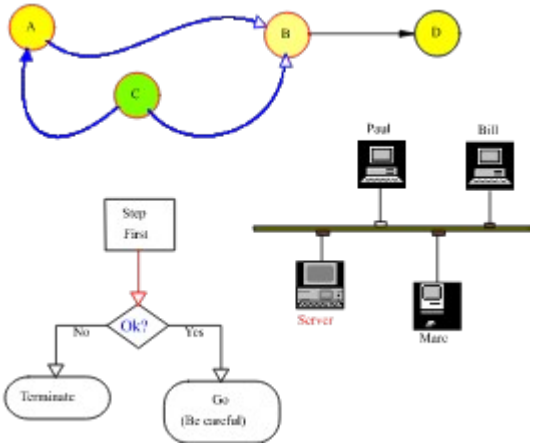
AddFlow ActiveX version 1.01



License Agreement

ADDFLOW ACTIVEX CONTROL LICENSE AGREEMENT
IMPORTANT: PLEASE READ CAREFULLY

This is a legal License Agreement between you and Lassalle Technologies. Carefully read all the terms and conditions of this agreement prior to using the AddFlow ActiveX control. By using this product, you are agreeing to be bound by the following terms and conditions:

1.OWNERSHIP

The AddFlow ActiveX Control program (the "SOFTWARE") and accompanying written materials are owned by Lassalle Technologies (the "LICENSOR") and are protected by copyright law and international treaties.

2. LICENSE FOR EVALUATION VERSION

When you install a copy of the SOFTWARE without providing the appropriate license number (also called "serial number"), the SOFTWARE will be installed as an Evaluation Version. When you use an Evaluation version, you will be prompted by a dialog box explaining that you are using an Evaluation Version of the SOFTWARE.

If you generate an executable application with an Evaluation Version, then any attempt to use executable application will display a dialog box explaining that it has been generated without license file and that the SOFTWARE will not work.

You may use the Evaluation Version of the SOFTWARE for up to **30 days** in your design environment for evaluation purposes only. You may copy and distribute it freely as long as all the files in the package, including the demo programs are distributed with it and no changes or additions of any kind are made to the original package.

3.FULL LICENSE

To get a full license of the SOFTWARE that allows to generate executable applications that work correctly, you have to order a full license of the SOFWARE.

If you pay the license fee and if you agree to the terms of this agreement, the LICENSOR grants to you a license to:

- use the SOFTWARE on a single terminal connected to a single computer by one person at a time for the purpose of development and creation of executable applications.

- have the royalty-free right to distribute executable applications that use the SOFTWARE as a runtime component.
- have the right to a technical support for a period of 12 months from the date of purchase of the license.
- have the right,during a period of 12 months, to download from LICENSOR's Internet site all maintenance releases of the SOFWARE, without having to pay any fees. Maintenance release mean bug corrections and minor improvements.

The SOFTWARE is intended to be sold to individuals. This is a single user license. Any company wishing to purchase it would need to purchase one copy for each person using it. You may not network the SOFTWARE or otherwise use it on more than one computer or computer terminal at the same time.
You may make one copy of the SOFTWARE for backup purposes only. Making copies for any other purpose violates international copyright laws. You can just copy the ActiveX file as a runtime component of one of your applications.
You are not allowed to distribute the license file (.lic) with any application that you distribute.
You agree to treat the license number as strictly confidential except to the LICENSOR or your authorized AddFlow distributor.

4.OTHER RESTRICTIONS

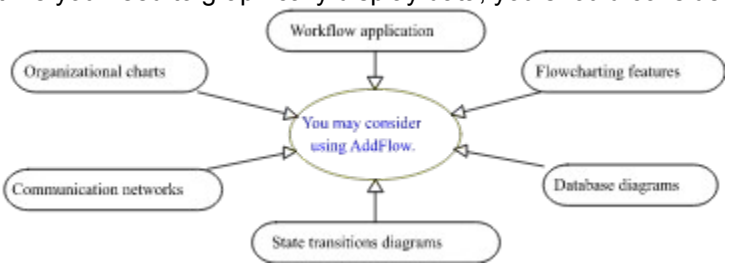
It is stricly prohibited to reverse engineer, decompile, or disassemble the SOFTWARE. You are not allowed to rent or lease the SOFTWARE, but you may transfer it and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement.
You are not allowed to expose directly or indirectly the properties and methods of the SOFTWARE.
You may not repackage the SOFTWARE for sale as a competing product.

5.DISCLAIMER OF WARRANTY

THIS SOFTWARE AND THE ACCOMPANYING FILES ARE SOLD "AS IS" WITHOUT WARRANTY OF ANY KIND EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
Good data processing procedure dictates that any program be thoroughly tested with non-critical data before relying on it. The user must assume the entire risk of using the program.

Introduction

AddFlow is a 32 bits ActiveX control that lets you quickly build flowchart-enabled applications. Each time you need to graphically display data, you should consider using AddFlow.



AddFlow provides two ways to build diagrams:
1) interactive way (with the mouse): see **Quick Start** in order to know how to interactively draw a diagram.
2) programmatic way with the AddFlow API (Application Programmatic Interface). This API is very easy to use since it is based upon OLE automation and collections. It is a set of properties, events and methods that allow to manipulate Node and Link objects and to customize your application. The **Tutorial** provides a full description of this API.

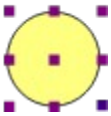
Quick Start under Visual Basic

- **Add the AddFlow ActiveX** to your project by selecting "Custom Control..." from Visual Basic's "Tools" menu.

- **Drag an AddFlow control** from the toolbox to your form. If you have not a license file, an "About" dialog box appears and you have to click Ok.

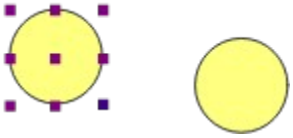
- **Launch** the program by selecting "Start" from the "Run" menu (or do F5).

- **Draw a node**: bring the mouse cursor into the control, press the left button, move the mouse and release the left button. You have created an elliptic node. This node is selected: that's why 9 handles (little squares) are displayed.

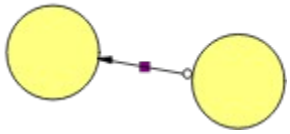


The handle at the center of the node is used to draw a link. The 8 others allow to **resize the node**. If you want to **move the node** you bring the mouse cursor into the node, press the left button, move the mouse and release the left button.

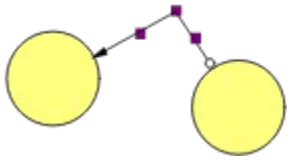
- **Draw a second node**...(same method)



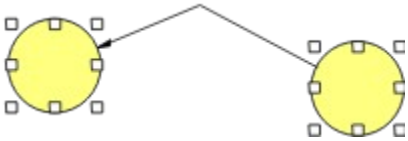
- **Draw a link**: bring the mouse cursor into the handle at the center of the selected node, press the left button, move the mouse towards the other node. When the mouse cursor is into the other node, release the left button. The link has been created. And it is selected since a handle is displayed at the center of this link.



- **You may stretch this link**: bring the mouse cursor into the link handle, press the left button, move the mouse and release the left button. You have created a new link segment. It has 3 handles allowing you to add or remove segments. (The handle at the intersection of two segments allows you to remove a segment : you move it with the mouse so that the two segments are aligned and when these two segments are approximately aligned, release the left button).



- You can select several nodes by clicking them with the mouse and simultaneously pressing the shift or control key.



There is another way to perform multiselection : see SelectMode property.

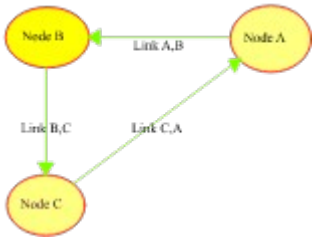
- **Now, you may return to the Visual Basic design-time mode** in order to change control properties. For instance you may change the default node filling color with FillColor property, the default node shape

(Shape property) or the default drawing color (DrawColor property).

Tutorial

AddFlow diagrams

An AddFlow diagram is a set of objects that can be linked each other. Following figure shows a typical example of such a diagram.



