

IniControls Overview

[Support](#) [Licence](#) [Components](#)

IniControls is an independently developed set of Delphi components that allows you to directly link edit boxes, list boxes, checkboxes and so on to an initialization file (an `.ini` file). You can do this by hand, of course, simply by using `TIniFile.WriteString` to transfer the values from a standard `TEdit` control to the `.ini` file. This is easy, but makes for a lot of very tedious code.

Ini-aware controls are similar to data-aware controls, but much easier to use. You can display the settings in a complete `.ini` file very simply:

Step 1: Assign a section name and a keyword to each control, telling it where its data is coming from.

Step 2: Link each control to a `TIniSource` component that knows the name of the file.

To save the settings is just as simple.

Step 3: Create a Save button and in its `OnClick` method, call `TIniSource.Save`.

With IniControls, configuration programs just write themselves!

IniControls can handle bitmapped options (for example `[Compatibility]` in `win.ini`), ordered n-tuples (for example, `[PrinterPorts]` in `win.ini`) and more. It has a complete repertoire of controls: edit boxes (alpha and numeric), check boxes, radio buttons, combo boxes and even grids.



This program is produced by a member of the Association of Shareware Professionals (ASP). If you are unable to resolve a shareware-related problem with an ASP member, ASP may be able to help. Click on the ASP logo for details.

This is \$Revision:: 1.23 \$ of this helpfile.

List of Components Available



TIniSource Provides the link between an .ini-aware control and an .ini file.



TIniEdit Text edit box.



TIniNEdit Numeric-only edit box.



TIniCheckbox Checkbox for yes-no options.



TIniRadioButton Radio buttons for multiple-choice options.



TIniCombobox For choosing an option from a fixed or drop-down list.



TIniGrid For displaying structured data.



TIniSource Component

[See also](#) [Properties](#) [Methods](#) [Events](#) [Tasks](#)

Unit

IniCtrls

Description

TIniSource is the interface between an initialization file (.ini file) and file aware controls on forms. TIniSource attaches to an .ini file through the [IniFilename](#) property. Initialization-file aware controls, such as .ini file edit boxes and .ini file grids, attach to a TIniSource through their [DataSource](#) properties.

Usually there is only one data source for each file: it does not make sense to have several. Usually, there will be several .ini-aware controls linked to the same TIniSource.

The IniFilename property identifies the file from which the data is obtained. When the form is created, the TIniSource component will load the values from the .ini file into its linked controls automatically.

Set the data in the edit controls to the saved values in the .ini file using the [Refresh](#) method. Save the data in the edit controls to the .ini file using the [Save](#) method. Set the data in the edit controls to their programmer-defined default values (which may be different from the saved values in the .ini file) using the [RestoreDefaults](#) method.

There are also [RefreshSection](#) and [SaveSection](#) and [RestoreSectionDefaults](#) methods to do the same for individual sections of the .ini file.

Delphi provides (and TIniSource uses) a [TIniFile](#) object, but you do not need to create one, because TIniSource does this for you. You should not normally operate on the TIniFile object directly, unless you need to enumerate section names or keywords, but it is available at run time in the [IniFile](#) property.

In addition to these properties, methods, and events, this component also has the properties and methods that apply to [all components](#).

See Also

[Overview of .INI-aware controls](#)

Properties

▶ Run-time only

🔑 Key properties


Enabled

▶ IniFile

🔑 IniFileName

▶ Modified

Methods


 Key methods

Activate


MapToRegistry

 Refresh

RefreshSection


 RestoreDefaults

RestoreSectionDefaults

 Save

SaveSection

Events

 Key events

OnActivate

OnDeactivate

Enabled Property

Declaration

```
property Enabled: boolean;
```

Description

Enabled is true when the `.ini`-aware controls are actively linked to the `.ini` file. Enabling takes place automatically when the `TIniSource` component is created, and when any linked control is created.

IniFile Property

Declaration

```
property IniFile: TIniFile;
```

Description

TIniFile is a Delphi-supplied object that simplifies operations on `.ini` files by providing cover methods for the API functions `GetPrivateProfileString`, `WritePrivateProfileString` and `GetPrivateProfileInt`.

TIniSource is a visual, non-windowed component that allows you to work with TIniFile objects in Designer.

The IniFile property is a run-time, read-only property that allows you access to the TIniFile object that TIniSource creates for you. You need to access it only if you need to access its `EraseSection`, `ReadSection` or `ReadSectionValues` methods.

Do not confuse the IniFile property with the [IniFilename](#) property.

IniFileName Property

Declaration

```
property IniFileName: TFilename;
```

Description

Specifies the name of the `.ini` file that `TIniSource` links to. The file is assumed to be in the Windows directory unless you specify a full path. Placing the `.ini` file in another directory is not recommended. Must have the extension `.ini`.

Modified Property

Applies to

TIniSource component, TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

```
property Modified: boolean;
```

Description

An `.ini`-aware component is Modified if the user has made any changes to the control since the last time it was refreshed or saved.

A `TIniSource` is Modified if any of the linked controls are Modified.

Typically referenced in an OnDeactivate method to issue a warning if the user exits the configuration screen without saving.

DefaultValue Property

Applies to

TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

Depends on component.

Description

At design time, the developer can specify a default value for an `.ini`-aware component. This is the value that will appear in the control if there is no entry in the `.ini` file for that component.

Hierarchical Property

Applies to

TIniGrid component.

Declaration

```
property Hierarchical: boolean;
```

Description

Consider a grid on your form containing the following:

	<u>Column</u>	<u>Column2</u>
	<u>1</u>	
RowA	This	is
RowB	a	grid

When Hierarchical is false (the default), the elements of the TIniGrid are written out to the `.ini` file against a single keyword, reading from left to right and top to bottom. The elements of the grid will be separated by commas. Any row and column headings in the grid will be ignored.

For example, the grid above will be written out as

```
MyGrid=This,is,a,grid
```

When Hierarchical is true, then there must be both row and column headings in the grid. You obtain row and column headings by setting FixedRows and FixedCols to greater than zero.

Each element in the grid will be written out as a separate entry in the `.ini` file.

