EasyNet/OCX version1.1

EasyNet/OCX is a control that lets you quickly build flowchart-enabled applications.

File Name

EZNET16.OCX, EZNET32.OCX

Class Name

EasyNet

Contents

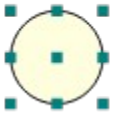
Quick Start under Visual Basic 4

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

- **Drag an EasyNet 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 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 (little squares) are displayed.

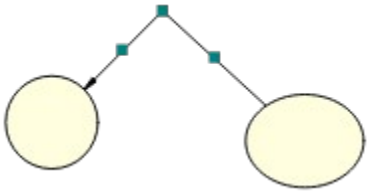


The handle at the center of the node is used to draw a link. The 8 others allow to **resize the node**. If you want to **move the node** you bring the mouse cursor into the node, press the left button, move the mouse and release the left button.

- **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).

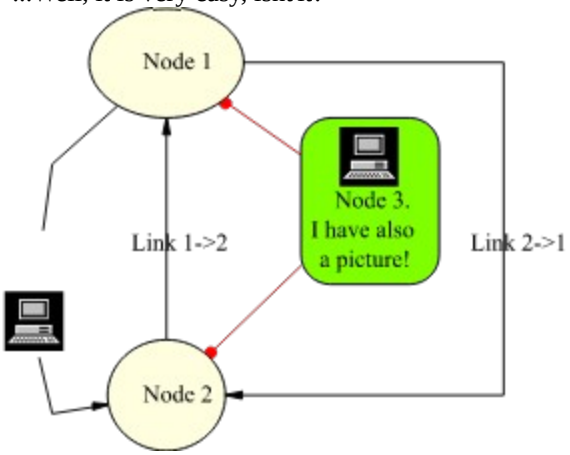


- **Now, you may return to the Visual Basic design-time mode** in order to change EasyNet control properties. For instance you may change the node filling color with FillColor property, the node shape (Shape property), the

drawing color (DrawColor property).

You may also create items programmaticaly with EditAction property or with AddNodeItem or AddLinkItem methods. Or copy the diagram to clipboard as a metafile, save its image to a file as a metafile, zoom the diagram, etc....

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



Why EasyNet?

If you need flowcharting features
If you want to implement a workflow applications
If you wish to draw organizational charts
If you have to draw communication networks
If you plan to draw state transitions diagrams
If you need to display relationships between entities (database diagrams)

then EasyNet is indispensable. GET IT!!

It allows you to draw diagrams interactively or programmaticaly in minutes.

EasyNet is powerful, opened and customizable:

- *allows to associate your own data to each item (node or link).*
- *allows navigation.*
- *offers many properties allowing you to "customize" your diagramming application.*
- *includes Royalty free runtime distribution*

Overview

EasyNet/OCX allows to draw flowchart diagrams. A flowchart diagram is a set of nodes that can be linked. Therefore, 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.

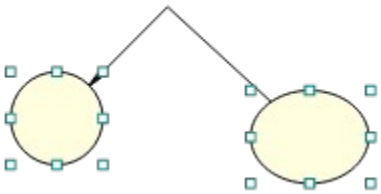
Drawings can be made interactively (with the mouse) or programmaticaly. See Quick Start to see how to interactively draw nodes, resize nodes, move nodes, stretch links, select one item or multiselect items.

By exploring following topics, you will discover all features of EasyNet/OCX.

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 (or with SetCurItem method)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.

Sleeping property allows to specify if an item is active or not. If it sleeps, the user cannot interactively make it current or selected.

Owner property allows to define an owner node for a node. When a node is created, it is free and its Owner property is 0. But if you set its Owner property, then the node will have to follow its owner node when it will be interactively moved with the mouse by the user. A node may have several owned nodes that follow it. And if those owned nodes are sleeping, they may be used to implement stubs or pins inside the owner node. Their role is just to receive links.

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. However, this behaviour may be changed with Owner property.

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.

ItemZOrder places current item at the front or back of the z-order.

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

```
Dim curLink&

With EasyNet1
  If .IsLink = True Then
    ' Save current item
    curLink = .Item

    ' Make origin node be the current item
    ' in order to work with it
    .Item = .Org

    ' Change node filling color
    .FillColor = RGB(255, 0, 0)

    ' Restore current item
    .Item = curLink
  End If
End With
```

Drawing

You can change colors, styles and shapes of each item either with Drawing Methods, either with following properties:

- X1, X2, Y1, Y2 properties allows to set or get position and size of each item.
- Picture property allows to associate a picture to each node.
- AutoSize property allows to adjust node size to picture size or adjust picture size to node size.
- Shape property allows to specify a shape for a node.
- DrawColor, DrawStyle 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.
- ForeColor property allows to specify the item text color.
- Text property associates a string that is displayed inside the node at a position depending on Alignment property (if item is a node) or near the link (if item is a link).
  The EasyNet control maintains the memory for the strings associated to items.
- Alignment sets or returns the alignment of text in 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.
- LinkHead property the arrowhead shape for a link.
- Transparent property specifies if a node is transparent or not.
- Hiding property specifies if an item (node or link) is visible or not.
- You can create items, delete items and do other edit actions (like copying the network diagram onto the clipboard in a metafile format) with either Edition Methods, either EditAction property.

Example:

Creates 3 nodes and 2 links. Each node has a text. Two are rectangles and the other is an ellipse. The links are oriented.

```
Sub Exercice ()
 Dim n1&, n2&, n3&

 ' Cause current item to be null
 ' Therefore, following property settings apply
 ' to next created items.
With Easynet1
 .Item = 0
 .Shape = 1  'Default shape = Rectangle.
 .FillColor = RGB(255, 255, 192) 'Default Fill color
 .DrawColor = RGB(0, 0, 128)     'Default Draw color
 .Oriented = True                'Oriented links

 ' Create first node. It has a rectangular shape.
 .EditAction = 0
 .X1 = 100
 .Y1 = 100
```

```
  .X2 = 2000
  .Y2 = 500
  .Text = "A network to implement ?"
  n1 = .Item

  ' Create second node. It has a rectangular shape.
  .EditAction = 0
  .X1 = 2200
  .Y1 = 300
  .X2 = 3600
  .Y2 = 700
  .Text = "FlowChart needs ?"
  n2 = .Item

  ' Create a third node. No shape is indicated.
  ' Therefore its shape is the default one: ellipse.
  .EditAction = 0
  .Shape = 0 ' Ellipse
  .X1 = 1100
  .Y1 = 1500
  .X2 = 3000
  .Y2 = 2000
  .Text = "Use EasyNet !!"
  n3 = .Item

  ' Create first link
  .Org = n1
  .Dst = n3
  .EditAction = 1

  ' Create second link with an extra point (2 segments)
  .Org = n2
  .Dst = n3
  .EditAction = 1
  .PointCount = 1
  .PointX(0) = 3200
  .PointY(0) = 1000

  ' Unselect last created link
  .Item = 0
 End With
End Sub
```

Metafile support

EasyNet/OCX offers a perfect metafile support:

- **Metafile copy**: you may copy an EasyNet diagram onto the clipboard and 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...

- **Metafile save**: you may save an image of your EasyNet diagram on disk as a metafile.

User Data Association

You can associate data to each item (node or link) either with User Data Methods, either with following properties:

- ItemTag property associates a string that is NOT displayed.
  The EasyNet control maintains the memory for the tags associated to items.
  This tag can be used to store user data.
- 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.

Navigation

You can navigate in an EasyNet/OCX diagram either with Navigation Methods, either with the three following 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.

LoopScope property allows to apply item property settings to all items involved in the loop.

