing** property is read/write.

Data Type Integer (Boolean)

Value True turns space shaping on; False turns it off.

Flow Equivalent The **SmartShapeSpacing** property is equivalent to clicking Options on the Tools menu, clicking the Alignment tab, and selecting the Shape Spacing option (True) or deselecting the option (False).

{button Related Topics,PI(``,`IDH_RT_SmartShapeSpacing_Property')}

SSSHorizontal Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_ALIGN');CW(`concfull')}

Usage *PreferencesObject*.**SSSHorizontal** = *HorizontalValue*

Description The **SSSHorizontal** property lets you find or set the horizontal value used by Shape Spacing. The **SSSHorizontal** property is read/write.

Data Type Double

Value The number of inches or centimeters that Shape Spacing is to use horizontally

Flow Equivalent The **SSSHorizontal** property is equivalent to clicking Options on the Tools menu, clicking the Alignment tab, and entering a value for the horizontal component used by Shape Spacing.

{button Related Topics,PI(``,`IDH_RT_SSSHorizontal_Property')}

Alignment Options
Example

AlignToRulers Property
ChannelAlignment Property
ShowRulers Property
SmartShapeSpacing Property
SSSVertical Property
TouchAlignment Property
Units Property (Chart Object)
Units Property (Preferences Object)

Preferences Object

SSSVertical Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_ALIGN');CW(`concfull')}

Usage *PreferencesObject*.**SSSVertical** = *VerticalValue*

Description The **SSSVertical** property lets you find or set the vertical value used by Shape Spacing. The **SSSVertical** property is read/write.

Data Type Double

Value The number of inches or centimeters that Shape Spacing is to use vertically

Flow Equivalent The **SSSVertical** property is equivalent to clicking Options on the Tools menu, clicking the Alignment tab, and entering a value for the vertical component used by Shape Spacing.

{button Related Topics,PI(``,`IDH_RT_SSSVertical_Property')}

Alignment Options
Example

AlignToRulers Property
ChannelAlignment Property
ShowRulers Property
SmartShapeSpacing Property
SSSHorizontal Property
TouchAlignment Property
Units Property (Chart Object)
Units Property (Preferences Object)

Preferences Object

TouchAlignment Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_ALIGN');CW(`concfull')}

Usage *PreferencesObject*.**TouchAlignment** = {True | False}

Description The **TouchAlignment** property lets you find or set whether touch alignment is turned on. The **TouchAlignment** property is read/write.

Data Type Integer (Boolean)

Value True turns touch alignment on; False turns it off.

Flow Equivalent The **TouchAlignment** property is equivalent to clicking Options on the Tools menu, clicking the Alignment tab, and selecting the Touch Alignment option (True) or deselecting the option (False).

{button Related Topics,PI(``,`IDH_RT_TouchAlignment_Property')}

Units Property (Preferences Object)

Usage *PreferencesObject*.**Units** = *UnitsIndicator*

Description You use the **Units** property of the Preferences object to set whether positions are measured in inches or centimeters. The default is inches. The **Units** property is read/write.

Data Type Integer

Value The units used for measurements are listed in the table below.

UnitsIndicator	**Description**
0	Inches
1	Centimeters

Flow Equivalent The **Units** property is equivalent to dragging the Inches or Centimeters button from the View category in the Customize dialog box to a toolbar, and then clicking it.

{button Related Topics,PI(``,`IDH_RT_Units_Property_Preferences_Object')}

Shape Object

Description The Shape object is below the Object object. You can have only one Shape object for each Object object.

Properties	Methods
Application	DeleteLines
BorderColor	FitShapeToText
BorderStyle	Launch
BorderWidth	Link
FillColor	Renumber
FillPattern	ReplaceShape
IsLaunched	
IsLinked	
LaunchCommand	
LinkedChartName	
LinkFields	
NoteFont	
NoteText	
NoteTextLF	
Number	
NumberShown	
Parent	
ShadowColor	
ShadowOffset	
ShadowStyle	
ShapeName	

{button Related Topics,PI(`',`IDH_RT_Shape_Object')}

BorderColor Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**BorderColor** = *Color*

Description You use the **BorderColor** property to set the border color for shapes using the **MakeRGB** method. A shape border includes not only the outside part of the shape, but also any interior lines used in the shape. For example, it includes the concentric circles on the inside of a 5.25" floppy disk shape. The **BorderColor** property is read/write.

Data Type Long

Value The color of the shape border

Flow Equivalent The **BorderColor** property is equivalent to selecting a shape, clicking the Line Color button on the Formatting toolbar, and then clicking the color you want.

{button Related Topics,PI(``,`IDH_RT_BorderColor_Property')}

Setting Shape Colors
Fill, Border, and Shadow Colors
Example

BasicColor Method
Color Property (Object Object)
FillColor Property
MakeRGB Method
ShadowColor Property