For example, the grid above will be written out as

```
MyGrid\RowA\Column1=This
```

```
MyGrid\RowA\Column2=is
```

```
MyGrid\RowB\Column1=a
```

```
MyGrid\RowB\Column2=grid
```

The row and column headings must be valid Delphi identifiers. Any row or column that does not meet this requirement will be skipped. If you have more than one fixed row or column, the highest-numbered one (that is, the one nearest the data) will be used.

Irrespective of whether Hierarchical is true or false, a TIniGrid's DefaultValue is always specified as a comma-delimited list.

SectionName Property

Applies to

TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

```
property SectionName: string;
```

Description

Windows initialization files are plain ASCII files divided into sections. Each section has the following structure:

```
[sectionname]  
keyword=string
```

```
.  
. .  
.
```

Any value in the file is uniquely identified by its section name and its keyword. Section names and keywords are not case sensitive and may contain embedded spaces.

Where sections and keywords do not exist, an `.ini`-aware control when created or refreshed will display its default value, if any. At save time, Windows will create the keyword entry, the section, and even the entire `.ini` file if necessary.

```
.
```

Keyword Property

Applies to

TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

```
property Keyword: string;
```

Description

Windows initialization files are plain ASCII files divided into sections. Each section has the following structure:

```
[sectionname]  
keyword=string  
.  
.  
.
```

Any value in the file is uniquely identified by its SectionName and its keyword. Section names and keywords are not case sensitive and may contain embedded spaces.

Within a section, each of the following kinds of component must have its own keywords (Designer will not enforce this): TIniEdit, TIniNEdit, TIniGrid, TIniComboBox.

A group of TIniRadioButton components needs to have the same section name and keyword. Which button is "on" is determined by the CheckedValue property.

A group of TIniCheckBox components may, but need not, have the same section name and keyword. If they do have the same section name and keyword, the values of the checkboxes are bitmapped, and you will need to set the CheckedValue property.

Where sections and keywords do not exist, an `.ini`-aware control when created or refreshed will display its default value, if any. At save time, Windows will create the keyword entry, the section, and even the entire `.ini` file if necessary.

DataSource Property

Applies to

TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

```
property DataSource: TIniSource;
```

Description

Typically, a form in a configuration program will have many controls, but all of them will refer to a single `.ini` file. The link to the `.ini` file itself is handled by the Delphi TIniFile object. TIniFile objects can't be used in Designer: they are code-only. A TIniSource is a visual, non-windowed component that allows you to work with a TIniFile object at design time. The `DataSource` property of an `.ini`-aware control links the control to the TIniSource component, which in turn uses a TIniFile component to link to the `.ini` file itself.

WriteAsHex Property

Applies to

TIniNEdit component.

Declaration

```
property WriteAsHex: boolean;
```

Description

Normally, numeric values are written to the .ini file in decimal notation. It is conventional for some values to be written as hexadecimal, in other words `MyKeyword=0x000F` instead of `MyKeyword=15`.

The Delphi method TIniFile.ReadInteger and the Windows API function GetPrivateProfileInt are both unaffected by the choice of notation. It is only of interest for applications that read their configuration data as hexadecimal strings and decode them themselves, rather than as integer values.

CheckedValue Property

Applies to

[TIniCheckBox component](#), [TIniRadioButton component](#)

Description

The CheckedValue property permits you to associate several controls with a single SectionName and Keyword combination.

[CheckedValue property for TIniRadioButton components](#)

[CheckedValue property for TIniCheckbox components](#)

CheckedValue Property

Declaration

```
property CheckedValue: word;
```

Description

A group of TIniRadioButton components will all be controlled by the same keyword in an `.ini` file. The CheckedValue property links the value that appears against the keyword with a particular radio button. When the values match, that button is "on"; all the other buttons are off.

CheckedValue Property

Declaration

```
property CheckedValue: 0..31;
```

Description

A group of TIniCheckbox components can (but need not) all be controlled by the same keyword in an `.ini` file. The CheckedValue property links the value that appears against the keyword with a particular checkbox button.

In order to achieve this, the value in the `.ini` file is bitmapped, that is, each checkbox in the group is assigned a value that is a power of 2 (1, 2, 4, 8, 16, 32 and so on). The value that is written to the `.ini` file is the sum of the values of the individual checkboxes.

CheckedValue can take on the values 0 through 15. This will assign the checkbox the value 2^0 (=1) through 2^{15} (=2 147 483 648).

If AllowGrayed is true, then CheckedValue must be even: in other words 0, 2, 4 ... 30. This is because the cbGrayed attribute takes an additional bit position, and IniControls uses the odd bit positions for this.

Note: There is usually little point in bitmapping checkbox values. This is provided solely to support existing applications.

AllowGrayed Property

Applies to

TIniCheckBox component

Declaration

```
property AllowGrayed: boolean;
```

Description

See TCheckbox for a description of this property.

If AllowGrayed is true, then CheckedValue must be even.

Bitmapped Property

Applies to

TIniRadioButton component

Declaration

```
property Bitmapped: boolean;
```

Description

A set of radio buttons normally all use the same keyword in an .ini file. Which radio button is set is indicated by the value of the keyword. Usually (and this is strongly recommended) each set of radio buttons will have its own keyword. **You need to read what follows only if you choose to ignore this recommendation and combine several sets of radio buttons into the same keyword.**

To combine more than one set of radio buttons into a single keyword:

1. Set Bitmapped to true on all participating radio buttons.
2. Choose a value for each radio button that is either zero, or a power of two: that is, drawn from the set 0, 1, 2, 4, 8, 16, 32, ... 16384. These numbers must be unique across all participating radio buttons, not just unique inside that button's set, with one exception: you may have one zero-valued button in each set.

Inverted Property

Applies to

TIniCheckBox component

Declaration

```
property Inverted: boolean;
```

Description

It often happens, especially in older programs, that the `.ini` file says something like

```
[Preferences]  
ShowIcon=1
```

while the program says `if HideIcon then...`

Setting `Inverted` to true causes a 1 to be written to the `.ini` file if the checkbox is unchecked, and a zero to be written if it is checked.

Activate Method

Declaration

```
procedure Activate(DesiredEnabled: boolean);
```

Description

Calling Activate is equivalent to assigning a value to Enabled.

MapToRegistry Method

Declaration

```
procedure MapToRegistry;
```

Description

Windows 3.x keeps configuration data such as preferences in .ini files. These have some disadvantages: they are hard to manage centrally, and are available for editing by potentially naive users.

