Leong Profile Controls

Welcome to the ***Leong Profile Controls***. They are a set of Visual Basic Custom Controls that make it easy to use INI files with your Visual Basic programs.

There are four controls:


the Leong Ini Color Control,


the Leong Ini Font Control,


the Leong Ini String Control. You can use the String control to store and retrieve many types of data in INI files, including numbers, dates and times.


the Leong Expose Control, which makes the Captions of all the controls in your program accessible via INI files

Please accept our invitation to try the Leong Profile Controls yourself. They are distributed as shareware, so you can evaluate them with no charge. But they are not free! You must pay for them before using them in earnest. See the Licence Conditions for full details.

We hope you like the controls.

This is version 1.01. See the Release Notes for a version history.

Contents of the Release

Description

These custom controls provide an interface to Windows Profiles, or INI files. This lets Visual Basic programs store and retrieve parameter settings from one invocation to the next.

File Names

LEONGINI.VBX	contains the actual controls
LEONGINI.TXT	contains Visual Basic constant definitions for setting the properties of the controls, and may be included in your programs.
LEONGINI.HLP	is this help file
READ.ME	contains the latest news of the product and how to obtain it.

Object Type

LeongIniColor

LeongIniFont

LeongIniString

LeongExpose

Remarks

These custom controls are distributed as shareware. There is a runtime version of the controls for use when you distribute applications. See the Licence Conditions for full details.

Distribution Note When you create and distribute applications that use these controls you should install the runtime version of the LEONGINI.VBX file in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

Version

This help file is version 1.01 and describes version 1.01 of the controls.

The changes from version 1.00 are:

The Expose control has been added to save Captions, Tags and Text from other controls.

The IniString control has been fixed to allow the default value to be returned if the Key, IniSection and IniFile are set to values that referred to an actual entry in an INI file and then changed to some values that did not refer to an actual entry (the control kept the most recently accessed entry as its value).

The Align property has been added so the controls can be placed directly on an MDI form.

The dynamic creation of additional controls at run time is fixed. Dynamically created controls do not have any Attached controls (Professional Edition).

The Help file, LEONGINI.HLP, now includes the correct numerical values for the OnChange and OnUnload values of the Write property.

It is now possible to delete entries in INI files by setting the Write property to 4 (Delete Entry) or 5 (Delete Section).

Distribution & Licensing

The Leong Profile Controls are distributed as shareware.

You can evaluate them for thirty days completely free of charge. After that, you must pay for them, or stop using them. Click here for details of how to register. See the READ.ME file for the latest update on support and registration.

You are encouraged to distribute the complete shareware package to others for them to evaluate, but not parts of the package.

Registration Benefits

When you register, you receive these benefits:

Full on-line support via CompuServe.

Run-time package, so you can distribute the controls as part of your programs.

Royalty-free distribution licence.

Registered development version of the controls.

Run-time Package

The run-time version of the controls may be distributed royalty-free with your programs, and the users have unlimited rights to use them with your programs.

The run-time version is smaller than the development version, saving valuable memory.

There is no notice to your users, though we always appreciate a mention in your *About..* box. Thanks!

Shareware Package

The shareware package contains the items in the shipping list.

You can develop programs in Visual Basic and run them. A notice reminds you of your rights to evaluate these controls when you first attach a control to Visual Basic.

You can build EXE files of your programs. When you run an EXE file that uses the shareware controls, a notice appears to remind you that the controls are included and that a royalty is payable if the program is used for any purpose apart from evaluationg these controls. The royalty is 10% of the price of the program (highest of list price of a product, registration fee or professional fees for custom development).

Registered Developer Version

The registered developer version gives you the right to develop programs for commercial, educational and other uses.

Professional Version

The Professional version of the controls offers all the same benefits as the Registered version, and supports *Attached Controls* as well.

Ask for the professional version when you register the Leong Ini Controls. This supports attached controls in addition to all the other features. You can save more programming by linking the values of several control's properties directly to an Ini control at design time, by using the Attach property.

The Professional version is not available as shareware.

Getting Support and Registering

You can get support via CompuServe ID 100332,3614 - Mabel Leong. This is the best method; please use it if you can.

You can register by using the CompuServe Software Registration Forum (GO SWREG) and quoting #2313 for the registered developer version or #2328 for the Professional version. You can also register by sending payment to:

Mabel Leong, 21 Meadowbank, Hitchin, Herts SG4 0HX, UK

See the README.TXT file for more details, including price. You can also get support by writing to this address.

We're sorry its not possible to accept phone calls.

