Registration Wizard Pro v5.01 User's Guide

What is Registration Wizard Pro?

Registration Wizard Pro is an ActiveX control that is designed for use with Visual Basic 5.0 to allow programmers to easily implement characteristics of shareware applications without the need to code and test such routines. Registration Wizard handles the task of implementing an evaluation period by limiting the period of time a user can use an application without registering as well as the registration code system to unlock your application.

What is new to version 5.01?

- Registration Wizard Pro v5.01 fixes a problem with the evaluation period expiring instantly. This was caused by the different in the format of the date by various countries (*i.e.* MM-DD-YY and DD-MM-YY).

Registration Wizard Pro Properties

• AppTitle Property

The AppTitle Property allows you to set the caption that will be displayed on any dialog boxes that Registration Wizard Pro will show to the user (*i.e.* Registration Code Dialog, Unregistered Delay, etc.). This property should resemble the title of your application.

Example: **RegWizardPro1.AppTitle = "Registration Wizard Pro"**

• AutoReRegister Property

The AutoReRegister Property allows you to have Registration Wizard Pro automatically re-register a user with the new correct registration code when their is a change in their SystemID number. This property is only used if you are using System Dependant registration codes. Upon a user being "re-registered", the UserReRegistered event will be fired. This event should be used to inform the user of the change and, if you desire, tell them their new registration code for future references.

Example: **RegWizardPro1.AutoReRegister = True**

• DateSensitive Property

The DateSensitive Property allows you to generate registration codes that are date sensitive (*i.e.* the system's date must match the date that the registration code was generated in order for the registration code to work.)

Examples:

'Enable date sensitive registration codes

RegWizardPro1.DateSensitive = True

'Disable date sensitive registration codes
RegWizardPro1.DateSensitive = False

- **DisableInEnvironment Property**

The DisableInEnvironment Property allows the programmer to disable the control while it is running in the Visual Basic 5.0 Development Environment. If enabled, the control will report that the application is registered even though there may not be any registered users. **Note:** This property does not affect Registration Wizard Pro's functionality in your compiled applications.

Examples:

'Enabled the DisableInEnvironment Property to always report the application is registered
RegWizardPro1.DisableInEnvironment = True

'Disable the DisableInEnvironment Property to report the correct registered status of the application.
RegWizardPro1.DisableInEnvironment = False

- **EncryptionKey Property**

The EncryptionKey Property contains your unique key which used in the calculation of registration codes. This allows you to use the control in multiple applications -- yet have different registration codes for the same name. The EncryptionKey must be a minimum of 5 characters and it is recommended to use a key containing 10 to 20 characters. **Note:** Registration Wizard Pro automatically generates a random encryption key upon control creation. If you would like to use your own key, simply change the property.

Example:

'Set the EncryptionKey to "AbCdEfGhIjKlMnOpQrStUvWxYz"
RegWizardPro1.EncryptionKey = "AbCdEfGhIjKlMnOpQrStUvWxYz"

- **EvaluationType Property**

The EvaluationType Property is used to set the type of evaluation period you wish to implement in your application. To set the amount of time for the evaluation period, see *ValidPeriod Property*.

TimesRun - Limit the number of times that the user can run your application before expiration.

DaysRemaining - Set the number of days that your application will run for before expiration.

- **FileName Property**

The FileName Property is used to set the path and filename of the file that will be used if evaluation information is stored also stored in a file. File storage can be enabled or disabled with the *FileStorage Property*. The file will be set to hidden and read-only.

RegWizardPro1.FileName = App.Path & "\Reg.Dat"

- **FileStorage Property**

The FileStorage Property is used to enable or disable the evaluation period information being stored in a file. The evaluation period information will be stored in the registry either way.

'Enable File Storage
RegWizardPro1.FileStorage = True

'Disable File Storage
RegWizardPro1.FileStorage = False

- **Language Property**

The Language Property is used to set which language will be displayed on any forms that Registration Wizard Pro displays to the user.

