EASYNET 1.0

EasyNet is a Custom Control for Microsoft Visual Basic for Windows (*). It helps you quickly draw and manage network diagrams.

Quick Start

- **Add the EasyNet VBX** to your project by selecting "Add File..." from Visual Basic's "File" menu. If this is the demonstration version, an "About" dialog box appears and you have to click Ok.

- **Drag an EasyNet control** from the toolbox to your form.

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

- **Draw a node**: bring the mouse cursor into the EasyNet 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 (black 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).

- **Now, you may return to the Visual Basic design-time mode** in order to change EasyNet control properties. For instance you may change the node filling color with FillColor property, the node shape (Shape property), the drawing color (DrawColor property). Yoy may allow multiselection (MultiSel property), add scrollbars (ScrollBars property), etc...

...Well, it is very easy, isnt'it?

Enjoy!

Why use EasyNet?

EasyNet control is useful in following applications:

- workflow diagrams
- data base diagrams
- organizational flowcharts
- communication networks
- ... and all application that needs to manage a network diagram.

Overview

This Custom Control allows to draw network diagrams. Drawings can be made interactively with the mouse (See Quick Start) or programmaticaly.

A network diagram is a set of nodes that can be linked. So an EasyNet control contains items that can be nodes or links. You can associate data to each item and you can navigate in the network diagram.

By exploring following topics, you'll discover all features of the EasyNet control.

Items

Items are nodes or links. Two nodes can be linked with a link. A link cannot exist without its origin and destination nodes. If one of these two nodes is deleted, the link is also deleted.

You can make an item be the current one either with the mouse or with Item property, allowing you to work with it, get or set its properties. You can also select several items with the mouse if multiselection is allowed (in such a case MultiSel and SelectMode properties are true).

IsLink property allows to know if current item is a link or not.

Example:*If current item is a link, make its origin node be red.*

```
Dim curLink As Long
If Net1.IsLink = True Then
  ' Save current item
  curLink = Net1.Item
  ' Make origin node be the current item
  ' in order to work with it
  Net1.Item = Net1.Org
  ' Change node filling color
  Net1.FillColor = RGB(255, 0, 0)
  ' Restore current item
  Net1.Item = curLink
End If
```

Drawing

You can change colors, styles and shapes of each item:

- X1, X2, Y1, Y2 properties allows to set or get position and size of each item.
- Picture property allows to associate a bitmap or an icon to each node.
- Shape property allows to specify a shape for a node (ellipse or rectangle)
- DrawColor and DrawWidth properties allow to specify the color and width of the pen used to draw nodes or links.
- FillColor property allows to specify the color used inside a node.
- PointCount, PointX, PointY properties allow to have a link composed of several segments.
- Oriented property specifies if a link is oriented or not. If the link is oriented, it has an arrowhead.

You can create items, delete items and do other edit actions (like copying the network diagram onto the clipboard in a metafile format) with EditAction property.

Example:

Creates two nodes and link them. Each node has a text. One is a green rectangle and the other is a yellow ellipse. The link is oriented.

```
Sub Exercice ()
  net1.Item = 0  'No currentitem

  ' Following properties will apply to next created node
  ' since there is no current item
  net1.Shape = 0   'Ellipse
  net1.FillColor = RGB(255, 255, 0)  'Yellow color
  net1.X1 = 2800
  net1.Y1 = 1600
  net1.X2 = 4400
  net1.Y2 = 2200

  ' Create a node with previous defined position and size
  net1.EditAction = 0

  ' The node is created: associate a text to it.
  net1.Text = "Use EASYNET.VBX!!"

  ' Following property will apply to next created link
  net1.Dst = net1.Item  'destination node of link

  ' Create a second node
  net1.Item = 0
  net1.Shape = 1   'Rectangle
  net1.FillColor = RGB(0, 255, 0)  'Green color
  net1.X1 = 200
  net1.Y1 = 200
```

```
  net1.X2 = 2000
  net1.Y2 = 800
  net1.EditAction = 0
  net1.Text = "A network to implement?"

  ' Following property settings will apply to next created link
  net1.Org = net1.Item  'origin node of link
  net1.Oriented = true  'link will have an arrowhead

  ' Create a link
  net1.EditAction = 1

  ' Make this link be not current
  net1.Item = 0
End Sub
```

Data Association

You can associate data to each item (node or link) with following properties:

- Text property associates a string. The EasyNet control maintains the memory for the strings associated to items.
- Data property associates a long integer that can be used to store a reference to a user data.
- Type property associates an integer that can be used to store an identifier or a type.
- Picture property associates a picture (bitmap or icon). The EasyNet control does not maintain the memory for that picture.

