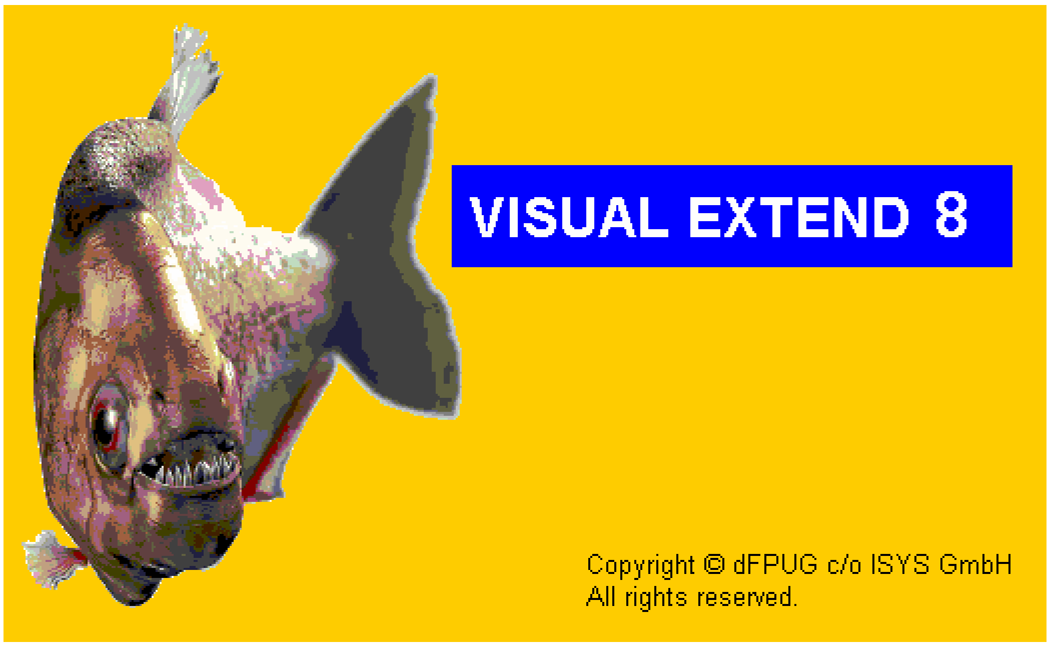


VFX 8 – New Features

August - September 2003

Table of contents

New properties of goProgram class

New properties used to manage application’s behavior when an error is raised, using of product activation features as well as the name of the printer driver that will be installed if the application needs to use PS Printer.

nAppOnErrorBehavior – this property determines what will happen when an application error occurs.
0 - always ignore (recommended not to be used)
1- show error message (default value)
2- cancel application execution

ErrorDetailLevel – this property determines what information will be logged on error.
0 - Error message only - no call stack information
1 - Write call stack information only (default value)
2 - Full detailed information

PSPrinterToInstall – the property holds the name of the standard PS Printer driver which will be installed at the time when the application needs a PS Printer driver to create a PDF file (default value "HP DeskJet 1200C/PS").

cConnectionCheckURL – the property holds the URL which will be used to check whether an Internet connection exists. This property is used when GhostScript is not installed on the computer, but is it needed to prepare PDF files. In this case GhostScript installation file is downloaded and installation is automatically started. If an internet connection does not exist a dial-up connection is established, asking for phone number to be dialed, user name and password to access the network resources.

lUseActivation – this property determines whether product activation using the cActivation class is enabled. You can set this property in the VFX Application Wizard when you create a new application and later in vfxmain.prg.

lActivationType – this property determines whether the cActivation class will check for the “First Install” file when the application starts for the first time. You can set this property in the VFX Application Wizard when you create a new application and later in vfxmain.prg.

New VFX 8 Developer Tools

New power builders:
Builders for cTreeViewForm, cTreeViewOneToMany and cPickAlternate classes.
Product activation wizards:
Define Activation Rules – defines system specific values to be used for product activation
Create Activation Key – creates activation key using user specific installation key.
(See application protection using activation key)
SQL Server database structure Updating.
Create meta data definition

cTreeViewForm Class

Main purpose of this class is to represent the data contained in a particular table (Master Table) in a tree structure. The tree-view structure gives the end-user complete overview of the hierarchical data relations contained in the particular table.
The class is based on cTreeView (vfxappl.vcx) and cDataFormPage (vfxform.vcx). It combines the functionality of cDataFormPage and the advanced data presentation in a hierarchical tree structure.