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%
  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 leaving (out) links of the current node
    Net1.LoopAction = 4

    ' Get count of selected nodes
    nblink = Net1.LoopCount

    ' For each link leaving the current node...
    For j = 1 To nblink
      Net1.Item = Net1.LoopItem(j - 1)
      Net1.DrawColor = RGB(255, 0, 0)
    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
- DisplayHandles
- DoAddLink
- DoAddNode
- DoSelChange
- MultiSel
- ReadOnly
- ScrollBars
- ShowGrid
- xGrid
- yGrid
- Zoom

Saving/Loading

Saving an EasyNet diagram is under the responsability of the VB application that uses an EasyNet control. The ImageFile property used in conjunction with EditAction property only allows to save an image of the EasyNet diagram. This image file can be used by other drawing applications but it cannot be loaded up again by EasyNet.

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

Performance tuning

Setting following properties to False allows to increase speed dramatically:

- DoAddLink
- DoAddNode
- DoSelChange
- Repaint
- CheckItem

Example:

You may insert this portion of code each time you need to do a time consuming task like saving an EasyNet diagram or navigating in the diagram.

```
' Setting those properties to False improve speed
With Easynet1
  .Repaint = False
  .DoSelChange = False
  .DoAddNode = False
  .DoAddLink = False
  .CheckItem = False
End With
```

When you have terminated your task, you may reset those properties to True.

```
With Easynet1
  .Repaint = True
  .DoSelChange = True
  .DoAddNode = True
  .DoAddLink = True
  .CheckItem = True
End With
```

Limits

For one EasyNet control:

- the maximum number of items (nodes + links) is **16376**.
- the maximum number of link points is **254**.
  (therefore, the maximum number of link segments is **255**).

For each item, the Text setting is approximately **65,500** characters. (same setting for ItemTag property).

Properties

All the properties are listed below. Properties that apply only to EasyNet/OCX, 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)
Alignment
BackColor
AutoSize
BackPicture
BorderStyle
Caption
CanDrawNode
CanDrawLink
CanMoveNode
CanSizeNode
CanStretchLink
CanMultiLink
CheckItem
DisplayHandles
DoAddLink
DoAddNode
DoSelChange
Data
Dst
DragIcon
DragMode
DrawColor
DrawStyle
DrawWidth
EditAction
Enabled
FillColor
Font
ForeColor
Height
HelpContextId
Hiding
Hwnd
ImageFile
Index
IsLink
Item
ItemTag
ItemZOrder
Left
LinkHead
LoopAction
LoopCount
LoopItem
LoopScope
MouseIcon
MousePointer
MultiSel

Oriented
Org
Owner
Parent
Picture
PointCount
PointedArea
PointedItem
PointX
PointY
ReadOnly
Repaint
ShowGrid
ScrollBars
SelectMode
Shape
Sleeping
TabIndex
TabStop
Tag
Text
Top
Transparent
Type
Version
Visible
Width
X1
X2
xGrid
xScroll
Y1
Y2
yGrid
yScroll
Zoom

Events

All the events are listed below. Events that apply only to EasyNet/OCX, 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
Click
DblClick
DragDrop
DragOver
GotFocus
KeyDown
KeyPress
KeyUp
LostFocus
MouseDown
MouseMove
MouseUp
SelChange

Custom Methods

AddLinkItem
AddNodeItem
CopyAll
CopySel
DeleteAll
DeleteItem
DeleteSel
DoClick
GetDstNodesArray
GetDstNodesCount
GetInAndOutLinksArray
GetInAndOutLinksCount
GetInLinksArray
GetInLinksCount
GetItemDrawColor
GetItemDrawStyle
GetItemDrawWidth
GetItemForeColor
GetItemLong
GetItemsArray
GetItemsCount
GetItemShort
GetItemTag
GetItemText
GetLinkArrowHead
GetLinkDst
GetLinkedNodesArray
GetLinkedNodesCount
GetLinkOrg
GetLinkPointCount
GetLinkPointX
GetLinkPointY
GetLinksArray
GetLinksCount
GetNodeAlignment
GetNodeAutoSize
GetNodeFillColor
GetNodeOwner
GetNodePicture
GetNodeRect
GetNodesArray
GetNodesCount
GetNodeShape
GetOrgNodesArray
GetOrgNodesCount
GetOutLinksArray
GetOutLinksCount
GetOwnedNodesArray
GetOwnedNodesCount
GetSelNodesArray
GetSelNodesCount
IsChanged
IsItemHiding

IsItemLink
IsItemSleeping
IsLinkOriented
IsNodeTransparent
Refresh
SaveImage
SelectAll
SetChangedFlag
SetCurItem
SetItemDrawColor
SetItemDrawStyle
SetItemDrawWidth
SetItemForeColor
SetItemHiding
SetItemLong
SetItemShort
SetItemSleeping
SetItemTag
SetItemText
SetLinkArrowHead
SetLinkOriented
SetLinkPointCount
SetLinkPointX
SetLinkPointY
SetNodeAlignment
SetNodeAutoSize
SetNodeFillColor
SetNodeOwner
SetNodePicture
SetNodeRect
SetNodeShape
SetNodeTransparent
UnSelect
ZOrderItem

Navigation Methods

Following methods offers features equivalent to Navigation properties:

GetDstNodesArray	GetDstNodesCount
GetInAndOutLinksArray	GetInAndOutLinksCount
GetInLinksArray	GetInLinksCount
GetItemsArray	GetItemsCount
GetLinkedNodesArray	GetLinkedNodesCount
GetLinksArray	GetLinksCount
GetNodesArray	GetNodesCount
GetOrgNodesArray	GetOrgNodesCount
GetOutLinksArray	GetOutLinksCount
GetOwnedNodesArray	GetOwnedNodesCount
GetSelNodesArray	GetSelNodesCount

Edition Methods

Following methods offers features equivalent to EditAction and Item properties:

AddLinkItem
AddNodeItem
CopyAll
CopySel
DeleteAll
DeleteItem
DeleteSel
IsChanged
SaveImage
SelectAll
SetChangedFlag
SetCurItem
UnSelect

Drawing Methods

Following methods offers features equivalent to Drawing properties.

GetItemDrawColor
GetItemDrawStyle
GetItemDrawWidth
GetItemForeColor
GetItemText
GetLinkArrowHead
GetLinkDst
GetLinkOrg
GetLinkPointCount
GetLinkPointX
GetLinkPointY
GetNodeAlignment
GetNodeAutoSize
GetNodeFillColor
GetNodeOwner
GetNodePicture
GetNodeRect
GetNodeShape
IsItemHiding
IsItemLink
IsItemSleeping
IsLinkOriented
IsNodeTransparent
SetItemDrawColor
SetItemDrawStyle
SetItemDrawWidth
SetItemForeColor
SetItemHiding
SetItemSleeping
SetItemText
SetLinkArrowHead
SetLinkOriented
SetLinkPointCount
SetLinkPointX
SetLinkPointY
SetNodeAlignment
SetNodeAutoSize
SetNodeFillColor
SetNodeOwner
SetNodePicture
SetNodeRect
SetNodeShape
SetNodeTransparent
ZOrderItem

User Data Methods

Following methods offers features equivalent to User Data Association properties.

GetItemLong
GetItemShort
GetItemTag
SetItemLong
SetItemShort
SetItemTag

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

EasyNet1.**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)
7	copy all the diagram onto the clipboard in a metafile format.
8	the image of the diagram is written to disk as a metafile (.WMF). For this option to work, the ImageFile property must be set to provide a name for the file.

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 those nodes 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, Edition Methods

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.
If LoopScope property is True, writing applies to every nodes involved in a call to LoopAction property.
This property has no effect if Transparent property is set.