The objects are called nodes and the lines that connect objects are called links. Nodes and links are AddFlow objects and those objects have many attributes that can be changed: colors, text, drawing styles, fonts, etc...

Nodes may be moved or resized whereas links may be stretched into several segments.

A link cannot exist without its origin and destination nodes. If one of these two nodes is removed, the link is also removed. For instance, if you remove the node C , the diagram becomes the following:

Node objects

You may create a Node object programmaticaly with the Add method of the nodes collection object. This method allows to specify the position and the size of the node that is created.

The node object type name is **afNode**.

When it is created, the node receives default values for the following attributes: colors (Drawing color, text color, filling color), drawing pen style and width, shape, text alignment, autosize mode, transparency, font and hidden flag.

Then you may alter the.position, the size and all default attributes of the node. You can also associate a text, a picture or user data to this node. You have to do it since you cannot define default values for those attributes (See Node Properties).

Example

Link objects

A Link object allows to link two nodes. It is a line that leaves the origin node and comes to the destination node. You may create a link programmaticaly with the Add method of the links collection object. When creating a link, you have to define its origin and its destination node.

The link object type name is **afLink**.

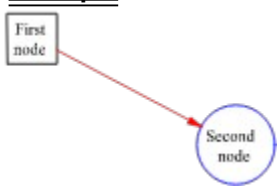
When it is created, the link receives default values for the following attributes: colors (Drawing color, text color), drawing pen style and width, arrow shapes, font and hidden flag.

Then you may alter the default attributes of the link. You can also associate a text or user data to this link. You have to do it since you cannot define default values for those attributes (See Link Properties).

A link may have several segments but the first segment is always directed towards the center of the origin node and the last segment is always directed towards the center of the destination node (See ExtraPoints, PointOrg, and PointDst properties).

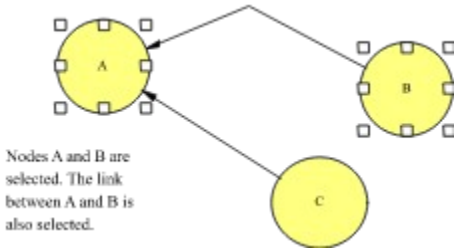
However, this behaviour may be changed with the Rigid property that allows to define ownership between nodes. Instead of creating a link between node A and node B, you create a link between A and a little node C inside B and owned by B. C may be inactive and hidden: it is used as a kind of pin or stub inside B.

Example



Selection

You can select a node either interactively by clicking it with the mouse either programmaticaly with SelectedNode property. You can also select several nodes with the mouse if multiselection is allowed (in such a case MultiSel and SelectMode properties are true).



You can select a link interactively by clicking it (see the note at the end of this topic) with the mouse either programmaticaly with SelectedLink property.
You can also know what object is under the mouse with PointedNode and PointedLink properties that return the reference of the node or the link under the mouse. If several objects are under the mouse, the returned object is the one that is at the top of the Z-order list. You may change this order with the Zorder property.

Note on selecting a link with the mouse:

If the link is made of one or several segments, then if you want to select it with the mouse, you have just to click near one of its segments. If the link is a Bezier curve (see LinkStyle property), then you have just to click near the curve.

Navigation

The 5 properties described here allow to access and manipulate every nodes and links of a diagram.

The Nodes property allows you to manipulate each node of the AddFlow diagram.

A Node object has two properties that allow to manipulate its links : InLinks and OutLinks.

You can retrieve the origin and the destination node of a link with Org and Dst properties.

Example 1

```
' Make all nodes rectangular
Dim node as afNode
For Each node In AddFlow1.Nodes  ' Nodes enumeration
  node.Shape = afRectangle
Next
```

Example 2

```
' Make all links leaving the selected node red.
Dim link as afLink
If AddFlow1.SelectedNode Is Nothing then
  ' Do Nothing
Else
  For Each link In AddFlow1.SelectedNode.OutLinks
    link.DrawColor = RGB(255, 0, 0)
  Next
End If
```

Example 3

```
' Make all links red.
Dim node as afNode, link as afLink
For Each node In AddFlow1.Nodes ' Nodes enumeration
  ' For each node, perform a « leaving link » enumeration.
  For Each link In node.OutLinks
    link.DrawColor = RGB(255, 0, 0)
  Next
Next
```

Example 4

```
' Make all destination nodes of the selected node Transparent.
Dim link as afLink
If AddFlow1.SelectedNode Is Nothing then
  ' Do Nothing
Else
  For Each link In AddFlow1.SelectedNode.OutLinks
    link.Dst.Transparent = True
  Next
End If
```

Capabilities

Following properties allow to set capabilities for an AddFlow control and therefore to customize it. For instance, if you wish to allow only one link between two nodes, you have just to unset the **CanMultiLink** property.

AutoScroll	determines whether automatic scrolling is allowed or not.
CanDrawNode	determines whether interactive creation of Node objects is allowed or not.
CanDrawLink	determines whether interactive creation of Link objects is allowed or not.
CanMoveNode	determines whether interactive dragging of Node objects is allowed or not.
CanSizeNode	determines whether interaction resizing of Node objects is allowed or not.
CanStretchLink	determines whether interactive stretching of Link objects is allowed or not.
CanMultiLink	determines whether you can create several links between two nodes.
DisplayHandles	determines whether the handles used for selection are displayed or not.
MultiSel	determines whether multiselection of Node objects is allowed or not.
ReadOnly	determines whether user interaction is allowed or not.
Repaint	not really a "capability". This property aims at improving performance: setting it to False allows to increase the speed when creating programmaticaly a whole diagram.
ScrollBars	allows to add scrollbars for the control.
ShowGrid	determines whether the grid is displayed or not.
xGrid	returns/sets the horizontal grid.
xZoom	returns/sets the horizontal zooming factor.
yGrid	returns/sets the vertical grid.
yZoom	returns/sets the vertical zooming factor.

Metafile support

AddFlow offers a Windows standard metafile support. SaveImage method allows to save the diagram on disk as a metafile. This method allows also to copy the diagram onto the clipboard. Then you may paste it in Window Write, in PaintBrush, Excel, Winword, WordPerfect, in a VB picture, etc... And the result can be resized. For instance, you may paste the metafile in a Winword document, double-click on the picture, adjust the margins so that there's room for other drawing objects, use the drawing tools to draw some lines, circles, etc, close the picture, select it, copy it to the clipboard, etc...

All the diagrams in this current help file have been made with AddFlow and copied using the SaveImage method.

Installation

Floppy disk Install

To install AddFlow on your system, follow these steps :

1) Make sure Windows NT or Windows 95 is running.

2) Insert the disk 1 into drive A or B.

3) Launch the setup program “A:\SETUP” or “B:\SETUP” depending on where you inserted the setup disk.

4) Follow the instructions given in the installation program.

Evaluation version Install

Generally, this demo version is packaged in a single zip file: ADDFLO.ZIP. You will find such a file on Compuserve or Internet or on some CD-ROMS provided by authorized AddFlow distributors. In such a case, you have just to unzip this file, then launch the setup program and follow the instructions given in the installation program.

Notes

1) When installing the product and if it is not an evaluation version, a license file **ADDFLOW.LIC** is copied in the same directory as the ActiveX file ADDFLOW.OCX.

2) The setup program should register the ActiveX file. Anyway you can do it yourself with the regsrv32 tool program. The command is the following:

REGSVR32 /U ADDFLOW.OCX.

3) AddFlow has been compiled with Microsoft VC++ 5.0. To work correctly, it requires that many shared DLLs from Microsoft to be present on your computer. Those DLLs are the following:

MFC42.DLL
MSVCRT.DLL
OLEPRO32.DLL

4) When you create and distribute applications that use AddFlow, you could install the ActiveX file in the same directory as the application.

5) **You are not allowed to distribute the license file ADDFLOW.LIC with any application that you distribute.**

Short description

AddFlow is a 32 bits ActiveX control that lets you quickly build flowchart-enabled applications. Each time you need to graphically display data, you should consider using AddFlow. Many features: distinct colors, fonts, shapes, styles, pictures, text, etc... for each item. Nodes stay connected when moved. Navigation and user data allowed, programmatic or interactive drawings, metafiles, zoom, Bezier curves. Easy, fast and small: only 120 Ko of code. Runtime royalty-free.