Under Windows NT and Windows 95, the recommended location for this kind of information is the Registry.

16-bit applications can of course continue to use .ini files. However, a more elegant solution is to map the .ini file to the Registry. This is done by creating a subkey for the .ini file in the Registry under HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\IniFileMapping. The effect of this is to cause all calls to the Windows API functions to be redirected to the Registry, transparently to the application program.

The MapToRegistry method creates this subkey for the .ini file specified in the IniFilename property.

Typically, you will call this method only once, in an installation program. There is no point in your application doing it repeatedly.

Under Windows 3.x, writing an IniFileMapping entry to the registry has no effect.

Refresh Method

Declaration

```
procedure Refresh;
```

Description

This method sets all of the linked controls to the values in the `.ini` file. Typically it will be used to provide a Revert to Saved menu option. Calling Refresh sets the Modified property of all linked controls to false.

Refresh Method

Applies to

TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

```
procedure Refresh (var Message: TMessage); message im_Refresh;
```

Description

Not intended for normal use. Use the Refresh or the RefreshSection methods of the TIniSource component instead. This will call each linked component's Refresh method.

Writes the values in a single control to the `.ini` file. Sets the Modified property to false.

Comment

Message `im_Refresh` is declared as `wm_User + 801`.

RefreshSection Method

Declaration

```
procedure RefreshSection (Sections: array of string);
```

Description

This method sets some of the linked controls to the values in the `.ini` file. You specify one or more section names in the parameter. Other sections are unaffected.

Typically it will be used to provide a Revert to Saved menu option.

Calling RefreshSection sets the Modified property of all affected controls to false.

RestoreDefaults Method

Declaration

```
procedure RestoreDefaults;
```

Description

This method sets all of the linked controls to the values specified by the developer in the DefaultValue property of each control: it restores the "factory settings". Calling RestoreDefaults sets the Modified property of all linked controls to true.

RestoreDefault Method

Applies to

TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

```
procedure RestoreDefault (var Message: TMessage); message im_RestoreDefault;
```

Description

Not intended for normal use. Use the RestoreDefaults or RestoreSectionDefaults methods of the TIniSource component instead. This will call each linked component's RestoreDefaults method.

Writes the values in a single control to the `.ini` file. Sets the Modified property to true.

Comment

Message `im_RestoreDefault` is declared as `wm_User + 802`.

RestoreSectionDefaults Method

Declaration

```
procedure RestoreSectionDefaults (Sections: array of string);
```

Description

This method sets some of the linked controls to the values specified by the developer in the DefaultValue property of each control: it restores the "factory settings".. You specify one or more section names in the parameter. Other sections are unaffected.

Calling RestoreSectionDefaults sets the Modified property of all affected controls to true.

Save Method

Declaration

```
procedure Save;
```

Description

Writes the values in all linked controls to the `.ini` file. Sets the Modified property of all linked controls to false.

Save Method

Applies to

TIniEdit component, TIniNEdit component, TIniCheckBox component, TIniRadioButton component, TIniComboBox component, TIniGrid component.

Declaration

```
procedure Save (var Message: TMessage); message im_Save;
```

Description

Not intended for normal use. Use the Save or the SaveSection methods of the TIniSource component instead. This will call each linked component's Save method.

Writes the values in a single control to the `.ini` file. Sets the Modified property to false.:

Comment

Message `im_Save` is declared as `wm_User + 800`.

SaveSection Method

Declaration

```
procedure SaveSection (Sections: array of string);
```

Description

Writes the values in some of the linked controls to the `.ini` file. You specify one or more section names in the parameter. Other sections are unaffected.

Sets the Modified property of all affected controls to false.

OnActivate Event

Declaration

```
property OnActivate: TNotifyEvent;
```

Description

A TIniSource component calls this event handler after the component has refreshed its linked controls, and just after Enabled is set to true.

Use this event handler to notify your application that the TIniSource component has been activated.

Your application may trigger this event by setting the Enabled property itself. In addition, Enabled is or may be set by changing IniFilename; creating an .ini-aware control on the fly, or changing any of a control's properties that determine the displayed value: DataSource, SectionName, Keyword.

Cautions

Your event handler should not perform operations that change the linked controls. In particular:

Avoid calling a control's Refresh method.

Avoid changing a control's DataSource property.

Do not test the Modified property.

All of these actions may trigger OnActivate, resulting in infinite recursion (reported as stack overflow).

OnDeactivate Event

Declaration

```
property OnDeactivate: TNotifyEvent;
```

Description

A TIniSource component calls this event handler after the component has discarded its list of linked controls, and after Enabled is set to false.

Use this event handler to notify your application that the TIniSource component has been deactivated.

Your event handler should not perform operations that change the linked controls, or create new linked controls on the fly. This is because any operation on a linked control will cause it to re-link itself to the TIniSource component. This in turn will set the control Enabled again.

Your application may trigger this event by setting the Enabled property itself. In addition, Enabled is set to false by changing IniFilename.


Caution

Do not test the Modified property in this method. It will always return false. If you wish to trap an exit-without-save, test Modified in the Close button's OnClick event.



Using the TIniSource Component

TIniSource Reference

Purpose

TIniSource  is a visual component that corresponds to a Delphi TIniFile object. It allows you to specify the name of the .ini file and thereafter takes charge of all movement of data between the .ini-aware controls on the form and the .ini file.

To Link a Control to an .ini File

1. Place a TIniSource component  on the form. (This will be invisible at runtime.) Its name will be `Inisource1` but you can change this. Set the IniFilename property to the name of the .ini file.
2. Place an .ini-aware control, for example a TIniEdit component  on the form. Set its DataSource property to `Inisource1` (or whatever name you chose in Step 1.) Set its SectionName and Keyword properties to the desired values. If desired, set the DefaultValue property.

To Save and Load Settings

1. Create buttons on form labelled Save and Load.
2. Create OnClick event handlers for each button. In these event handlers, call `Inisource1.Save` and `Inisource1.Refresh`.

TIniEdit Component

See also [Properties](#) [Methods](#) [Tasks](#)

Description

A TIniEdit component is an `.ini`-aware edit box with all the capabilities of an ordinary edit box (a [TEdit](#) component).

Unlike an ordinary edit box, the `.ini`-aware edit box is linked to a particular data item in an `.ini` file. You create this link by specifying three properties.