Using the Controls

Its very simple to start using the profile controls. Just follow these steps:

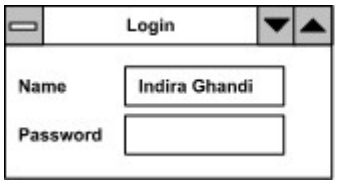
1. Ensure your form has a meaningful Name.
2. Create a Leong Profile Control.
3. Change its Name to represent the value you want to initialise.
4. Give it a suitable default Value by typing in the Property Box.
5. Assign its Value to your variables in your program.

Lets look at an example.

Example of Use

See Also

Suppose you're writing an application that needs the user to log in. You've designed a simple form for this and you'd like the program to remember the user's name from one session to the next:



The form is called **Login**

The input box is called **NameBox**

What to do

1. Check your form has a meaningful name. **Login** is just fine.
2. Add a String Profile Control to the form. Use the File Add menu (or CTRL+D) to select the LEONGINI.VBX file then double-click the icon in the toolbox .

   
3. Use the property box to change the control's name to **UserName**.
4. You don't need to set a default, since there isn't a meaningful one for a person's name.
5. Add a line to the Form_Load procedure to initialise the input box with the name from the INI file:

   ```
   NameBox = UserName
   ```

   and a line to the NameBox_LostFocus procedure to record any different name that the user types in the INI file:

   ```
   UserName = NameBox
   ```

That's it!

What the Control does

The default properties of the Profile Control mean that everything else is automatic. Your program will now create and use a file C:\WINDOWS\PROGNAME.INI. It will contain an entry like this:

```
[Login]
UserName=Indira Ghandi
```

Each entry is in a section of the INI file headed by the name of the form. This groups together all the initialisation for the form, producing a tidy INI file. You can change this using containers or the IniSection property if necessary. You can also change the name or location of the INI file (see Where to Store the INI File)

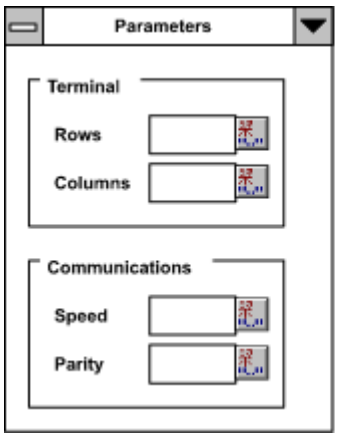
Using Containers

(Creating Sections in INI files)

You can use container controls to collect together related entries in the INI file.

Suppose you are writing a program to access remote computers. You have several parameters that all appear on a form; some concern the communications line, whilst others depend on the type of remote computer. You group these settings together using Frame controls so the user can understand it.

Now add profile controls in the same way as for the Login form, but put the controls in the frames as well.



The form is called **Parameters**
The frame is called **Terminal**
The profile control is called **Rows**
etc

The INI file contains:

```
[Terminal]
Rows=24
Columns=80
[Communications]
Speed=2400
Parity=even
```

This will create an INI file like that shown.

If you need, you can get even more control by setting the IniSection property.

Container Controls

Visual Basic controls that can contain other controls are called containers. These include the frame and panel controls.

Where to Store the INI File

See Also

The controls let you keep INI files in any directory you want. There are various factors that will influence your choice.

Windows Directory

Microsoft recommend that all files are kept in the personal Windows directory (often c:\windows), because when Windows is loaded from a network server, this is the only place guaranteed to be private to each user rather than shared between all users. INI files need to be able to be customised for individual users.

Program Directory

Some people prefer to keep INI files in the same directory as the program itself. This keeps everything to do with the program in one place, and works well on stand-alone systems. It makes it easy to copy the program onto a diskette without forgetting anything.

Other Directories

A site may have a network configuration where there is some other private directory for each user that is preferred to the Windows directory.

There may be a need to keep a profile for one group of users, and a different profile for another group. Or some things might depend on the individual user whilst others depend on what particular machine he's logged on.

The Profile Controls can handle all of these requirements, as simply as possible. They use a search strategy, a set of rules to adapt to different circumstances.

Search Strategy

See Also

The Profile Controls have a search strategy to find the INI file. The goal is to give preference to decisions taken closer to the point of use. For example, if the user takes a decision about how things are arranged on his system, it should generally overrule one that you took back in your development shop.

1. A value assigned to the IniFile property at run time.
2. A file named in WIN.INI by your application EXE name.
3. A file named in WIN.INI by your application title.
4. A value given to the IniFile property at design time.
5. An INI file in the Windows directory with the same name as your program.
6. An INI file in the same directory as your program, with its name.