Shape Object

BorderColor, BorderStyle, BorderWidth, FillPattern, FillColor Properties Example

This example uses the **BorderColor** property, **BorderStyle** property, **BorderWidth** property, **FillPattern** property, and **FillColor** property of the Shape object to set the border and fill of a shape.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim NewShape1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set NewShape1 = Chart.DrawShape("Decision")            ' Create a Decision shape
NewShape1.Shape.BorderColor = ABC.MakeRGB(0, 0, 255)      ' Make the border blue
NewShape1.Shape.BorderStyle = 1                        ' Make the border a solid line
NewShape1.Shape.BorderWidth = 3                        ' Give the border a medium width
NewShape1.Shape.FillPattern = 23                       ' Fill the shape with a brick pattern
NewShape1.Shape.FillColor = ABC.MakeRGB(255, 255, 0)       ' Fill the shape with yellow
```

BorderStyle Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**BorderStyle** = *StyleNumber*

Description You use the **BorderStyle** property to find or set the line style for shape borders. A shape border includes not only the outside edge of a shape, but also any interior lines used in the shape (for example, the concentric circles on the inside of a 5.25" floppy disk shape). FlowCharter provides many useful border styles, including solid and dashed lines and an invisible border. The BorderStyle property is read/write.

Data Type Integer

Value Set the BorderStyle property to 0 for an invisible border and to 1 for a solid border. The following illustration shows the values of the BorderStyle property for each available style.

Flow Equivalent The **BorderStyle** property is equivalent to selecting a shape, clicking the Line Style button on the Formatting toolbar, and then clicking the border style you want.

{button Related Topics,PI(`',`IDH_RT_BorderStyle_Property')}

Border Style and Width
Example

Shape Object

BorderWidth Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**BorderWidth** = *WidthValue*

Description You use the BorderWidth property to find or set the width of the border of a shape. The BorderWidth property is read/write.

Data Type Integer

Value The **BorderWidth** property can have a value ranging from 1 to 5, with 1 a hairline (the thinnest possible line) and 5 the thickest line width.

Flow Equivalent The **BorderWidth** property is equivalent to selecting a shape, and then clicking the arrows next to the Line Weight button on the Formatting toolbar.

{button Related Topics,PI(`',`IDH_RT_BorderWidth_Property')}

Border Style and Width
Example

Shape Object

FillColor Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**FillColor** = *Color*

Description The **FillColor** property lets you find or set the fill color (interior color) for shapes (see the **MakeRGB** method). The **FillColor** property is read/write.

Data Type Long

Value The fill color of the shape

Flow Equivalent The **FillColor** property is equivalent to selecting a shape, clicking the Fill Color button on the Formatting toolbar, and then clicking the color you want.

{button Related Topics,PI(``,`IDH_RT_FillColor_Property')}

Setting Shape Colors
Fill, Border, and Shadow Colors
Example1
Example2

BasicColor Method
BorderColor Property
Color Property (Object Object)
MakeRGB Method
ShadowColor Property

Shape Object

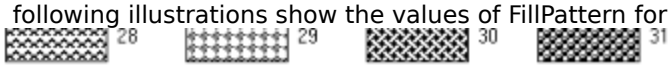
FillPattern Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**FillPattern** = *PatternNumber*

Description The **FillPattern** property lets you find or set a shape's fill pattern. The FillPattern property is read/write.

Data Type Integer

Value Set the FillPattern property to 0 for a transparent fill and to 1 for a solid fill. The following illustrations show the values of FillPattern for each available pattern



Flow Equivalent The **FillPattern** property is equivalent to clicking a shape, clicking Fill on the Format menu, selecting the Fill option, and then clicking the fill pattern you want.

{button Related Topics,PI(`',`IDH_RT_FillPattern_Property')}

IsLaunched Property

Usage *ShapeObject*.**IsLaunched**

Description You can arrange to launch a program using a shape. You use the **IsLaunched** property to find if a shape has an associated program that it can launch. The **IsLaunched** property is read only.

Data Type Integer (Boolean)

Value True means the shape has an associated launch; False means it does not.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_IsLaunched_Property')}

IsLinked Property

Usage *ShapeObject*.**IsLinked**

Description You can link shapes between charts. After the shapes are linked, you can double click a designated shape in one chart to open the linked chart. You use the **IsLinked** property to determine if a shape is linked to another chart. The **IsLinked** property is read only.

Data Type Integer (Boolean)

Value True means the shape is linked; False means the shape is not linked.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_IsLinked_Property')}

Linking Shapes to Other Charts
Example

Link Method
LinkedChartName Property
LinkFields Property
LinkIndicator Property
LinkShadow Property

Shape Object

Number Property

Usage *ShapeObject*.**Number** = *ShapeNumber*

Description You use the **Number** property to find or set the number of a shape. The **Number** property is read/write.

Data Type String

Value The number of a shape

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Number_Property')}

Numbering Shapes
Example

NextNumber Property
NumberShown Property
Renumber Method

Shape Object

NumberShown Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Shape_Numbers_Overview');CW(`concfull')}

Usage *ShapeObject*.**NumberShown** = {True | False}

Description You use the NumberShown property to display or hide shape numbers. The NumberShown property is read/write.

Data Type Integer (Boolean)

Value True means the shape number is shown; False means it is not shown.

Flow Equivalent The **NumberShown** property is equivalent to selecting a shape, clicking Shape Numbering on the Format menu, and then clicking Display Shape Numbers on the submenu.

{button Related Topics,PI(``,`IDH_RT_NumberShown_Property')}

Hiding Shape Numbers
Example

NextNumber Property
Number Property
Renumber Method

Shape Object

LaunchCommand Property

Usage *ShapeObject*.**LaunchCommand** = *Command*
The *Command* element is the command that you want executed when the shape is launched, such as the path for the program to run.

Description You use the **LaunchCommand** property to set a command to launch for the object. The LaunchCommand property is read/write.

Data Type String

Value The path and name of the program to run

Flow Equivalent The **LaunchCommand** property is equivalent to clicking the Selector tool in the toolbox, selecting the shape you want to use, clicking the Link button in the standard toolbar, and specifying the command line in the dialog box.

{button Related Topics,PI(`',`IDH_RT_LaunchCommand_Property')}

LinkedChartName Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_LINKDB');CW(`concfull')}

Usage *ShapeObject*.**LinkedChartName** = *ChartName*

Description You use the **LinkedChartName** property to provide a full pathname and filename for a chart and link the shape to the chart. Quotation marks should be used whenever long filenames or long pathnames are used. The **LinkedChartName** property is read/write.

Data Type String

Value The full pathname and filename of the chart linked to the object

Flow Equivalent The **LinkedChartName** property is equivalent to clicking the Selector tool in the toolbox, selecting the shape you want to use, clicking the Link button in the standard toolbar, and entering the pathname in the Link dialog box.

{button Related Topics,PI(`',`IDH_RT_LinkedChartName_Property')}

Linking Shapes to Other Charts
Example

IsLinked Property
Link Method
LinkFields Property
LinkIndicator Property
LinkShadow Property

Shape Object

LinkFields Property {button Flow Equivalent,JI(`FLOW.HLP>dialog',`IDH_LINKDB');CW(`concfull')}

Usage *ShapeObject*.**LinkFields** = {True | False}

Description The **LinkFields** property lets you choose whether to accumulate the linked chart's field data as the object's field information if this object is linked to another chart with field information. The **LinkFields** property is read/write.

Data Type Integer (Boolean)

Value True accumulates the linked chart's field data as the object's field information; False does not accumulate it.

Flow Equivalent The **LinkFields** property is equivalent to clicking the Selector tool in the toolbox, selecting the shape you want to use, clicking the Link button in the standard toolbar, and selecting or clearing the Show Accumulated Data option.

{button Related Topics,PI(`',`IDH_RT_LinkFields_Property')}

NoteFont Property

Usage *ShapeObject*.**NoteFont**

Description The **NoteFont** property lets you find the font object for notes. The **NoteFont** property is read only, but all the properties from the object it returns are read/write.

Data Type Object

Value The Font object for notes

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_NoteFont_Property')}

NoteText Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_NOTES');CW(`concfull')}

Usage *ShapeObject*.**NoteText** = *Note*

Description You use the NoteText property to find or set notes for a shape. You do not need to open the Note window to attach notes to a shape. If you wish, however, you can show or hide the FlowCharter Note Viewer using the **NoteViewerVisible** property. You set the font for notes using the **NoteFont** property. If you wish to preserve Returns when reading a note, you should use the **NoteTextLF** property. The NoteText property is read/write.

Data Type String

Value The note associated with a shape

Flow Equivalent The **NoteText** property is equivalent to selecting a shape and entering text in the Note window.

{button Related Topics,PI(`',`IDH_RT_NoteText_Property')}

Attaching a Note to a Shape
Adding Notes to a Shape
Example

NoteFont Property
NoteIndicator Property
NoteShadow Property
NoteTextLF Property
NoteViewerVisible Property

Shape Object

NoteText, NoteFont Properties Example

This example uses the **NoteText** property and **NoteFont** property of the Shape object to enter note text for a shape and change the text to red.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.New                                    ' Get the active chart

Set Shape1 = Chart.DrawShape("Storage")                ' Draw a Storage shape
Shape1.Shape.NoteText = ("Shape #" + Shape1.Shape.Number + " is a " +
Shape1.Shape.ShapeName + " shape.")                    ' Enter text for the shape's note
Shape1.Shape.NoteFont.Color = ABC.MakeRGB(255, 0, 0)       ' Make the note text red
```

ShadowColor Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**ShadowColor** = *Color*

Description The **ShadowColor** property lets you find or set the shadow color for shapes (see the **MakeRGB** method). The **ShadowColor** property is read/write.