Navigation

You can navigate in the network diagram with LoopAction, LoopCount and LoopItem properties:

- LoopAction property has to be called first in order to indicate the type of navigation to perform.
- Then, a call to LoopCount gives the count of items involved in this navigation.
- Then, you get each item with LoopItem property.

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

Oriented property specifies if a link is oriented or not.

Example:
Makes color of all "out" links of all selected nodes be red.
Two calls to LoopAction property cannnot be cascaded so you have first to memorize the selected nodes in an array in order to work with them.

```
Sub Exercice ()
  Dim nbnode, nblink, i, j As Integer
  Dim Node() As Long

  ' Do a loop with selected nodes
  Net1.LoopAction = 2

  ' Get count of selected nodes
  nbnode = Net1.LoopCount

  ' If no selected nodes, nothing to do
  If nbnode = 0 Then Exit Sub

  ' Memorize selected nodes in a dynamic array.
  ReDim Node(1 To nbnode)
  For i = 1 To nbnode
    Node(i) = Net1.LoopItem(i - 1)
  Next i

  ' For each node of our array...
  For i = 1 To nbnode
    ' ... makes it be the current item
    Net1.Item = Node(i)

    ' Do a loop with all links of the current node
    Net1.LoopAction = 3

    ' Get count of selected nodes
    nblink = Net1.LoopCount

    ' For each link of the current node...
    For j = 1 To nblink
      Net1.Item = Net1.LoopItem(j - 1)
      '... if this link is an "out" link changes its color
      If Net1.Org = Node(i) And Net1.Oriented = True Then
        Net1.DrawColor = RGB(255, 0, 0)
```

```
      End If
    Next j
  Next i

 ' Don't forget to delete the array
 Erase Node
End Sub
```

Capabilities

Following properties allow to set capabilities for an EasyNet control:

- CanDrawNode
- CanDrawLink
- CanMoveNode
- CanSizeNode
- CanStretchLink
- CanMultiLink
- MultiSel
- ReadOnly
- ScrollBars
- xGrid
- yGrid

Limits

For one EasyNet control:

- the maximum number of nodes is **120**.
- the maximum number of links is **120**.
- the maximum length of an item text is **50** characters.
- the maximum number of link points is **8**. (therefore, the maximum number of link segments is **9**).

Of course, one application can manage simultaneously several EasyNet controls (for instance, in MDI child windows).

Properties

All the properties are listed below. Properties that apply only to the EasyNet Custom Control, 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.

CanDrawNode	CanDrawLink	CanMoveNode	CanSizeNode
CanStretchLink	CanMultiLink	CtlName	BackColor
BorderStyle	Caption	Data	Dst
DragIcon	DragMode	DrawColor	DrawWidth
EditAction	Enabled	FillColor	FontBold
FontItalic	FontName	FontSize	FontStrike
FontUnder	Height	HelpContextId	Hwnd
Index	IsLink	Item	Left
LoopAction	LoopCount	LoopItem	MousePointer
MultiSel	Oriented	Org	Parent
Picture	PointCount	PointX	PointY
ReadOnly	ScrollBars	SelectMode	Shape
TabIndex	TabStop	Tag	Text
Top	Type	Visible	Width
xGrid	X1	X2	Y1
Y2	yGrid		

Events

All the events are listed below. Events that apply only to the EasyNet Custom Control, 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 events.

