Contents

Introduction

This ActiveX control is an encryption control based upon the WWII German Enigma encryption machine. It is one of the world's most known and popular ways to encrypt data. Enigma has multiple rotors, ring settings, start positions, and an advanced rotor and reflector generator using numeric seeds. This gives it the most advanced and strongest protection against decryption methods such as the brute force attack.

Properties

RotorOrder1..
RingSetting1..
StartPosition1..
SourceFile
TargetFile
Text
Index
Left
Name
Top
Tag
ReflectorSeed
RotorSeed

RotorOrder1..

RotorOrder1..10 refers to all of the rotor order properties from 1 to 10. RotorOrder1 is the property that stores the numer that sets that particular ring.

Syntax

object.RotorOrder1 [= expression]

The RotorOrder1 property syntax has these parts:

object An object expression that evaluates to an object in the Applies To list.
expression A double expression identifying the object. The default is the number 1.

Reference

Rotor orders are used to change the way the algorithm encrypts data.

See Also

RingSetting1..
StartPosition1..

RingSetting1..

RingSetting1..10 refers to all of the ring setting properties from 1 to 10. RingSetting1 is the property that stores the numer that sets that particular ring.

Syntax

object.RingSetting1 [= expression]

The RingSetting1 property syntax has these parts:

object An object expression that evaluates to an object in the Applies To list.
expression A double expression identifying the object. The default is the number 1.

Reference

Ring Settings are used to change the way the algorithm encrypts data.

See Also

RotorOrder1..
StartPosition1..

StartPosition1..

StartPosition1..10 refers to all of the rotor order properties from 1 to 10. StartPosition1 is the property that stores the numer that sets that particular ring.

Syntax

object.StartPosition1 [= expression]

The StartPosition1 property syntax has these parts:

object An object expression that evaluates to an object in the Applies To list.
expression A double expression identifying the object. The default is the number 1.

Reference

Start positions are used to change the way the algorithm encrypts data.

See Also

RotorOrder1..
RingSetting1..

SourceFile

SourceFile property stores the path of the file that is to be encrypted.

Syntax

object.SourceFile [= expression]

The SourceFile property syntax has these parts:

object An object expression that evaluates to an object in the Applies To list.
expression A string expression identifying the object. The default is a zero-length string ("").

Reference

Example:

```
Form1.EnigmaControl1.SourceFile = "c:\windows\temp\file1"
Form1.EnigmaControl1.TargetFile = "c:\windows\temp\file2"
Form1.EnigmaControl1.EncryptFile Progressbar1
```

See Also

TargetFile
EncryptFile

TargetFile

TargetFile property stores the path of the file that is to be written as encrypted.

Syntax

object.TargetFile [= expression]

The TargetFile property syntax has these parts:

object An object expression that evaluates to an object in the Applies To list.
expression A string expression identifying the object. The default is a zero-length string ("").

Reference

Example:

```
Form1.EnigmaControl1.SourceFile = "c:\windows\temp\file1"
Form1.EnigmaControl1.TargetFile = "c:\windows\temp\file2"
Form1.EnigmaControl1.EncryptFile Progressbar1
```

Heading

SourceFile
EncryptFile

EncryptFile

Syntax

Object.EncryptFile ProgressBar

Object: refers to the enigma control.
ProgressBar: refers to a progressbar control. (Optional)

Reference

This method encrypts a file where the path of the file is placed in the sourcefile property and writes the result file using the path placed in the targetfile property.
An additional control can be used to display the progress of the encryption. This method is compatible with Microsoft's Windows progress bar and Sheridan's SSPanel control.

Example:

```
Form1.EnigmaControl1.SourceFile = "c:\windows\temp\file1.txt"
Form1.EnigmaControl1.TargetFile = "c:\windows\temp\file2.txt"
Form1.EnigmaControl1.EncryptFile ProgressBar1
```

See Also

Encrypt
SourceFile
TargetFile

Encrypt

Object.Encrypt

Object: is the enigma control.