Data Type Long

Value The shadow color for the shape

Flow Equivalent The **ShadowColor** property is equivalent to selecting a shape, clicking Effects on the Format menu, selecting the Shadow option, clicking the down arrow next to the color box, and then clicking the color you want.

{button Related Topics,PI(``,`IDH_RT_ShadowColor_Property')}

Setting Shape Colors
Fill, Border, and Shadow Colors
Example

BasicColor Method
BorderColor Property
Color Property (Object Object)
FillColor Property
MakeRGB Method

Shape Object

ShadowOffset Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**ShadowOffset** = *OffsetAmount*

Description You use the **ShadowOffset** property to find or set the width of a shadow (the distance the shadow appears away from its shape). The ShadowOffset property is read/write.

Data Type Integer

Value The **ShadowOffset** property can have a value ranging from 1 to 5, with 1 a hairline (the thinnest possible shadow) and 5 the thickest shadow.

Flow Equivalent The **ShadowOffset** property is equivalent to clicking a shape, clicking Effects on the Format menu, selecting the Shadow option, and clicking the arrows next to the width box.

{button Related Topics,PI(`',`IDH_RT_ShadowOffset_Property')}

Shadow Style and Width
Example
ShadowStyle Property
Shape Object

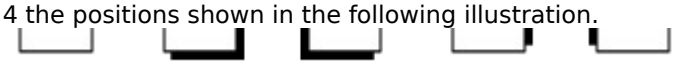
ShadowStyle Property {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**ShadowStyle** = *ShadowStyleNumber*

Description You use the **ShadowStyle** property to find or set the position of the shadow on a shape. The ShadowStyle property is read/write.

Data Type Integer

Value ShadowStyle can have a value from 0 to 4, with 0 being no shadow and 1 through 4 the positions shown in the following illustration.



Flow Equivalent The **ShadowStyle** property is equivalent to clicking a shape, clicking the Shadow button on the Formatting toolbar, selecting the Shadow option, and then clicking the shadow position you want.

{button Related Topics,PI(`',`IDH_RT_ShadowStyle_Property')}

Shadow Style and Width
Example
ShadowOffset Property
Shape Object

ShapeName Property

Usage *ShapeObject*.**ShapeName**

Description The **ShapeName** property lets you find the name of a shape, such as "Process" or "Decision." The **ShapeName** property is read only.

Data Type String

Value The name of the shape

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_ShapeName_Property')}

ShapeName, ShadowStyle, ShadowColor, ShadowOffset Properties Example

This example uses the **ShapeName** property, **ShadowStyle** property, **ShadowColor** property, and **ShadowOffset** property of the Shape object to identify a shape and apply a shadow in the desired location, color, and offset. To show the proper result, FlowCharter should be open and contain shapes, at least one of which is a Decision shape.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim CurrentShape As Object
Dim SelectedShapes As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                               ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set SelectedShapes = Chart.Objects

Do
   Set CurrentShape = SelectedShapes.ItemFromShapes       ' Check all shapes in the chart
   If CurrentShape.Shape.ShapeName = "Decision" Then      ' If a shape is a Decision
        CurrentShape.Shape.ShadowStyle = 1                ' Apply shadow to bottom right
        CurrentShape.Shape.ShadowColor = ABC.MakeRGB(0, 0, 255)   ' Make shadow blue
        CurrentShape.Shape.ShadowOffset = 5               ' Give the shadow a big offset
   End If
Loop While CurrentShape.Valid
```

DeleteLines Method

Usage *ShapeObject*.**DeleteLines**

Description You use the **DeleteLines** method of the Shape object to delete all the lines attached to a specific shape. Deleting lines with this method does not place the lines in the Windows Clipboard.

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_DeleteLines_Method')}

DeleteLines, Repaint Methods Example

This example uses the **Repaint** method of the Object object and the **DeleteLines** method of the Shape object to refresh the screen and delete lines connected to a shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, NewLine As Object
Dim Counter As Single

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape("Document")               ' Draw a Document shape
For Counter = 0 To 3                                   ' To draw 4 lines to the shape
   Chart.DrawPositionY = Counter + 1                   ' Change the lines' starting points
   Set NewLine = Chart.DrawLineToOneObject(Shape1, Counter)        ' Draw a line to the shape
   NewLine.Repaint                                     ' Refresh the lines on screen
Next Counter

ABC.MsgBox "Are you ready to delete the lines?"
Shape1.Shape.DeleteLines                               ' Delete all lines connected to Shape1
```

FitShapeToText Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Shapes_Look');CW(`concfull')}

Usage *ShapeObject*.**FitShapeToText**

Description You use the FitShapeToText method to expand or contract a shape around its center so that the text in it fits. This is useful when the length of the text string may vary and you want to avoid hiding text that will not fit within the shape.

Flow Equivalent The **FitShapeToText** method is equivalent to selecting the shape, and then clicking Fit to Text on the Arrange menu.

{button Related Topics,PI(`',`IDH_RT_FitShapeToText_Method')}

FitShapeToText Method Example

This example uses the **FitShapeToText** method of the Shape object to change the size of a shape so the text in it fits correctly.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim NewShape1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set NewShape1 = Chart.DrawShape("Document")        ' Draw a Document shape
NewShape1.Text = "Antidisestablishmentarianism"    ' Enter text in the shape
NewShape1.Shape.FitShapeToText                     ' Resize the shape to fit the text
```

Launch Method

Usage	*ShapeObject*.**Launch**
Description	You use the **Launch** method to execute the shape's launch.
Data Type	Integer (Boolean)
Value	True means the launch was successful; False means the launch was not successful.
Flow Equivalent	The **Launch** method is equivalent to double clicking the shape you set for launching.

{button Related Topics,PI(``,`IDH_RT_Launch_Method')}

Launch Example

This example uses the **Launch** method, the **IsLaunched** property and the **LaunchCommand** property of the Shape object to determine if a shape has a launch attached to it and, if it does not, to define the program to be launched and launch the application.