Set the [DataSource](#) property to a [TIniSource](#) component. Set the [SectionName](#) property to the section in the `.ini` file where the data resides. Set the [Keyword](#) property to the keyword that identifies the data item.

TIniEdit is a descendant of TEdit. All [TEdit](#) properties, methods and events are available.

See Also

[Overview](#) of .INI-aware controls

[TIniNEdit](#) component

TIniEdit Properties

▶ Run-time only

🔑 Key properties

🔑 DataSource


DefaultValue

🔑 Keyword

🔑 Modified

🔑 SectionName

TiniEdit Methods

 Key methods

Refresh

RestoreDefault

Save

Using the TIniEdit Component

TIniEdit Reference

Purpose

TIniEdit is an `.ini` file-aware version of the TEdit component.


Use the TIniEdit component to read or write a single string from a specific section and keyword in an `.ini` file.

To read and write a comma-delimited string with multiple independent values, use the TIniGrid component.

To limit user input to numbers, use the TIniNEdit component.

Tasks

To link the TIniEdit component to an `.ini` file:

1. Add a TIniSource  component to your form, if you haven't already (you only need one for each `.ini` file).
2. Specify the name of the TIniSource component as the value of the DataSource property.
3. Specify the SectionName and Keyword properties to identify the data item within the `.ini` file that you want to display.

Association of Shareware Professionals (ASP)

This program is produced by a member of the Association of Shareware Professionals (ASP). ASP wants to make sure that the shareware principle works for you. If you are unable to resolve a shareware-related problem with an ASP member by contacting the member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute or problem with an ASP member, **but does not provide technical support for members' products**. Please write to the ASP Ombudsman at 545 Grover Road, Muskegon, MI USA 49442-9427, Fax 616-788-2765, or send a CompuServe message via CompuServe Mail to ASP Ombudsman 70007,3536 (Internet: 70007.3536@compuserve.com).

How to Contact Us

You can contact us by mail, by fax, or by email. We don't offer telephone support because the timezones don't fit (our office hours are 10:30 PM to 06:30 AM PST).

Defect Reports

If you report problems with IniControls, we will be glad to try and resolve them. If you have suggestions about how the components could be improved to be more useful to you, they will be welcomed.

When you report problems, it is *really important* for us to know what release of IniControls you are running (the problem you report may already have been identified and fixed in a later release). The revision number of the code is returned by the string function `IniCtrls.Revision`.

When you buy a developer licence, you get guaranteed support for 90 days. The guarantee means that if IniControls does not work in your application, we will do our best to fix it so that it does work. If we can't fix the defect we will refund your licence fee.

We will support you even if you have not yet licensed IniControls, but the support will be limited to getting the components installed and working to allow you to evaluate them, and if your problem requires program changes, they may be accorded a low priority.

Support by Mail

Paul Keating
Prodigy Computing
PO Box 2194
Cramerview
South Africa 2060

Please be patient if you use the mail. Airmail delivery from the US takes 3–5 weeks. Surface mail takes up to 4 months.

Support by Fax

Paul Keating

Prodigy Computing

+27-11-888-2370 OR +27-11-792-9512.

*

Support by Email

Email Paul Keating:

CompuServe **73770,660**

Internet **keating@acm.org**

Licensing IniControls

See also

IniControls is distributed as shareware. It is not in the "public domain". It is not free software. However, you do have the opportunity to try it for 90 days before you pay for it, subject to the terms of the evaluation licence.

You may not distribute programs of your own that incorporate `inictrls.dcu` without a developer licence. A developer licence costs \$17-50 per developer workstation (or \$37-50 if you want the source code). Generous site licences are available. This is a one-off fee: there are no royalties to pay.

You may distribute copies of the IniControls package to other developers if you wish: please read the terms of the distribution licence first.

How to Obtain a Developer Licence

Benefits of Obtaining a Developer Licence

Developer Licence Terms

See Also

[How to Obtain a Developer Licence](#)

[Benefits of Obtaining a Developer Licence](#)

[Developer Licence Terms](#)

[Evaluation Licence Terms](#)

[Distribution Licence Terms](#)

[Source Code Licence Terms](#)

[Warranty](#)

[Copyright](#)

How to Obtain a Developer Licence

See also

The licence form is in this helpfile, and Help will print it for you, or copy it to the Windows clipboard. Just select your preferred method of payment and follow the prompts.



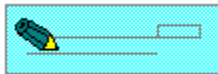
If you can pay by **credit card**, you can obtain a licence by email or by fax or by mail.



We accept NetCash for email registrations. NetCash is quick and it's fun!



If you have a **CompuServe account**, the quickest and easiest way to buy a licence is to use CompuServe's online shareware registration service. This way, the whole process takes place online, you don't have to bother with a form, and the licence fee is simply added to your regular monthly CompuServe bill.



If you need to send a **cheque**, then you will need to mail your licence fee to us.



Some companies make it difficult to send payment with order. Corporate users who have this problem may issue a purchase order.

Benefits of Obtaining a Developer Licence

See also

1. *Deployability.* Without a developer licence, you can't deploy applications containing `inictrls.dcu` to end-user sites because the programs won't work. When we receive your licence form we will send you a licence number that will permit you to deploy unlimited copies of your programs.
2. *Support.* Developer licensees get guaranteed support for 90 days. If IniControls does not work on your system we will do our best to fix it so that it does work. We can't promise to make it work in all cases. If we can't fix the defect we will refund your fee.
3. *Nag-free usage.* After your evaluation licence expires, the program will nag you to license it.

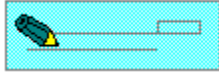
Developer Licence Terms


See also

In return for the developer licence fee, Prodigy Computing grants you a non-exclusive licence to use `inictrls.dcu` indefinitely without further payment and to distribute programs that contain it freely and without restraint.