'Set the Language Property to display English Forms
RegWizardPro1.Language = English

'Set the Language Property to display Spanish Forms
RegWizardPro1.Language = Spanish

- **Mask Property**

The Mask Property is used to format the generated registration code to a specific format. **Note:** Any settings to the CharType Property, CodeHyphenated Property and MaxLength Property will be ignored if you are using a mask.

Mask Character	Description
#	Digit placeholder - possible replacements are 0 to 9
&	Character placeholder - possible replacements are "A" to "Z" and "a" to "z"

^ Forced Uppercase Character placeholder - possible replacements are "A" to "Z"

* Forced Lowercase Character placeholder - possible replacements are "a" to "z"

Any Other Character Any other character will be a constant and will remain the same in the generated registration code.

Example:

```
'Set the Mask Property to "RWP-#####-&&&&&-^^**"
RegWizardPro1.Mask = "RWP-#####-&&&&&-^^**"
```

The foregoing mask would generated a code resembling: RWP-23948-GmqYx-CExk

- **RegInfo Property**

The RegInfo Property is used to enabled/disable the "Registration Information" button on the English registration dialog forms. If this property is set to True, the button will be displayed on the form. When a user clicks this button, the registration dialog form will be unloaded and the ShowRegInfo Event is fired. After this, the registration dialog is displayed again.

For more information on the ShowRegInfo Event, please scroll down to the Events section.

- **SystemDependent Property**

The SystemDependent Property is used to enable or disable the use of system dependent registration codes. System Dependent registration codes allow your application to calculate information related to the user's system into the registration code; therefore, limiting the number of systems that the registration code will work on. **Note:** If a user upgrades their system and your applications is using system dependent codes, the registration code will no longer work and the user will need to contact you so you can issue them a new registration code. New to version 5.00 - please see the AutoReRegister Property.

Examples:

```
'Enable System Dependent Registration Codes
RegWizardPro1.SystemDependent = True

'Disable System Dependent Registration Codes
RegWizardPro1.SystemDependent = False
```

- **TitleBars Property**

The TitleBars Property is used to enable or disable the titlebars on any forms that Registration Wizard Pro displays to the users.

'Enabled TitleBars
RegWizardPro1.TitleBars = True

'Disable TitleBars
RegWizardPro1.TitleBars = False

- **UserName Property**

The UserName Property is used to set the name for which the registration code will be generated.

Example:

'Set the UserName Property to "Russell Anderson"
RegWizardPro1.UserName = "Russell Anderson"

- **ValidPeriod Property**

The ValidPeriod Property is used to set the length of time that your evaluation period is valid for. If the *EvaluationType Property* is set to *TimesRun*, this property's value will represent the number of times you are allowing the user to run your application before it expires. If the *EvaluationType Property* is set to *DaysRemaining*, this property's value will represent the number of days your application will run after the first day it was run.

Example:

'Set the ValidPeriod Property to 30
RegWizardPro1.ValidPeriod = 30

Registration Wizard Pro Methods

- **DeleteUser Method**

The DeleteUser Method is used to delete a user from the system registry that was stored with the *RegisterUser Method*. If the user is successfully deleted, the *UserDeleted Event* is fired.

Example:

'Delete user Russell Anderson from system registry
RegWizardPro1.DeleteUser "Russell Anderson"

- **GetUserName Method**

The GetUserName Method is used to retrieve the name(s) of the registered users that is/are stored in the system registry.

Examples:

```
'Retrieve the first registered user's name
MsgBox RegWizardPro1.GetUserName(1)

'Retrieve the last registered user's name
MsgBox RegWizardPro1.GetUserName(RegWizardPro1.TotalUsers)
```

- **ProgramRun Method**

The ProgramRun Method is used to update the evaluation period information. This method should be run **ONCE** at the beginning of your application after you have set the *EvaluationType Property* and *ValidPeriod Property*, as well as any options you want to set for formatting your registration codes (*i.e. CharType, Mask, etc.*).