```
Dim ABC As Object, Chart As Object, CurrentShape As Object, Shapes As Object
Dim NewShape As Object
Dim CmdStr As String

Set ABC = CreateObject("ABCFlow.application")' Start ABC
ABC.Visible = True                                  ' Make ABC visible
If ABC.Charts.Count < 1 Then                        ' See if a chart is already open
    Set Chart = ABC.New                             ' Create a new chart
Else
    Set Chart = ABC.ActiveChart                     ' Get the already open chart for use
End If

Set NewShape = Chart.DrawShape                      ' Draw a new shape

Set Shapes = Chart.Objects

Do
    Set CurrentShape = Shapes.ItemFromShapes

    If CurrentShape.Shape.IsLaunched = False Then   ' If shape does not have an associated
                                                    ' launch then set up one
    CmdStr = "C:\windows\notepad.exe\ " + Chart.Name + ".txt"        ' Append chart name for the
                                                    ' launch command
    CurrentShape.Text = "Double-click me to launch Notepad!" ' Show that there is a launch now
    CurrentShape.Shape.FitShapeToText               ' Resize so the text is visible
    CurrentShape.Shape.LaunchCommand = CmdStr       ' Set command line for launch command
    Else ABC.MsgBox "This shape's launch is already set to '" & CurrentShape.LaunchCommand & "'."
    End If
Loop While CurrentShape.Valid

NewShape.Shape.Launch                               ' Launch the new shape we added
```

Link Method {button Flow Equivalent,JI(`FLOW.HLP>procedur','`IDH_LINKOPENP');CW(`concfull')}

Usage *ShapeObject*.**Link**

Description You use the **Link** method to open the chart linked to a shape. If you have not yet set a value for the **LinkedChartName** property, this method creates a new chart with an automatically generated filename using the **DefaultFilePath** property.

Data Type Object

Value The linked chart

Flow Equivalent The **Link** method is equivalent to double-clicking the linked shape. The linked chart opens and becomes the active chart.

{button Related Topics,PI(`','`IDH_RT_Link_Method')}

Opening a Linked Chart
Example

GroupAndLink Method

DefaultFilePath Property
IsLinked Property
LinkedChartName Property
LinkFields Property
LinkShadow Property

Shape Object

Link Method, LinkFields Property, and LinkShadow Property Example

This example uses the **Link** method and the **LinkFields** property of the Shape object and the **LinkShadow** property of the Chart object to link a shape to show a shadow on linked shapes, link a shape to another chart, and clear the Show Accumulated Data option in the Link dialog box.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Link1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.CloseAll                                       ' Close all charts
ABC.New                                                  ' Add a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Chart.LinkShadow = True                            ' Show a shadow on linked shapes
Set Shape1 = Chart.DrawShape("Decision")           ' Draw a shape
Set Link1 = Shape1.Shape.Link                      ' Link the shape to a new chart
ABC.TileCharts                                     ' Tile all chart windows

Shape1.Shape.LinkFields = 0                        ' Clear Show Accumulated Data box
ABC.MsgBox "Accumulations of fields from the linked chart will not appear in the top chart."
```

Renumber Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Shape_Numbers_Overview');CW(`concfull')}

Usage *ShapeObject*.**Renumber**

Description You use the **Renumber** method to replace the current shape number with the value in the **NextNumber** property and increment the value in the NextNumber **property**.

Flow Equivalent The **Renumber** method is equivalent to clicking the Renumber tool in the toolbox, and then clicking the shape.

{button Related Topics,PI(`',`IDH_RT_Renumber_Method')}

Numbering Shapes
Example

NextNumber Property
Number Property
NumberShown Property

Shape Object

ReplaceShape Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_PLACESHAPES');CW(`concfull')}

Usage *ShapeObject*.**ReplaceShape** [*ShapeName*]
The *ShapeName* element is the name of the shape to put in place of the selected shape.

Description You use the ReplaceShape method of the Shape object to replace shapes. The new shape connects to the lines of the old shape. You can replace shapes with the chart's current shape or can specify a shape.

Data Type Integer (Boolean)

Value True means the shape was replaced successfully; False means the replacement was not successful. The *ShapeName* element is a string.

Flow Equivalent The **ReplaceShape** method is equivalent to selecting a shape, choosing the new shape in the Shape palette, and then clicking Replace Shape button on the Arrange+ toolbar.

{button Related Topics,PI(``,`IDH_RT_ReplaceShape_Method')}

ReplaceShape Method Example

This example uses the **ReplaceShape** method of the Shape object to replace one shape with another.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Shape1 = Chart.DrawShape("Terminal")           ' Draw a Terminal shape
Shape1.Shape.ReplaceShape "Delay"                  ' Change Shape1 into a Delay shape
```

NoteTextLF Property

Usage *ShapeObject*.**NoteTextLF** = *Note*

Description You use the **NoteTextLF** property to find or set notes for a shape. When adding a note, the property is identical to the **NoteText** property. When reading the text in a note, the property does not substitue spaces for Returns as the **NoteText** property does. If you do not wish to preserve Returns, you should use the **NoteText** property.
You do not need to open the Note window to attach notes to a shape. If you wish, however, you can show or hide the FlowCharter Note Viewer using the **NoteViewerVisible** property. You set the font for notes using the **NoteTextLF** property. The **NoteTextLF** property is read/write.

Data Type String

Value The note associated with a shape with Returns preserved

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_NoteTextLF_Property')}

Attaching a Note to a Shape
Adding Notes to a Shape
Example

NoteFont Property
NoteIndicator Property
NoteShadow Property
NoteText Property
NoteViewerVisible Property

Shape Object

NoteTextLF Property Example

This example uses the **NoteTextLF** property of the Shape object to read note text and preserve the Returns in the text.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, NoteText As String

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.New                                    ' Get the active chart

Set Shape1 = Chart.DrawShape("Storage")                ' Draw a Storage shape
Shape1.Shape.NoteText = ("Shape #" + Shape1.Shape.Number + " is a " +
Shape1.Shape.ShapeName + " shape. " + CHR$(13) + "This is a note.")          ' Enter text for the
shape's note
NoteText = Shape1.Shape.NoteTextLF                     ' Read the note, preserving Returns
```

TextBlock Object

Description The TextBlock object is below the Object object. You can have only one TextBlock object for each Object object.