AddLink	AddNode	Change	Click
DblClick	DragDrop	DragOver	ErrSpace
GotFocus	LostFocus	MouseDown	MouseMove
MouseUp	SelChange		

FillColor Property

Description

If current item is 0, sets or returns the "current" filling node color (the filling color used for next created nodes).
If current item is a node, sets or returns its color (the color with which the node is filled).
If current item is a link, writing has no effect and reading returns 0.

Usage

[*form*.]*NET*.**FillColor**[= *color* &]

Settings

The FillColor property settings are:

Setting	Description
Normal RGB Colors	Color set with RGB or QBColor function in code
System Default Colors from	Colors specified with the system color constants 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)

Data Type

Long

See Also

Drawing

DrawColor Property

Description

If current item is 0, sets or returns the "current" drawing color (the drawing color used for next created items).
If current item is not 0, sets or returns its drawing color.

Usage

[*form*.]*NET*.**DrawColor**[= *color* &]

Settings

The DrawColor property settings are:

Setting	**Description**
Normal RGB Colors	Color set with RGB or QBColor function in code
System Default Colors from	Colors specified with the system color constants 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)

Data Type

Long

See Also

Drawing

DrawWidth Property

Description

If current item is 0, sets or returns the "current" drawing pen width (the drawing pen width used for next created items).
If current item is not 0, sets or returns the item drawing pen width.

Usage

[*form*.]*NET*.**DrawWidth**[= *size*]

Setting

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

Data Type

Integer

See Also

Drawing

Shape Property

Description

If current item is 0, sets or returns the "current" node shape (the shape used for next created nodes).
If current item is a node, sets or returns its shape (ellipse, rectangle).
If current item is a link, writing has no effect and reading returns 0.

Usage

[*form*.]*NET*.**Shape**[= *shape*]

Settings

The Shape property settings are:

Setting	Description
0	Ellipse
1	Rectangle

By default, Shape is set to 0 (ellipse)

Data Type

Integer

See Also

Drawing

Oriented Property

Description

If current item is 0, specify if next created links will be oriented or not.
If current item is a link, specify if it is oriented or not.
If current item is a node, writing has no effect and reading returns 0.
When a link is oriented, is is displayed with an arrowhead at its destination node.

Usage

[*form*.]*NET*.**Oriented**[= {True | False}]

Settings

The Oriented property settings are:

Setting	**Description**
False	no arrowhead
True	(default) one arrowhead

Data Type

Integer (Boolean)

See also

Navigation

X1, Y1, X2, Y2 Property

Description

If current item is 0, sets or returns the coordinates of upper left point (X1, Y1) or lower right point (X2, Y2) of the bounding rectangle of next created node.

If current item is a node, sets or returns the coordinates of upper left point (X1, Y1) or lower right point (X2, Y2) of its bounding rectangle.

If current item is a link, writing those properties has no effect and reading returns the coordinates of upper left point (X1, Y1) or lower right point (X2, Y2) of its bounding rectangle.

Not available at design time.

Usage

[*form*.]*NET*.**X1**[= *numeric expression*]

[*form*.]*NET*.**Y1**[= *numeric expression*]

[*form*.]*NET*.**X2**[= *numeric expression*]

[*form*.]*NET*.**Y2**[= *numeric expression*]

Data Type

Long

See Also

Drawing

xGrid, yGrid Property

Description

Sets or returns the grid values in twips.

Usage

[*form*.]*NET*.**xGrid**[= *numeric expression*]

[*form*.]*NET*.**yGrid**[= *numeric expression*]

Data Type

Long

See Also

Capabilities

Org Property

Description

Sets the origin node of next created links (The value of the current item has no effect when writing this property).
If current item is 0, or if it is not a link, returns the origin node of next created links.
If current item is a link, returns its origin node.
Not available at design time.

Usage

[*form*.]*NET*.**Org**[= *idNode*]

Data Type

Long

Remarks

It is not possible to change directly the origin node 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 node and sets previous saved properties.

See Also

Navigation

Dst Property

Description

Sets the destination node of next created links (The value of the current item has no effect when writing this property).

If current item is 0, or if it is not a link, returns the destination node of next created links.

If current item is a link, returns its destination node.

Not available at design time.