Modified Search during Development

The strategy is slightly modified whilst you are developing the program. The profile controls don't use rules 2, 5 and 6 when you are in the Visual Basic development environment.

This is because the program's EXE name is "VB.EXE" if you run it from within the Visual Basic development environment. This would cause the profile controls to use Visual Basic's own INI file, "C:\WINDOWS\VB.INI", so the strategy is modified.

Rules 2, 5 and 6 are only used after you have made a freestanding EXE version of your program.

See Also

About INI Files
Where to Store the INI File

1 If your program sets the IniFile property at run time, the value you set will be used as the name and path of the INI file. Your program has taken the appropriate decision on this occasion, probably in consultation with the user.

2 The file given in an entry `INI=filename` in a section in WIN.INI labelled with the name of your program's EXE file, if present. The user can customise this entry to meet his needs. He can also make copies of your program with different names, if he needs multiple profiles.

3 The file given in an entry `INI=filename` in a section in WIN.INI labelled with your applications's title, if present. You set the application title when you make the EXE file. The user can customise the setting in WIN.INI to meet his needs.

4 The value you gave the IniFile property when you were designing it, if any. This is a default in case there are no entries in WIN.INI.

5 A file in the Windows directory with the same name as your application's EXE file, but with a .INI extension (eg C:\WINDOWS\PROGNAME.INI). This is a good place in normal circumstances.

6 A file with the same name as your application's EXE file and in the same directory as it, but with a .INI extension (eg C:\YOURAPP\PROGNAME.INI). This is intended as a backstop since its easy to install extra files in the same directory as your program.

See Also

IniFile property

You can use a Leong Ini String Control to build this ability into your program, if you want.

Windows Directory

The **personal** Windows Directory is the place where the user's personal Windows files are kept. Typically this is **c:\windows** on a stand-alone machine. On a network, it might still be on the local disk, but in many cases it will be on a network server. This is the directory that contains a user's personal profile (eg. **u:\winuser**) and is an appropriate place for INI files.

The **shared** Windows Directory is where the Windows executables are kept. Typically this is **c:\windows** on a stand-alone machine. On a network, it may be a shared directory on a server (eg. **w:\winshare**).

About INI Files

Microsoft™ Windows® allows applications to keep profiles for individual users. These profiles are commonly stored in files with names that end in `.INI`, and are often referred to as *INI files* or initialization files

Most applications use an INI file with the same name as the program, but with a .INI extension. Thus, the Visual Basic program is VB.EXE and its profile is VB.INI. INI files are usually stored in the shared Windows directory.

Entries and Sections

Each line in these text files defines the *value* of an *entry*, or key. The value is expressed as a string. Each line is set out like this:

```
entry=text string representing value
```

The files can be divided into different sections by header lines:

```
[section name]
```

All entries following a header line belong to that section.

Case Insensitive

The names of entries and sections are not case-sensitive. They can be spelled in lower case or capital letters, or any mixture, without changing their meaning. Thus [section name], [Section Name] and [SECTION NAME] all refer to the same section.

Profile Controls

The Leong Profile Controls provide an easy way to use INI files in your Visual Basic programs. Each control represents one entry in an INI file. The name of the entry corresponds to the Key property of the control (which is usually also the control's name). The value in the file corresponds to the Value property. The IniSection property indicates the section of the file and the filename itself can be determined by the IniFile property (there are other factors as well).

Updating INI Files

You can change the value of a profile control by setting its Value property:

```
control.Value = "new value"
```

or even

```
control = "new value"
```

since the Value property is the default property of the control. The control will immediately save the new value in the INI file so it will be remembered next time the program is run.

You may not want the value to be automatically written to the file. Perhaps you want the user to confirm the changes first, or perhaps the change shouldn't be recorded at all.

You can achieve this using the Write property. By default, it is set to 0-OnChange; change it to 2-Manual. The control won't write its value to the INI file now unless you instruct it. You do this by assigning 3-Now to the Write property. This won't change the property, it will still be set to 2-Manual, but it will cause the control to write its value to the INI file.

If you don't want your program to alter the INI file ever, just set the Write property to 2-Manual at design-time and don't refer to it at run-time.

"default property" or "value of the control"

Visual Basic controls can have a default property. You can omit the name of this property when getting or setting its value. The default property of the Profile Controls is the Value property, because this leads to the shortest code. So you can write:

```
myProfile = "new value"
```

instead of:

```
myProfile.Value = "new value"
```

The default property is also called *the value of the control*, because of this abbreviation.

Refer to the Visual Basic Programming Guide - Code in Procedures (in the topic Using the Value of a Control) for more detail.