Example:

```
'Invoke the ProgramRun Method

'RegWizardPro1.EncryptionKey = "AbCdEfGhIjKlMnOpQrStUvWxYz"
'RegWizardPro1.EvaluationType = TimesRun
'RegWizardPro1.ValidPeriod = 30
'RegWizardPro1.Mask = "RWP-#####-&&&&&-^^**"
RegWizardPro1.ProgramRun
```

- **RegCode Method**

The RegCode Method is used to obtain the registration code for a user's name. The optional Override parameter is used to generate a registration code for a specific system id number if you are using system dependent registration codes. This parameter should only be used in your registration code generater and **never** in your application.

Examples:

```
'Get registration code for user's name
MsgBox "The Registration Code Is:  " & RegWizardPro1.RegCode
```

If your application uses system dependent codes (*i.e.* the *SystemDependent Property* is set to True), the user must provide you with their System ID Number. In this example, we will assume the user told you that their System ID Number is 2934853193.

```
'Get system dependent registration code for user's name
MsgBox "The RegistrationCode Is: " &
RegWizardPro1.RegCode("2934853193")
```

- **Registered Method**

The Registered Method returns True/False based upon whether there are any registered users stored in the system registry.

Example:

```
'If/Then Statement using registered status
If RegWizardPro1.Registered = True Than
    ...
Else
    ...
End If
```

- **RegisterUser Method**

The RegisterUser Method is used to store users in the system registry. Both the user's name and registration code parameters must be passed to this method. The RegisterUser Method then tests the registration code and, if valid, stores the user's name in the system registry. If the User's Name is successfully stored in the system registry, the *RegisterUserPassed Event* is fired; If the User's Name was not successfully stored in the system registry, the *RegisterUserFailed Event* is fired.

Hint: If you want to have a place on your own forms for the user to enter their registration name and code, use two text boxes and the RegisterUser Method (*i.e. RegWizardPro1.RegisterUser Text1.Text, Text2.Text*).

Example:

```
'Store user name and registration code in the system registry
RegWizardPro1.RegisterUser "Russell Anderson", "RWP-19384-
JfnVp-DYwq"
```

- **ResetLimit Method**

The ResetLimit Method allows you to reset the amount of time remaining in the system registry. Most applications will not use this method; however, I provided this method as there may be a need to reset the remaining time.

Example:

'Reset the amount of time remaining in the evaluation period to 30
RegWizardPro1.ResetLimit 30

- **ShowDelay Method**

The ShowDelay Method is used to display the unregistered delay dialog box. The integer value represents the number of seconds for the delay.

Example:

'Implement 15 second unregistered delay if application is unregistered
If RegWizardPro1.Registered = False Then
RegWizardPro1.ShowDelay 15

- **ShowRegDialog Method**

The ShowRegDialog Method is used to display the registration dialog to the user. This dialog prompts the user to enter their registration name and code. It then tests the registration code and, if valid, stores the user in the system registry.

Example:

'Display the ShowRegDialog prompt to the user
RegWizardPro1.ShowRegDialog

- **SystemID Method**

The SystemID Method returns the unique system id number of the user's system. This system id number is needed if your application is using system dependent registration codes.

Example:

'Display System ID Number
MsgBox "You System ID # is " & RegWizardPro1.SystemID

- **TimesAppRan Method**

The TimesAppRan Method returns the number of times that your application has been run.

- **TotalUsers Method**

The TotalUsers Method returns the total number of registered users that are stored in the system registry.

Example:

```
'Return total number of users
MsgBox "Total Number of Users:    " &
Str(RegWizardPro1.TotalUsers)
```

Registration Wizard Pro Events

- **EvaluationExpired Event**

The EvaluationExpired Event is fired when you application runs the *ProgramRun Method* and Registration Wizard Pro detects that your application is unregistered and the evaluation period has expired. This event will also be fired if a user tried to "edit" the system registry and get past the evaluation period.

This event should contain code that displays to the user the registration dialog box so they can enter a registration code and should then terminate your application.

- **RegisterUserFailed Event**

The RegisterUserFailed Event is fired when the *RegisterUser Method* is used to store a user in the system registry but the registration code is incorrect for the provided name.