Usage

[*form*.]*NET*.**Dst**[= *idNode*]

Data Type

Long

Remarks

It is not possible to change directly the destination node 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 destination node and sets previous saved properties.

See Also

Navigation

Item Property

Description

Sets or returns the current item (node or link). The current item is the selected one. Making an item be the current one allows to work with it (setting or getting its properties: position ,size, text, colors, etc).

Setting this property causes previous selection to disappear.

Not available at design time.

Usage

[*form*.]*NET*.**Item**[= *item*]

Data Type

Long

See Also

Items

IsLink Property

Description

Indicates if the current item is a link.
Not available at design time; read only at run time.

Usage

[*form*.]*NET*.**IsLink**

Settings

The IsLink property settings are:

Setting	Description
False	current item is 0 or it is a node
True	current item is not 0 and it is a link

Data Type

Integer (Boolean)

See Also

Items

LoopAction Property

Description

Specifies the type of item navigation to perform.
Not available at design time; write only at run time.

Usage

[*form*.]*NET*.**LoopAction**[= *setting*]

Settings

The LoopAction property settings are:

Setting	Description
0	all nodes
1	all links
2	all selected nodes
3	all links of a node

Data Type

Integer (enumerated)

Remarks

1. This property is to be used in conjonction with LoopCount and LoopItem properties:

- LoopAction specifies the type of loop to do: for instance a loop among all current node links (LoopAction = 3).
- After a call to LoopAction, LoopCount indicates the number of items involved in this loop.
- Finally, LoopItem allows to read each item and to perform any work with it.

2. Two calls to LoopAction property cannnot be cascaded.

See Also

Navigation

LoopCount Property

Description

Specifies the count of items involved in a navigation action performed by a call to LoopAction property.
Not available at design time; read only at run time.

Usage

[*form*.]*NET*.**LoopCount**

Data Type

Integer

Remarks

This property has to be called just after a call to LoopAction property.

See Also

Navigation

LoopItem Property

Description

Returns an item selected in a navigation action performed by a call to LoopAction property.

Not available at design time; read only at run time.

Usage

[*form*.]*NET*.**LoopItem**(index)

Data Type

Long

See Also

Navigation

PointCount Property

Description

If current item is 0 or is a node, writing this property has no effect and reading it returns 0.
If current item is a link, sets or returns the number of its points.
Not available at design time.

Usage

[*form*.]*NET*.**PointCount**[= *numeric expression*]

Data Type

Integer

Remarks

A link point is the point that joins two segments of a link. If a link has **n** points, it is composed of **n + 1** segments. The maximum value for the number of link points is **8**.

See Also

Drawing

PointX Property

Description

If current item is 0 or is a node, writing this property has no effect and reading it returns 0.
If current item is a link, sets or returns a long integer value that identifies an x position of a specified link point.
Not available at design time.

Usage

[*form*.]*NET*.**PointX**(index)[= *numeric expression*]

Data Type

Long

Remarks

If current item is a link reading this property has special meanings if index has a negative value between -1 and -4:

* **-1**: returns x position of intersection point between origin node border and link.
* **-2**: returns x position of intersection point between destination node border and link
* **-3:** if link is oriented, returns x position of one arrowhead point. If link is not oriented, it has the same effect as the case -2.
* **-4**: if link is oriented, returns x position of the other arrowhead point. If link is not oriented, it has the same effect as the case -2.