When a particular node in the TreeView is clicked, the record pointer is moved to the corresponding data row in the main table. The user can view and modify the data on the right part in the form.

VFX provides you with a builder that you can use to rapidly create and tune a form based on cTreeViewForm class.

The builder works similar to the builder of the cDataFormPage class. You should make your setting in the EditPage and FormOptions tab in the same way, which you already know for cDataFormPage based forms. In addition to this you should make your setting for the cTreeView class in the page TreeViewOptions. There are two types of settings for the TreeView control:

Data binding settings

IDFieldName - here you should fill the name of the primary key field in your main table

ParentIDFieldName - this is the name of the field that holds information about the parent id for each data record. In other words this is the key value of the parent record.

NodeText - here you can either fill the name of the field that holds a description text or an expression, which will be evaluated and the result will be used as NodeText in the tree structure. If you use a concrete field name, you can allow your end-user to edit the description text directly into the TreeView control. This also depends of the Allow Node Rename property. If Allow Node Rename is set, the user can edit labels in the TreeView and this will automatically update the corresponding data field.

AllowNodeRename - This property defines if user can edit description text in the TreeViewControl. Editing the description in the TreeView when this description is content of a single data field, it will automatically be updated.
Remark: Allowed only if Node Text points to a single data field.

TreeView Control Settings:

These settings are analogous to the TreeView ActiveX control
Style : 0 - tvwStyleText
1 - tvwStylePictureText
2 - tvwStylePlusMinusText
3 - tvwStylePlusMinusPictureText
4 - tvwStyleLinesText
5 - tvwStyleLinesPictureText
6 - tvwStyleLinesPlusMinusText
7 – tvwStyleLinesPlusMinusPictureText

Appearance : 0 - ccFlat
1 - cc3D

Border Style: 0 - ccNone
1 – ccFixedSingle

Indentation – This property sets the width of the indentation of nodes in the TreeView control

cTreeViewOneToMany Class

Main purpose of this class is to represent the data contained in a particular table (main table) in a tree structure along with the powerful functionality that cOneToMany class gives the developers. The tree-view structure gives the end-user complete overview of the hierarchical data relations contained in the particular table.
The class is based on cTreeView (vfxappl.vcx) and cOneToMany (vfxform.vcx). It combines the functionality of cOneToMany and the advanced data presentation in a hierarchical tree structure. When a particular node in the TreeView is clicked, the record pointer is moved to the corresponding data row in the main table. On the right part in the form the user can view and modify the corresponding data in the main table and related child tables.

VFX provides you with a builder that you can use to rapidly create and tune a form based on the cTreeViewForm class.

The builder works similar to the builder of the cOneToMany class. You should make your settings in the EditPage, FormOptions and ChildGrid tabs in the same way, which you already know for cOneToMany based forms. In addition to this you should make your settings for the cTreeView class in the page TreeViewOptions. There are two types of settings for the TreeView control:

Data binding settings

IDFieldName - here you should fill the name of the primary key field of your main table

ParentIDFieldName - this is the name of the field that holds information about the parent id for each data record. In other words this is the key value of the parent record.

NodeText - here you can either fill the name of the field that holds a description text or an expression, which will be evaluated and the result will be used as NodeText in the tree structure. If you use a concrete field name, you can allow your end-user to edit the description text directly into the TreeView control. This also depends of the Allow Node Rename property. If Allow Node Rename is set, the user can edit labels in the TreeView and this will automatically update the corresponding data field.

AllowNodeRename - This property defines if the user can edit the description text in the TreeView control. Editing the description in the TreeView when this description is content of a single data field, it will automatically be updated.
Remark: Allowed only if Node Text points to a single data field.

TreeView Control Settings:

These settings are analogous to the TreeView ActiveX control
Style : 0 - tvwStyleText
1 - tvwStylePictureText
2 - tvwStylePlusMinusText
3 - tvwStylePlusMinusPictureText
4 - tvwStyleLinesText
5 - tvwStyleLinesPictureText
6 - tvwStyleLinesPlusMinusText
7 – tvwStyleLinesPlusMinusPictureText

Appearance : 0 - ccFlat

1 - cc3D

Border Style: 0 - ccNone
1 – ccFixedSingle

Indentation – This property sets the width of the indentation of nodes in the TreeView control

cPickAlternate Class

Similar to the cPickField control, the cPickAlternate class can be used for validating user input against a table as well as to provide the call of a pick list dialog where the user can pick a record. It allows the end-user to enter and choose two fields from the child table (PickList table) and to store quite different values in the parent table (main table).