Frequently Asked Questions

How to associate data to a node or a link ?

Following properties allow to associate user data to a node or a link:

UserData: this property allows to associate a numeric (long integer) value to a node or a link.

Tag: this property allows to associate a string to a node or a link. This string may be a comment, a help tip, an URL or anything you need for your application.

Marked: this property allows to associate a flag to a node or a link. This flag may be used with the **DeleteMarked** method. But you can also use this flag for your own purpose.

How to save a diagram ?

Saving an AddFlow diagram is under the responsability of the VB application that uses an AddFlow control.

You may see **Editor** sample that is supplied with the package in order to see a way to save and load an AddFlow diagram. It is just an example but you may consider it as a starting point to write your own Saving/Loading procedures.

How to print a diagram ?

Printing is under the responsability of the VB program that uses AddFlow. To print an AddFlow diagram today, you have 4 methods:

1) copy it to clipboard via Metafile format (SaveImage method). Than paste it in Winword or Excel or PaintBrush. Then you may resize it, change it and print it.

2) save it to disk as a metafile (SaveImage method) then the problem is to print a metafile.

3) use VB code. You have to retrieve attributes of each Node or Link object and use VB code to draw those objects. It is not very easy but feasable.

4) use the **PrintForm** method. This method allows to print the visible part of the diagram. But you can use it in conjunction with xScroll and yScroll properties in order to print the whole diagram.

How to detect a user action (node or link creation, node resizing, etc...) ?

This can be done when receiving the **MouseUp** event. You have to use the LastUserAction method. See **Example**

This allows to avoid what we call the "events proliferation syndrom". All AddFlow events are standards.

I am using your previous flowcharter OCX: EasyNet. Why should I change to use AddFlow ?

If you are developping a 16 bits application then you should keep on using EasyNet since AddFlow exists only in 32 bits flavor.

Now, if you are developing a 32 bits application then you should adopt AddFlow without any doubt for the following reasons:

1) AddFlow is easier to use: its API (Application programming Interface) has been reduced dramatically without losing any features. For instance, EasyNet has approximately 70 methods whereas AddFlow has only 11 methods!

The reason for this is OLE automation: this mechanism is one of the most powerful features of OLE. It allows objects to expose methods and properties to other objects and applications. Such objects are called automation servers. Under AddFlow, nodes and links are automation servers. This mechanism used in conjunction with collections allows to write very clear and elegant code like the following:

```
' Make all nodes rectangular
Dim node as afNode
```

```
For Each node In AddFlow1.Nodes  ' Nodes enumeration
  node.Shape = afRectangle
Next
```

2) AddFlow is faster. You may verify this when creating many items (for instance 20000 nodes).

3) AddFlow has new interesting features like Bezier curves (for displaying links), rigid links, or bidirectional links.

You should carefully see the **readme** file in order to know what are the differences between EasyNet and AddFlow.

Why should I use AddFlow instead of another product?

AddFlow is RUNTIME ROYALTY FREE product. It is not expensive and you only need to purchase one license per developper.

AddFlow is easy, fast and small (only 120 K of code).

What kind of policy you have regarding access to the source code? Is it possible to get the sources for maintenance purposes?

We haven't decided as of yet to sell any of the source code. However, we will take it under consideration for the future.

Reference Guide

The AddFlow API offers a set of properties, events and methods that may apply to a whole AddFlow control object, a node object, a link object, a collection of nodes, a collection of links or a collection of extra points of a link

AddFlow control

Some properties allow to customize the control according to your needs.
Some properties allow to set default values to Node and Link attributes. For instance, the FillColor property of the AddFlow control allows to set the default filling color of nodes. When a node is created, it receives this default filling color. Then, you may change the filling color of this node by using the FillColor of the Node object.

Properties

Events

Methods

Constants

Node and Link objects

Some properties apply only to Node objects (for instance FillColor, Picture, Shape, etc, ...).
Some properties apply only to Link objects (for instance, ArrowOrg, ArrowDst, Org, Dst, etc...).
Some properties apply to both types of object (for instance, DrawColor, Text, Tag, etc...).
See following topics to know what you can do with each object type.

Node Properties

Link Properties

Collections

An AddFlow diagram is a collection of Node objects. Each Node object allows to access the collection of the Link objects that leave the node and the collection of the Link objects that come to the Node. Each Link object allows to access the collection of its extra points.Therefore we have 3 types of collection objects:

Nodes collection

Links collection

LinkPoints collection

Properties

All the properties are listed below. Properties that apply only to AddFlow/ActiveX, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties.

(About)	displays the AddFlow About dialog box.
Alignment	returns/sets the default Node object text alignment style.
ArrowOrg	returns/sets the default Link object origin arrow shape.
ArrowDst	returns/sets the default Link object destination arrow shape.
AutoSize	returns/sets the default Node object AutoSize style.
AutoScroll	determines whether automatic scrolling is allowed or not.
BackColor	returns/sets the control background color
BackPicture	allows to display a bitmap inside the control
BorderStyle	returns/sets the border style of the control.
CanDrawNode	determines whether interactive creation of Node objects is allowed or not.
CanDrawLink	determines whether interactive creation of Link objects is allowed or not.
CanMoveNode	determines whether interactive dragging of Node objects is allowed or not.
CanSizeNode	determines whether interaction resizing of Node objects is allowed or not.
CanStretchLink	determines whether interactive stretching of Link objects is allowed or not.
CanMultiLink	determines whether you can create several links between two nodes.
DisplayHandles	determines whether the handles used for selection are displayed or not.
DragIcon	returns/sets the icon used to be displayed as a pointer in a drag-and-drop operation.
DragMode	returns/sets a value that determines whether manual or automatic drag mode is used.
DrawColor	returns/sets the default pen color used to draw objects (Node or Link).
DrawStyle	returns/sets the default pen style used to draw objects (Node or Link).
DrawWidth	returns/sets the default pen width used to draw objects (Node or Link).
Enabled	returns/sets a value that determines whether an object is enabled or not.
FillColor	returns/sets the default color used to fill Node objects.
Font	returns/sets the default font used to display text.
ForeColor	returns/sets the default foreground color used to display text.
Height	returns/sets the height of an object.
HelpContextId	specifies the default Help file context ID for an object.
Hidden	determines whether objects (Node or Link) are by default visible or hidden.
Hwnd	returns a window handle to the control
Index	returns/sets the number identifying a control in a control array.
LastUserAction	returns the last interactive user action.
Left	returns/sets the distance between the internal left edge of an object and the left

	edge of its container.
LinkStyle	returns/sets the default style (polyline, bezier) used to draw Link objects.
MouseIcon	sets a custom mouse icon
MousePointer	returns/sets the type of mouse pointer displayed when over control.
MultiSel	determines whether multiselection of Node objects is allowed or not.
Nodes	returns a reference to the collection of all Node objects of the diagram.
Parent	returns the form on which an object is located.
PointedArea	returns the type of the area pointed by the mouse.
PointedLink	returns a reference to the Link object pointed by the mouse.
PointedNode	returns a reference to the Node object pointed by the mouse.
ReadOnly	determines whether user interaction is allowed or not.
Repaint	determines whether repainting the control is allowed or not.
Rigid	determines whether Link objects are by default rigid or not.
ScrollBars	allows to add scrollbars for the control.
SelectedLink	returns/sets a value which determines if a Link object is selected..
SelectedNode	returns/sets a value which determines if a Node object is selected.
SelectMode	determines whether selection mode is set instead of drawing mode.
Shape	returns/sets the default Node object shape.
ShowGrid	determines whether the grid is displayed or not.
TabIndex	returns/sets the tab order of an object within its parent form.
TabStop	returns/sets a value indicating whether a user can use the TAB key to give the focus to an object.
Tag	stores any extra data needed for your program.
Top	returns/sets the distance between the internal top edge of an object and the top edge of its container.
Transparent	determines whether Node objects are transparent or not.
Visible	returns/sets a value that determines whether an object is visible or hidden.
Width	returns/sets the width of an object.
xGrid	returns/sets the horizontal grid.
xScroll	returns/sets the horizontal scrolling offset.
xZoom	returns/sets the horizontal zooming factor.
yGrid	returns/sets the vertical grid.
yScroll	returns/sets the vertical scrolling offset.
yZoom	returns/sets the vertical zooming factor.