Usage

EasyNet1.**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	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)

Data Type

Long

See Also

Drawing

ForeColor Property

Description

If current item is 0, sets or returns the "current" item text color (the text color used for next created items).
If current item is not 0, sets or returns its text color.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.

Usage

EasyNet1.**ForeColor**[= *color* &]

Settings

The ForeColor property settings are:

Setting	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)

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.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.

Usage

EasyNet1.**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	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)

Data Type

Long

See Also

Drawing

DrawStyle Property

Description

If current item is 0, sets or returns the "current" drawing style (the drawing style used for next created items).
If current item is not 0, sets or returns the item drawing style.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.

Usage

EasyNet1.**DrawStyle**[= *size*]

Setting

The DrawStyle property settings are:

Setting	Description
0	(Default) Solid
1	Dash
2	Dot
3	Dash-Dot
4	Dash-Dot-Dot
5	Transparent
6	Inside Solid

Data Type

Integer (enumerated)

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.

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.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.

Usage

EasyNet1.**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.
If current item is a link, writing has no effect and reading returns 0.
If LoopScope property is True, writing applies to every nodes involved in a call to LoopAction property.

Usage

EasyNet1.**Shape**[= *shape*]

Settings

The Shape property settings are:

Setting	Description
0	Ellipse
1	Rectangle
2	Round rectangle
3	Diamond
4	North triangle
5	South triangle
6	East triangle
7	West triangle
8	Hexagon

By default, Shape is set to 0 (ellipse)

Data Type

Integer (enumerated)

See Also

Drawing

LinkHead Property

If current item is 0, sets or returns the "current" link arrowhead shape
(the arrowhead used for next created links).
If current item is a node, writing has no effect and reading returns 0.
If current item is a link, sets or returns its arrowhead
If LoopScope property is True, writing applies to every links involved in a call to LoopAction property.

Usage

EasyNet1.**LinkHead**[= *shape*]

Settings

The LinkHead property settings are:

Setting	**Description**
0	Filled arrow 15°
1	Filled circle
2	Empty arrow 15°
3	Empty circle
4	Filled arrow 30°
5	Empty arrow 30°
6	Filled arrow 45°
7	Empty arrow 45°

By default, LinkHead is set to 0

Data Type

Integer (enumerated)

See Also

Drawing

Alignment Property

Description

If current item is 0, sets or returns the "current" text alignment style
(the text alignment style used for next created nodes).
If current item is a node, sets or returns its text alignment style.
If current item is a link, writing has no effect and reading returns 0.
If LoopScope property is True, writing applies to every nodes involved in a call to LoopAction property.

Usage

EasyNet1.**Alignment**[= *alignment*]

Settings

The Alignment property settings are:

Setting	Description
0	Left - TOP
1	Left - MIDDLE
2	Left - BOTTOM
3	Right - TOP
4	Right - MIDDLE
5	Right - BOTTOM
6	Center - TOP
7	Center - MIDDLE
8	Center - BOTTOM

Data Type

Integer (enumerated)

See Also

Drawing

AutoSize Property

Description

Allows to adjust node size to picture size or adjust picture size to node size.
If current item is 0, sets or returns the "current" node autosize style
(the autosize style used for next created nodes).
If current item is a node, sets or returns its autosize style.
If current item is a link, writing has no effect and reading returns 0.
If LoopScope property is True, writing applies to every nodes involved in a call to LoopAction property.

Usage

EasyNet1.**Autosize**[= *autosize*]

Settings

The AutoSize property settings are:

Setting	Description
0	None
1	Adjust picture size to node size
2	Adjust node size to picture size

Data Type

Integer (enumerated)

See Also

Drawing

Transparent Property

Description

If current item is 0, specify if next created nodes will be transparent or not.
If current item is a node, specify if it is transparent or not.
If current item is a link, writing has no effect and reading returns 0.
If LoopScope property is True, writing applies to every nodes involved in a call to LoopAction property.

Usage

EasyNet1.**Transparent**[= {True | False}]

Settings

The Transparent property settings are:

Setting	Description
False	(default) Opaque
True	Transparent

Data Type

Integer (Boolean)

See also

Drawing

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.

If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.

Not available at design time.

Usage

EasyNet1.**X1**[= *numeric expression*]

EasyNet1.**Y1**[= *numeric expression*]

EasyNet1.**X2**[= *numeric expression*]

EasyNet1.**Y2**[= *numeric expression*]

Data Type

Long

See Also

Drawing

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

EasyNet1.**PointCount**[= *numeric expression*]

Data Type

Integer

Remarks

A link point is a 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 **254**.

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

EasyNet1.**PointX**(index)[= *numeric expression*]

Data Type

Long

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

EasyNet1.**PointY**(index)[= *numeric expression*]

Data Type

Long

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, it is displayed with an arrowhead at its destination node.
If LoopScope property is True, writing applies to every links involved in a call to LoopAction property.

Usage

EasyNet1.**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

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

EasyNet1.**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

EasyNet1.**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

EasyNet1.**Item**[= *item*]

Data Type

Long

See Also

Items , SetCurItem method

IsLink Property

Description

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

Usage

EasyNet1.**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 , IsItemLink method

Sleeping Property

Description

If current item is 0, it has no effect.
If current item is not 0, specify if it is in "sleeping mode" or not.
Not available at design time
When an item is in "sleeping mode", it is inactive and the user cannot interactively make it current or selected. He can do this only programmaticaly by saving its identifier in a global variable. Such an item can be used to display a bitmap or a text but the user cannot move, stretch or resize it with the mouse.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.

Usage

EasyNet1.**Sleeping** [= {True | False}]

Settings

The Sleeping property settings are:

Setting	Description
False	(default) The item is active.
True	The item is sleeping.

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

EasyNet1.**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
4	all links leaving current node (out links)
5	all links coming to current node (in links)
6	all nodes connected to a node (in and out nodes)
7	all destination nodes of current node
8	all origin nodes of current node
9	all owned nodes of current node
10	all items (nodes and links).

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 , Navigation Methods

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

EasyNet1.**LoopCount**

Data Type

Integer

Remarks

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

See Also

Navigation , Navigation Methods

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

EasyNet1.**LoopItem**(*index*)

Data Type

Long

See Also

Navigation , Navigation Methods

LoopScope Property

Description

When set to True, this property indicates that next item property settings will apply to all items involved in a call to LoopAction property.

Not available at design time

Usage

EasyNet1.**LoopScope**[= {True | False}]

Settings

The LoopScope Property settings are:

Setting	Description
False	(Default) No loop scope .
True	Loop scope is performed.

Data Type

Integer (Boolean)

Remark

Properties that may have a loop scope are the following:

Alignment	Data	DrawColor	DrawStyle
DrawWidth	FillColor	ForeColor	Hiding
LinkHead	Oriented	Owner	Picture
Shape	Sleeping	Transparent	Type
X1	Y1	X2	Y2

See Also

Navigation , Navigation Methods

Example:
Makes all selected nodes transparent.

```
With EasyNet1
  ' Do a loop with selected nodes
  .LoopAction = 2
  ' Indicates that next item property settings apply
  ' to all items in the loop.
  .LoopScope = True
  ' Cause all selected nodes to be transparent.
  .Transparent = True
  ' Reset loop scope to false
  .LoopScope = False
End With
```

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.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.
Not available at design time.

Usage

EasyNet1.**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.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.
Not available at design time.

Usage

EasyNet1.**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 EasyNet control maintains the memory for the strings associated to items.

Not available at design time.

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.

Usage

EasyNet1.**Text**[= *string expression*]

Data Type

String

Remarks

The Text setting is approximately **65,500** characters.

See Also

Drawing

ItemTag 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 a tag associated with this item. The EasyNet control maintains the memory for the tags associated to items.
Not available at design time.