Properties	Methods
Application	UnattachFromLine
AttachedToLine	
Parent	

{button Related Topics,PI(`',`IDH_RT_TextBlock_Object')}

AttachedToLine Property

Usage	*TextBlockObject*.**AttachedToLine**
Description	The **AttachedToLine** method lets you find whether a text block is attached to a line. The **AttachedToLine** method is read only.
Data Type	Integer (Boolean)
Value	True means the text block is attached to a line; False means it is not.
Flow Equivalent	None

{button Related Topics,PI(`',`IDH_RT_AttachedToLine_Property')}

Unattaching Text from a Line Example

TextBlock Property
UnattachFromLine Method

TextBlock Object

UnattachFromLine Method {button Flow Equivalent,JI(`FLOW.HLP>large',`IDH_Determining_How_Lines_Look');CW(`concfull')}

Usage *TextBlockObject*.**UnattachFromLine**

Description The **UnattachFromLine** method lets you detach a text block from all lines.

Flow Equivalent The **UnattachFromLine** method is equivalent to dragging a text block away from lines to detach it from the lines.

{button Related Topics,PI(`',`IDH_RT_UnattachFromLine_Method')}

Unattaching Text from a Line
Example
AttachedToLine Property
TextBlock Object

UnattachFromLine Method Example

This example uses the **UnattachFromLine** method of the TextBlock object to detach text from a line.

```
Dim ABC As Object, Chart As Object
Dim NewLine As Object
Dim NewText As Object
DIM X AS STRING
DIM Y AS STRING

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set NewLine = Chart.DrawFreeLine(4,4)                  ' Draw a line
Set NewText = Chart.DrawTextBlock("Hello there!")      ' Draw a text block
NewLine.Line_.AttachText NewText                       ' Attach the text to the line
ABC.MsgBox "The text is attached to the line. When you click OK, it will be detached."
X = NewText.TextBlock.AttachedToLine
ABC.MsgBox "Is it true that the line is attached to thetext block? -- " + X
NewText.TextBlock.UnattachFromLine                     ' Detach text from the line
Y = NewText.TextBlock.AttachedToLine
ABC.MsgBox "Is it true that the line is not attached to thetext block? -- " + Y
```

Resizing Windows {button OLE Automation,JI(`>concfull',`AUTOMATE.HLP',IDH_Bottom_Property_Application_Object);CW(`large')}

You can make the FlowCharter window larger or smaller, resizing it in any direction. With the mouse, you can resize horizontally and vertically at the same time from the corner of a window.

To resize the window with the mouse

1 Point to a border or corner and press and hold the left mouse button.

2 Drag the border or corner until the new border indicates the desired size.

3 Release the mouse button.

To resize the window with the keyboard

1 Press **Alt+Spacebar** and then **S** to choose the Size command.

2 Press an **Arrow** key to move the four-headed arrow to the border you want to move. To move to a corner, press the two **Arrow** keys that point to that corner.

3 Press the **Arrow** keys repeatedly to change the window to the desired size.

4 Press **Enter**. The active window changes to the new size.

To switch among application windows {button OLE Automation,JI(`>concfull',`AUTOMATE.HLP',IDH_Activate_Method_Application_Object);CW(`procedur')}

1 Press **Alt + Tab**.

A window shows icons for all the active applications.

2 Continue to hold down **Alt** and press **Tab** until the desired application is highlighted.

Restore Command (Control Menu) {button OLE Automation,JI(`>concfull',`AUTOMATE.HLP',IDH_Restore_Method_Application_Object);CW(`command')}

Use the Restore command in the Control menu to return the active window to its size and position before you chose the Maximize or Minimize command.

Clicking the Restore button in the upper-right corner of a maximized window is the same as choosing the Restore command.

Tip

- Double click the title bar to restore the window quickly.

Minimize Command (Control Menu) {button OLE Automation,JI(`>concfull',`AUTOMATE.HLP',IDH_Minimize_Method_Application_Object);CW(`command')}

Use the Minimize command in the Control menu to reduce the FlowCharter window to an icon.

Clicking the Minimize button in the upper-right corner of the window is the same as choosing the Minimize command.

Maximize Command (Control Menu) {button OLE Automation,JI(`>concfull',`AUTOMATE.HLP',IDH_Maximize_Method_Application_Object);CW(`command')}

Use the Maximize command in the Control menu to enlarge the active window to fill the available space. For example, a chart window expands to fill the FlowCharter window. The FlowCharter window expands to fill the entire screen.

Clicking the maximize button in the upper-right corner of the window is the same as choosing the Maximize command.

Tip

- Double click the title bar to maximize the window quickly.

StatusBarVisible Property {button Flow Equivalent,JI(`FLOW.HLP>command',`IDH_Status_Bar_Command');CW(`concfull')}

Usage *ApplicationObject*.**StatusBarVisible** = {True | False}

Description You use the **StatusBarVisible** property of the application object to find or set whether the status bar is displayed. The **StatusBarVisible** property is read/write.

Data Type Integer (Boolean)

Value True means the status bar is visible; False means it is not.

Flow Equivalent The **StatusBarVisible** property is equivalent to clicking Status Bar on the View menu.

{button Related Topics,PI(``,`IDH_RT_StatusBarVisible_Property')}

StatusBarVisible Property Example

This example uses the **StatusBarVisible Property** to turn the status bar on if it is off, and vice versa.

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC Visible
ABC.New                                              ' Create a new chart
If ABC.StatusBarVisible Then                         ' Is the status bar visible?
   ABC.MsgBox "Status Bar is visible"
   ABC.StatusBarVisible= False                       ' Toggle the status bar off
Else
   ABC.MsgBox "Status Bar is not visible"
   ABC.StatusBarVisible = True                       ' Toggle the status bar on
End If
```

ZoomWindowVisible Property

Usage *ApplicationObject*.**ZoomWindowVisible** = {True | False}

Description You use the **ZoomWindowVisible** property of the application object to find or set whether the QuickZoom window is visible. The **ZoomWindowVisible** property is read/write.

Data Type Integer (Boolean)

Value True means the QuickZoom window is visible; False means it is not.

Flow Equivalent The **ZoomWindowVisible** property is equivalent to clicking QuickZoom on the View menu.

{button Related Topics,PI(``,`IDH_RT_ZoomWindowVisible_Property')}

ZoomWindowVisible Property Example

This example uses the **ZoomWindowVisible** property to display the Quick Zoom window if it is not visible, and vice versa.

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC Visible
ABC.New                                              ' Create a new chart
If ABC.ZoomWindowVisible Then                        ' Is the zoom window visible?
   ABC.MsgBox "Quick Zoom Window is visible"
   ABC.ZoomWindowVisible = False                     ' Toggle the zoom window off
Else
   ABC.MsgBox "Quick Zoom Window is not visible"
   ABC.ZoomWindowVisible = True                      ' Toggle the zoom window on
End If
```

PrintPreview Method

Usage *ChartObject*.**PrintPreview**

Description You use the **PrintPreview** method of the chart object to display the print preview window

Flow Equivalent The **PrintPreview** method is equivalent to clicking Print Preview on the File menu.