Events

All AddFlow events are standards and are listed below. See the Visual Basic *Language Reference* or online Help for documentation of the those events.

Click

DblClick
DragDrop
DragOver
GotFocus
KeyDown
KeyPress
KeyUp
LostFocus
MouseDown
MouseMove
MouseUp

Custom Methods

DeleteSel	Removes all selected nodes and their associated links.
DeleteMarked	Removes all marked nodes (and their associated links) and all marked links.
DoClick	Fires a click event.
GetVersion	Returns control version.
IsChanged	Returns the change flag.
IsSelChanged	Returns the selection change flag.
Refresh	Force a complete repaint of the control.
SaveImage	Save the diagram in metafile format.
SelectAll	Select all nodes.
SetChangedFlag	Reset the change flag.
SetSelChangedFlag	Reset the selection change flag.

Constants

Alignment Constants

Constants	Value	Description
afLeftJustifyTOP	0	Left Justify - TOP
afLeftJustifyMIDDLE	1	Left Justify - MIDDLE
afLeftJustifyBOTTOM	2	Left Justify - BOTTOM
afRightJustifyTOP	3	Right Justify - TOP
afRightJustifyMIDDLE	4	Right Justify - MIDDLE
afRightJustifyBOTTOM	5	Right Justify - BOTTOM
afCenterTOP	6	Center - TOP
afCenterMIDDLE	7	Center - MIDDLE
afCenterBOTTOM	8	Center - BOTTOM

Arrow Constants

Constants	Value	Description
afNoArrow	0	None
afFilledCircle	1	Filled Circle
afEmptyCircle	2	Empty Circle
afFilledArrow15	3	Filled Arrow 15
afEmptyArrow15	4	Empty Arrow 15
afFilledArrow30	5	Filled Arrow 30
afEmptyArrow30	6	Empty Arrow 30
afFilledArrow45	7	Filled Arrow 45
afEmptyArrow45	8	Empty Arrow 45

AutoSize Constants

Constants	Value	Description
afNoAutoSize	0	None
afAdjustPictureSizetoNode	1	Adjust Picture Size to Node
afAdjustNodeSizetoPicture	2	Adjust Node Size to Picture

BorderStyleConstants

Constants	Value	Description
afNoBorder	0	None
afFixedSingle	1	Fixed Single

DrawStyle Constants

Constants	Value	Description
afSolid	0	Solid
afDash	1	Dash

afDot	2	Dot
afDashDot	3	Dash-Dot
afDashDotDot	4	Dash-Dot-Dot
afTransparent	5	Transparent
afInsideSolid	6	Inside Solid

LinkStyle constants

Constants	Value	Description
afPolyline	0	Polyline
afBezier	1	Bezier curve

MousePointer Constants

Constants	Value	Description
afDefault	0	Default
afArrow	1	Arrow
afCross	2	Cross
afIBeam	3	I-Beam
afIcon	4	Icon
afSize	5	Size
afSizeNESW	6	Size NE SW
afSizeNS	7	Size N S
afSizeNWSE	8	Size NW SE
afSizeEW	9	Size W E
afUpArrow	10	Up Arrow
afHourglass	11	Hourglass
afNoDrop	12	No Drop
afArrowHourglass	13	Arrow and Hourglass
afArrowQuestion	14	Arrow and Question mark
afSizeAll	15	Size all
afCustom	99	Custom

PointedArea Constants

Constants	Value	Description
afNWSEsizeHandle	0	NW-SE Size handle area
afNSsizeHandle	1	North-South Size handle area
afNESWsizeHandle	2	NE-SW Size handle area
afWEsizeHandle	3	West-East Size handle area
afStretchingHandle	4	Stretching handle area
afLinkingHandle	5	Linking handle area
afNodeArea	6	Node area
afOutSide	7	Outside area
afLinkArea	8	Link area

ScrollBars Constants

Constants	Value	Description
afNoScroll	0	None
afHorizontal	1	Horizontal
afHorizontal	2	Vertical
afBoth	3	Both

Shape Constants

Constants	Value	Description
afEllipse	0	Ellipse
afRectangle	1	Rectangle
afRoundRect	2	RoundRect
afDiamond	3	Diamond
afNorthTriangle	4	North Triangle
afSouthTriangle	5	South Triangle
afEastTriangle	6	East Triangle
afWestTriangle	7	West Triangle
afHexagon	8	Hexagon

UserAction Constants

Constants	Value	Description
afNoUserAction	0	None
afNodeCreation	1	Node creation action
afLinkCreation	2	Link creation action
afNodeDragging	3	Node dragging action
afNodeResizing	4	Node resizing action
afLinkStretching	5	Link stretching action
afNodeSelection	6	Node selection action

Node Properties

Alignment	sets or returns the alignment of text in a node.
AutoSize	allows to adjust node size to picture size or adjust picture size to node size.
DrawColor	returns/sets the pen color used to draw the node.
DrawStyle	returns/sets the pen style used to draw the node.
DrawWidth	returns/sets the pen width used to draw the node.
FillColor	returns/sets the color used to fill the node.
Font	returns/sets the font used to display the node text.
ForeColor	returns/sets the foreground color used to display the node text.
Height	returns/sets the height of the bounding rectangle of a node.
Hidden	determines whether the node is visible or hidden.
Index	returns the index of the node in the nodes collection.
InLinks	returns a reference to the collection of links that come to the node.
Left	returns/sets the left position of the bounding rectangle of a node.
Marked	returns/sets a flag.
OutLinks	returns a reference to the collection of links that leave the node.
Picture	returns/sets the picture to be displayed in a node.
Selectable	determines if the node is active or readonly.
Selected	determines if the node is selected or not.
Shape	returns/sets the node shape.
Tag	returns/sets the tag associated with a node.
Text	returns/sets the text associated with a node
Top	returns/sets the top position of the bounding rectangle of a node.
Transparent	determines whether a node is transparent or not.
UserData	returns/sets a numeric data associated with a node.
Width	returns/sets the width of the bounding rectangle of a node
ZOrder	places a node at the front or back of the z-order.

Link Properties

ArrowDst	returns/sets the link destination arrow shape.
ArrowOrg	returns/sets the link origin arrow shape.
DrawColor	returns/sets the pen color used to draw the link.
DrawStyle	returns/sets the pen style used to draw the link.
DrawWidth	returns/sets the pen width used to draw the link.
ExtraPoints	returns a reference to the collection of a link extra points.
Dst	returns the reference of the destination node of a link.
Font	returns/sets the font used to display the link text.

ForeColor	returns/sets the foreground color used to display the link text.
Hidden	determines whether the link is visible or hidden.
LinkStyle	returns/sets the style (polyline, bezier) used to draw the link.
Marked	returns/sets a flag.
Org	returns the reference of the origin Node object of a link
PointDst	returns the last point of a link, i.e the point at the intersection between the destination node and the link.
PointOrg	returns the first point of a link, i.e the point at the intersection between the origin node and the link.
Rigid	determines if the link is rigid or not.
Selectable	determines if the link is active or readonly.
Selected	determines if the link is selected or not.
Tag	returns/sets the tag associated with a link.
Text	returns/sets the text associated with a link.
UserData	returns/sets a numeric data associated with a link.
ZOrder	places a link at the front or back of the z-order.

Nodes collection

Add	add a node to a collection of nodes.
Clear	erase all the nodes of a collection of nodes.
Count	returns the number of nodes in a collection of nodes.
Item	returns the reference to a node object of a collection of nodes.
Remove	remove a node from a collection of nodes.

Links collection

Add	add a link to a collection of links.
Clear	erase all the links of a collection of links.
Count	returns the number of links in a collection of links.
Item	returns the reference to a link object of a collection of links.
Remove	remove a link from a collection of links.

LinkPoints collection

Add	add a point to a collection of points.
Clear	erase all the points of a collection of points.
Count	returns the number of points in a collection of points.
Item	returns the reference to a point object of a collection of points.
Remove	remove a point from a collection of points.