Control Arrays

Example

Sometimes you want to keep a list of values until the program is run again. One example would be to provide a list of the last four files that your program had opened.

The profile controls support the use of control arrays to help you do this.

You can create arrays of profile controls in the normal way, and you're free to set their values as you wish. Initially, each control will set its Key property to a unique value that is made by combining the name of the array with the Index of this particular control. Thus the third element of a control array called myProfile, which is referred to as `myProfile(2)` in the program, will use an entry of `myProfile_2` in the INI file.

Using the Expose Control

Expose Control

Set the **Property** property to indicate which property you want to expose in an INI file. For example, *0 - Caption*.

Set the **IniSection** property to indicate which section in the INI file you want to use. For example, *"Captions"*.

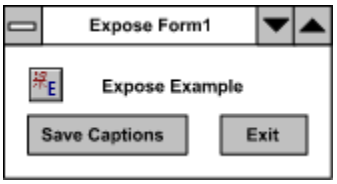
Set the **IniFile** property to choose which INI file to use. For example, *"Deutsch.ini"*.

All the above settings can be made at design time or run time. Then at run time, either:

Set the **Read** property to *2 - ReadNow* to (re)read the values of the properties of all controls, or

Set the **Write** property to *2- WriteNow* to store the values of the properties of all controls.

For example, if your application looks like this:



and you expose the captions, the INI file might look like this:

```
[Expose Captions]
Form1.Form1=Expose Form1
Form1.Command2=Exit
Form1.Command1=Save Captions
Form1.Label1=Expose Example
```

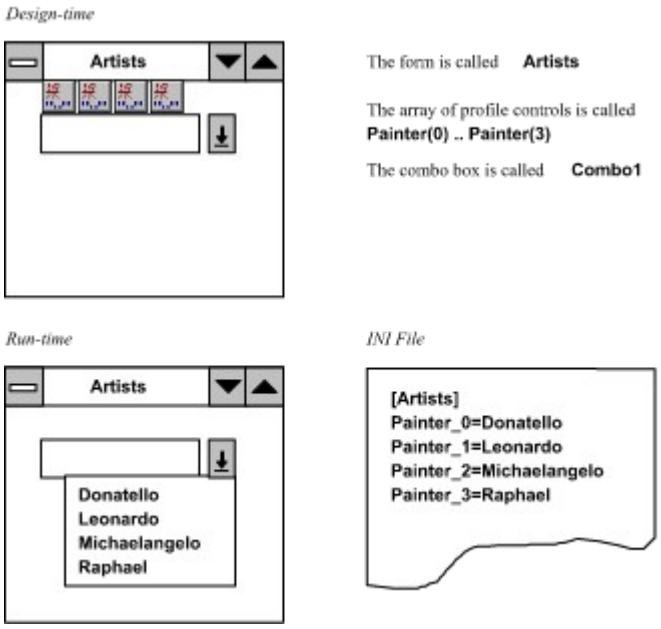
See the EXPOSE.MAK example program also.

Control Array Example

See Also

This example illustrates how you can use an array of profile controls to store a list of values until the next time the program is run.

A combo box called Combo1 on a form called Artists displays a list of painters' names. Up to four of these names are remembered by an array of four Profile controls called Painter. The form is shown below with a typical section from the INI file.



Just ten lines of code support this functionality, in the Form's Load and Unload procedures. Load just adds any non-empty values from the profile array to the combo box's list:

```
Sub Form_Load ()
    For i = 0 To 3
        If Painter(i) <> "" Then
            Combo1.AddItem Painter(i)
        EndIf
    Next i
End Sub
```

Unload copies the first four items from the combo's list to the profile array. This automatically causes the profile controls to save the values to the INI file:

```
Sub Form_Unload (Cancel As Integer)
    n = Combo1.ListCount
    If n > 4 Then n = 4
    For i = 0 To n - 1
        Painter(i) = Combo1.List(i)
    Next i
End Sub
```

control arrays

A control array is made up of a group of controls of the same type (for example, all list boxes) that share a common control name and a set of event procedures. Each control in the array has a unique index number. All items in a control array must have the same setting of the Name property. (See Visual Basic Help - Creating a Control Array or the Visual Basic Programmer's Guide - Working with Control Arrays).

See Also

Simple Example
Container Example
Control Array Example

Index property

Specifies the number that uniquely identifies a control in a control array. Available only if the control is part of a control array; read-only at run time. Because control array elements share the same Name property setting, you must use the Index property in code to refer to a particular control in the array. Indexes start at zero by default. The Index must appear as a number in parentheses next to the control array name, for example:

```
MyProfile(3)
```

See Visual Basic Help - Index Property (Control Arrays).