Usage

EasyNet1.**ItemTag**[= *string expression*]

Data Type

String

Remarks

The Itemtag setting is approximately **65,500** characters.

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.
If LoopScope property is True, writing applies to every nodes involved in a call to LoopAction property.
Not available at design time.

Usage

EasyNet1.**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

Drawing

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 items. You bring the mouse cursor into the EasyNet 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 items by clicking them with the mouse and simultaneously pressing the shift or control key. You can select them again by using the same method.

Usage

EasyNet1.**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 interactively.

Usage

EasyNet1.**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 interactively.

Usage

EasyNet1.**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 interactively.

Usage

EasyNet1.**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 resize nodes interactively.

Usage

EasyNet1.**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) interactively.

Usage

EasyNet1.**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

EasyNet1.**CanMultiLink**[= {True | False}]

Settings

The CanMultiLink Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

MultiSel Property

Description

Specify if multiselection mode is possible or not.

Usage

EasyNet1.**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. In such a mode user interaction is not allowed.

Usage

EasyNet1.**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

EasyNet1.**ScrollBars**[= *setting*]

Settings

The ScrollBars Property settings are:

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

Data Type

Integer (Enumerated)

See Also

Capabilities

xGrid, yGrid Property

Description

Sets or returns the grid values in twips.

Usage

EasyNet1.**xGrid**[= *numeric expression*]

EasyNet1.**yGrid**[= *numeric expression*]

Data Type

Long

See Also

Capabilities

ShowGrid Property

Description

Specify if the grid is displayed or not.

Usage

EasyNet1.**ShowGrid**[= {True | False}]

Settings

The ShowGrid Property settings are:

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

Data Type

Integer (Boolean)

See Also

Capabilities

xScroll, yScroll Property

Description

Sets or returns the scroll values in twips.

Not available at design time.

Usage

EasyNet1.**xScroll**[= *numeric expression*]

EasyNet1.**yScroll**[= *numeric expression*]

Data Type

Long

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

Usage

EasyNet1.**PointedArea**

Settings

The PointedArea property settings are:

Setting	Description
0	Size NW-SE square area
1	Size N-S square area
2	Size NE-SW square area
3	Size W-E square area
4	Stretching square area
5	Linking square area
6	Node area
7	No special area.
8	Link area

Data Type

Integer

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.

PointedItem Property

Description

Returns the item identifier pointed by the mouse.

Not available at design time; read only at run time

Usage

EasyNet1.**PointedItem**

Data Type

Long

BackPicture Property

Description

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

DoAddLink Property

Description

Specify if AddLink event can be fired. Setting this property to False increases speed performance.

Usage

EasyNet1.**DoAddLink**[= {True | False}]

Settings

The DoAddLink Property settings are:

Setting	Description
False	AddLink event cannot be fired
True	(Default) AddLink event can be fired

Data Type

Integer (Boolean)

See Also

Capabilities
Performance tuning

DoAddNode Property

Description

Specify if AddNode event can be fired. Setting this property to False increases speed performance.

Usage

EasyNet1.**DoAddNode**[= {True | False}]

Settings

The DoAddNode Property settings are:

Setting	Description
False	AddNode event cannot be fired
True	(Default) AddNode event can be fired

Data Type

Integer (Boolean)

See Also

Capabilities
Performance tuning

DoSelChange Property

Description

Specify if SelChange event can be fired. Setting this property to False increases speed performance.

Usage

EasyNet1.**DoSelChange**[= {True | False}]

Settings

The DoSelChange Property settings are:

Setting	Description
False	SelChange event cannot be fired
True	(Default) SelChange event can be fired

Data Type

Integer (Boolean)

See Also

Capabilities
Performance tuning

Repaint Property

Description

Specify if repainting the EasyNet 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

Usage

EasyNet1.**Repaint**[= {True | False}]

Settings

The Repaint Property settings are:

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

Data Type

Integer (Boolean)

See Also

Performance tuning

CheckItem Property

Description

Specify if item checking is performed or not. Setting this property to False increases speed performance.

Important: Setting this property to False requires to be very cautious when using Item, Org and Dst properties.Setting wrong values to those properties when CheckItem is False may result in a General Protection Fault .

The same problem may happen with methods that have an item parameter. If this parameter is wrong (the item does not exist), you may have unpredictable results if CheckItem property is False.

Not available at design time.

Usage

EasyNet1.**CheckItem**[= {True | False}]

Settings

The CheckItem Property settings are:

Setting	Description
False	Item checking is not performed.
True	(Default) Item checking is performed

Data Type

Integer (Boolean)

See Also

Performance tuning

Version Property

Description

Returns the version of the EasyNet control currently loaded in memory.

Read only.

Usage

EasyNet1.**Version**

Data Type

Integer

Remarks

The version number is a three digit integer where the first digit is the major version number and the last two represent the minor version number. For example, if current version is 1.60, then this property returns 160.

Hiding Property

Description

If current item is 0, specify if next created items will be visible or not
If current item is not 0, specify if it is visible or not.
If LoopScope property is True, writing applies to every items involved in a call to LoopAction property.
Not available at design time

Usage

EasyNet1.**Hiding** [= {True | False}]

Settings

The Hiding property settings are:

Setting	Description
False	(default) The item is visible.
True	The item is not visible.

Data Type

Integer (Boolean)

See also

Drawing

ImageFile Property

Description

Sets a file name to which the metafile is written when EditAction is set to 8.
If a path is not specified, the current directory is used.

Usage

EasyNet1.**ImageFile** [= filename$]

Data Type

String

Remarks

The appropriate extension (.WMF) is appended automatically.

See also

EditAction

DisplayHandles Property

Description

Specify if handles are displayed. The handles are the little black squares on the selected item.

Usage

EasyNet1.**DisplayHandles**[= {True | False}]

Settings

The DisplayHandles Property settings are:

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

Data Type

Integer (Boolean)

Zoom 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.

Usage

EasyNet1.**Zoom**[= *setting*]

Data Type

Integer

ItemZOrder Property

Description

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

Usage

EasyNet1.**ItemZOrder** = *setting*

Settings

The ItemZOrder property settings are:

Setting	Description
0	Send item Front
1	Send item Back

Data Type

Integer

Remarks

If you perform a loop among all items (Net1.LoopAction = 10), items sent back will be at the beginning of the list whereas items sent front will be at the end of the list.

See also

Items

Owner Property

Description

If current item is a node, sets or returns its owner node.
If current item is 0 or is a link, writing has no effect and reading returns 0.
If LoopScope property is True, writing applies to every nodes involved in a call to LoopAction property.
Not available at design time.

Usage

EasyNet1.**Owner**[= *idNode*]

Data Type

Long

Remarks

- A node follows its owner. 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 X1 or Y1 properties), owned nodes do not move.
- A node cannot be an owner node if it is owned by another node.
- You can get each owned node of current node with LoopAction property.
- A node cannot owns itself.
- 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. You may make those little nodes sleeping (see Sleeping property) so that the user cannot select it, size it or move it.

SelChange Event

Description

Occurs when selection is changed.

Syntax

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

Remarks

- This event is not fired if DoSelChange property is False.
- Important: Actions that change selection (i.e. using Item Property) should not be used within this event as you will encounter unexpected results

AddNode Event

Description

Occurs when a node is added.

Syntax

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

Remarks

- This event is not fired if DoAddNode property is False.
- Important: Actions that create nodes (i.e. using EditAction Property) should not be used within this event as you will encounter unexpected results.
- Typically, this event allows the user to change a property of the node just after its creation and just before it is displayed. For instance, if you need fixed size nodes, you have just to give values to X1, X2, Y1, Y2 properties:

```
Sub Net1_AddNode ()
    With EasyNet1
      .X2 = .X1 + 500
      .Y2 = .Y1 + 500
    End With
End Sub
```

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

  SelChange

  AddNode

AddLink Event

Description

Occurs when a link is added.