Add Method (Links collection)

Description

Creates a link, adds it to a collection of links and returns a reference to this link.

Syntax

*Set link = collection.***Add***(node)*

The arguments are:

Arguments	**Description**
link	a reference to the created lin.
collection	a reference to a Links collection.
node	a reference to a node.

Add Method (LinkPoints collection)

Description

Adds a LinkPoint object to a collection of LinkPoint objects.

Syntax

*collection.***Add***(point)*

The arguments are:

Arguments	**Description**
collection	a reference to a LinkPoints collection.
point	a reference to a LinkPoint object.

Add Method (Nodes collection)

Description

Creates a node, adds it to a collection of nodes and returns a reference to this node.

Syntax

*[node =] collection.***Add***(left, top, width, height)*

Arguments	**Description**
node	a reference to the created node.
collection	a reference to a Nodes collection.
left	Single
top	Single
width	Single
height	Single

Alignment Property

Description

If applied to a node, it returns/sets its text alignment style.
If applied to an AddFlow control, it returns/sets the default node text alignment style. When a node is created, it has this alignment value.

Syntax

*object.***Alignment**[= *alignment*]

Arguments	Description
object	Object expression that may reference an AddFlow control or a node.
alignment	the node text alignment.

Settings

Constants	Value	Description
afLeftJustifyTOP	0	Left Justify - TOP
afLeftJustifyMIDDLE	1	Left Justify - MIDDLE
afLeftJustifyBOTTOM	2	Left Justify - BOTTOM
afRightJustifyTOP	3	Right Justify - TOP
afRightJustifyMIDDLE	4	Right Justify - MIDDLE
afRightJustifyBOTTOM	5	Right Justify - BOTTOM
afCenterTOP	6	Center - TOP
afCenterMIDDLE	7	Center - MIDDLE (default)
afCenterBOTTOM	8	Center - BOTTOM

ArrowOrg, ArrowDst Property

Description

If applied to a link, it returns/sets its origin arrow shape or its destination arrow shape.

If applied to an AddFlow control, it returns/sets the default link origin or destination arrow shape. When a link is created, it receives those default link shapes.

Syntax

object.**ArrowOrg**[= *shape*]

object.**ArrowDst**[= *shape*]

Arguments	Description
object	Object expression that may reference an AddFlow control or a link.
shape	the link origin or destination shape

Settings

Constants	Value	Description
afNone	0	None
afFilledCircle	1	Filled Circle
afEmptyCircle	2	Empty Circle
afFilledArrow15	3	Filled Arrow 15
afEmptyArrow15	4	Empty Arrow 15
afFilledArrow30	5	Filled Arrow 30
afEmptyArrow30	6	Empty Arrow 30
afFilledArrow45	7	Filled Arrow 45
afEmptyArrow45	8	Empty Arrow 45

By default, ArrowDst is set to 4 and ArrowOrg is set to 0.

AutoScroll Property

Description

Determines whether automatic scrolling is allowed or not. Automatic scrolling means for instance that when the user moves a node outside the visible part of the control, the control scrolls automatically so that the node stays visible.

Syntax

AddFlow1.**AutoScroll**[= {True | False}]

Settings

Value	Description
False	Automatic scrolling is not allowed.
True	(Default) Automatic scrolling is allowed.

See Also

Capabilities

AutoSize Property

Description

This property allows to adjust node size to picture size or adjust picture size to node size.
If applied to a node, it returns/sets its AutoSize style.
If applied to an AddFlow control, it returns/sets the default node AutoSize style. When a node is created, it has this AutoSize value.

Syntax

*object.***AutoSize**[= *autosize*]

Arguments	Description
object	Object expression that may reference an AddFlow control or a node.
autosize	the node AutoSize style.

Settings

Constants	Value	Description
afNone	0	None (default)
afAdjustPictureSizetoNode	1	Adjust Picture Size to Node
afAdjustNodeSizetoPicture	2	Adjust Node Size to Picture

BackPicture Property

Description

This property is the same as the standard Visual Basic Picture property except that it only supports bitmap (.BMP) files.

CanDrawNode Property

Description

Determines whether you can create nodes interactively or not

Syntax

AddFlow1.**CanDrawNode**[= {True | False}]

Settings

Value	Description
False	Drawing nodes is not allowed.
True	(Default) Drawing nodes is allowed.

See Also

Capabilities

CanDrawLink Property

Description

Determines whether you can create links interactively or not. If the value of this property is False, then the handle at the middle of the node is not displayed.

Syntax

AddFlow1.**CanDrawLink**[= {True | False}]

Settings

Value	Description
False	Drawing links is not allowed.
True	(Default) Drawing links is allowed.

Note

You'll have to do a refresh just after changing the value of this property.

Example 1

```
AddFlow1.CanDrawLink = False  'or True
AddFlow1.Refresh
```

See Also

Capabilities

CanMoveNode Property

Description

Determines whether you can move (drag) nodes interactively or not

Syntax

AddFlow1.**CanMoveNode**[= {True | False}]

Settings

Value	Description
False	Moving nodes is not allowed.
True	(Default) Moving nodes is allowed.

See Also

Capabilities

CanMultiLink Property

Description

Determines whether you can create several links between two nodes or not.

Syntax

AddFlow1.**CanMultiLink**[= {True | False}]

Settings

Value	Description
False	Multi links is not allowed.
True	(Default) Multi links is allowed.

See Also

Capabilities

CanSizeNode Property

Description

Determines whether you can resize nodes interactively or not. If the value of this property is False, then the node sizing handles are white, which mean you cannot use them to resize the node. The handles are still visible since they allow to know that the node is selected. See DisplayHandles property in order to avoid displaying handles.

Syntax

AddFlow1.**CanSizeNode**[= {True | False}]

Settings

Value	Description
False	Sizing nodes is not allowed.
True	(Default) Sizing nodes is allowed.

Note

You'll have to do a refresh just after changing the value of this property.

Example 1

```
AddFlow1.CanSizeNode = False  'or True
AddFlow1.Refresh
```

See Also

Capabilities

CanStretchLink Property

Description

Determines whether you can "stretch" links (i.e add or remove segments) interactively or not. If the value of this property is True, then the link handles are white, which mean you cannot use them to strech the link. The handles are still visible since they allow to know that the link is selected. See DisplayHandles property in order to avoid displaying handles.

Syntax

AddFlow1.**CanStretchLink**[= {True | False}]

Settings

Value	Description
False	Stretching links is not allowed.
True	(Default) Stretching links is allowed.

Note

You'll have to do a refresh just after changing the value of this property.

Example 1

```
AddFlow1.CanStretchLink = False  'or True
AddFlow1.Refresh
```

See Also

Capabilities

Clear Method

Description

Remove all objects of a collection.

Syntax

collection.**Clear**

Arguments	**Description**
collection	Object expression that may reference a collection of nodes, links or points.

Count Method

Description

Returns the number of objects in a collection.

Syntax

number = *collection*.**Count**

Arguments	**Description**
collection	Object expression that may reference a collection of nodes, links or points.
Number	the returned number of objects.

DeleteMarked Method

Description

Deletes marked nodes (and their associated links) and marked links.

Syntax

AddFlow1.**DeleteMarked**

See Also

Marked property.

DeleteSel Method

Description

Deletes selected nodes (and their associated links)

Syntax

AddFlow1.**DeleteSel**

DisplayHandles Property

Description

Determines whether handles are displayed. The handles are the little black squares on the selected node or link. If handles are not displayed, then the user cannot interactively resize nodes, create or stretch links. In such a situation, if you (the developper that is using AddFlow) need to emphasize the selected node or link, you have to do it yourself (by changing its color for instance).

Syntax

AddFlow1.**DisplayHandles**[= {True | False}]

Settings

Value	Description
False	Handles are not displayed.
True	(Default) Handles are displayed.

Note

You'll have to do a refresh just after changing the value of this property.

Example 1

```
AddFlow1.DisplayHandles = False  'or True
AddFlow1.Refresh
```

See Also

Capabilities

DoClick Method

Description

Fires a click event.