Use the cPickAlternate control rather than the Ccombobox Control, whenever you have a table with a lot of records and if you want to validate user keyboard input directly as well as when you need to give the user the ability to enter a value, that is not the relation key between both tables. The purpose of the class is to provide the end-user with an easy-to-use interface allowing well known values ("logical" keys) to be entered instead of program generated id keys. The user fills this logical key value and it is used to navigate to the correspondent record in the PickList table. When the searched record is found the Return Field value is passed back and is used to update the related data in the main table.

This class is based on cPickField and inherits all of its properties and methods. In addition to them it has a new property cControlSourceInternalKey where you have to specify the name of the field in the main table which is a foreign key to the PickList table.

You can use the cPickAlternate builder to easily set the cPickAlternate class properties:

Pick Table Name - here you should specify the name of the PickList table.

Pick Table Index Tag – this is the name of the index which will be used when searching in the PickList table. Do not forget that this index should be on the logical key field.

CpickAlternate.txtField.ControlSource – The name of the "logical" key field in PickList Table where value entered by user will be searched for validation.

CpickAlternate.txtDesc.ControlSource – The name of the description field. Its value will be displayed in the description text box by the cPickAlternate class when validation is successful.

Return Field Name (Code) Use STR() for Num. Fields. – The name of the "logical" key field in the PickList Table, returned from cPickDialog form or expression (It must always result to character data type). This field is then filled into textfield.

Return Field Name (Description) – The name of the description field returned from the cPickDialog form or expression (It must always result to character data type). This field is then shown in description field.
Return Field Name (Internal Key) – the name of the field in the PickList table which value uniquely identifies the chosen record. Usually, this field is used to make relation from MainTable to PickList table.

Control Source Internal Key – the name of the field in the main table, where the cPickAlternate class will store the key value. Usually this is the foreign key to the PickList table.

Application Protection using Activation Key

Main purpose of using activation keys is to prevent unauthorized use of your application on unregistered computers just by copying it.

Developers can choose whether to protect the application, by checking ´Enable product activation´ check box on the third page "Options" in the VFX Application Wizard or later by setting goProgramm.lUseActivation property to .T.

When goProgramm.lUseActivation is .F. no protection actions are performed.

When protection using Activation key is enabled, on startup of the application the goProgram.SecurityRights object is created and it has child properties named according rights, that the developer has defined. Every property could have 3 different values:

(-1) – application is not registered – in this case the developer can choose what action to perform, so the user can have only limited access while the application is still not registered

(0) - application is registered, but the user is not allowed to perform this action

(1) - application is registered and user has right to perform that action

For further information please see *Defining rights that will be used for restricting user access to particular modules* topic below.

When protection using Activation key is enabled, the activation key along with data of the first installation is stored in an INI file. The developer can choose the name for this INI file (default value is "VFX.INI"), so that every application uses its own INI file. The Activation key is encrypted using a developer-defined combination of different hardware parameters along with character constants, specific registry values and creation datetime of a specific file. This combination can be different for every particular application, so that every application can have its own unique activation rules. The process of creating Security key pattern will be described later in this document.

In addition to this setting, the developer can also choose the type of protection:

Standard protection creates the INI file at the time when the application is started for the first time. At this moment the system date is stored in the INI file. This date is then available for usage as the value of goProgram.InstallationDate property and the developer can restrict application execution according to it.

The weakness of this protection is that the end-user can delete the created INI file and then the INI file will be created again and will contain a subsequent date. To avoid such users action, the developer can choose to use an additional protection level, using a special file, deployed along with installation files. This file is called "First Install file". Its name can also be set using cFirstInstal ptoperty of cActivation class (appl.vcx).

When the developer chooses "Use First Install file" activation type, the application will run in the following way:

When the application is started it checks for the INI file. If the file exists, the application uses it to set goProgram.InstallationDate and all the rest of the user rights according to the activation key.

If the file does not exist, then it is assumed that this is the first run of a newly installed application. To verify this fact, the cActivation class makes an additional check if the “First Install file” exists.

If the file exists, it is considered that the application is really started for the first time. Then the installation date is stored into the INI file and the “First Install file” is deleted. If an unfair user tries to re-activate the application by INI file deletion. cActivation key will stop application execution, if “First Install file” does not exists. This protection level ensures a higher security level, but you must not forget to include the “First Install file” in your installation files and to set it to be placed into the Windows folder.