Properties & Events

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

Properties

Enabled
Index
IniFile
IniSection
Key
Left
Name
Parent
Read
Tag
Top
Value
Write

Leong Ini Color Control

Properties

Description

The Color control stores and retrieves color values from the INI profile.

Usage

[*form*.]*IniColor*.**Value**[= *color expression*]

Remarks

The control understands colors represented as hex numbers, or using an RGB(r,g,b) expression or by name. Thus the following entries in an INI file all represent the same color:

```
color_1=0x000000FF
color_2=&H000000FF&
color_3=RGB(255,0,0)
color_4=red
```

The control always writes colors to the file in the first format.

You can easily create matched color schemes by attaching controls to a color control, if you have the Professional Version. All the attached controls will automatically change color together whenever the color control's properties are altered.

Color Names

The Color Control recognises three groups of colors by name. These are the Visual Basic colors, Quick Basic colors and the Windows system colors:

Black
Orange
Gray
Light Gray
Blue
Green
Cyan
Red
Magenta
Yellow
White

QB Black
QB Blue
QB Green
QB Cyan
QB Red
QB Magenta
QB Yellow
QB White
QB Gray
QB Light Blue
QB Light Green
QB Light Cyan
QB Light Red
QB Light Magenta
QB Light Yellow
QB Bright White

SCROLL BARS
DESKTOP
ACTIVE TITLE BAR
INACTIVE TITLE BAR
MENU BAR
WINDOW
WINDOW FRAME
MENU TEXT
WINDOW TEXT
TITLE BAR TEXT
ACTIVE BORDER
INACTIVE BORDER
APPLICATION WORKSPACE
HIGHLIGHT
HIGHLIGHT TEXT
BUTTON FACE
BUTTON SHADOW
GRAY TEXT
BUTTON TEXT
INACTIVE TITLE BAR TEXT
BUTTON HIGHLIGHT

Color Control Properties

All of the properties for the color control are listed below. Properties that apply only to the Leong Ini controls, or require special consideration when used with them, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties.

Align
Enabled
Index
IniFile
IniSection
Key
Left
Name
Parent
Read
Tag
Top
Value
Write

Leong Ini Font Control

Properties

Description

The Font control stores and retrieves fonts from the INI profile.

It understands the same attributes of fonts as Visual Basic. It represents a font with a textual description in the INI file as in the following examples:

```
font_1= 10 point Arial
font_2= 24 point Bold Italic Times New Roman
```

Note that the font name, such as 'Times New Roman', must exactly match one that the system recognises. In particular, there must be just one space between words.

Usage

[*form*.]*IniFont.***FontBold**[= *expression*]

[*form*.]*IniFont.***FontItalic**[= *expression*]

[*form*.]*IniFont.***FontName**[= *expression*]

[*form*.]*IniFont.***FontSize**[= *expression*]

[*form*.]*IniFont.***FontStrikethru**[= *expression*]

[*form*.]*IniFont.***FontUnderline**[= *expression*]

Remarks

The font control uses the same six font properties as other Visual Basic controls. This makes it slightly different to the other Leong Profile Controls, because it doesn't have a Value property.

You can save many lines of code by attaching controls to a font control, if you have the Professional Version. All the attached controls will automatically change font whenever the font control's properties are altered.

Font Control Properties

All of the properties for the font control are listed below. Properties that apply only to the Leong Ini controls, or require special consideration when used with them, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties.

Align
Enabled
FontBold
FontItalic
FontName
FontSize
FontStrikethru
FontUnderline
Index
IniFile
IniSection
Key
Left
Name
Parent
Read
Tag
Top
Write

Leong Ini String Control

Properties **Example**

Description

The String control stores and retrieves strings from the INI profile.

Usage

[*form*.]*IniString*.**Value**[= *string expression*]

Remarks

You can use the string control with other data types as well:

Number Just assign the number to the String control to store it and assign it to a numeric variable or property to retrieve it. Visual Basic takes care of the conversions.

Date Use the **DateValue()** function when you want to retrieve a date from an INI file:

```
myDateVariable = DateValue(IniString.Value)
```

and use the **Format$()** function when you want to store it:

```
IniString.Value = Format$(myDateVariable, "Medium Date")
```

Time Use the **TimeValue()** function when you want to retrieve a time from an INI file:

```
myTimeVariable = TimeValue(IniString.Value)
```

and use the **Format$()** function when you want to store it:

```
IniString.Value = Format$(myTimeVariable, "Long Time")
```

You can also combine dates and times together:

```
myEpoch = DateValue(IniString.Value)
          + TimeValue(IniString.Value)
IniString.Value = Format$(myEpoch, "d mmm yyyy hh:mm:ss")
```

String Control Properties

All of the properties for the string control are listed below. Properties that apply only to the Leong Ini controls, or require special consideration when used with them, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties.