Reference

This method encrypts the string placed in the text property of the control

Example:

```
Form1.EnigmaControl1.Text = Form1.Text1.Text
Form1.EnigmaControl1.Encrypt
Form1.Label1.Caption = Form1.EnigmaControl1.Text
```

See Also

EncryptFile
Text

Text

ComboBox control (Style property set to 0 [Dropdown Combo] or to 1 [Simple Combo]) and TextBox controlreturns or sets the text contained in the edit area.

ComboBox control (Style property set to 2 [Dropdown List]) and ListBox controlreturns the selected item in the list box; the value returned is always equivalent to the value returned by the expression List(ListIndex). Read-only at design time; read-only at run time.

Grid controlreturns or sets the text contained in a cell or range of cells. Not available at design time.

Syntax

object.Text [= string]

The Text property syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
string A string expression specifying text.

Reference

At design time only, the defaults for the Text property are:

ComboBox and TextBox controlsthe control's Name property.
ListBox controla zero-length string ("").

For a ComboBox with the Style property set to 0 (Dropdown Combo) or to 1 (Simple Combo) or for a TextBox, this property is useful for reading the actual string contained in the edit area of the control.
For a ComboBox or ListBox control with the Style property set to 2 (Dropdown List), you can use the Text
property to determine the currently selected item.
The Text setting for a TextBox control is limited to 2048 characters unless the MultiLine property is True, in which case the limit is about 32K.
For a Grid control, you can add text to a single cell by setting the Text property. This property applies to the cell defined by the current values of the Grid control's Row and Col properties.

You can use the Text and FillStyle properties to add the same text to a highlighted range of cells. When FillStyle = 0, the text assigned to the Text property is added only to the cell defined by the current Row and Col property values. When FillStyle = 1, the text is added to all cells whose CellSelected property setting is True.
You can also use the Clip property to fill a range of cells. For example, you might want to paste a large block of information from the Clipboard into a Grid control.

Index

Returns or sets the number that uniquely identifies a control in a control array. Available only if the control is part of a control array.

Syntax

object[(number)].Index

The Index property syntax has these parts:

object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression that evaluates to an integer that identifies an individual control within a control array.

Settings

The settings for number are:

No value	(Default) Not part of a control array.
0 to 32,767	Part of an array. Specifies an integer greater than or equal to 0 that identifies a control within a control array. All controls in a control array have the same Name property. Visual Basic automatically assigns the next integer available within the control array.

Reference

Because control array elements share the same Name property setting, you must use the Index property in code to specify a particular control in the array. Index must appear as an integer (or a numeric expression evaluating to an integer) in parentheses next to the control array namefor example, MyButtons(3). You can also use the Tag property setting to distinguish one control from another within a control array.

When a control in the array recognizes that an event has occurred, Visual Basic calls the control array's event procedure and passes the applicable Index setting as an additional argument. This property is also used when you create controls dynamically at run time with the Load statement or remove them with the Unload statement.

Although Visual Basic assigns, by default, the next integer available as the value of Index for a new control in a control array, you can override this assigned value and skip integers. You can also set Index to an integer other than 0 for the first control in the array. If you reference an Index value in code that doesn't identify one of the controls in a control array, a Visual Basic run-time error occurs.

Note To remove a control from a control array, change the control's Name property setting, and delete the control's Index property setting.

Left

Leftreturns or sets the distance between the internal left edge of an object and the left edge of its container.

Topreturns or sets the distance between the internal top edge of an object and the top edge of its container.

Syntax

object.Left [= value]
object.Top [= value]

The Left and Top property syntaxes have these parts:

objectAn object expression that evaluates to an object in the Applies To list.
value A numeric expression specifying distance.