When your user wants to activate the installed application he must send you his installation key. The installation key can be sent in three different ways. This depends on the value of the nRegWay property:

(0) – Installation key is displayed in a message box and the user can copy it and paste wherever needed.

(1) - Installation key is stored into a file and the user can send it to the developer later. The name of the file must be filled in cParamFile property.

(2) - Installation key is stored into a file and sent as an e-mail attachment to the developer. The name of the file must be filled in the cParamFile property and the recipient´s e-mail address (the address where the e-mail will be sent) must be filled in the cRegEMail property.

The installation key can be of different length. It depends of your activation key pattern (see later - *Defining Security key patter*). The end-user should send you the installation key by e-mail or could enter it in your registration web page. This installation key is used by the ***Define user rights*** wizard to create an activation key for that user. When the activation key is prepared, it is sent to the user and he can activate his application by entering the key in the registration window.
Below you will read how you can define activation rules for your application and how to generate activation keys for your end-users, having their installation key.

List of used terms

System specific value - System specific value is called a particular serial number for some hardware device or creation datetime value for particular file (chosen by the developer) or the value of a particular registry key (chosen again by the developer).
Activation key - This is a character string that holds a concrete right that you have assigned to a particular user. It depends on the user’s system specific values that you specified.
Installation key - This is the character string that contains user’s hardware details. It is used to prepare the Activation key for the particular user. This Activation key is useless for any other computer but the user’s.
Security key pattern- When you define system specific values that will be used to uniquely identify the user’s machine, you can choose any system specific value in various order and also expressions containing text proceeding functions. (The result is always of character type.) This concrete combination will be unique for your application only and is called Security key pattern. You can define separate patterns for each of your applications or use only one for all of them.

Installation date - This is the date, when the application was installed or when the application is started for the first time. Which date is used depends on your settings.

cVFXActivation class properties

cFirstInstall - Name of a file which will be checked to determine whether this application starts for the first time. If it is blank, no First install file will be used. In this case, when application starts and no installation date is stored in the INI file, the date is filled with no additional checks.

cINIFileName - Name of the INI file which will be used to store the application installation date and activation information. Default value is "VFX.INI".

cParamFile - Name of a file which will be created to hold the installation key. Depending of nRegWay value, this file can be sent by e-mail or processed in other ways.

cRegMail - E-mail address of the developer, where the installation key file will be sent, if nRegWay property is set to 2.

cRegFileName - You can point here one of your files, that your installation will create. Its creation date will be used to determine installation datetime. If this property is left blank system datetime is used on first start of the application.

nRegWay - This property specifies how installation key will be sent back to the developer:

(0) – Installation key is displayed in a message box and the user can copy it and paste wherever needed.

(1) - Installation key is stored into a file and the user can send it to the developer later. The name of the file must be filled in cParamFile property.

(2) - Installation key is stored into a file and sent as an e-mail attachment to the developer. The name of the file must be filled in the cParamFile property and the recipient´s e-mail address (the address where the e-mail will be sent) must be filled in the cRegEMail property.

How it works

Main activation data is stored into the INI file which name you have to specify in the cINIFileName property of the cVFXAcvtivation class (default value – VFX.INI).

You can choose whether your application will use the activation feature at all and you can also choose whether a special file called "First Install File" will be used. The name of the file is up to you and it is hold in cFirstInstall property of the class.

If you choose to use the First Install File, you have to deploy this file along with your applications' files. The installation must store it in the Windows folder and the cActivation object will erase the file when the application is started for the first time. At this time the installation date is stored into the INI file. Later on every application start the INI file will be checked and if the stored installation date is missing, and if First Install file is also missing, the object assumes that the installation is not valid and stops application execution.

If you choose not to use the First Install File, the application will create the INI every time, when it is missing.

The installation date can be specified in two ways: System date-time can be used or creation date-time of a particular file. If you want to use the creation date-time of a file its name should be stored in the cRegFileName property of the cActivation class.

Defining activation Rules

Defining Security key pattern - When you start the Define Activation Rules builder for the first time for a particular project, you are asked whether you want to create a new pattern for this project.

You can choose Cancel to return back in VFP if you do not want to enter Define Activation Rules builder right now.

If you answer Yes, a small window appears where you should enter a name for the new pattern and when you write your name for the new pattern the Activation Security Key and Rights window loads.