- **RegisterUserPassed Event**

The RegisterUserPassed Event is fired when the *RegisterUser Method* is used to store a user in the system registry and storage is successful.

- **ShowRegInfo Event**

This event will be fired when a user clicks on the "Registration Information" button on the registration dialog. This event should contain code to load a form of your choice and display modal. It is mandetory that you display your form modal so that the registration dialog will not be displayed until your form in hidden or unloaded.

- **SystemIDChanged Event**

This event will be fired anytime your application run the *ProgramRun Method* and Registration Wizard Pro detects that the SystemID number has changed. This event will only be fired if the *SystemDependent Property* is set to True. If this event is fired, it means that the user has in some way upgraded their system and will need to contact you for a new registration code.

- **UserDeleted Event**

The UserDeleted Event is fired anytime a user is successfully deleted from the system registry by using the *DeleteUser Method*.

- **UserReRegistered Event**

The UserReRegistered Event is fired anytime Registration Wizard Pro re-registers a user with a new registration code because of a change to their SystemID. This event can be used to inform the user of the change and also to tell them their new registration code for future references. This event has passed into it the User's Name, the old registration code as well and the new registration code for you to use as you desire.

Basic Implementation Theory

- **Generating Registration Codes**

To generate registration codes for your application, you need to begin by setting all the properties that you set in your application exactly as they are in your application. This includes the *DateSensitive Property, EncryptionKey Property, SystemDependant Property and the Mask Property*.

RegWizardPro1.DateSensitive = (Same as in application)
RegWizardPro1.EncryptionKey = (Same as in application)
RegWizardPro1.Mask = (Same as in application)
RegWizardPro1.SystemDependant = (Same as in application)

Once these properties have been set to the same values as they are in your application you need to set the User Name for which the registration code is to be generated by setting the *UserName Property*.

RegWizardPro1.UserName = "Russell Anderson"

After the username is set, you can obtain the registration code by using the *RegCode Method*:

MsgBox "The Registration Code Is: " & RegWizardPro1.RegCode

If your application uses system dependent registration codes (*i.e.* the SystemDependent Property must be set to True in your application), you can obtain the user's registration code through the *RegCode Method*; however, you will use the optional parameter to overide the SystemID Number. This overide should only be used in your registration code generater program and **never** in your application.

Assume the user tells you that their system id number is: 3921938604

You would obtain their registration code through the following code:

MsgBox "The Registration Code Is: " & RegWizardPro1.RegCode("3921938604")

- **Implementing a shareware evaluation period**

Here are the basic steps to implement a shareware evaluation period into your application. This code should be placed at the beginning of your applications source code so it is ran first.

First, set the control to use the encryption key of your choice should you not want to use the random key:

RegWizardPro1.EncryptionKey = "YourKeyHere"

Next, determine the type of evaluation period you want to implement (for more information, see the *EvaluationType Property*).

RegWizardPro1.EvaluationType = TimesRun
or
RegWizardPro1.EvaluationType = DaysRemaining

After setting the type of evaluation period you want to use, set the amount of time that the evaluation period is valid for. In this example we will set it to 30.

RegWizardPro1.ValidPeriod = 30

You can now run the *ProgramRun Method*. This will check the status of the evalatuion period and update it.

RegWizardPro1.ProgramRun

You have now implemented the registration period. You should now add code to deal with the status of the evaluation period.

We will now add code to the *EvaluationExpired Event* which will run in the event the expiration has expired. This event should contain code to display the registration dialog to the user and then terminate the application.

```
Private Sub RegWizardPro1_EvaluationExpired()
        RegWizardPro1.ShowRegDialog
        End
End Sub
```

That's about it to implementing an evaluation period into your application.

You can also implement code to do things based upon the registered status of your applications. Here's a quick sample using an If/Then Statement:

If RegWizardPro1.Registered = True Then

```
        Load MainMenu
        ...
Else
        Load UnRegisteredNotice
        ...
End If
```

or

```
If RegWizardPro1.Registered = False Then RegWizardPro1.ShowDelay 15
```

If you still have questions, please feel free to email Russell Anderson at randersn@pair.com

Thanks!