Syntax

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

Remarks

- This event is not fired if DoAddLink property is False.
- Important: Actions that create links (i.e. using EditAction Property) should not be used within this event as you will encounter unexpected results.
- Typically, this event allows the user to change a property of the link just after its creation and just before it is displayed.
- In fact when a link is created, two events are generated in the following order:

  SelChange

  AddLink

AddLinkItem Method

Description

Creates a link and returns a long integer that identifies the created link.

Syntax

[*link* =] *EasyNet1*.**AddLinkItem** *origin*, *destination*

The arguments are:

Arguments	**Type**	**Description**
link	long	the returned created link
origin	long	the origin node of the link
destination	long	the destination node of the link

See Also

Drawing, Edition Methods

AddNodeItem Method

Description

Creates a node and returns a long integer that identifies the created node.

Syntax

[*node* =] *EasyNet1*.**AddNodeItem**

Arguments	**Type**	**Description**
node	long	the returned created node

See Also

Drawing, Edition Methods

CopyAll Method

Description

Copy all the diagram onto the clipboard in a metafile format.

Syntax

EasyNet1.**CopyAll**

See Also

Drawing, Edition Methods

CopySel Method

Description

Copy selected nodes onto the clipboard in a metafile format.

Syntax

EasyNet1.**CopySel**

See Also

Drawing, Edition Methods

DeleteAll Method

Description

Clears network diagram (all items are deleted)

Syntax

EasyNet1.**DeleteAll**

See Also

Drawing, Edition Methods

DeleteItem Method

Description

Deletes an item (node or link). If this item is a node, its links are also deleted.

Syntax

EasyNet1.**DeleteItem** *item*

The arguments are:

Arguments	**Type**	**Description**
item	long	the item to delete

See Also

Drawing, Edition Methods

DeleteSel Method

Description

Deletes selected nodes (and their links)

Syntax

EasyNet1.**DeleteSel**

See Also

Drawing, Edition Methods

DoClick Method

Description

Fires a click event.

Syntax

EasyNet1.**DoClick**

GetDstNodesArray Method

Description

Gets an array of all destination nodes of a node.

Syntax

EasyNet1.**GetDstNodesArray** *item*, *count*, *arrayDstNodes*

The arguments are:

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayDstNodes	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetDstNodesCount Method

Description

Gets count of all destination nodes of a node.

Syntax

[*count* =] *EasyNet1*.**GetDstNodesCount** *item*

Arguments	**Type**	**Description**
count	long	the returned count of all destination nodes
item	long	a node item identifier

See Also

Navigation , Navigation Methods

GetInAndOutLinksArray Method

Description

Gets an array of all links leaving from or going (in and out) to a node.

Syntax

EasyNet1.**GetInAndOutLinksArray** *item*, *count*, *arrayInAndOutLinks*

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayInAndOutLinks	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetInAndOutLinksCount Method

Description

Gets count of all links (In links and out links) of a node.

Syntax

[*count* =] *EasyNet1*.**GetInAndOutLinksCount** *item*

Arguments	**Type**	**Description**
count	long	the returned count of all links of a node
item	long	a node item identifier

See Also

Navigation , Navigation Methods

GetInLinksArray Method

Description

Gets an array of all links going (in links) to a node.

Syntax

EasyNet1.**GetInLinksArray** *item, count, arrayInLinks*

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayInLinks	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetInLinksCount Method

Description

Gets count of all entering links (In links) of a node.

Syntax

[*count* =] *EasyNet1*.**GetInLinksCount** *item*

Arguments	**Type**	**Description**
count	long	the returned count of all entering links
item	long	a node item identifier

See Also

Navigation , Navigation Methods

GetItemDrawColor Method

Description

Gets the drawing color of an item.

Syntax

[*color* =] *EasyNet1*.**GetItemDrawColor** *item*

Arguments	**Type**	**Description**
color	long	the returned drawing color
item	long	an item identifier

See Also

DrawColor property, SetItemDrawColor

GetItemDrawStyle Method

Description

Gets the drawing pen style used to draw an item.

Syntax

[*style* =] *EasyNet1*.**GetItemDrawStyle** *item*

Arguments	**Type**	**Description**
style	integer	the drawing pen style returned
item	long	a item identifier

See Also

DrawStyle property, SetItemDrawStyle

GetItemDrawWidth Method

Description

Gets the drawing pen width used to draw an item.

Syntax

[*width* =] *EasyNet1*.**GetItemDrawWidth** *item*

Arguments	**Type**	**Description**
width	integer	the returned drawing pen width
item	long	an item identifier

See Also

DrawWidth property, SetItemDrawWidth

GetItemForeColor Method

Description

Gets an item fore color.

Syntax

[*color* =] *EasyNet1*.**GetItemForeColor** *item*

Arguments	**Type**	**Description**
color	long	the returned fore color
item	long	an item identifier

See Also

ForeColor property, SetItemForeColor

GetItemLong Method

Description

Returns the long integer that is associated to an item.

Syntax

[*data* =] *EasyNet1*.**GetItemLong** *item*

Arguments	**Type**	**Description**
data	long	the returned long integer value associated to an item
item	long	an item identifier

See Also

Data property, SetItemLong, User Data Methods

GetItemsArray Method

Description

Gets an array of all items.

Syntax

EasyNet1.GetItemsArray *count, arrayItems*

Arguments	**Type**	**Description**
count	long	the size of the array.
arrayItems	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetItemsCount Method

Description

Gets count of all items.

Syntax

[*count* =] *EasyNet1*.**GetItemsCount**

Arguments	Type	Description
count	long	the returned count of all items.

See Also

Navigation , Navigation Methods

GetItemShort Method

Description

Returns the short integer that is associated to an item.

Syntax

[*data* =] *EasyNet1*.**GetItemShort** *item*

Arguments	Type	Description
data	integer	the returned integer value associated to an item
item	long	an item identifier

See Also

Type property, GetItemShort, User Data Methods

GetItemTag Method

Description

Returns associated tag of an item.

Syntax

[*tag* =] *EasyNet1*.**GetItemTag** *item*

Arguments	**Type**	**Description**
tag	string	the returned associated tag
item	long	an item identifier

See Also

ItemTag property, SetItemTag, User Data Methods

GetItemText Method

Description

Returns the associated text of an item.

Syntax

[*text* =] *EasyNet1*.**GetItemText** *item*

Arguments	**Type**	**Description**
text	string	the returned associated text
item	long	an item identifier

See Also

Text property, SetItemText

GetNodeAlignment Method

Description

Gets the text alignment style of a node

Syntax

[*alignment* =]*EasyNet1*.**GetNodeAlignment** *item*

Arguments	**Type**	**Description**
alignment	integer	the returned text alignment style
item	long	a node item identifier

See Also

Alignment property, SetNodeAlignment method.

GetNodeAutoSize Method

Description

Gets the node AutoSize mode.

Syntax

[*autosize* =] *EasyNet1*.**GetNodeAutoSize** *item*

Arguments	**Type**	**Description**
autosize	integer	the returned autosize mode
item	long	an item identifier

See Also

AutoSize property, SetNodeAutoSize method.

GetLinkArrowHead Method

Description

Gets the arrow shape of a link.

Syntax

[*arrowhead* =] *EasyNet1*.**GetLinkArrowHead** *item*

Arguments	**Type**	**Description**
arrowhead	integer	the returned arrow shape of a link
item	long	a link item identifier

See Also

LinkHead property, SetLinkArrowHead method.

GetLinkDst Method

Description

Returns destination node identifier of a link.