See Also

Drawing

Example *Print an arrow*

```
Dim i, nbpoint As Integer
Dim l, ptx1, pty1, ptx2, pty2, ptx3, pty3 As Long
Dim ptx(), pty() As Long

'Number of extra points
nbpoint = Net1.PointCount

'Alocate an array of nbpoint + 2
ReDim ptx(0 To nbpoint + 1)
ReDim pty(0 To nbpoint + 1)

'First point (intersection between origin node border and link)
ptx(0) = Net1.PointX(-1)
pty(0) = Net1.PointY(-1)

' Normal extra points
For l = 1 To nbpoint
  ptx(l) = Net1.PointX(l - 1)
  pty(l) = Net1.PointY(l - 1)
Next l

'Last point (intersection between destination node border and link)
```

```
ptx(nbpoint + 1) = Net1.PointX(-2)
pty(nbpoint + 1) = Net1.PointY(-2)

' Draw all link segments
For l = 0 To nbpoint
  printer.Line (ptx(l), pty(l))-(ptx(l+1), pty(l+1)), Net1.DrawColor
Next l

'Get point arrow head
ptx1 = Net1.PointX(-3)
pty1 = Net1.PointY(-3)
ptx2 = Net1.PointX(-4)
pty2 = Net1.PointY(-4)
ptx3 = ptx(nbpoint + 1)
pty3 = pty(nbpoint + 1)

'Draw arrow head
printer.Line (ptx1, pty1)-(ptx2, pty2), Net1.DrawColor
printer.Line (ptx1, pty1)-(ptx3, pty3), Net1.DrawColor
printer.Line (ptx3, pty3)-(ptx2, pty2), Net1.DrawColor
```

PointY Property

Description

If current item is 0 or is a node, writing this property has no effect and reading it returns 0.
If current item is a link, sets or returns a long integer value that identifies an y position of a specified link point.
Not available at design time.

Usage

[*form*.]*NET*.**PointY**(index)[= *numeric expression*]

Data Type

Long

Remarks

If current item is a link, reading this property has special meanings if index has a negative value between -1 and -4:

* **-1**: returns y position of intersection point between origin node border and link.
* **-2**: returns y position of intersection point between destination node border and link
* **-3:.**if link is oriented, returns y position of one arrowhead point. If link is not oriented, it has the same effect as the case -2.
* **-4**: if link is oriented, returns y position of the other arrowhead point. If link is not oriented, it has the same effect as the case -2.

See Also

Drawing

Example *Print an arrow*

```
Dim i, nbpoint As Integer
Dim l, ptx1, pty1, ptx2, pty2, ptx3, pty3 As Long
Dim ptx(), pty() As Long

'Number of extra points
nbpoint = Net1.PointCount

'Alocate an array of nbpoint + 2
ReDim ptx(0 To nbpoint + 1)
ReDim pty(0 To nbpoint + 1)

'First point (intersection between origin node border and link)
ptx(0) = Net1.PointX(-1)
pty(0) = Net1.PointY(-1)

' Normal extra points
For l = 1 To nbpoint
  ptx(l) = Net1.PointX(l - 1)
  pty(l) = Net1.PointY(l - 1)
Next l

'Last point (intersection between destination node border and link)
```

```
ptx(nbpoint + 1) = Net1.PointX(-2)
pty(nbpoint + 1) = Net1.PointY(-2)

' Draw all link segments
For l = 0 To nbpoint
  printer.Line (ptx(l), pty(l))-(ptx(l+1), pty(l+1)), Net1.DrawColor
Next l

'Get point arrow head
ptx1 = Net1.PointX(-3)
pty1 = Net1.PointY(-3)
ptx2 = Net1.PointX(-4)
pty2 = Net1.PointY(-4)
ptx3 = ptx(nbpoint + 1)
pty3 = pty(nbpoint + 1)

'Draw arrow head
printer.Line (ptx1, pty1)-(ptx2, pty2), Net1.DrawColor
printer.Line (ptx1, pty1)-(ptx3, pty3), Net1.DrawColor
printer.Line (ptx3, pty3)-(ptx2, pty2), Net1.DrawColor
```

Type Property

Description

If current item is 0, writing this property has no effect and reading it returns 0.
If current item is not 0, sets or returns its associated integer data.
Not available at design time.

Usage

[*form*.]*NET*.**Type**[= *setting*]

Data Type

Integer

Remarks

Typically, this property allows the user to define node or link types. Like Data property, the value of Type property is not used by the EasyNet control but only stored. The meaning of this property depends on the application that uses it.

See Also

Data Association

Data Property

Description

If current item is 0, writing this property has no effect and reading it returns 0.
If current item is not 0, sets or returns its associated long data.
Not available at design time.

Usage

[*form*.]*NET*.**Data**[= *setting*]

Data Type

Long

Remarks

Like Type property, the value of Data property is not used by the EasyNet control but only stored. The meaning of this property depends on the application that uses it.