Syntax

AddFlow1.**DoClick**

DrawColor Property

Description

If applied to a node or a link, it returns/sets the pen color used to draw it.
If applied to an AddFlow control, it returns/sets the default color used to draw nodes or links. When a node or a link is created, it is displayed with this drawing color.

Syntax

*object.***DrawColor**[= *color* &]

Arguments	**Description**
object	Object expression that may reference an AddFlow control, a node or a link.
color	the pen drawing color

Settings

Value	**Description**
Normal RGB Colors	Color set with RGB or QBColor function in code
System Default Colors	Colors specified with the system color constants from CONSTANT.TXT, a Visual Basic file that you can load into a project's global module. Window's substitutes the user's choices, as specified through the user's Control Panel Settings.

By default, DrawColor is set to 0 (black)

DrawStyle Property

Description

If applied to a node or a link, it returns/sets the pen style used to draw it.
If applied to an AddFlow control, it returns/sets the default pen style used to draw nodes or links. When a node or a link is created, it is displayed with this drawing pen style.

Syntax

*object.***DrawStyle**[= *style*]

Arguments	Description
object	Object expression that may reference an AddFlow control, a node or a link.
style	the pen drawing style

Setting

Constants	Value	Description
afSolid	0	Solid
afDash	1	Dash
afDot	2	Dot
afDashDot	3	Dash-Dot
afDashDotDot	4	Dash-Dot-Dot
afTransparent	5	Transparent
afInsideSolid	6	Inside Solid

Remarks

If DrawWidth is set to a value greater than 1, then DrawStyles 1 through 4 produce a solid line (the DrawStyle property value is not changed). If DrawWidth is set to 1, DrawStyle produces the effect described above for each setting.

DrawWidth Property

Description

If applied to a node or a link, it returns/sets the pen width used to draw it.
If applied to an AddFlow control, it returns/sets the default pen width used to draw nodes or links When a node or a link is created, it is displayed with this drawing pen style.

Syntax

*object.***DrawWidth**[= *size*]

Arguments	**Description**
object	Object expression that may reference an AddFlow control, a node or a link.
size	the pen drawing size

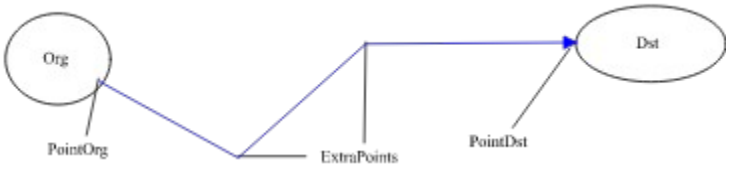
Setting

You can set DrawWidth to a value of 1 to 8 (pixels).

ExtraPoints Property

Description

Returns a reference to the collection of extra points of a Link object
Not available at design time. Read-only at run-time.



Syntax

*Object.***ExtraPoints**

Notes

You may manipulate LinkPoint objects by using the standard methods of a collection (for instance, **Add** and **Remove**). It is possible to access each element of the collection with **Item** method or in a **For Each** loop.

See Also

PointOrg, PointDst properties.

FillColor Property

Description

If applied to a Node object, it returns/sets the color used to fill the node.
If applied to an AddFlow control, it returns/sets the default color used to fill node objects. When a node is created, it is filled with this color.

Syntax

*object.***FillColor**[= *color* &]

Arguments	**Description**
object	Object expression that references an AddFlow control or a node.
color	the filling color

Settings

Value	**Description**
Normal RGB Colors	Color set with RGB or QBColor function in code
System Default Colors	Colors specified with the system color constants from CONSTANT.TXT, a Visual Basic file that you can load into a project's global module. Window's substitutes the user's choices, as specified through the user's Control Panel Settings.

By default, FillColor is set to 0 (black)

Font Property

Description

Sets or returns the font used to display the text associated to a node or a link.
If applied to a node or a link, it returns/sets its font used to display its text.
If applied to an AddFlow control, it returns/sets the default font used to display the text of a node or a link. When a node or a link is created, it has this default font.

Syntax

object.**Font**[= *font*]

Note

If you need to change the text color, you have to use the ForeColor property

ForeColor Property

Description

If applied to a node or a link, it returns/sets the foreground color used to display its text.
If applied to an AddFlow control, it returns/sets the default foreground color used to display the text of next created nodes or links.

Syntax

*object.***ForeColor**[= *color* &]

Arguments	**Description**
object	Object expression that references an AddFlow control, a node or a link.
color	the foreground color

Settings

Value	**Description**
Normal RGB Colors	Color set with RGB or QBColor function in code
System Default Colors	Colors specified with the system color constants from CONSTANT.TXT, a Visual Basic file that you can load into a project's global module. Window's substitutes the user's choices, as specified through the user's Control Panel Settings.

By default, ForeColor is set to 0 (black).

GetVersion Method

Description

Returns control version.

Syntax

AddFlow1.**GetVersion** *major, major*

Arguments	Description
major	major version number
minor	minor version number

For instance, if the control version number is 1.31, then major = 1 and minor = 31.

Hidden Property

Description

If applied to a node or a link, it determines whether it is visible or hidden.
If applied to an AddFlow control, it determines whether nodes and links are by default visible or hidden

Syntax

Object.**Hidden** [= {True | False}]

Arguments	**Description**
object	Object expression that may reference an AddFlow control, a node or a link.

Settings

Value	**Description**
False	(default) The object (node or link) is visible.
True	The object is hidden.

Index Property

Description

Returns the index of a node, i.e its position in the Nodes collection.
Not available at design time; read only at run time.

Syntax

*object.***Index**

Item Property

Description

Returns an object from a collection.
Not available at design time; read only at run time.

Syntax

object = *collection.***Item***(index)*

Arguments	**Description**
collection	object expression that references a collection.
object	object expression that references an object in a collection.
index	index of an object in a collection.

InLinks, OutLinks Property

Description

InLinks returns a reference to the collection of Link objects that comes to a node.
OutLinks returns a reference to the collection of Link objects that leaves a node.
Not available at design time. Read-only at run-time.

Syntax

*object.***InLinks**

*object.***OutLinks**

Notes

You may manipulate Link objects by using the standard methods of a collection (for instance, **Add** and **Remove**). It is possible to access each element of the collection with **Item** method or in an enumeration loop (**For Each ... In**)

IsChanged Method

Description

Returns an integer value that is nonzero if the AddFlow diagram has changed, otherwise zero.

Call this method to determine if the AddFlow diagram has changed.

Syntax

[*changed* =] *AddFlow1*.**IsChanged**

Arguments	**Description**
changed	True if a change has occured, False elsewhere.

See Also

SetChangedFlag method.

IsSelChanged Method

Description

Returns an integer value that is nonzero if the selection in the AddFlow diagram has changed, otherwise zero.

Call this method to determine if the selection has changed.

Syntax

[*selchanged* =] *AddFlow1*.**IsSelChanged**

Arguments	Description
selchanged	True if a selection change has occured, False elsewhere.

See Also

SetSelChangedFlag method.

LastUserAction Property

Description

Rreturns the last user action.

Syntax

object.**LastUserAction**[= *action*]

Arguments	**Description**
object	Object expression that may reference an AddFlow control.
action	the last user action

Setting

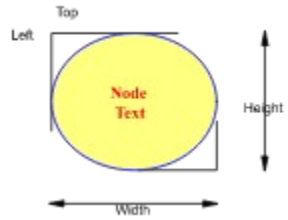
Constants	**Value**	**Description**
afNone	0	None
afNodeCreation	1	Node creation action
afLinkCreation	2	Link creation action
afNodeDragging	3	Node dragging action
afNodeResizing	4	Node resizing action
afLinkStretching	5	Link stretching action
afNodeSelection	6	Node selection action

See **Example**

Left, Top, Width, Height Property

Description

Sets or returns the position (Left, Top) and size (Width, Height) of the bounding rectangle of a node. Not available at design time.



Syntax

Node.**Left**[= *numeric expression*]

Node.**Top**[= *numeric expression*]

Node.**Width**[= *numeric expression*]

Node.**Height**[= *numeric expression*]

LinkStyle Property

Description

If applied to a Link object, it returns/sets the style (polylines or curves) used to draw the link.
If applied to an AddFlow control, it returns/sets the default style used to draw links.