If you answer No, this will mean that you intend to use an existing pattern that you already have defined for another project.

In the Security key page of the Activation Security Key and Rights window you can see a DropDown combo that you can use to change the pattern for the current active project and a grid, where you can add as many rows as you need. Every row defines a part of the Installation key, which will be used later to make sure that your application is used on the right computer and only rights, that you had assigned to this user are in effect.

In the first column of the grid you can choose the System specific value which you want to be used. In the Drop-Down combo are listed all possible hardware parameters that you can use to build your installation key. In addition to this you can define additional string operations for this value.

For example you may decide that you do not need to use the full HDD factory serial number and you want to use only the last 4 digits to generate your activation key. You will choose the row “HDD Factory Serial Number” in the ComboBox in the first column. Then the system variable that corresponds to this parameter - “HDDFactoryNumber” - automatically will be filled into the second column. To use only the last 4 digits, you have to write the following expression in this column: RIGHT(ALLTRIM(HDDFactoryNumber), 4).

If you want to use one of these system specific values: File Creation Date or Registry Key Value, you also have to set additional parameters. For File Creation Date you have to write the name of the file which creation date will be read and for Registry Key Value you have to write the registry key that should be accessed to obtain the desired value.

Every hardware parameter that you choose (or expression of it) is concatenated to build one common installation key. This key is used when end-user registers his application. If just one of these parameters changes the application registration becomes invalid and the end-user will need to obtain a new activation key, corresponding to the new hardware details.

You can add as many rows as you wish. You can also reorder rows using the buttons in the right part of the screen.

Be careful - as many rows you are adding for the Security key pattern, the longer will be the resulting activation key string!

When you save defined activation rules this pattern is saved in the property cActPattern of cVFXActivation class (in Appl.vcx) for later use when the application runs – DO NOT DELETE IT!

Defining rights that will be used for restricting user access to particular modules - You can define up to 32 different user rights. They will depend on your application structure and your needs. For example you can define rights as: RunDataForms, RunReports, EditData, ViewData etc. Later in your application you can check any of them to decide whether a particular form can be run or a particular report can be viewed.

All of these values are available as properties of the global object goProgram.SecurityRights, so they can be checked at any time as needed.

While the application is running in unregistered mode their values are -1 (all of them). Later when the user receives the activation key from the developer and activates his application for every particular right the value will be: 1 – for allowed and 0 – for not allowed. Using the upper example if you give to the particular user only rights to manipulate data, but not to access reports you will have these values:

```
goProgram.SecurityRights.RunDataForms = 1
goProgram.SecurityRights.RunReports = 0
goProgram.SecurityRights.EditData = 1
goProgram.SecurityRights.ViewData = 1
```

How to define these rights for a particular end-user will be described later in Defining User Rights section.

To enable a particular right you first need to check the checkbox in the first column in the grid, that shows that you will use this right and then to fill its name into the third column. Be careful – the name of the right should conform to rules for variables naming, because this name is used to create every property of the object goProgram.SecurityRights!

Remark: Application rights are specific for every different application. You cannot use rights that are already defined for another application. If you need similar rights, you will have to define them again. This is because these rights are hold in the vfxapprights.dbf table, that is application specific. This table should be defined for every application project.

Defining user rights for every particular end-user

Now, when your end-user sends you his installation key, you are about to make an activation key. This Activation key is responsible to tell your application whether this user is allowed to execute a particular action (when the corresponding right is 1) or not.

When you run Create activation key from your VFX 8.0 menu, Rights, defined for the active project are loaded in the form.

You have to load user’s hardware details (*System specific values)* using the button Read User Hardware details. In the form that opens you can either paste or load from a file the user’s installation key.

When you define the value for every right, you are ready to generate the activation key. The generated key is stored in the file <ProjectName>.XAK in the home folder of the active project. This Activation key should be sent to the user for activation.

When the end-user starts an application that requires registration (and when the application is still not registered), the installation key is automatically generated. Depending of the nRegWay property, the generated installation key is either shown in a form or a file is created and sent via e-mail. If the user has received his activation key, he can enter it in this window and thus register the application.

Later the user can access this windows using Tool/Register… menu. In this case the generated installation key is only shown in the form regardless of the nRegWay property value.

SQL Server Database Structure Uupdate

This wizard helps you to prepare meta-data definition of your currently used database structure and to populate it over the user’s database.

