INI.VBX
By
Walter F. Dexter
CIS ID: 76450,663

This Visual Basic custom control provides a somewhat easier mechanism for accessing .INI files from your visual basic program. It was written purely as a programming exercise, and has never been put to any extensive use. Full source code is included, and the control may be modified and distributed freely. No charge beyond reasonable media costs may be made for this software, and the entire software package should be distributed, although the source code portion may be withheld at your discretion. All copyright and authorship notices, both in the binary and the source code, must remain in place. I will attempt to provide limited support for this control via CompuServe mail or on the MSBASIC forum. My CompuServe ID is 76450,663. I will support ONLY the binary file provided in this package. If you even recompile the control, I absolve myself completely. If you're going to try this, note that the compiler used was Borland C++ version 3.1. At least one change, related to FP_SEG, would probably have to be made to use a Microsoft compiler.

The standard legal disclaimers also apply; I accept no responsability or liability for anything resulting from the use, misuse, or even mere presence of this control. I do not guarantee it does anything except use disk space. Although my intention, obviously, is for it to work properly, it is a programming exercise, not a commercial product. Remember: you get what you pay for (if you're lucky) and I'm not accepting payment. If you manage to wipe out a vital configuration file, don't come whining to me. I'll be sympathetic, helpful if I can, but I don't accept any responsability.

If you use this software, I would like to hear about it at the above CompuServe address, especially if you use it in a commercial application of any kind. (I'm especially interested in why you wouldn't just use GetPrivateProfileString and WritePrivateProfileString.)

This control was written for, and only tested with, Visual Basic 3.0.

Microsoft, Windows, and Visual Basic are trademarks of Microsoft Corporation, which undoubtedly reserves all rights to their use. Borland C++ is a trademark of Borland International, which also probably reserves all rights. I may have casually used other trademarked names in this document; if so, they belong to whoever owns them, who also probably reserves all rights.

Using INI.VBX

As you are no doubt aware, .INI files are of the format

[section]
entry=string

The INI.VBX custom control is used by filling in the necessary properties with values and assigning a value to the "Action" property. If the action succeeds, the "Action" property will be set to 0 following your assignment; if it fails, its value will be the value assigned to it.

Only those properties required for a given action need be set.
Properties:

Entry   corresponds to the "entry" portion of the .INI file entry. Doing a write with this assigned to a null string will **not** delete the section; use the **Delete Section** action instead. Doing a read with this assigned to a null string will **not** list all entries; that functionality is not supported. (At least, I don't think it is. Remember: lightly tested.) Type: string. Required for Actions: 1, 2, 3.

FileName   correspnds to the .INI file name. If no path name is given, the file is assumed to be in the windows directory. If a full path is given, that directory is used. Type: string. Required for Actions: 1, 2, 3, 4.

Text   corresponds to the "string" portion of the .INI file entry. This should be the control's default property. (I say should be because I haven't actually tested this. Like I said before, lightly tested.) Doing a write with this assigned to a null string will **not** delete the entry; use the **Delete Entry** action instead. Type: string. Required for Actions: 1.

Section   corresponds to the "section" portion of the .INI file entry. Type: string. Required for Actions: 1, 2, 3, 4.

Default   is used only for Read actions. If no entry is found, the Text property is assigned the value in the Default property. Type: string. Required for Actions: 2.

Action   is used to trigger the desired action by assigning it a value. The following values apply:
  0 - No Action
  1 - Write
  2 - Read
  3 - Delete Entry
  4 - Delete Section

If an action is successful, the Action property's value after being assigned one of these values will be 0. If it is not successful, the property will be the value it was actually assigned.

The following standard properties are also supported:

Name
Index
Left
Top

Since the control is visible only at design time, the positioning properties are not especially useful. Although provided, I do not expect the Index property to be especially useful either; I can think of no good reason for an array of these controls.

To gain further understanding of this control, I suggest consulting documentation of the GetPrivateProfileString and WritePrivateProfileString API calls. You could also play around with the enclosed demonstration/test VB 3.0 code. Or, you could even inspect my source code, which is mostly derived from the "generic" custom control code provided by Microsoft Corporation. Your best bet may be to play with the demo form while having notepad open on the .ini file you're using. You need to re-open the file with notepad after each change in order to see the change.

Known Limitation

1. Although the length of the string that can be written is limited only (presumably limited) by VB and Windows, only 256 characters can be read. This is a VBX-induced limitation.