Syntax

*object.***LinkStyle**[= *style*]

Arguments	Description
object	Object expression that may reference an AddFlow control or a link.
style	the link style.

Setting

Constants	Value	Description
afPolyline	0	Polyline
afBezier	1	Bezier curve

Marked Property

Description

Returns/sets a flag indicating if a node or a link is marked or not. This property can be used in conjunction with the DeleteMarked method.

Syntax

Object.**Marked** [= {True | False}]

Settings

Value	Description
False	(default) The Node or Link Object is unmarked.
True	The Node or Link Object is marked.

MultiSel Property

Description

Determines whether multiselection mode is possible or not.

Syntax

AddFlow1.**MultiSel**[= {True | False}]

Settings

Value	Description
False	Multi selection is not allowed.
True	(Default) Multi selection is allowed.

See Also

Capabilities

Nodes Property

Description

Returns a reference to the collection of Node objects.
Not available at design time. Read-only at run-time.

Syntax

AddFlow1.**Nodes**

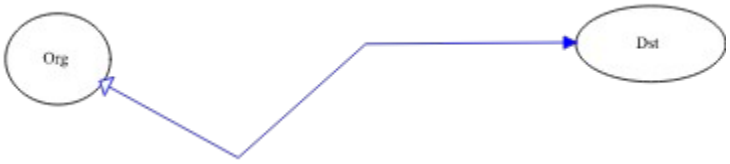
Notes

You may manipulate Link objects by using the standard methods of a collection (for instance, **Add** and **Remove**). It is possible to access each element of the collection with **Item** method or in an enumeration loop (**For Each ... In**)

Org, Dst Property

Description

Return the reference of the origin and destination nodes of a link.
Not available at design time. Read-only at run-time.



Syntax

Link.**Org**

Link.**Dst**

Remarks

It is not possible to change directly the origin or destination nodes of a link. If you want to do that, you have to memorize the link properties, destroy it, create a new one with the new origin or destination node and sets previous saved properties.

See Also

Navigation

Picture Property

Description

Sets or returns the picture to be displayed in a node. This picture can be a bitmap or an icon.

Syntax

Node.**Picture**[= *picture*]

Settings

Value	Description
(none)	(Default)
(bitmap, icon)	Specifies a picture. You can also set this property using the LoadPicture function on a bitmap or an icon.

PointedArea Property

Description

Returns the type of the area pointed by the mouse (sizing square, stretching square, linking square, node, over no special area).

Not available at design time; read only at run time

Syntax

AddFlow1.**PointedArea**

Settings

Constants	Value	Description
afNWSEsizeHandle	0	NW-SE Size handle area
afNSsizeHandle	1	North-South Size handle area
afNESWsizeHandle	2	NE-SW Size handle area
afWEsizeHandle	3	West-East Size handle area
afStretchingHandle	4	Stretching handle area
afLinkingHandle	5	Linking handle area
afNodeArea	6	Node area
afOutSide	7	Outside area
afLinkArea	8	Link area

Remarks

This property allows to change dynamically the mouse pointer BEFORE the user clicks anywhere, to indicate what actions are possible.

For example, when the pointer is over one of the corner points of a node, it should change to the standard NE/SW or NW/SE diagonal arrow. When it is over a side node, it would be the N/S or E/W arrow.

See Also

PointedNode, PointedLink properties.

PointedNode, PointedLink Property

Description

Returns the object pointed by the mouse :
- If it is a node, PointedNode returns a reference to this node and PointedLink returns Nothing.
- If it is a link, PointedNode returns Nothing and PointedLink returns a reference to this link.
- if the mouse is over nothing, both properties returns Nothing.

Not available at design time; read only at run time

Syntax

AddFlow1.**PointedNode**

AddFlow1.**PointedLink**

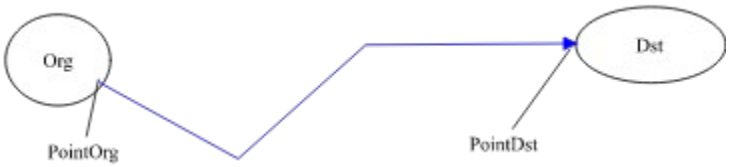
See Also

PointedArea property.

PointDst, PointOrg Property

Description

PointDst returns a reference to the last point of a link.
PointOrg returns a reference to the first point of a link.
Not available at design time. Read-only at run-time.



Syntax

*object.***PointDst**

*object.***PointOrg**

See Also

ExtraPoints property.

ReadOnly Property

Description

Set "read only" mode. In such a mode user interaction is not allowed.

Syntax

AddFlow1.**ReadOnly**[= {True | False}]

Settings

Value	Description
False	(Default) "Read only" mode is set.
True	"Read only" mode is not set.

Note

You'll have to do a refresh just after changing the value of this property.

Example 1

```
AddFlow1.ReadOnly = False  'or True
AddFlow1.Refresh
```

See Also

Capabilities

Refresh Method

Description

Forces a repaint of the OLE control.

Syntax

AddFlow1.**Refresh**

Remove Method

Description

Remove an object from a collection

Syntax

collection.**Remove** *object*
collection.**Remove** *index*

Arguments	**Description**
collection	object expression that may reference a collection of nodes, links or points.
object	object expression that may reference a node or a link.
index	index of a point in a collection of points.

Repaint Property

Description

Determines whether repainting the AddFlow control is allowed or not.Setting this property to False increases speed performance. Setting this property to True causes a refresh.

Not available at design time

Syntax

AddFlow1.**Repaint**[= {True | False}]

Settings

Value	Description
False	Repainting not allowed.
True	(Default) Repainting allowed

Example

Rigid Property

Description

Determines whether a Link object is rigid or not.

If a link is rigid, it follows (without being stretched) its origin or destination node when this origin or destination node is being dragged. A consequence of this rigid behaviour is that the origin node follows the destination node (if it is this one that is moving) or the destination node follows the origin node (if it is this one that is moving). If all nodes are linked each other with rigid links then, all the nodes grape is moving if the user drags one node.

If it is not rigid, it does not move when its origin or destination node is moved. Only its first point (if origin node moved) or its last point (if destination node moved) follows the node.

Syntax

*Object.***Rigid** [= {True | False}]

Arguments	**Description**
object	Object expression that may reference an AddFlow control, or a link.

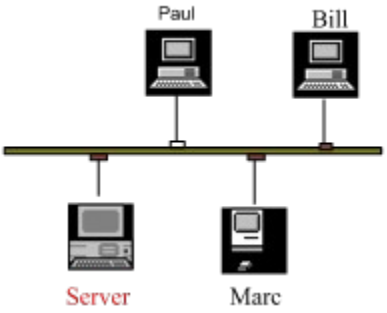
Settings

Value	**Description**
False	(default) The link is not rigid.
True	The link is rigid.

Remarks

- This property allows to define a kind of ownership between 2 nodes. If there is a rigid link from node A to node B, then if the user drags A, B follows it. We may say that A owns B. When an owner node is moved, all its owned nodes are also moved. This happens only when the user moves the node interactively with the mouse (dragging). If the node is moved programmaticaly (i.e changing its Left or Top properties), owned nodes do not move.

- You may use this property in a recursive way: an owned node may be itself owner of other nodes. This property may be used to implement stubs or pins, allowing a node to have several owned nodes inside itself and those owned nodes can be used as stubs receiving links. For instance, in the following diagram,the flat rectangular node is the owner of 4 little nodes used as stubs. The rigid links are hidden (see Hidden property) and readonly (see Selectable property). You may make the little nodes readonly too so that the user cannot select, resize or move them.



SaveImage Method

Description

Save an image of the diagram using the Windows standard metafile format. It does not save the diagram itself. It saves only an image of this diagram. This image can be used in every application that accept the Windows standard metafile format.

Syntax

AddFlow1.**SaveImage** *tymed, format, file*

Arguments	Description
tymed	type of medium.
format	format of data. This value must be 0 (Windows Metafile format).
file	name of file

Settings

Value	Description
0	The diagram is stored in a file.
1	The diagram is copied to the clipboard.