See Also

Data Association

Text Property

Description

If current item is 0, writing this property has no effect and reading it returns an empty string.

If current item is not 0 (node or link), sets or returns the text associated with this item. The maximum length of an item text is **50** characters. The EasyNet control maintains the memory for the strings associated to items.

Not available at design time.

Usage

[*form*.]*NET*.**Text**[= *string expression*]

Data Type

String

Remarks

The string is truncated if it is longer than the maximum number of bytes allowed: **50**.

See Also

Data Association

Picture Property

Description

If current item is 0, sets or returns the picture to be displayed in next created nodes.

If current item is a node, sets or returns the picture to be displayed in this node. This picture can be a bitmap or an icon.

If current item is a link, writing this property has no effect and reading it returns 0.

Not available at design time.

Usage

[*form*.]*NET*.**Picture**[= *picture*]

Settings

The Picture Property settings are:

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

Data Type

Integer

See Also

Data Association

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.

Usage

[*form*.]*NET*.**SelectMode**[= {True | False}]

Settings

The SelectMode Property settings are:

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

Data Type

Integer (Boolean)

CanDrawNode Property

Description

Specify if you can create nodes.

Usage

[*form*.]*NET*.**CanDrawNode**[= {True | False}]

Settings

The CanDrawNode Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

CanDrawLink Property

Description

Specify if you can create links.

Usage

[*form*.]*NET*.**CanDrawLink**[= {True | False}]

Settings

The CanDrawLink Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

CanMoveNode Property

Description

Specify if you can move (drag) nodes.

Usage

[*form*.]*NET*.**CanMoveNode**[= {True | False}]

Settings

The CanMoveNode Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

CanSizeNode Property

Description

Specify if you can size (change dimensions) nodes.

Usage

[*form*.]*NET*.**CanSizeNode**[= {True | False}]

Settings

The CanSizeNode Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

CanStretchLink Property

Description

Specify if you can "stretch" links (i.e add or remove segments)

Usage

[*form*.]*NET*.**CanStretchLink**[= {True | False}]

Settings

The CanStretchLink Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

CanMultiLink Property

Description

Specify if you can have several links between two nodes.

Usage

[*form*.]*NET*.**CanMultiLink**[= {True | False}]

Settings

The CanMultiLink Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

MultiSel Property

Description

Specify that multiselection mode is possible or not.

Usage

[*form*.]*NET*.**MultiSel**[= {True | False}]

Settings

The MultiSel Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

ReadOnly Property

Description

Set "read only" mode.

Usage

[*form*.]*NET*.**ReadOnly**[= {True | False}]

Settings

The ReadOnly Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

ScrollBars Property

Description

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

Usage

[*form*.]*NET*.**ScrollBars**[= *setting*]

Settings

The ScrollBars Property settings are:

Setting	Description
0	(Default) No scrollbar.
1	Horizontal scrollbar.
2	Vertical scrollbar.
3	Both Horizontal and Vertical scrollbars.

Data Type

Integer (Enumerated)

See Also

Capabilities

EditAction Property

Description

Specifies an action that applies to selected items or that allows to select or unselect items.
Not available at design time; write only at run time.

Usage

[*form*.]*NET*.**EditAction**[= *setting*]

Settings

The EditAction property settings are:

Setting	Description
0	create a node
1	create a link
2	delete selected nodes (and their links)
3	select all nodes.
4	unselect.
5	copy selected nodes onto the clipboard in a metafile format.
6	clear network diagram (all items are deleted)

Data Type

Integer (enumerated)

Remarks

Link creation: The link that is created when setting EditAction to 1 is a link that links the nodes specified by Org and Dst properties. If one of this node is not valid, the link is not created.

Selection: Only nodes can be selected.by the user.

Delete: When a node is deleted, all its links are also deleted. A link cannot exist without its origin and destination nodes. If one of these two nodes is deleted, the link is also deleted.

See Also

Drawing

Change Event

Description

Occurs when a change is made. (For instance, an item is added, moved, deleted or one of its properties is changed).