{button Related Topics,PI(``,`IDH_RT_PrintPreview_Method')}

Example
Chart Object

PrintPreview Method Example

This example uses the **PrintPreview** method to display the print preview.

```
Dim ABC As Object, Chart As Object, NewObj As Object
Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the new chart object
   Set NewObj = Chart.DrawShape("Process")           ' Draw a new shape
   NewObj.Color = ABC.RED                            ' Change the objects color
   Chart.PrintPreview                                ' Activate the print preview
                                                     ' User must close manually
```

ReplaceText Method

Usage *ChartObject*.**ReplaceText** (*FindText*, *ReplacementText* [,*MatchCase*] [,*WholeWord*])

Description The **ReplaceText** method of the Chart object looks through the chart and replaces all occurrences of *FindText* with *ReplacementText* . The *MatchCase* and *WholeWord* parameters are optional; they are FALSE if not provided.

Data Type Long. *FindText* and *ReplacementText* are strings. *MatchCase* and *WholeWord* parameters are optional.

Value The number of replacements made

Flow Equivalent The **ReplaceText** method is equivalent to clicking Replace on the Edit menu and completing the dialog box.

{button Related Topics,PI(``,`IDH_RT_ReplaceText_Method')}

Example
Chart Object

ReplaceText Method Example

This example uses the **ReplaceText** method to replace four shapes with text blocks.

```
Dim ABC As Object, Chart As Object
Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC Visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the new chart object

Chart.DrawPositionX = 1                              ' Set next shape X position
Chart.DrawPositionY = 1                              ' Set next shape Y position
Chart.DrawSpacingX = 1                               ' Set X spacing for new shapes
Chart.DrawSpacingY = 1                               ' Set Y spacing for new shapes

For Shapes = 1 To 4
   Msg = "Shape " + Str(Shapes)
   Chart.DrawTextBlock Msg                           ' Draw four text blocks
Next Shapes

ABC.MsgBox "Notice Shape names"
NumReplaced = Chart.ReplaceText("Shape", "Text Block", False, False)     ' Perform Replace
Msg = "Replaced " + Str(NumReplaced) + " strings"    ' Report number of replacements
ABC.MsgBox Msg
```

ImportShape Method

Usage *ChartObject*.**ImportShape** (*PathName*)

Description You use the **ImportShape** method of the Chart object to import a graphics file into a new shape. A shape is created and the graphics file is inserted into it. The extension on the filename determines the graphics filter used. For example, if the filename has the extension .DRW, the Draw/Designer filter is used. Quotation marks should be used whenever long filenames or long pathnames are used.

Data Type Object

Value It returns the shape object that was drawn; it returns not valid if the import failed.

Flow Equivalent The **ImportShape** method is equivalent to clicking Import on the File menu in FlowCharter.

{button Related Topics,PI(``,`IDH_RT_ImportShape_Method')}

Example
Chart Object

ImportShape Method Example

This example uses the **ImportShape** method to import a DRW file and place it as a shape.

```
Dim ABC As Object, Chart As Object, NewObject As Object

Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the new chart object
Chart.DrawPositionX = 1                                ' Set next shape X position
Chart.DrawPositionY = 1                                ' Set next shape Y position
Chart.DrawSpacingX = 1                                 ' Set X spacing for new shapes
Chart.DrawSpacingY = 1                                 ' Set Y spacing for new shapes

Set NewObject = Chart.ImportShape("C:\TestObj.Drw")          ' Import shape and return object
If NewObject.Valid Then                                ' Verify import was successful
   ABC.MsgBox "Successfully imported shape"
Else
   ABC.MsgBox "Failed to import shape"
End If
```

Export Method

Usage *ChartObject*.**Export** (*PathName*)

Description You use the **Export** method of the Chart object to export the chart to a graphics file. The extension on *PathName* determines the filter used. The extension on the filename determines the graphics filter used. Quotation marks should be used whenever long filenames or long pathnames are used.

Data Type Integer (Boolean)

Value True means the export was successful; False means the export was not successful.

Flow Equivalent The **Export** method is equivalent to clicking Export on the File menu in FlowCharter.

{button Related Topics,PI(``,`IDH_RT_Export_Method')}

Example
Chart Object

Export Method Example

This example uses the **Export** method to export a shape to a DRW file.

```
    Dim ABC As Object, Chart As Object, NewObj As Object
Set ABC = CreateObject("ABCFlow.application")        ' Start ABC
ABC.Visible = True                                   ' Make ABC visible
ABC.New                                              ' Create a new chart
Set Chart = ABC.ActiveChart                          ' Get the new chart object
Set NewObj = Chart.DrawShape("Process")              ' Draw a new shape
NewObj.Color = ABC.RED                               ' Change the shape color
If Chart.Export("C:\Process.drw") Then               ' Attempt the export to a .DRW file
    ABC.MsgBox "Successfully exported chart."        ' Export returned true
Else
    ABC.MsgBox "Failed to export chart."             ' Export returned false
End If
```

SpaceEvenly Method

Usage *ChartObject*.**SpaceEvenly** (*Direction*)

Description The **SpaceEvenly** method lets you space currently selected objects evenly either across or down, based on their centers or edges.

Data Type Integer (Boolean). *Direction* is an integer.

Value True means the spacing was successful; False means the spacing was not successful. The **SpaceEvenly** method uses the values in the following table.

Direction	**Description**
0	Across, centers
1	Down, centers
2	Across, edges
3	Down, edges

Flow Equivalent The **SpaceEvenly** method is equivalent to selecting objects and then clicking a Space Evenly command on the Arrange menu or a Space Evenly button on the Arrange toolbar.

{button Related Topics,PI(``,`IDH_RT_SpaceEvenly_Method')}

Example
Chart Object

SpaceEvenly Method, Align Method, and MakeSameSize Method Example

This example uses the **MakeSameSize** method to make shapes the same size, then uses the **Align** method to align them along the left edges, and finally uses the **SpaceEvenly** method to space them evenly down based on the centers.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC visible
ABC.New                                            ' Create a new chart

Set Chart = ABC.ActiveChart                        ' Get the new chart object
For I = 1 To 4
   Chart.DrawPositionX = I * .6                    ' Position shapes in a staircase
   Chart.DrawPositionY = I * .6
   Set Obj1 = Chart.DrawShape("Process")           ' Draw shape
   Obj1.Height = Obj1.Height / I                   ' Decrease shape height
   Obj1.Width = Obj1.Width / I                     ' Decrease shape width
Next I

Chart.Select (0)                                   ' Select all objects
Chart.MakeSameSize (2)                             ' Make all same size
Chart.Align (0)                                    ' Align along left edges
Chart.SpaceEvenly (1)                              ' Space evenly down, centers