Reference

For a form, the Left and Top properties are always expressed in twips; for a control, they are measured in units depending on the coordinate system of its container. The values for these properties change as the object is moved by the user or by code. For a Timer control, these properties aren't available at run time.
For both properties, you can specify a single-precision number.
Use the Left, Top, Height, and Width properties for operations based on an object's external dimensions, such as moving or resizing. Use the ScaleLeft, ScaleTop, ScaleHeight, and ScaleWidth properties for operations based on an object's internal dimensions, such as drawing or moving objects that are contained within the object. The scale-related properties apply only to PictureBox controls and Form and Printer objects.

Name

Returns the name used in code to identify a form, control, or data access object.

Syntax

object.Name

The object placeholder represents an object expression that evaluates to an object in the Applies To list. If object is omitted, the form associated with the active form module is assumed to be object.

Reference

The default name for new objects is the kind of object plus a unique integer. For example, the first new Form object is Form1, a new MDIForm object is MDIForm1, and the third TextBox control you create on a form is Text3.

An object's Name property must start with a letter and can be a maximum of 40 characters. It can include numbers and underline (_) characters but can't include punctuation or spaces. Forms can't have the same name as another public object such as Clipboard, Screen, or App. Although the Name property setting can be a keyword, property name, or the name of another object, this can create conflicts in your code.

You can use a form's Name property with the Dim statement at run time to create other instances of the form. You can't have two forms with the same name at design time.

You can create an array of controls of the same type by setting the Name property to the same value. For example, when you set the name of all option buttons in a group to MyOpt, Visual Basic assigns unique values to the Index property of each control to distinguish it from others in the array. Two controls of different types can't share the same name.

For the Application object that is exposed or supplied by Visual Basic to add-ins, the Name property setting is always Microsoft Visual Basic. For a Property object that is exposed or supplied by Visual Basic to add-ins, the Name property setting is the same as the name listed in the Visual Basic (VB) object library in the Object Browser.

Note Although Visual Basic often uses the Name property setting as the default value for the Caption, LinkTopic, and Text properties, changing one of these properties doesn't affect the others.

Top

Leftreturns or sets the distance between the internal left edge of an object and the left edge of its container.

Topreturns or sets the distance between the internal top edge of an object and the top edge of its container.

Syntax

object.Left [= value]
object.Top [= value]

The Left and Top property syntaxes have these parts:

objectAn object expression that evaluates to an object in the Applies To list.
value A numeric expression specifying distance.

Reference

For a form, the Left and Top properties are always expressed in twips; for a control, they are measured in units depending on the coordinate system of its container. The values for these properties change as the object is moved by the user or by code. For a Timer control, these properties aren't available at run time.
For both properties, you can specify a single-precision number.
Use the Left, Top, Height, and Width properties for operations based on an object's external dimensions, such as moving or resizing. Use the ScaleLeft, ScaleTop, ScaleHeight, and ScaleWidth properties for operations based on an object's internal dimensions, such as drawing or moving objects that are contained within the object. The scale-related properties apply only to PictureBox controls and Form and Printer objects.

Tag

Returns or sets an expression that stores any extra data needed for your program. Unlike other properties, the value of the Tag property isn't used by Visual Basic; you can use this property to identify objects.

Syntax

object.Tag [= expression]

The Tag property syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
expression A string expression identifying the object. The default is a zero-length string ("").

Reference

You can use this property to assign an identification string to an object without affecting any of its other property settings or causing side effects. The Tag property is useful when you need to check the identity of a control or MDIForm object that is passed as a variable to a procedure.

Tip When you create a new instance of a form, assign a unique value to the Tag property.

ReflectorSeed

ReflectorSeed is the property that stores the numeric seed needed for the random number generator to generate a new reflector.

Syntax

object.ReflectorSeed [= expression]

The ReflectorSeed property syntax has these parts:

object An object expression that evaluates to an object in the Applies To list.
expression A double expression identifying the object. The default is the number 1.

Reference

Example:

Form1.EnigmaControl1.ReflectorSeed = 972942
Form1.EnigmaControl1.GenReflector

See Also

GenReflector
RotorSeed

GenReflector

Object.GenReflector

Object: refers to the Enigma control.

Reference

GenReflector generates a new reflector rotor based upon the ReflectorSeed property.