Syntax

Sub *NET*_**Change** ()

SelChange Event

Description

Occurs when selection is changed.

Syntax

Sub *NET*_**SelChange** ()

AddNode Event

Description

Occurs when a node is added.

Syntax

Sub *NET*_**AddNode** ()

Remarks

Typically, this event allows the user to change a property of the node just after its creation and just before its display.

In fact when a node is created, three events are generated in the following order:

SelChange

AddNode

Change

AddLink Event

Description

Occurs when a link is added.

Syntax

Sub *NET*_**AddLink** ()

Remarks

Typically, this event allows the user to change a property of the link just after its creation and just before its display.

In fact when a link is created, three events are generated in the following order:

SelChange

AddLink

Change

ErrSpace Event

Description

Occurs when no more memory is available.

Syntax

Sub *NET*_**ErrSpace** ()

Remarks

This event will never occur in the current version of EasyNet control.
In fact, this event will be used in later versions.

Registration

The demonstartion version of the EasyNet control is fully functional but may only be used in the development environment. Any attempt to use this version in conjunction with an EXE file will display a dialog box identifying this as a demonstration version and the control will fail to load.

If you like EasyNet control then you can receive a full version by registering as follows:

1) Either in the SWREG forum on Compuserve. The fee is **$99** and the Registration ID number is **2547.**Then you will receive EasyNet by Compuserve E-Mail and the registration fee will be billed to your Compuserve Account.

2) Either by completing and sending the Order Form, along with a check for **FF 590** (French currency - includes s&h) or **$117** (includes cost of conversion + s&h) to:

Patrick Lassalle
2 , rue Gutenberg
92100, Boulogne
FRANCE

Then, you will receive the EasyNet control on diskette.

Registration benefits: In return for your registration you receive these benefits:

- the latest version of the EasyNet control **development** version.
- **license** file giving you the right to use EasyNet control development version. *This file is not for distribution.*
- full product **support** (via the MSBASIC forum on Compuserve) and free product upgrades for a period of 12 months.
- a royalty-free **run-time** version of EasyNet control, so that you can distribute the controls as part of your program. This run-time version is smaller than the development version.

License

The EasyNet control is not public domain or free software.

The EasyNet control is copyrighted, and all rights are reserved by its author: Patrick Lassalle.

You are granted a license to use EasyNet control for an evaluation period of 30 days. After 30 days, registration is required. You can distribute the demonstration programs of the distribution package. You may NOT distribute any other programs that utilizes the EasyNet control without obtaining a Registered User License for the EasyNet.

If you pay the registration fee, you may distribute the run-time version of EasyNet control with any or all products that use it with the exception that those products must contain a notice stating Patrick Lassalle's copyright of EasyNet control.

Disclaimer: This software is 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. The authors assume no liability for any alleged or actual damages arising from the use of this software. (Some states do not allow the exclusion of implied warranties, so the exclusion may not apply to you.)

Your use of this product indicates that you have read and agreed to these terms.

EasyNet Order Form (Select "Print Topic" from the File menu to print this order).

Date of order: _____

SHIPPING ADDRESS

Name _____

Company _____

Address _____

_____

_____

_____

Phone _____ **FAX** _____

PAYMENT ADRESS: **Patrick Lassalle**
2, rue Gutenberg
92100, Boulogne
FRANCE

Please send me:

EasyNet Custom Control Diskette:.....$117 (or FF 590) x _____

TOTAL _____

All payment must be by check in U.S. funds or French funds.
The US price includes our cost of conversion.
Sorry, at this time we cannot accept credit cards or "bill-me" orders.
Please make the check payable to Patrick Lassalle.

Installation

Demonstration version:The files **easynet.vbx** and **easynet.hlp** should be copied in your WINDOWS\SYSTEM directory.

Registered version:The files **easynet.vbx, easynet.hlp** and **easynet.lic** should be copied in your WINDOWS\SYSTEM directory.

Distribution note: When you create and distribute applications that use the EasyNet control you should install the file **easynet.vbx** (run-time version) in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install you applications correctly.

*You are not allowed to distribute **easynet.lic** file with any application that you distribute.*