ScrollBars Property

Description

Allows to add scrollbars for the AddFlow control. Read-only at run time.

Syntax

AddFlow1.**ScrollBars**[= *setting*]

Settings

Constants	Value	Description
afNone	0	None
afHorizontal	1	Horizontal
afHorizontal	2	Vertical
afBoth	3	Both

See Also

Capabilities

Selectable Property

Description

Determines whether a node or a link is selectable by clicking on it with the mouse or unselectable (readonly or inactive).

When an object is unselectable,the user cannot interactively make it current or selected. He can do this only programmaticaly. Such an object may be used to display a bitmap or a text but the user cannot move, stretch or resize it with the mouse.

Syntax

Object.**Selectable** [= {True | False}]

Settings

Value	Description
False	The Node or Link Object is unselectable.
True	(default) The Node or Link Object is selectable.

SelectAll Method

Description

Selects all nodes

Syntax

AddFlow1.**SelectAll**

Selected Property

Description

Returns a flag indicating if a node or a link is selected or not.

Syntax

Object.**Selected**

Settings

Value	Description
False	(default) The Node or Link Object is not selected.
True	The Node or Link Object is selected.

SelectedLink Property

Description

Sets or returns the reference to the current Link object.
Setting this property causes previous selection to disappear.
Not available at design time.

Syntax

Set AddFlow1.**SelectedLink**[= *Link*]

SelectedNode Property

Description

Sets or returns the reference to the current Node object.
Setting this property causes previous selection to disappear.
Not available at design time.

Syntax

Set AddFlow1.**SelectedNode**[= *Node*]

SelectMode Property

Description

Allow to enter in selection mode instead of drawing mode. This property has no effect if MultiSel property is not set.

Not available at design time.

The **selection mode** allows to select several nodes. You bring the mouse cursor into the AddFlow control, press the left button, move the mouse and release the left button. All nodes inside the selection rectangle are selected. Then you can unselect some nodes by clicking them with the mouse and simultaneously pressing the shift or control key. You can select them again by using the same method.

Syntax

AddFlow1.**SelectMode**[= {True | False}]

Settings

Value	Description
False	(Default) Drawing mode.
True	Select mode is set.

SetChangedFlag Method

Description

Allow to set a flag indicating that the diagram has changed or not. Typically, you should set this flag to False just after having saved the diagram.

Syntax

AddFlow1.**SetChangedFlag** *changed*

Arguments	Description
changed	boolean value.

See Also

IsChanged method.

SetSelChangedFlag Method

Description

Allow to set a flag indicating that the selection has changed or not.

Syntax

AddFlow1.**SetSelChangedFlag** *selchanged*

Arguments	**Description**
selchanged	boolean value.

See Also

IsSelChanged method.

Shape Property

Description

If applied to a node, it returns/sets its shape.
If applied to an AddFlow control, it returns/sets the default node shape. When a node is created, it has this shape.

Syntax

*object.***Shape**[= *shape*]

Arguments	Description
object	Object expression that may reference an AddFlow control or a node
shape	the node shape

Settings

Constants	Value	Description
afEllipse	0	Ellipse (default)
afRectangle	1	Rectangle
afRoundRect	2	RoundRect
afDiamond	3	Diamond
afNorthTriangle	4	North Triangle
afSouthTriangle	5	South Triangle
afEastTriangle	6	East Triangle
afWestTriangle	7	West Triangle
afHexagon	8	Hexagon

ShowGrid Property

Description

Specify if the grid is displayed or not.

Syntax

AddFlow1.**ShowGrid**[= {True | False}]

Settings

Value	Description
False	(Default) The grid is not displayed.
True	The grid is displayed.

See Also

Capabilities

Tag Property

Description

Sets or returns a tag associated with a node or a link.

Syntax

Object.**Tag**[= *string expression*]

Text Property

Description

Sets or returns the text associated with a node or a link.

The text associated to a node is displayed inside the node. It is a multiline display. The text is wrapped automatically inside the node. Linefeed and carriage return characters are supported.

The text associated to a link is displayed at the middle of its segment number n/2 + 1 (n is the number of segments). This text is displayed in a single line.

Syntax

Object.**Text**[= *string expression*]

Remarks

The font and the color of the text are determined by Font and ForeColor properties.

If the object is a node, then the position of the text is determined by the Alignment property.

Transparent Property

Description

Determines whether next created nodes will be transparent or not.
If applied to a node, it determines whether it is transparent or not.
If applied to an AddFlow control, it determines whether next created nodes will be transparent or not.

Syntax

*object.***Transparent**[= {True | False}]

Arguments	**Description**
object	Object expression that may reference an AddFlow control or a node.

Settings

Value	**Description**
False	(default) Opaque
True	Transparent

UserData Property

Description

Sets or returns a numeric data associated with a node or a link.

Syntax

Object.**UserData**[= *value*]

xGrid, yGrid Property

Description

Sets or returns the grid values in twips.

Syntax

AddFlow1.**xGrid**[= *numeric expression*]

AddFlow1.**yGrid**[= *numeric expression*]

See Also

Capabilities

xScroll, yScroll Property

Description

Sets or returns the scroll values in twips.
Not available at design time.

Syntax

AddFlow1.**xScroll**[= *numeric expression*]

AddFlow1.**yScroll**[= *numeric expression*]

xZoom, yZoom Property

Description

Specify a zoom factor which can be a value between 0 and 1000.
Setting it to 0 display the diagram so that it fits in the control area.
Setting it to 100% display the diagram at its normal size.
Setting it to a value higher than 100% expands the diagram
Setting it to a value less than 100% shrinks the diagram.

Syntax

AddFlow1.**xZoom***[= setting]*

AddFlow1.**yZoom***[= setting]*

ZOrder Property

Description

Places current Node or Link object at the front or back of the z-order.
Not available at design time; write only at run time.

Syntax

object.**ZOrder** = [*setting*]

Settings

Value	Description
0	Send object (node or link) Front
1	Send object (node or link) Back

LastUserAction, property, example

```
' The following code allows to associate text to nodes and links
' as soon as those objects are created by the user with the mouse.
' You should copy this code and paste it into the MouseUp procedure of
' an AddFlow control.
Dim Action As Long, node as afNode, link as afLink

With AddFlow1
  Action = .LastUserAction()
  Select case Action
    Case afNodeCreation  ' Node creation
      ' Display the node index
      Set node = .SelectedNode
      node.Text = Str(node.Index())
    Case afLinkCreation  ' Link creation
      ' Display the link origin and destinations nodes index
      Set link = .SelectedLink
      link.Text = Str(link.Org.Index()) + "," + Str(link.Dst.Index())
  End Select
End With
```

Repaint, property, example

```
' Make all nodes and links red.
Dim node as afNode, link as afLink

With AddFlow1
  .Repaint = False    ' Avoid any repaintings during process

  ' Nodes enumeration
  For Each node In .Nodes
    node.DrawColor = RGB(255, 0, 0)

    ' For each node, perform a « leaving link » enumeration.
    For Each link In node.OutLinks
      link.DrawColor = RGB(255, 0, 0)
    Next
  Next

  .Repaint = True     ' Cause a refresh
End With
```

Node and link creation example

```
' Create 2 nodes and link them.
Dim node1 as afNode, node2 as afNode
Dim link as afLink

' Define a default filling color for next created nodes and links
AddFlow1.DrawColor = RGB(0, 0, 0)  ' Default Drawing color = Black
' Define a default shape for next created nodes
AddFlow1.Shape = afRectangle  ' Default shape = rectangle

' Create a blue rectangular node and associate a text to this node
Set node1 = AddFlow1.Nodes.Add(100, 100, 500, 500)
node1.Text = "First node"

' Create a blue elliptic node and associate a text to this node
Set node2 = AddFlow1.Nodes.Add(2000, 1000, 800, 800)
node2.DrawColor = RGB(0, 0, 255)
node2.Shape = afEllipse
node2.Text = "Second node"

' Create a red link between node1 and node2
Set link = node1.OutLinks.Add(node2)
link.DrawColor = RGB(255, 0, 0)
```