Syntax

[*dst* =] *EasyNet1*.**GetLinkDst** *item*

Arguments	**Type**	**Description**
dst	long	the returned destination node identifier
item	long	a link item identifier

GetLinkedNodesArray Method

Description

Gets an array of all linked nodes of a node.

Syntax

EasyNet1.**GetLinkedNodesArray** *item, count, arrayLinkedNodes*

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayLinkedNodes	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetLinkedNodesCount Method

Description

Gets the count of all linked nodes of a node.

Syntax

[*count* =] *EasyNet1*.**GetLinkedNodesCount** *item*

Arguments	Type	Description
count	long	the returned count of all linked nodes of a node
item	long	a node item identifier

See Also

Navigation , Navigation Methods

GetLinkOrg Method

Description

Returns origin node identifier of a link.

Syntax

[*org* =] *EasyNet1*.**GetLinkOrg** *item*

Arguments	**Type**	**Description**
org	long	the returned origin node identifier
item	long	a link item identifier

GetLinkPointCount Method

Description

Returns the number of points of a link.

Syntax

[*count* =] *EasyNet1*.**GetLinkPointCount** *item*

Arguments	**Type**	**Description**
count	long	the returned count of points of a link
item	long	a link item identifier

See Also

SetLinkPointCount

GetLinkPointX Method

Description

Returns a long integer value that identifies an x position of a specified link point.

Syntax

[*xPos* =] *EasyNet1*.**GetLinkPointX** *item*, *index*

Arguments	**Type**	**Description**
xPos	long	the returned x position of the specified point
item	long	a link item identifier
index	integer	the index of the link point. The first point has index 1 and the last has an index equal to the point count.

See Also

SetLinkPointX method

GetLinkPointY Method

Description

Returns a long integer value that identifies an y position of a specified link point.

Syntax

[*yPos* =] *EasyNet1*.**GetLinkPointY** *item, index*

Arguments	**Type**	**Description**
yPos	long	the returned y position of the specified point
item	long	a link item identifier
index	integer	the index of the link point. The first point has index 1 and the last has an index equal to the point count.

See Also

SetLinkPointY method

GetLinksArray Method

Description

Gets an array of all links

Syntax

EasyNet1.**GetLinksArray** *count, arrayItems*

Arguments	**Type**	**Description**
count	long	the size of the array.
arrayItems	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetLinksCount Method

Description

Gets count of all links.

Syntax

[*count* =] *EasyNet1*.**GetLinksCount**

Arguments	**Type**	**Description**
count	long	the returned count of all link.

See Also

Navigation , Navigation Methods

GetNodeFillColor Method

Description

Gets a node fill color.

Syntax

[*color* =] *EasyNet1*.**GetNodeFillColor** *item*

Arguments	**Type**	**Description**
color	long	the returned filling color
item	long	a node item identifier

See Also

FillColor property, SetNodeFillColor method.

GetNodeOwner Method

Description

Gets the owner node of a node.

Syntax

[*owner* =] *EasyNet1*.**GetNodeOwner** *item*

Arguments	**Type**	**Description**
owner	long	the returned owner node identifier
item	long	a node item identifier

See Also

Owner propety, SetNodeOwner method.

GetNodePicture Method

Description

Gets a node picture reference.

Syntax

[*picture* =] *EasyNet1*.**GetNodePicture** *item*

Arguments	**Type**	**Description**
picture		the returned picture reference
item	long	a node item identifier

Picture property, SetNodePicture method.

GetNodeRect Method

Description

Gets node upper left point and lower right point.

Syntax

EasyNet1.**GetNodeRect** *item*, *X1*, *Y1*, *X2*, *Y2*

Arguments	**Type**	**Description**
item	long	a node item identifier
X1	long	upper left point x coordinate
Y1	long	upper left point y coordinate
X2	long	lower right point x coordinate
Y2	long	lower right point y coordinate

See Also

X1, Y1, X2, Y2 properties, SetNodeRect method.

GetNodesArray Method

Description

Gets an array of all nodes.

Syntax

EasyNet1.**GetNodesArray** *count, arrayItems*

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayItems	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetNodesCount Method

Description

Gets count of all nodes.

Syntax

[*count* =] *EasyNet1*.**GetNodesCount**

Arguments	**Type**	**Description**
count	long	the returned count of all nodes.

See Also

Navigation , Navigation Methods

GetNodeShape Method

Description

Gets node shape.

Syntax

[*shape* =] *EasyNet1*.**GetNodeShape** *item*

Arguments	**Type**	**Description**
shape	integer	the returned node shape
item	long	a node item identifier

See Also

Shape property, SetNodeShape method.

GetOrgNodesArray Method

Description

Gets an array of all origin nodes of a node.

Syntax

EasyNet1.**GetOrgNodesArray** *item*, *count*, *arrayDstNodes*

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayDstNodes	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetOrgNodesCount Method

Description

Gets count of all origin nodes.

Syntax

[*count* =] *EasyNet1*.**GetOrgNodesCount** *item*

Arguments	**Type**	**Description**
count	long	the returned count of all origin nodes
item	long	a node item identifier

See Also

Navigation , Navigation Methods

GetOutLinksArray Method

Description

Gets an array of all links leaving (out links) a node.

Syntax

EasyNet1.**GetOutLinksArray** *item*, *count*, *arrayOutLinks*

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayOutLinks	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetOutLinksCount Method

Description

Gets count of all links leaving (out links) of a node.

Syntax

[*count* =] *EasyNet1*.**GetOutLinksCount** *item*

Arguments	**Type**	**Description**
count	long	the returned count of all leaving links
item	long	a node item identifier

See Also

Navigation , Navigation Methods

GetOwnedNodesArray Method

Description

Gets an array of all owned nodes of a node.

Syntax

EasyNet1.**GetOwnedNodesArray** *item, count, arrayOwnedNodes*

Arguments	**Type**	**Description**
item	long	a node item identifier
count	long	the size of the array.
arrayOwnedNodes	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetOwnedNodesCount Method

Description

Gets count of all owned nodes of a node.

Syntax

[*count* =] *EasyNet1*.**GetOwnedNodesCount** *item*

Arguments	**Type**	**Description**
count	long	the returned count of all owned nodes
item	long	a node item identifier

See Also

Navigation , Navigation Methods

GetSelNodesArray Method

Description

Gets an array of all selected nodes

Syntax

EasyNet1.**GetSelNodesArray** *count*, *arraySelNodes*

Arguments	**Type**	**Description**
count	long	the size of the array.
arraySelNodes	array of long	the array used to store the item identifiers. (under VB, pass the first element of the arrray)

See Also

Navigation , Navigation Methods

GetSelNodesCount Method

Description

Gets count of all selected nodes

Syntax

[*count* =] *EasyNet1*.**GetSelNodesCount**

Arguments	**Type**	**Description**
count	long	the returned count of all selected nodes.

See Also

Navigation , Navigation Methods

IsChanged Method

Description

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

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

Syntax

[*changed* =] *EasyNet1*.**IsChanged**

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

See Also

SetChangedFlag method.

IsItemHiding Method

Description

Indicates if an item is hidden.

Syntax

[*hiding* =] *EasyNet1*.**IsItemHiding** *item*

Arguments	**Type**	**Description**
hiding	boolean	True if the item is hidden, False elsewhere.
item	long	an item identifier

IsItemLink Method

Description

Indicates if an item is a link.

Syntax

[*islink* =] *EasyNet1*.**IsItemLink** *item*

Arguments	**Type**	**Description**
islink	boolean	True if the item is a link, False elsewhere.
item	long	an item identifier

See Also

Items

IsItemSleeping Method

Description

Indicates if an item is in sleeping mode.