1. You may install `inictrls.dcu` on one Delphi developer's workstation: that is, any computer where there is also installed a licensed copy of Borland International's Delphi development environment.
2. You may write and compile your own application programs using `inictrls.dcu`.
3. You may use, reproduce, give away or sell any program you write using `inictrls.dcu`, without additional licence or fees.
4. All copies of the programs you create must bear a valid copyright notice.
5. The terms of Prodigy Computing's warranty to you are as set out in this helpfile. You agree to those terms. Prodigy Computing provides no warranty to anyone else: you are solely responsible to anyone receiving your programs for support, service, upgrades, or technical or other assistance.
6. You will indemnify and hold Prodigy Computing harmless from and against any claims arising out of the use, reproduction or distribution of your programs.
7. The file `inictrls.dcu` is and remains the property of Prodigy Computing.
8. This licence is to be construed according to the laws of South Africa. You consent to the jurisdiction of the Magistrate's Court, Randburg, Gauteng in any legal action that is brought by or against you in terms of this licence.

Licensing IniControls by Mail



On the licence form you will see a button like this:  Press that button to send the form to your printer. Fill it in and mail it to the address shown on the form.

You can pay by credit card (we accept MasterCard and Visa) or by cheque. We can accept cheques in US, Canadian and South African currency. Our agents will process cheques in other currencies. Please don't draw a cheque in a currency not your own. It costs us more than their face value to cash them.

Sending cash through the mail is risky, but if you accept the risk, we'll happily accept the banknotes.

If you pay by credit card, we ask for your address, because our bank requires us to mail your copy of the completed credit card voucher to you. This address will be used for no other purpose.

Mail is slow. Surface mail (one way) from the US to Southern Africa takes anything up to three months.


Airmail takes 2-4 weeks. Surface mail (one way) from Europe takes 3-5 weeks. Airmail takes 8 days.

We will fax your licence number to you (if you provide a fax number) or email it (if you provide an email address), and after some weeks you will receive a credit card voucher in the mail. If you do not provide a fax number or an email address, we will send you your licence number and credit card voucher by airmail.

Another currency

If your bank account is in a currency other than US or Canadian dollars or South African rands, this is unlikely to be a problem, but please contact us before sending payment for pricing and payment instructions.

Licensing IniControls by Fax


On the licence form you will see a button like this:  Press this button to send the form to your printer. Fill it in and fax it to the number shown on the form.

When you pay by fax you can choose one of two methods. You can pay by credit card (we accept MasterCard and Visa), or send us a company purchase order.

If you use your credit card, we ask for your address because our bank requires us to mail your copy of the completed credit card voucher to you. This address will be used for no other purpose.

We will fax or email your licence number back to you, and after some weeks you will receive a credit card voucher in the mail.

Licensing IniControls by Email

On the [licence form](#) you will see a button like this:  Press that button to copy the form to the clipboard. Use a text editor or the message editor of your communications program to fill it in. Then email it to us: email addresses are on the form.

When you pay by email you can choose one of two methods. You can pay by credit card (we accept MasterCard and Visa), or send us a [NetCash](#) voucher.

We will email your licence number back to you.

If you use your credit card, we ask for your address because our bank requires us to mail your copy of the completed credit card voucher to you. This address will be used for no other purpose.

If you are concerned about sending your credit card number over the Internet (and you should be), then either use the [split-number technique](#) or [encrypt your message](#) using PGP.

Split-Number Technique

Credit card numbers are easy to pick out of a mail message automatically because they have a very fixed structure: 1111-2222-3333-4444 05/96. Packet-sniffers can steal your number by scanning messages and looking for this pattern. If you send two registration messages with the first half of your number in the first message, and the second half of your number in the second message, your message will be proof against pattern-recognizers.

If you're unconvinced, spend three minutes thinking about how you would write a program that would not be fazed by this technique.

Encrypting Your Message using PGP

If you have a copy of PGP, you can use it to encrypt your message. Included in the IniControls package is a file called `public.key`, which contains Prodigy Computing's PGP public key. To add this key to your keyring, use the command `pgp -ka public.key`. To encrypt your registration form, save it as a text file (for example `regform.txt`), then use the command `pgp -ea regform.txt Prodigy`. Import or paste the resulting file into an email message.

PGP is short for "Pretty Good Privacy", a public-key cryptosystem using the Rivest-Shamir-Adleman algorithm. PGP is a trademark of Philip Zimmermann and Phil's Pretty Good Software, 3021 Eleventh Street, Boulder, Colorado 80304 USA. PGP is © copyright Philip R. Zimmermann, 1990-1993.

Please don't ask Prodigy Computing or Philip Zimmermann to send you a copy of PGP. Both will refuse, for legal reasons (see below). However, you can probably obtain PGP from your nearest bulletin board, or via anonymous ftp from `nic.funet.fi`, `ghost.dsi.unimi.it` or `src.doc.ic.ac.uk`.

Notice: If you as a private citizen use or distribute PGP within the US, you may be infringing US Patent 4405829, held by Public Key Partners. If you import PGP into the US from another country, you may be infringing US laws regarding the import of munitions. If you import PGP into your own country from the US, you (and the source you import it from) are almost certainly infringing US laws regarding the export of munitions.

Licensing IniControls Online

CompuServe offers a way to license shareware online. When you do this, the licence fee is added to your CompuServe charges and CompuServe will bill you in the normal way.

To license the program, first find its description in the CompuServe's Shareware database.

To begin, GO SWREG and select "Register Shareware" from the menu. CompuServe calls the process of buying a licence "registration". A list of search criteria will be displayed to you. CompuServe has assigned the object-only IniControls package the **Registration ID 8864**. The with-source option has the **Registration ID 8865**. Enter one of these IDs under option #1 on the search criteria menu. You will then navigate directly to a description of IniControls. This is the easiest and fastest method of licensing a program.

If you have any questions or concerns about the service, send a message to the Shareware Administrator by selecting the "Provide Feedback" option at the main SWREG menu.

We will send you your licence number (and the program source, if applicable) by CompuServe email.

Purchase Orders

Corporate users who find it administratively difficult to send payment with order may issue a purchase order.

Because of the administration involved, we only accept corporate purchase orders for four copies (\$70) or more. We will invoice your company in US dollars (or whatever currency you prefer) and issue the licence numbers on receipt of payment.

Generous site licence discounts are available: contact one of the [support](#) addresses.

Mail your purchase order to: Prodigy Computing, PO Box 2194 Cramerview, South Africa 2060.

NetCash

NetCash is a new form of online currency that may be used by anyone who has an electronic mail address.

NetCash coupons are essentially a string of numbers. You pay cash, or provide services, to obtain them. You trade them simply by including them in an e-mail message. Like real cash, there are no transaction fees when NetCash changes hands. For this reason, NetCash is ideal for paying small amounts of money over the Internet.