Example:

```
Form1.EnigmaControl1.ReflectorSeed = 4923864
Form1.EnigmaControl1.GenReflector
```

See Also

GenRotors
ReflectorSeed
RotorSeed

GenRotors

Object.GenRotors

Object: refers to the Enigma control.

Reference

GenRotors generates a new set of rotors based upon the RotorSeed.

Example:

```
Form1.EnigmaControl1.RotorSeed = 835983
Form1.EnigmaControl1.GenRotors
```

See Also

GenReflector
RotorSeed
ReflectorSeed

RotorSeed

RotorSeed is the property that stores the numeric seed needed for the random number generator to generate new rotors.

Syntax

object.RotorSeed [= expression]

The RotorSeed property syntax has these parts:

object An object expression that evaluates to an object in the Applies To list.
expression A double expression identifying the object. The default is the number 1.

Reference

Example:

```
Form1.EnigmaControl1.RotorSeed = 972942
Form1.EnigmaControl1.GenReflector
```

See Also

GenRotors
ReflectorSeed

Events

OnRotorOrder1..10Change
OnRingSetting1..10Change
OnStartPosition1..10Change
PostEncryption
PreEncryption
PostFileEncryption
PreFileEncryption
OnReflectorSeedChange
OnRotorSeedChange

OnRotorOrder1..Change

This section includes events from OnRotorOrder1Change to OnRotorOrder..Change.
OnRotorOrder1Change indicates that a ring setting has changed.

Syntax

Private Sub object_OnRotorOrder1Change([index As Integer])

The OnRotorOrder1Change event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

The OnRotorOrder1Change event procedure can synchronize or coordinate data display among controls.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the Text property setting for a TextBox control. To prevent a cascading event:

If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.

Avoid creating two or more controls whose Change event procedures affect each other, for example, two TextBox controls that update each other during their Change events.

Avoid using a MsgBox function or statement in this event for HScrollBar and VScrollBar controls.

See Also

OnStartPosition1..10Change
OnRingSetting1..10Change

OnStartPosition1..Change

This section includes events from OnStartPosition1Change to OnStartPosition..Change. OnStartPosition1Change indicates that a ring setting has changed.

Syntax

Private Sub object_OnStartPosition1Change([index As Integer])

The OnStartPosition1Change event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

The OnStartPosition1Change event procedure can synchronize or coordinate data display among controls.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the Text property setting for a TextBox control. To prevent a cascading event:

If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.

Avoid creating two or more controls whose Change event procedures affect each other, for example, two TextBox controls that update each other during their Change events.

Avoid using a MsgBox function or statement in this event for HScrollBar and VScrollBar controls.

See Also

OnRingSetting1..10Change
OnRotorOrder1..10Change

OnRingSetting1..Change

This section includes events from OnRingSetting1Change to OnRingSetting..Change.
OnRingSetting1Change indicates that a ring setting has changed.

Syntax

Private Sub object_OnRingSetting1Change([index As Integer])

The OnRingSetting1Change event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

The OnRingSetting1Change event procedure can synchronize or coordinate data display among controls.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the Text property setting for a TextBox control. To prevent a cascading event:

If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.

Avoid creating two or more controls whose Change event procedures affect each other, for example, two TextBox controls that update each other during their Change events.

Avoid using a MsgBox function or statement in this event for HScrollBar and VScrollBar controls.

See Also

OnStartPosition1..10Change
OnRotorOrder1..10Change

PostEncryption

This event is fired after encryption has taken place.

Syntax

Private Sub object_PostEncryption([index As Integer,]keyascii As Integer)

The PostEncryption event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

Example:

```
Private Sub EnigmaControl1_PostEncryption()
   MsgBox "Encryption complete..."
End Sub
```

See Also

PreEncryption

PreEncryption

This event is fired before encryption has taken place.