Align
Enabled
Index
IniFile
IniSection
Key
Left
Name
Parent
Read
Tag
Top
Value
Write

Leong Expose Control

Properties Using the Expose Control

Description

The Expose control makes property values of all the controls in your application accessible via INI files. You can save and load all the Caption, Tag or Text properties in a file.

One use for this is to assist in translation of your application into other languages without making any changes to the program.

Usage

[*form*.]*Expose*.**Read** = **ReadNow** ' Load values

[*form*.]*Expose*.**Write** = **WriteNow** ' Save values

Remarks

Most of the static text visible in your application is the value of the Caption property of the controls in your application. For example, the Caption property of a Form is displayed as the title of the window, the Caption properties of menu items appear as the legends in the menu on screen, etc.

The Expose control can save the value of ***all*** the Captions to an INI file. The INI file can be changed, perhaps translated into a foreign language, and the Expose control can load the new Captions so they are shown on the screen.

The Expose control can also load and save all the Tags. The Tag property is often used to store help prompts. The ability to list them all together in a file helps with management of the program as well as permitting translation.

Finally, the Expose control can save and restore all Text property values.

Expose Control Properties

All of the properties for the Expose control are listed below. Properties that apply only to the Leong Ini controls, or require special consideration when used with them, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties.

Align
Enabled
Index
IniFile
IniSection
Left
Name
Property
Read
Tag
Top
Write

Font Properties, Leong Profile Controls

Description

Visual Basic's standard font properties are used as the value of the Font Control.

Usage

[*form*.]*IniFont.***FontBold**[= *boolean*]

[*form*.]*IniFont.***FontItalic**[= *boolean*]

[*form*.]*IniFont.***FontName**[= *string expression*]

[*form*.]*IniFont.***FontSize**[= *numeric expression*]

[*form*.]*IniFont.***FontStrikethru**[= *boolean*]

[*form*.]*IniFont.***FontUnderline**[= *boolean*]

Data Type

FontBold : Integer (Boolean)

FontItalic : Integer (Boolean)

FontName : String

FontSize : Single

FontStrikethru : Integer (Boolean)

FontUnderline : Integer (Boolean)

Remarks

These properties are read from the INI file determined by the other properties. They are the value of the entry that matches the Key property, in a section headed with the IniSection property's value, in a file determined by the IniFile property. If no such entry can be located, they have the default values set at design-time.

Setting one of these properties causes the new value to be written to the INI file, subject to the value of the Write property.

See Visual Basic's Help for more details of these properties.

Changing Properties at Run Time

If you want to use a control to access several entries in one or more sections of one or more INI files, you will want to change the values of the IniFile, IniSection or Key properties at run time.

When you change these values, you also need to set the Read property to 2 - ReadNow afterwards. Setting the Read property is what causes the control to read the new value from the file. For example:

```
IniString1.Key = "MyNewKey"
IniString1.Read = ReadNow
Print IniString1.Value
```

IniFile Property, Leong Profile Controls

See Also

Description

The IniFile property is used to determine the name of the INI file to search for an entry for this control.

Usage

[*form*.]*INI*.**IniFile** [= *filename*]

Data Type

String (FileName)

Remarks

The control makes a sequence of attempts to find the appropriate INI file.

If the IniFile property is set at runtime, then the profile entry is sought in the file with that name (note that the Read property will need to be set to 2 - Manual also).

Otherwise, WIN.INI is examined for an entry containing the name and path of the INI file for this program.

If both these fail, a third try to find the INI file is made using the value of the IniFile property that was set at design-time, if any.

Finally, an attempt is made to find a file with the same name as the program, but with a .INI extension, in the Windows directory or the same directory as the program.

For more details of how the IniFile is found, see Search Strategy.

The IniFile property is evaluated relative to the user's personal Windows directory if it is not set to a full path.

Using Multiple INI Files

If you need to use more than one INI file with your program, set the IniFile property of the controls appropriately. You might have one set of controls that store their values in the program's main INI file, and another set of controls that use an auxiliary file (say its called **aux.ini**). The first set of controls will have the correct behaviour by default. Just set the IniFile for the auxiliary controls:

```
profile.IniFile = "aux.ini"
```

Expose Control

The Expose control does not use the search strategy for the INI file. It simply uses the named file.