Syntax

[sleeping =] *EasyNet1*.IsItemSleeping item

Arguments	**Type**	**Description**
sleeping	boolean	True if the item is sleeping, False elsewhere.
item	long	an item identifier

See Also

Items, SetItemSleeping method.

IsLinkOriented Method

Description

Indicates if a link is oriented.

Syntax

[*oriented* =]*EasyNet1*.**IsLinkOriented** *item*

Arguments	**Type**	**Description**
oriented	boolean	True if the link is oriented, False elsewhere.
item	long	a link item identifier

IsNodeTransparent Method

Description

Indicates if a node is transparent.

Syntax

[*transparent* =] *EasyNet1*.**IsNodeTransparent** *item*

Arguments	**Type**	**Description**
transparent	boolean	True if the node is transparent, False elsewhere.
item	long	a node item identifier

Refresh Method

Description

Forces a repaint of the OLE control.

Syntax

EasyNet1.**Refresh**

SaveImage Method

Description

Writes the image of the diagram to disk as a metafile (.WMF).

Syntax

EasyNet1.**SaveImage** *ImageFile*

Arguments	**Type**	**Description**
ImageFile	string	name of file used to store metafile image

See Also

Drawing, Edition Methods

SelectAll Method

Description

Selects all nodes

Syntax

EasyNet1.**SelectAll**

See Also

Drawing, Edition Methods

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

EasyNet1.**SetChangedFlag** *Changed*

Arguments	**Type**	**Description**
Changed	integer	False if no change.

See Also

IsChanged method.

SetCurItem Method

Description

Makes an item the current one.

Syntax

EasyNet1.**SetCurItem** *item*

Arguments	**Type**	**Description**
item	long	an item identifier

See Also

Item property

See Also

Items, Edition Methods

SetItemDrawColor Method

Description

Sets the drawing pen color used to draw an item.

Syntax

EasyNet1.**SetItemDrawColor** *item, color*

Arguments	Type	Description
item	long	an item identifier
color	long	Draw color

See Also

DrawColor property, GetItemDrawColor method

SetItemDrawStyle Method

Description

Sets the drawing pen style used to draw an item.

Syntax

EasyNet1.**SetItemDrawStyle** *item*, *style*

Arguments	**Type**	**Description**
item	long	an item identifier
style	integer	drawing style (See DrawStyle property settings)

See Also

DrawStyle property, GetItemDrawStyle method

SetItemDrawWidth Method

Description

Sets the drawing pen width used to draw an item.

Syntax

EasyNet1.**SetItemDrawWidth** *item, width*

Arguments	**Type**	**Description**
item	long	an item identifier
width	integer	the drawing pen width

See Also

DrawWidth property, GetItemDrawWidth method

SetItemForeColor Method

Description

Sets the fore color used to display the text of an item.

Syntax

EasyNet1.**SetItemForeColor** *item, color*

Arguments	**Type**	**Description**
item	long	an item identifier
color	long	fore color

See Also

ForeColor property, GetItemForeColor method

SetItemHiding Method

Description

Specifies if an item is hiding or not.

Syntax

EasyNet1.**SetItemHiding** *item, hiding*

Arguments	**Type**	**Description**
item	long	an item identifier
hiding	boolean	True if Hiding, False if not

See Also

Hiding property, IsItemHiding method

SetItemLong Method

Description

Associates a long integer to an item.

Syntax

EasyNet1.**SetItemLong** *item, data*

Arguments	Type	Description
item	long	an item identifier
data	long	a long integer value

Remark

This long integer is not used by the EasyNet control but only stored. Its meaning depends on the application that uses it.

See Also

Data property, GetItemLong method, User Data Methods

SetItemShort Method

Description

Associates a short integer to an item.

Syntax

EasyNet1.**SetItemShort** *item, data*

Arguments	**Type**	**Description**
item	long	an item identifier
data	integer	an integer value

Remark

Typically, this method allows the user to define node or link types. This short integer is not used by the EasyNet control but only stored. Its meaning depends on the application that uses it.

See Also

Type property, GetItemShort method, User Data Methods

SetItemSleeping Method

Description

Specifies if an item is sleeping or not.

Syntax

EasyNet1.**SetItemSleeping** *item*, *sleeping*

Arguments	**Type**	**Description**
item	long	an item identifier
sleeping	boolean	True if Sleeping, False if not

See Also

Sleeping property, IsItemSleeping method

SetItemTag Method

Description

Associates a string to an item. This string is just stored, not displayed.

Syntax

EasyNet1.**SetItemTag** *item, tag*

Arguments	**Type**	**Description**
item	long	an item identifier
tag	string	the tag string

See Also

ItemTag property, GetItemTag method, User Data Methods

SetItemText Method

Description

Associates a string to an item. This string is stored, and displayed.

Syntax

EasyNet1.**SetItemText** *item, text*

Arguments	**Type**	**Description**
item	long	an item identifier
text	string	the text string

See Also

Text property, GetItemText method

SetLinkArrowHead Method

Description

Specifies the arrow shape of a link. This arrow is displayed only if the link is oriented.

Syntax

EasyNet1.**SetLinkArrowHead** *item*, *head*

Arguments	**Type**	**Description**
item	long	an item identifier
head	integer	the link arrow head (See LinkHead property settings)

See Also

LinkHead property, GetLinkArrowHead method.

SetLinkOriented Method

Description

Specifies if a link is oriented or not.

Syntax

EasyNet1.**SetLinkOriented** *item, oriented*

Arguments	Type	Description
item	long	an item identifier
oriented	boolean	True if Oriented, False if not

See Also

Oriented property, IsLinkOriented method

SetLinkPointCount Method

Description

Sets the number of points of a link.

Syntax

EasyNet1.**SetLinkPointCount** *item*, *count*

Arguments	**Type**	**Description**
item	long	an item identifier
count	integer	the count of points

See Also

GetLinkPointCount method

SetLinkPointX Method

Description

Sets a long integer value that identifies an x position of a specified link point.

Syntax

EasyNet1.**SetLinkPointX** *item, index, xpoint*

Arguments	**Type**	**Description**
item	long	an item identifier
index	integer	the index of the link point. The first point has index 1 and the last has an index equal to the point count.
xpoint	long	x position of link point

See Also

GetLinkPointX method

SetLinkPointY Method

Description

Sets a long integer value that identifies an y position of a specified link point.

Syntax

EasyNet1.**SetLinkPointY** *item*, *index*, *ypoint*

Arguments	Type	Description
item	long	an item identifier
index	integer	the index of the link point. The first point has index 1 and the last has an index equal to the point count.
ypoint	long	y position of link point

See Also

GetLinkPointY method

SetNodeAlignment Method

Description

Sets the text alignment style of a node.

Syntax

EasyNet1.***SetNodeAlignment*** *item, alignment*

Arguments	**Type**	**Description**
item	long	an item identifier
alignment	integer	Alignment value See Alignment property settings

See Also

Alignment property, GetNodeAlignment method.

SetNodeAutoSize Method

Description

Allows to adjust node size to picture size or adjust picture size to node size.

Syntax

EasyNet1.**SetNodeAutoSize** *item*, *autosize*

Arguments	**Type**	**Description**
item	long	an item identifier
autosize	integer	AutoSize value See AutoSize property settings

See Also

AutoSize property, GetNodeAutoSize method.

SetNodeFillColor Method

Description

Sets the filling color of a node.

Syntax

EasyNet1.**SetNodeFillColor** *item*, *color*

Arguments	**Type**	**Description**
item	long	an item identifier
color	long	filling color

See Also

FillColor property, GetNodeFillColor method.

SetNodeOwner Method

Description

Sets the owner node of a node.