Syntax

Private Sub object_PreEncryption([index As Integer,]keyascii As Integer)

The PreEncryption event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

Example:

```
Private Sub EnigmaControl1_PreEncryption()
   Response% = MsgBox("Encrypt text?",36,"Encrypt")
   If Response% = VBYes Then
      ....
   End If
End Sub
```

See Also

PostEncryption

PostFileEncryption

This event is fired after a file has been encrypted.

Syntax

Private Sub object_PostFileEncryption([index As Integer,]keyascii As Integer)

The PostFileEncryption event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

Example:

```
Private Sub EnigmaControl1_PostFileEncryption()
   MsgBox "File encrypted..."
End Sub
```

See Also

PreFileEncryption

PreFileEncryption

This event is fired before a file has been encrypted.

Syntax

Private Sub object_PreFileEncryption([index As Integer,]keyascii As Integer)

The PreFileEncryption event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

Example:

```
Private Sub EnigmaControl1_PreFileEncryption()
   Response% = MsgBox("Encrypt this file?",36,"Encrypt File")
   If Response% = VBYes Then
      ...
   End If
End Sub
```

See Also

PostFileEncryption

OnReflectorSeedChange

Indicates that the ReflectorSeed property value has changed.

Syntax

Private Sub object_OnReflectorSeedChange([index As Integer])

The OnReflectorSeedChange event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

The OnReflectorSeedChange event procedure can synchronize or coordinate data display among controls.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the Text property setting for a TextBox control. To prevent a cascading event:

If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.

Avoid creating two or more controls whose Change event procedures affect each other, for example, two TextBox controls that update each other during their Change events.

Avoid using a MsgBox function or statement in this event for HScrollBar and VScrollBar controls.

See Also

OnRotorSeedChange

OnRotorSeedChange

Indicates that the contents of a control have changed.

Syntax

Private Sub object_OnRotorSeedChange([index As Integer])

The OnRotorSeedChange event syntax has these parts:

objectAn object expression that evaluates to an object in the Applies To list.
index An integer that uniquely identifies a control if it's in a control array.

Reference

The OnRotorSeedChange event procedure can synchronize or coordinate data display among controls.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the Text property setting for a TextBox control. To prevent a cascading event:

If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.

Avoid creating two or more controls whose Change event procedures affect each other, for example, two TextBox controls that update each other during their Change events.

Avoid using a MsgBox function or statement in this event for HScrollBar and VScrollBar controls.

See Also

OnReflectorSeedChange

Methods

Registering

To register, do one of the following:

1 Go to our on-line ordering page at http://www.dlcomputing.com.

2 E-mail us the completed registration form to DLCSales@Juno.com. The registration form is located under the File Lock 98 start-up menu.

3 Fax us the completed registration form to (205) 350-4366.

4 Request registration by voice at (205) 350-5024 and a sales representative will return your call.

5 Send payment in U.S. currency to D & L Computing, Inc. at the following address:

Address: D & L Computing, Inc.
P.O. Box 6141
Huntsville, AL 35824

D & L Computing, Inc. accepts checks, money orders, purchase orders (with prior approval), American Express, MasterCard, and VISA. See our web site for our new, faster check processing capability that could prevent delay in mailing your personal check to us!

Note

Please note that the price, address and telephone numbers are subject to change. See our web site for the latest information. All orders are processed within 48-hours of receipt. All returned checks are subject to a $20 service charge. Purchase orders should be faxed to (205) 350-4366 to obtain prior approval to prevent delay in processing purchase.

About

This ActiveX control is based upon the W.W. II German enigma machine used by the German military. It is said that the enigma machine was the toughest encryption to break; however, the British and the Polish ser only able to break the code due to the capture of an actual enigma machine and a cypher book of settings. Using computers, we now can create unlimited virtual enigma machine configurations to give the most powerful encryption available.

Warning

The U.S. government has placed strict export laws on encryption routines. Please be advised that if a program is developed using this package, the programmer is solely responsible in adhereing to federal law. Please keep a close check on the updating of these laws.

D & L Computing, Inc.

Please check out our web site at http://www.dlcomputing.com.