See Also

Where to store the INI File
Changing the IniFile property at run time

IniSection Property, Leong Profile Controls

See Also

Description

The section in the INI file to look for this entry.

Usage

[*form*.]*INI*.**IniSection** [= *stringexpression*]

Remarks

The **IniSection** property determines which section in the INI file contains the entry. For example, suppose that the **Key** property of a control called **myProfile** is set to "`confusing`" and the **IniSection** property is set to "`look here`". If the INI file looks like this:

```
[look here]
confusing=Do you see what I mean?
[but not here]
confusing=unexpected answer
```

then **myProfile** will have a value of "`Do you see what I mean?`".

By default, **IniSection** is set to the name of the control's container. This is the name of the form if the profile control is placed directly on the form. There is no default value for the Expose control.

Data Type

String

See Also

Using ContainersChanging values at run time.

Key Property, Leong Profile Controls

Description

The name of the entry in the INI file.

Usage

[*form*.]*INI*.**Key** [= *stringexpression*]

Remarks

The **Key** property determines what entry is sought in the INI file. For example, if it is set to "`password`" and the INI file contains the following line in the appropriate section:

```
Password=unpredictable
```

then the value of the control will be "`unpredictable`".

Entries in INI files are not case sensitive, so the `password` key matches the `Password` file entry.

(Note that keeping passwords in INI files is not secure without encryption.)

By default, **Key** is set to the same value as the name of the control. If the control is an element of an array, an underscore and the control Index are added to the end (eg. The control `Date(2)` defaults to a key name of `Date_2` in the INI file).

Click here for a note about changing the Key property at run time.

Data Type

String

Value Property, Leong Profile Controls

Description

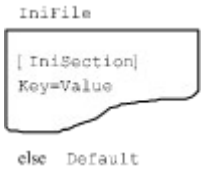
The value of the control, obtained from the INI file.

Usage

[*form*.]*INI*.**Value** [= *expression*]

Remarks

The **Value** property is the value read from the INI file, as determined by the other properties. It is the value of the entry that matches the Key property, in a section headed with the IniSection property's value, in a file determined by the IniFile property. If no such entry can be located, it has the default value set at design-time.



Setting the **Value** property causes the new value to be written to the INI file, subject to the value of the **Write** property.

Value is the default property of the control. So if the control is called `IniDate(2)`:

```
IniDate(2).Value = "31 Oct"
```

means the same as

```
IniDate(2) = "31 Oct"
```

Data Type

Color Control	*Color*	Initially it has the value 0 (black).
Font Control	Does not have a Value property. It has Font properties instead.	
String Control	*String*	Initially it has the value "". It has a maximum length of 1000 characters.

Property Property, Leong Expose Control

Expose Control

Description

The Property property determines the property that is read from or stored in the INI file.

Usage

[*form*.]*INI*.**Property** [= { **Caption** | **Tag** | **Text** }]

Settings

The Property property settings are:

Setting	Description
0 - Caption	All **Caption** properties of all controls in the application are saved to or read from the INI file.
1 - Tag	The **Tag** properties of all controls in the application are saved to or read from the INI file.
2 - Text	All **Text** properties of all controls in the application are saved to or read from the INI file.

By default, Property is set to 0 - Caption.

Remarks

The Caption, Tag or Text properties are saved to the INI file when the Write property of the Expose control is set to 2 - WriteNow at run time.

The Caption, Tag or Text properties are read from the INI file when the Read property of the Expose control is set to 2 - ReadNow at run time.

Data Type

Integer (Enumerated)

Read Property, Leong Profile Controls

Description

The Read property determines when the value is read from the INI file.

Usage

[*form*.]*INI*.**Read** [= { **OnLoad** | **Manual** | **ReadNow** }]

Settings

The Read property settings are:

Setting	Description
0 - OnLoad	The value is automatically obtained from the INI file when the form is loaded.
1 - Manual	The value is only obtained under program control by assigning 2 - Now to the **Read** property. The control does not read the value at startup.
2 - Now	Assigning this value to the property at runtime causes the **Value** to be (re)read from the INI file. It does ***not*** affect the value of the **Read** property, which keeps its previous value, normally 1 - Manual.

By default, Read is set to 0 - OnLoad.

Remarks

The OnLoad setting should be sufficient in normal use. The Manual setting provides more flexibility for special cases, such as when the value of the Key property is only known at run time.

The **Read** property of the Expose control is always set to 1 - Manual, never to 0 - OnLoad. Assigning 2 - ReadNow to the property at run time causes all the **Caption** (or **Tag** or **Text**) properties to be restored from the values stored in the INI file.