Syntax

EasyNet1.**SetNodeOwner item**, *owner*

Arguments	**Type**	**Description**
item	long	an item identifier
owner	long	the owner item identifier

See Also

Owner propety, GetNodeOwner method.

SetNodePicture Method

Description

Sets the node picture. This picture may be a bitmap or an icon.

Syntax

EasyNet1.**SetNodePicture** *item, picture*

Arguments	**Type**	**Description**
item	long	an item identifier
picture		a picture reference

See Also

Picture property, GetNodePicture method.

SetNodeRect Method

Description

Sets the coordinates of upper left point (X1, Y1) or lower right point (X2, Y2) of the bounding rectangle of a node.

Syntax

EasyNet1.**SetNodeRect** *item, X1, Y1, X2, Y2*

Arguments	**Type**	**Description**
item	long	an item identifier
X1	long	upper left point x coordinate
Y1	long	upper left point y coordinate
X2	long	lower right point x coordinate
Y2	long	lower right point y coordinate

See Also

X1, Y1, X2, Y2 properties, GetNodeRect method.

SetNodeShape Method

Description

Specifies a node shape.

Syntax

EasyNet1.**SetNodeShape** *item*, *shape*

Arguments	Type	Description
item	long	an item identifier
shape	integer	Shape value See Shape property settings

See Also

Shape property, GetNodeShape method.

SetNodeTransparent Method

Description

Specifies if a node is transparent or not.

Syntax

EasyNet1.**SetNodeTransparent** *item*, *transparent*

Arguments	**Type**	**Description**
item	long	an item identifier
transparent	boolean	True if Transparent, False if Opaque

See Also

Transparent property, IsNodeTransparent method.

UnSelect Method

Description

Unselects items.

Syntax

EasyNet1.**UnSelect**

See Also

Drawing, Edition Methods

ZOrderItem Method

Description

Places an item at the front or back of the z-order.

Syntax

EasyNet1.**ZOrderItem** *item*, *zorder*

Arguments	**Type**	**Description**
item	long	an item identifier
zorder	integer	Send item Back if 1, Front if 0

Registration

In the development environment, any attempt to use EasyNet/OCX without a license file will display a dialog box explaining that EasyNet/OCX is used without license.

If you generate an EXE file with EasyNet/OCX but without a license file, then any attempt to use this EXE file will display also a dialog box explaining that it has been generated without license file and that the EasyNet control will not work.

If you like EasyNet/OCX then you can receive the license file by registering as follows:

1) EITHER in the SWREG forum on Compuserve:

License type	SWREG id	Price
Single User	8739	$ 199

Then you will receive the EasyNet/OCX license file by Compuserve E-Mail and the registration fee will be billed to your Compuserve Account. This is a quick and easy way to register EasyNet/OCX.

2) EITHER by ordering with MC, Visa, Amex, or Discover from Public (software) Library by calling 800-2424-PsL or 713-524-6394 or by FAX to 713-524-6398 or by CIS Email to 71355,470. You can also mail credit card orders to PsL at P.O.Box 35705, Houston, TX 77235-5705. Ask for product # 11517 and say that you want the OCX version The cost is $ 202 (includes $3 s&h charge). Then, you will receive the EasyNet/OCX license file on diskette.

Note: THE ABOVE NUMBERS ARE FOR ORDERS ONLY. Please address any questions to Patrick Lassalle through CIS e-mail.

3) EITHER by completing and sending the Order Form, along with a check for the amount listed above (plus $3 s&h if a diskette is used instead of E-Mail)

to:

Patrick Lassalle
247, Avenue du Marechal Juin
92100, Boulogne
FRANCE

Then, you will receive the EasyNet/OCX license file either on diskette or via E-Mail if possible.

If you want to pay with french currency, prices are indicated in the Order Form.

Note: The documention consists in the help file: **EZNET.HLP**.

Registration benefits:

In return for your registration you receive these benefits:
- a **license** file giving a royalty-free right to reproduce and distribute the control file (16 bits version AND 32 bits version) with any application that you develop and distribute.
- full product Support for a period of 12 months.
- you are entitled to free upgrades for a period of 12 months.
- the right to use EasyNet/OCX in your design environment.

License Agreement

EasyNet/OCX is not public domain or free software.

Licensing:

1. shareware version

You may use the shareware version of EasyNet/OCX 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.

2. registered version

As a registered user, you can use EasyNet/OCX in your design environment and you have a royalty-free right to distribute executables that use EasyNet/OCX as a runtime component. Only registered users can distribute executables using EasyNet/OCX.

You may copy the software to facilitate your use of it on as many computers as there are licensed users specified in the license file. Making copies for any other purpose violates international copyright laws. In particular, you are prohibited from distributing a registered version of EasyNet/OCX, except as a runtime component of one of your applications.

The license file allows you to compile your applications with EasyNet/OCX. **You are not allowed to distribute the license file EZNET.LIC with any application that you distribute.**

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.

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

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

Date of order: _____

SHIPPING ADDRESS

Name _____

Company _____

Address _____

_____

_____

_____

Phone _____

FAX _____

E_Mail _____

PAYMENT ADRESS: **Patrick Lassalle**
247 , Avenue du Marechal Juin
92100, Boulogne
FRANCE

Please send me the EasyNet/OCX last version :

Single User License	**US $ 199**	(or FF 995)	**x**	_____
s & h. (if diskette required)	**US $ 3**	(or FF 15)		_____
		TOTAL		_____

The diskette contains the EasyNet/OCX package in a zip file and the license file.
Those files may be sent via e-mail. In such a case, s & h is not to be included.
All payment must be by check in U.S. funds or French funds.
Please make the check payable to Patrick Lassalle.
Prices and terms subject to change without notice.

Installation

To install EasyNet/OCX on your system, follow these steps :

1) copy all the following files into the same directory:

EZNET32.OCX	32-bit OLE Control (if you work under 32 bits environment
EZNET16.OCX	16-bit OLE Control (if you work under 16 bits environment)
EZNET.HLP	On-line documentation
EZNET.LIC	License file (if you have a registered version)

2) register the OCX. (with regsrv32, or regsrv16 tools or directly with VB4)

When you create and distribute applications that use EasyNet/OCX, you could install the OCX file in the same directory as the application.

Note: You are not allowed to distribute the license file EZNET.LIC with any application that you distribute.

Converting from VBX

EasyNet/OCX is compatible with the VBX version (EasyNet.vbx)

If you want existing VB3 projects using EasyNet/VBX to be automatically converted to use EasyNet/OCX, edit the VB.INI file and add the following lines (we suppose here that you have installed the OCX in your \win95\system directory) :

[VBX Conversions32]

easynet.vbx={54EFCB40-0A49-101D-A9C2-02D22A48786A}#1.1#0; C:\win95\system\eznet32.ocx

[VBX Conversions16]

easynet.vbx={54EFCB40-0A49-101D-A9C2-02D22A48786A}#1.1#0; C:\win95\system\eznet16.ocx

Differences between the VBX and OCX versions

1) General

In the OCX version, ScrollBars property is set to 3 by default (both Horizontal and Vertical scrollbars are set).

In the OCX version, the default value of CanMultiLink property is True.

2) Properties:

- DoChange property disappears in the OCX since the Change event is removed.
- MouseIcon property is a new OCX property.
- The VBX font properties (FontName, FontSize, etc...) are replaced by the OCX Font property.

3) Events:

- Change event is replaced by 2 methods: SetChangedFlag and IsChanged.
- ErrSpace event disappears in the OCX.

4) Methods:

Methods are a new OCX feature.

Support

EasyNet/OCX support can be obtained

- via Compuserve: **100325,725**
- via Internet: **100325.725@compuserve.com**
- at the address indicated in Registration

Thanks in advance for your feedbacks or questions!