The NetBank, operated by Software Agents Inc, manages the exchange of NetCash between users. The NetBank validates NetCash coupons and makes change.

You can buy NetCash from the NetBank by mail, fax or email. For full details, send an email message to netbank-info@agents.com. Here is a quick summary of how it works:

This is a NetCash coupon that represents a five dollar bill:

```
NetCash US$ 5.00 E123456H789012W
```

To pay for your licence, you simply send one or more NetCash coupons to us in the licence form. After you give the coupon to us, it is no longer yours. You may only spend a NetCash coupon once.

When we receive your coupon, we ask the NetBank if it's valid, by sending the following email message:

```
NetCash US$ 5.00 E123456H789012W /Accept
```

The NetBank will check that the NetCash is valid. If it is, the NetBank will mark the coupon as used up, and issue us a fresh NetCash coupon with a new number. We will keep the coupon in our "cash drawer" and ultimately deposit it into our NetBank account. When we have enough to make it worthwhile, we will turn the coupons into cash.

You pay 2% commission to buy NetCash coupons from the NetBank.

.Exchange Rate Fluctuations

The US dollar licence fee will be converted to South African rands at the ruling exchange rate on the day your order is processed, and that amount will be billed to your credit card. Because exchange rates fluctuate, your credit card company may use a slightly different rate when it converts the charge back into your own currency. So the amount actually charged to your card may be slightly lower or slightly higher.

South African users will be billed directly in rands. The rand licence fee is R72-50 (including 14% VAT), or R156-00 if the program source is included.

See Also

[Evaluation Licence Terms](#)

[Distribution Licence Terms](#)

[Source Code Licence Terms](#)

[Developer Licence Terms](#)

[Warranty](#)

[Copyright](#)

Evaluation Licence Terms

See also

These terms form a contract between us. Even though you have not signed the contract, using the product for an extended period indicates your assent.

1. You may install `inictrls.dcu` on a Delphi **developer's workstation**: that is, any computer where there is also installed a licensed copy of Borland International's Delphi development environment.
2. You may include `inictrls.dcu` in as many programs as required to evaluate the product. You may continue to use the unit `inictrls.dcu` without payment for **a trial period of 90 days**.
3. You may **not** distribute to end-users any program that uses `inictrls.dcu`.
4. At the end of the 90-day trial period you must either delete the program from your computer system or obtain a developer licence to use the program.
5. To legally distribute a program of your own that incorporates `inictrls.dcu`, you must likewise obtain a developer licence to use the program.
6. You agree to the terms of the warranty.

Shareware

Shareware means "try-before-you-buy" software.

Shareware is copyrighted software which is distributed by authors through bulletin boards, on-line services, disk vendors, and copies passed among friends. It is commercial software that you are allowed to try before you pay for it.

You do not have to pay for the software until you have had an opportunity to try it out for a reasonable period. You use the software on your own system, in your own work environment, for a fixed period, like 90 days. If you decide not to continue using it, you throw it away and forget all about it. You only pay for it if you continue to use it.

Shareware is a distribution method, not a type of software. There is good and bad shareware, just as there is good and bad retail software. If retail software turns out to be unsatisfactory, you might have trouble getting your money back. With shareware, you know if it's good or bad before you pay for it.

Source Code Licence Terms

See also

You may choose to license a copy of the source code along with your developer licence for `inictrls.dcu`. If you do, your use of the source code is subject to the following conditions:

1. You may write and compile your own application programs using the source code.
2. You may use, reproduce, give away or sell any program you write using the source code, in executable form only, without additional licence or fees; provided that the programs that you distribute may not be merely a subset of `inictrls.dcu`.
3. All copies of the programs you create must bear a valid copyright notice.
4. The terms of Prodigy Computing's warranty to you are as set out in this helpfile. You agree to those terms. Prodigy Computing provides no warranty to anyone else: you are solely responsible to anyone receiving your programs for support, service, upgrades, or technical or other assistance.
5. You will indemnify and hold Prodigy Computing harmless from and against any claims arising out of the use, reproduction or distribution of your programs.
6. The source code is and remains the property of Prodigy Computing. Regardless of any modifications that you make, you may not distribute the source code. You are not, of course, restricted from distributing source code that is entirely your own.
7. This licence is to be construed according to the laws of South Africa. You consent to the jurisdiction of the Magistrate's Court, Randburg, Gauteng in any legal action that is brought by or against you in terms of this licence.

The source code will be delivered to you by email after we receive your licence form. To keep costs (and licence fees) low, we don't mail out disks.

Distribution Licence Terms

See also

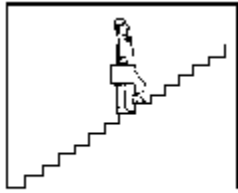
The following may distribute the IniControls package including `inictrls.dcu` and all of its supporting materials completely unaltered, without further permission:

- ◇ private individuals passing copies to friends without charge;
- ◇ bulletin board systems;
- ◇ bulletin board file distribution networks;
- ◇ disk vendors who are ASP Vendor Members; and
- ◇ disk vendors who are not ASP Vendor Members but who disclose to their customers prior to purchase in a visible fashion that the product is shareware, the nature of shareware, and that separate payment to the copyright owner is required if the product is used beyond the 90-day trial period.

For other channels of distribution or to distribute in modified form, you must consult the data record in the file VENDINFO.DIZ, which is included in the package, and which is hereby incorporated by reference. Any distribution satisfying all the distribution requirements expressed in that data record is hereby authorized. Distribution that does not conform to the requirements of this licence nor to the requirements expressed in the attached data record requires explicit written permission from the copyright owner in every case.

This distribution licence does not permit you to incorporate `inictrls.dcu` into an application program for distribution for end-users. To distribute the product in that form you must have a developer licence.

Copyright



PRODIGY

This unit was written by Paul Keating. The program and its supporting materials are copyright © 1995 by Prodigy Computing (Pty) Limited, PO Box 2194, Cramerview 2060, South Africa.

No Warranty

IniControls is made available *voetstoots* (a Roman-Dutch legal condition that excludes, among other things, all supplier's warranties of any kind, express or implied, against defects latent or patent). You assume all responsibility for the adverse consequences of any defects in IniControls, including any adverse consequences of including it in your own products. If IniControls does not work, or if it works differently from the way you expected or intended or were led to believe, your sole remedy is to stop using it, uninstall it from your Delphi environment, and remove all references to it from your own program code. You may in certain cases claim a refund of any licence fee you have paid.