Data Type

Integer (Enumerated)

Write Property, Leong Profile Controls

See Also

Description

The Write property determines when the value is written to the INI file.

Usage

[*form*.]*INI*.**Write** [= { **OnUnload** | **OnChange** | **Manual** | **WriteNow** | **DeleteEntry** | **DeleteSection** }]

Settings

The Write property settings are:

Setting	Description
0 - OnChange	The value of the control is automatically stored whenever the value is changed by the program
1 - Manual	The value is only stored under program control by assigning *2 - WriteNow* to the **Write** property.
2 - WriteNow	Assigning this value to the property at run time causes Causes a `Key = Value` line to be (re)written to the INI file. It does ***not*** affect the value of the **Write** property, which keeps its previous value, normally *1 - Manual*.
3 - OnUnload	The value of the control is automatically written to the INI file when the form is unloaded.
4 - DeleteEntry	The entry for the key in the INI file is deleted when this value is assigned to the property at run time. It does ***not*** affect the value of the **Write** property, which keeps its previous value.
5 - DeleteSection	All entries in the current section of the INI file are deleted when this value is assigned to the property at run time. It does ***not*** affect the value of the **Write** property, which keeps its previous value.

By default, Write is set to *OnChange*.

Remarks

Windows caches the most recently used INI file.

The **Write** property of the Expose control is always set to 1 - Manual. Assigning 2 - WriteNow to the property at run time causes all the **Caption** (or **Tag** or **Text**) properties to be saved in the INI file. The **Write** property of the **Expose** control cannot be set to any other value.

Data Type

Integer (Enumerated)

See Also

Updating INI Files

Attach Property, Leong Profile Controls

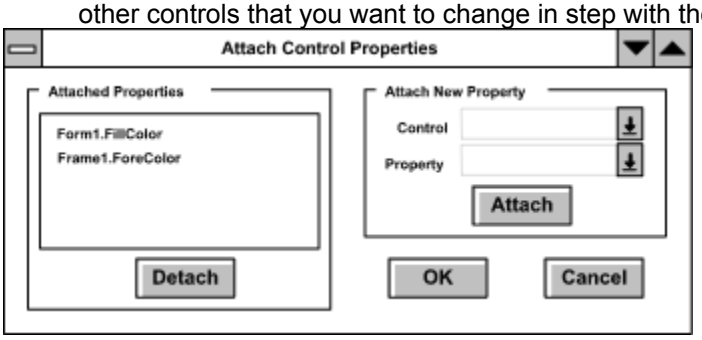
(Professional Version Only)

See Also

Description

The Attach property determines which other control properties are changed by the Ini Control.

You can attach other control's properties to the Ini Control at design time by double-clicking on the Attach property in the Property Box. This pops up a dialog box. Use it to choose properties from other controls that you want to change in step with the value of the Ini control.



For example, you could attach the ForeColor property of a number of controls to one Ini control. Then all these controls will change color whenever the Ini control gets a new value.

So you can allow the user to customise the appearance of your application very easily.

This is particularly valuable for font properties, because it saves setting each of the six font properties on every control each time a font is changed.

Usage

The Attach property is not available at runtime. Use the Attach function instead.

Attach Function

(Professional Version Only)

See Also

Description

The Attach function supplements the Attach property.

You can attach additional control's properties to the Ini Control at run time by calling the Attach function. Use it to choose properties from other controls that you want to change in step with the value of the Ini control.

Syntax

Attach(*IniControl, AttachedControl, AttachedProperty***)**

Remarks

The Attach function has these parts:

Part	Description
IniControl	The Leong Profile control to which another control is to be attached.
AttachedControl	The control that is to be attached.
AttachedProperty	String expression giving the name of the property to be attached. NB *Not* the property itself. Thus **"ForeColor"**, not **ForeColor**

The **Attach** function returns **True** if it is successful in attaching the control property. It returns **False** if it fails, perhaps because the *AttachedControl* does not have a property of the given name.

The Attach function is called like a function in a DLL (see Calling Procedures in DLLs in the Visual Basic Programmer's Guide). It is not exactly like a Visual Basic function call or a control method, because we haven't found a way to do this!

Example

This example attaches the foreground color of a frame control to a Leong Profile Color control, and then changes the color of the frame's foreground to blue.

```
status = Attach(IniColor1, Frame1, "ForeColor")
IniColor1 = &HFF0000&
```

Declaration

The **Attach** function is declared in the LEONGINI.TXT file as:

```
Declare Function Attach Lib "LEONGINI.VBX" (Ini as Any,
          attachedControl as Control, ByVal Property as String) as
          Integer
```

See Also

Attach Function
Attach Property