Chart.DeselectAll                                  ' Deselect all objects
```

Align Method

Usage *ChartObject*.**Align** (*By*)

Description You use the **Align** method to align selected objects.

Data Type Integer (Boolean)

Value True means the realignment was successful; False means the realignment was not successful. The **Align** method uses the values shown in the following table.

By	**Description**
0	Left edges
1	Centers (horizontal)
2	Right edges
3	Top edges
4	Middle (vertical)
5	Bottom edges

Flow Equivalent The **Align** method is equivalent to selecting the objects, and then clicking an Align command on the Arrange menu or an Align button on the Arrange toolbar.

{button Related Topics,PI(`',`IDH_RT_Align_Method')}

Example
Chart Object

MakeSameSize Method

Usage *ChartObject*.**MakeSameSize** (*AccordingTo*)

Description You use the **MakeSameSize** method of the Chart object to make the currently selected objects the same size.

Data Type Integer (Boolean). *AccordingTo* is an integer.

Value True means the resizing was successful; False means the resizing was not successful.

The following table describes the values for the **MakeSameSize** method.

Value	According To
0	Width
1	Height
2	Both
3	Fit to Text

Flow Equivalent The **MakeSameSize** method is equivalent to selecting objects, and then clicking a Make Same Size option on the Arrange menu or a Make Same Size button on the Arrange toolbar.

{button Related Topics,PI(`',`IDH_RT_MakeSameSize_Method')}

Example
Chart Object

FlippedVertical Property

Usage *ObjectObject*.**FlippedVertical** = {True | False}

Description You use the **FlippedVertical** property to find or set whether an object is flipped vertically (from top to bottom.) The **FlippedVertical** property is read/write.

Data Type Integer (Boolean)

Value True means the object is flipped vertically; False means it is not.

Flow Equivalent The **FlipVertical** property is equivalent to selecting the objects, and then clicking the Flip Vertical command from the Rotate/Flip submenu on the Arrange menu or the Flip Vertical button on the Arrange toolbar.

{button Related Topics,PI(``,`IDH_RT_FlippedVertical_Property')}

Example
Object Object

FlippedVertical, FlippedHorizontal, Rotation Properties Example

This example uses the **FlippedVertical**, **FlippedHorizontal**, and **Rotation** properties to change the orientation of three objects, and then returns them to their original orientation.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object, Obj3 As Object
Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                 ' Make ABC Visible
ABC.New                                            ' Create a new chart
Set Chart = ABC.ActiveChart                        ' Get the new chart object
Chart.DrawDirection = 2                            ' Draw the new shapes north to south
Chart.DrawPositionY = .5                           ' Initial Y position
Chart.DrawSpacingY = 1                             ' Vertical spacing between shapes
Set Obj1 = Chart.DrawShape("Merge")                ' Draw shapes
Set Obj2 = Chart.DrawShape("Display")
Set Obj3 = Chart.DrawShape("Collate")
ABC.MsgBox "Observe original orientations"         ' Show original orientations
Obj1.FlippedVertical = True                        ' Flip Merge shape vertically
Obj2.FlippedHorizontal = True                      ' Flip Display shape horizontally
Obj3.Rotation = 1                                  ' Rotate Collate shape 90 degrees
ABC.MsgBox "Observe new orientations"
Obj1.FlippedVertical = False                       ' Return to original orientations
Obj2.FlippedHorizontal = False
Obj3.Rotation = 3                                  ' Rotate 270 degrees
```

FlippedHorizontal Property

Usage *ObjectObject*.**FlippedHorizontal** = {True | False}

Description You use the **FlippedHorizontal** property to find or set whether an object is flipped vertically (from top to bottom.) The **FlippedHorizontal** property is read/write.

Data Type Integer (Boolean)

Value True means the object is flipped horizontally; False means it is not.

ABC Equivalent The **FlipHorizontal** property is equivalent to selecting the objects, and then clicking the Flip Horizontal command from the Rotate/Flip submenu on the Arrange menu or the Flip Horizontal button on the Arrange toolbar.

{button Related Topics,PI(``,`IDH_RT_FlippedHorizontal_Property')}

Example
Object Object

Rotation Property

Usage *ObjectObject*.**Rotation** = *Value*

Description You use the **Rotation** property to find or set the rotation of an object. All rotation is clockwise in 90 degree increments.

Data Type Integer

Value The following table describes the values for the **Rotation** property.

Value	Amount of Rotation
0	0
1	90
2	180
3	270

Flow Equivalent The **Rotation** property is equivalent to selecting the objects, and then clicking the Rotate Right command on the Rotate/Flip submenu on the Arrange menu or the Rotate button on the Arrange toolbar an approprioate number of times.

{button Related Topics,PI(``,`IDH_RT_Rotation_Property')}

Example
Object Object

ApplyDefaults Method

Usage *ObjectObject*.**ApplyDefaults**

Description You first use *ChartObject*.**SetDefaults** (*ObjectObject*) to define the default styling for shapes, lines, and textblocks. Then you use the **ApplyDefaults** property to apply the chart's default styling to this object.

Flow Equivalent The ApplyDefaults method is equivalent to using the Preset Styles toolbar.

{button Related Topics,PI(``,`IDH_RT_ApplyDefaults_Method')}

ApplyDefaults Method Example

This example uses the **ApplyDefaults** method to apply the styles of one shape to two other shapes.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object, Obj3 As Object
Set ABC = CreateObject("ABCFlow.application")          ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                ' Create a new chart
Set Chart = ABC.ActiveChart                            ' Get the new chart object
Chart.DrawSpacingX = 1.25                              ' Set horizontal spacing
Set Obj1 = Chart.DrawShape("Process")                  ' Draw shapes
Set Obj2 = Chart.DrawShape("Connector")
Set Obj3 = Chart.DrawShape("Decision")
Obj1.Color = ABC.RED                                   ' Change Process shape color to red
Obj1.Shape.ShadowStyle = 4                             ' Add shadow to Process shape
Chart.SetDefaults Obj1.Shape                           ' Copy Process shape format information
Obj2.ApplyDefaults                                     ' Apply Process format to Connector
Obj3.ApplyDefaults                                     ' Apply Process format to Decision shape
```

Routing Property

Usage *Line_Object*.**Routing** = *LineRoutingValue*

Description You use the **Routing** property to change the type of routing for existing lines. The **Routing** property is read/write.

Data Type Integer

Value The following table describes the values for the **Routing** property.

Value	Type of Line
0	Direct
1	Right angle
2	Curved
3	Organization chart
4	Cause-and-effect

Flow Equivalent None

{button Related Topics,PI(``,`IDH_RT_Routing_Property')}

Example
Line Object

Routing, CrossoverSize, CrossoverStyle Properties Example

This example uses the **Routing**, **CrossoverSize**, and **CrossoverStyle** properties of lines.