TIniNEdit Component

[See also](#) [Properties](#) [Methods](#) [Tasks](#)

Description

A TIniNEdit component is a numeric-only `.ini`-aware edit box with all the capabilities of an ordinary edit box (a [TEdit](#) component). Unlike an ordinary edit box, the `.ini`-aware edit box is linked to a particular data item in an `.ini` file. You create this link by specifying three properties.

Set the [DataSource](#) property to a [TIniSource](#) component. Set the [SectionName](#) property to the section in the `.ini` file where the data resides. Set the [Keyword](#) property to the keyword that identifies the data item.

TIniNEdit is a descendant of [TIniEdit](#). All TIniEdit and [TEdit](#) properties, methods and events are available.

See Also


[Overview of .INI-aware controls](#)


[TIniEdit](#) component


[TIniCheckbox](#) component

[TIniRadioButton](#) component

TIniNEdit Properties

 Run-time only


 Key properties

 DataSource

DefaultValue


 Keyword

 Modified

 SectionName

WriteAsHex

TIniNEdit Methods

 Key methods

Refresh

RestoreDefault

Save

Using the TIniEdit Component

TIniEdit Reference

Purpose


TIniEdit is an `.ini` file-aware version of the TEdit component that accepts numeric input only.

Use the TIniEdit component to read or write a single number from a specific section and keyword in an `.ini` file.

To read and write a comma-delimited string with multiple independent values, use the TIniGrid component.

Tasks

To link the TIniEdit component to an `.ini` file:

1. Add a TIniSource  component to your form, if you haven't already (you only need one for each `.ini` file).
2. Specify the name of the TIniSource component as the value of the DataSource property.
3. Specify the SectionName and Keyword properties to identify the data item within the `.ini` file that you want to display.

TIniCheckbox Component

[See also](#) [Properties](#) [Methods](#) [Tasks](#)

Description

A TIniCheckbox component is an `.ini`-aware checkbox with all the capabilities of an ordinary checkbox (a [TCheckbox](#) component). Unlike an ordinary checkbox, the `.ini`-aware checkbox is linked to a particular data item in an `.ini` file. You create this link by specifying three or four properties.

Set the [DataSource](#) property to a [TIniSource](#) component. Set the [SectionName](#) property to the section in the `.ini` file where the data resides. Set the [Keyword](#) property to the keyword that identifies the data item. Optionally, set the [CheckedValue](#) property to indicate the value that is to be written to the `.ini` file when the checkbox is checked. `CheckedValue` defaults to zero, which causes 1 (2^0) to be written to the `.ini` file when the checkbox is checked.


TIniCheckbox is a descendant of [TCheckbox](#). All TCheckbox properties, methods and events are available.


See Also

[Overview of .INI-aware controls](#)

[TIniRadioButton](#) component

TIniCheckbox Properties


 Run-time only

 Key properties

CheckedValue

 DataSource

DefaultValue


 Keyword

Inverted

 Modified

 SectionName

TIniCheckbox Methods

 Key methods

Refresh

RestoreDefault

Save

Using the TIniCheckbox Component

TIniCheckbox Reference


Purpose

TIniCheckbox is an `.ini` file-aware version of the TCheckbox component.

Use the TIniCheckbox component to read or write a single on/off value from a specific section and keyword in an `.ini` file.

Tasks

To link the TIniCheckbox component to an `.ini` file:

1. Add a TIniSource  component to your form, if you haven't already (you only need one for each `.ini` file).
2. Specify the name of the TIniSource component as the value of the DataSource property.
3. Specify the SectionName and Keyword properties to identify the data item within the `.ini` file that you want to display.
4. To have a zero value in the `.ini` file appear as a checked box instead of an unchecked box, set Inverted to true.
5. To combine the settings of several checkboxes into a single keyword in the `.ini` file, specify the same values for the SectionName and Keyword properties, and give each checkbox control a different value in the CheckedValue property.

TIniRadioButton Component

[See also](#) [Properties](#) [Methods](#) [Tasks](#)

Description

A TIniRadioButton component is an `.ini`-aware radio button with all the capabilities of an ordinary radio button (a [TRadioButton](#) component). Unlike an ordinary radio button, the `.ini`-aware radio button is linked to a particular data item in an `.ini` file. You create this link by specifying four properties.

Set the [DataSource](#) property to a [TIniSource](#) component. Set the [SectionName](#) property to the section in the `.ini` file where the data resides. Set the [Keyword](#) property to the keyword that identifies the data item. Set the [CheckedValue](#) property to the value that is to be written to the `.ini` file when the radio button is pressed.


TIniRadioButton is a descendant of [TRadioButton](#). All TRadioButton properties, methods and events are available.


See Also

[Overview](#) of .INI-aware controls

[TIniCheckbox](#) component


TIniRadioButton Properties

 Run-time only

 Key properties

Bitmapped

 CheckedValue


 DataSource
DefaultValue

 Keyword

 Modified

 SectionName

TIniRadioButton Methods

 Key methods

Refresh

RestoreDefault

Save

Using the TIniRadioButton Component

TIniRadioButton Reference


Purpose

TIniRadioButton is an `.ini` file-aware version of the TRadioButton component.

Use a group of TIniRadioButton components to read or write a single numeric value from a specific section and keyword in an `.ini` file.

Tasks

To link the TIniRadioButton component to an `.ini` file:

1. Add a TIniSource  component to your form, if you haven't already (you only need one for each `.ini` file).
2. Specify the name of the TIniSource component as the value of the DataSource property.
3. Specify the SectionName and Keyword properties to identify the data item within the `.ini` file that you want to display.
4. Specify the same values for the SectionName and Keyword properties for each radio button in the group.
5. Give each RadioButton control a different value in the CheckedValue property. This is the value that will be written out to represent the settings of the buttons in the group.

TIniGrid Component

See also [Properties](#) [Methods](#) [Tasks](#)

Description

For some purposes, having an `.ini`-aware edit box is insufficiently powerful. Some programs, and Windows itself, write out their `.ini` file settings as a set of ordered pairs, or triples, or even 16-tuples. For example, consider this entry in `win.ini`:

```
[PrinterPorts]
Epson FX-80=EPSON9,LPT2:,15,45
Generic / Text Only=TTY,FILE:,15,45
NEC Silentwriter2 90=pscript,LPT1:,45,90
```

To write out the four elements of each value under proper control is difficult with just an edit box. However, using a 1x4 grid, each individual value can be sensibly displayed and individually validated.

A TIniGrid component is an `.ini`-aware string grid with all the capabilities of an ordinary string grid (a [TStringGrid](#) component). Unlike an ordinary string grid, the `.ini`-aware string grid is linked to a particular data item (or items, if [Hierarchical](#) is true). You create this link by specifying three properties.


Set the [DataSource](#) property to a [TIniSource](#) component. Set the [SectionName](#) property to the section in the `.ini` file where the data resides. Set the [Keyword](#) property to the keyword that identifies the data item.


TIniGrid is a descendant of [TStringGrid](#). All TStringGrid properties, methods and events are available.

See Also

[Overview of .INI-aware controls](#)

TIniGrid Properties

 Run-time only


 Key properties

 DataSource

DefaultValue


 Keyword

 Modified

 SectionName

Hierarchical

TIniGrid Methods

 Key methods

Refresh

RestoreDefault

Save

Using the TIniGrid Component

TIniGrid Reference


Purpose

TIniGrid is an `.ini` file-aware version of the TStringGrid component.

Use the TIniGrid component to read or write ordered pairs of values from a specific section and keyword in an `.ini` file.

Tasks

To link the TIniGrid component to an `.ini` file:

1. Add a TIniSource  component to your form, if you haven't already (you only need one for each `.ini` file).
2. Specify the name of the TIniSource component as the value of the DataSource property.
3. Specify the SectionName and Keyword properties to identify the data item within the `.ini` file that you want to display.

TIniCombobox Component

[See also](#) [Properties](#) [Methods](#) [Tasks](#)

Description

A TIniComboBox component is an `.ini`-aware combo box with all the capabilities of an ordinary combo box (a [TComboBox](#) component). Unlike an ordinary combo box, you can use the `.ini`-aware combo box to enter data into an `.ini` file, or to simply display data from an `.ini` file. Link the `.ini`-aware combo box with an `.ini` file by specifying the `.ini` source component (TIniSource) that identifies the `.ini` file as the value of the combo box's [DataSource](#) property. Specify the section name and keyword in the `.ini` file you want to access as the value of the [SectionName](#) and [Keyword](#) properties.

TIniComboBox is a descendant of TComboBox. All [TComboBox](#) properties, methods and events are available.

Using the TIniCombobox Component

TIniCombobox Reference


Purpose

TIniComboBox is an `.ini` file-aware version of the TComboBox component.

Use the TIniComboBox component to read or write a single string from a specific section and keyword in an `.ini` file. The combo box, as its name suggests, combines the edit box and the list box into a single control so that the user doesn't have to type the value.

Tasks


To link the TIniComboBox component to an `.ini` file:


1. Add a TIniSource  component to your form, if you haven't already (you only need one for each `.ini` file).
2. Specify the name of the TIniSource component as the value of the DataSource property.
3. Specify the SectionName and Keyword properties to identify the data item within the `.ini` file that you want to display.
4. To limit user input to a specified list, change the Style property to `csDropDownList` and enter the permitted values in the Items property. If you do this, the value will be saved as a number indicating the position of the chosen value in `Items`. Otherwise, the value will be saved as a string.


See Also

[Overview](#) of .INI-aware controls

TIniCombobox Properties

 Run-time only

 Key properties

 DataSource


DefaultValue

 Keyword

 Modified

 SectionName

TIniCombobox Methods

 Key methods

Refresh

RestoreDefault

Save



IniControls Developer Licence

Mail: Cheques

I want a developer licence for my copy of IniControls. Please license the program in the name of

_____ and send me a licence number.

I enclose a cheque in favour of Prodigy Computing (Pty) Ltd for the amount shown below.

My bank account is in	Currency	Standard Fee	Fee Including Source Code
US dollars	USD	\$17-50	\$37-50
Canadian dollars	CAD	\$23-75	\$51-00
SA rands	ZAR	R72-50 inc VAT	R156-00 inc VAT

Another currency...

Email address or fax number or mailing address *(for notification of licence number)*:

NB: If you want the source code, please supply an email address.

Mail to:

Prodigy Computing
PO Box 2194
Cramerview 2060
South Africa



IniControls Developer Licence

Mail: Credit Cards

I want a developer licence for my copy of IniControls. Please license the program in the name of

and send me a licence number.

Charge the \$17-50 licence fee to my MasterCard Visa account.

I want the source code.

Charge the \$37-50 licence fee to my MasterCard Visa account.

I understand that exchange rate fluctuations may slightly affect the amount billed.

Number: _____

Expiry date: _____ / _____ *(must be supplied)*

Name on card *(if different from name above)*:

Postal Address *(for return of credit card voucher: our bank insists that we mail it to you)*

Signature

Email address or fax number *(for notification of licence number: if omitted you will be notified by mail)*:

NB: If you want the source code, please supply an email address.

Mail to:

Prodigy Computing
PO Box 2194
Cramerview 2060
South Africa



IniControls Developer Licence Form

Copy to Clipboard 

Fax/Email

I want a developer licence for my copy of IniControls. Please license the program in the name of

and send me a licence number.

I am paying the normal \$17-50 licence fee.

I want the source code and so I am paying the \$37-50 licence fee.

Here is a NetCash coupon for the licence fee: _____

Charge the licence fee to my MasterCard Visa account. I understand that exchange rate fluctuations may slightly affect the amount billed.

Note: If you are concerned about sending your credit card number over the Internet, then use either the split-number technique or encrypt your message using PGP.

Card number: _____

Expiry date: _____ / _____ (must be supplied)

Name on card (if different from name above):

Postal Address (for return of credit card voucher: our bank insists that we mail it to you)

Signature (for fax orders):

Email address or fax number (for notification of licence number):

NB: If you want the source code, please supply an email address.

Fax to:

Prodigy Computing, +27-11-888-2370 or +27-11-792-9512. The + means your international call prefix, such as 011 in the US or 00 in Europe.

Email to:

CompuServe: 73770,660

Internet: keating@acm.org