```
 Dim ABC As Object, Chart As Object
   Dim Obj1 As Object, Obj2 As Object, Obj3 As Object, Obj4 As Object
   Dim Obj5 As Object, Obj6 As Object, Obj7 As Object, Obj8 As Object
   Dim Line1 As Object, Line2 As Object, Line3 As Object, Line4 As Object, Line5 As
Object

   Set ABC = CreateObject("ABCFlow.application")     ' Start ABC
   ABC.Visible = True                                ' Make ABC visible
   Set Chart = ABC.New                               ' Make a new chart

   Chart.DrawPositionX = 2                           ' Set position
   Chart.DrawPositionY = 1
   Set Obj1 = Chart.DrawShape("Process")             ' Draw the shape
   Chart.DrawPositionX = 3                           ' Set position
   Chart.DrawPositionY = 6
   Set Obj2 = Chart.DrawShape("Document")            ' Draw the shape
   Set Line1 = Chart.DrawLine(Obj1, Obj2, 2, 0)      ' Draw a line from Obj1 to Obj2
   Line1.Line.Routing = 1                            ' Set line routing to "Right-
angle"
   Line1.Line.CrossOverStyle = 2                     ' Set line crossover style to
"Solid lines"

   Chart.DrawPositionX = 1                           ' Set position
   Chart.DrawPositionY = 2
   Set Obj3 = Chart.DrawShape("Decision")            ' Draw the shape
   Chart.DrawPositionX = 6                           ' Set position
   Chart.DrawPositionY = 3
   Set Obj4 = Chart.DrawShape("Terminal")            ' Draw the shape
   Set Line2 = Chart.DrawLine(Obj3, Obj4, 1, 3)      ' Draw a line from Obj3 to Obj4
   Line2.Line.Routing = 0                            ' Set routing to "Direct-line"
   Line2.Line.CrossOverStyle = 0                     ' Set line crossover style to
"Bunny hops"
   Line2.Line.CrossOverSize = 0                      ' Set line crossover size to
"Small", 0

   Chart.DrawPositionX = 4                           ' Set position
   Chart.DrawPositionY = 1
   Set Obj5 = Chart.DrawShape("Connector")           ' Draw the shape
   Chart.DrawPositionX = 5                           ' Set position
   Chart.DrawPositionY = 6
   Set Obj6 = Chart.DrawShape("Input/Output")        ' Draw the shape
   Set Line3 = Chart.DrawLine(Obj5, Obj6, 2, 0)      ' Draw a line from Obj5 to Obj6
   Line3.Line.Routing = 2                            ' Set line routing to "Curve"
   Line3.Line.CrossOverStyle = 1                     ' Set line crossover style to
```

```
"Broken lines"

   Chart.DrawPositionX = 1                          ' Set position
   Chart.DrawPositionY = 4
   Set Obj7 = Chart.DrawShape("Alternate Process")  ' Draw the shape
   Chart.DrawPositionX = 6                          ' Set position
   Chart.DrawPositionY = 5
   Set Obj8 = Chart.DrawShape("Process")            ' Draw the shape
   Set Line4 = Chart.DrawLine(Obj7, Obj8, 1, 3)     ' Draw a line from Obj7 to Obj8
   Line4.Line.Routing = 1                           ' Set line routing to "Right-
angle"
   Line4.Line.CrossOverStyle = 0                    ' Set line crossover style to
"Bunny hops"
   Line4.Line.CrossOverSize = 2                     ' Set line crossover size to
"Large", 2

   Set Line5 = Chart.DrawLine(Obj5, Obj7, 3, 0)     ' Draw a line from Obj5 to Obj7
   Line5.Line.Routing = 0                           ' Set line routing to "Direct-
line"
   Line5.Line.CrossOverStyle = 2                    ' Set line crossover style to
"Solid lines"
```

CrossoverSize Property

Usage *Line_Object*.**CrossoverSize** = *RelativeSize*

Description The **CrossoverSize** property lets you find or set the size of the crossover when one line crosses of another for a specific line. The setting applies to bunny hops and broken lines, but has no effect when the crossover style is solid lines. (See the CrossoverStyle property for information on the available styles.) The **CrossoverSize** property is read/write.

Data Type Integer

Value The values for the relative sizes for bunny hop crossovers are in the following table. The same relative sizes apply when the style is broken lines.

RelativeSize	**Description**
0	▪ Small
1	▪ Medium
2	▪ Large

Value The relative size of the crossover when one line crosses another

Flow Equivalent The **CrossoverSize** property is equivalent to selecting one or more lines, clicking Ends on the Format menu and setting the Crossovers Size.

{button Related Topics,PI(``,`IDH_RT_CrossoverSize_Property')}

Example
Line Object

CrossoverStyle Property

Usage *Line_Object*.**CrossoverStyle** = *Style*

Description The **CrossoverStyle** property lets you find or set the style of the crossover when one line crosses another for a specific line. The **CrossoverStyle** property is read/write.

Data Type Integer

Value The values for the styles are in the following table.

Style	**Description**
0	▪ Solid lines
1	▪ Bunny hops
2	▪ Broken lines

Value The style when one line crosses another

Flow Equivalent The **CrossoverStyle** property is equivalent to selecting a line, clicking Ends on the Format menu and setting the Crossovers Style.

{button Related Topics,PI(`',`IDH_RT_CrossoverStyle_Property')}

Example
Line Object

IndexWindowHandle Property

Description This command was in ABC FlowCharter 6.0, but is not appropriate in FlowCharter 7. It is not invalid; it is ignored.

IndexVisible Property

Description This command was in ABC FlowCharter 6.0, but is not appropriate in FlowCharter 7. It is not invalid; it is ignored.

LaunchFlags Property

Description This command was in ABC FlowCharter 4.0, but is not appropriate in FlowCharter 7. It is not invalid; it is ignored.

LaunchStartDir Property

Description This command was in ABC FlowCharter 4.0, but is not appropriate in FlowCharter 7. It is not invalid; it is ignored.

RestorePicture Method (OLE Object)

Description This command was in ABC FlowCharter 4.0, but is not appropriate in FlowCharter 7. It is not invalid; it is ignored.

ShapeSizing Property

Description This command was in ABC FlowCharter 4.0, but is not appropriate in FlowCharter 7. It is not invalid; it is ignored.

LaunchIndicator Property

Description This command was in ABC FlowCharter 4.0, but is not appropriate in FlowCharter 7. It calls **LinkIndicator** property.

{button Related Topics,PI(`',`IDH_RT_LaunchIndicator_Property')}

LinkIndicator Property

LaunchShadow Property

Description This command was in ABC FlowCharter 6.0, but is not appropriate in FlowCharter 7. It calls **LinkShadow** property.

{button Related Topics,PI(`',`IDH_RT_LaunchShadow_Property')}

LinkShadow Property