Using the menu “Create meta data definition” you can create a Datadict.dbf. This is a free table, where the structure of a database along with constraints, user defined data types, rules, views and stored procedures is stored. The wizard scans existing connections in the active project and retrieves data structures for every one of them. Later, when the table is deployed to end-users the data structure update will use it and again scanning existing connections will update user´s database structures.

cEmail Class

This class gives the developer the ability to send e-mails just by passing a few parameters to its Send_Email_Report method. In addition to this you can attach additional reports in PDF format to your e-mails.

Properties:

LastErrorNo – this property contains the number of the last error (if some were encountered). You can use it to check what is the reason for raised error.

LastErrorTest – if an error occurs, the text of the error is stored in this property.

oEmail_Attachment – internally used to hold a collection of added attachments.

Methods:

AddAttachment (tsAlias, tcFileName, tcReport, tcFor)
Adds an information about attachment that will be created and sent with next sent e-mail. The information for all PDF attachments, that have to be prepared is stored in the oEmail_Attachment property. If you pass data alias and report name the class automatically creates PDF file using this report and alias. You can also specify additional expressions that will be used to filter out reported records. If you do not specify data alias and no table is open

in the current work area, the class considers that you want to attach a file that you have prepared in advance. In this case the file must exist at the time when Send_Email_Report method is called.

Parameters

tcAlias – used only for PDF creation, data alias that will be exported to PDF

tcRezFile – name of the attached file (if PDF will be prepared – name of the PDF file that will be created)

tcFRXName – used only for PDF creation, report that will be used to prepare PDF file

tcFor – used only for PDF creation, expression to filter exported to PDF records

Send_Email_Report (tcEmail, tcSubject, tcText)

tcEmail – e-mail recipient address

tcSubject – subject of the e-mail

tcText – body text for the e-mail

AddAttachment (tsAlias, tcFileName, tcReport, tcFor)

Clears all currently assigned attachments.

You just need to call the AddAttachment method as many times as you need, passing the alias of your data tables (or cursors), name of the result file, formatting report name and expression to be used as FOR clause. Then call the Send_Email_Reports method. All PDF files will be created and attached to the created e-mail. Also files, which you have previously created and passed to the AddAttachment method without report name and alias name parameters will be attached.

cCreatePDF Class

This class is designated to prepare reports in PDF format. It receives as parameters the data alias, name of the resultant PDF file, name of FRX to be used for creating the report and an expression that will be used as FOR clause to filter out reported records.

To create a file in PDF format you need to have GhostScript and a PS Printer driver installed on the computer. The class is checking for GhostScript and if it is not installed, it can download and install GhostScript automatically. The cDownload class is used to perform download. The content of VFXSys.Install_GS memo fields describes necessary actions that will be executed. See cDownload for more details.

If there is no PS printer installed on the computer the class installs the standard Windows printer driver according to the value stored in the goProgram.PSPrinterToInstall property.

Properties:
LastErrorNo – this property contains the number of the last error (if some were encountered). You can use it to check what is the reason for raised error.

LastErrorTest – if an error occurs, the text of the error is stored in this property.

Methods:

AddAttachment (tsAlias, tcFileName, tcReport, tcFor)
Adds an information about attachment that will be created and sent with next sent e-mail.

Create_PDF(tcAlias, tcRezFile, tcFRXName, tcFor)

tcAlias – data alias to be exported in PDF file

tcRezFile – full path and name of the resulting PDF file

tcFRXName – report which will be used for formatting data to PDF file

tcFor – conditional expression to filter out exported data

Method returns .T. if the file is successfully created and .F. otherwise. The number and description text of encountered error are in LastErrorNo an LastErrorText properties of the object.

cDownload Class

This class allows you to download files from the Internet and if necessary, to run them waiting for special events and sending keystrokes to running application to drive its behavior.

The class can be used for many purposes through passing different execution macros to ExecMacro method.

Macros are strings containing consecutive commands, using defined macro language. You can write your own macros similar to the macro stored in vfxsys.install_gs field.

The class uses the goProgram.cConnectionCheckURL property to check if there is an existing Internet connection and if the connection is not found, a Dial-Up connection is established. In this case the user must provide a phone number, a user name and a password for the new connection.

Properties:

LastErrorNo – this property contains the number of the last error (if some were encountered). You can use it to check what is the reason for raised error

LastErrorTest – if an error occurs, the text of the error is stored in this property

Methods:

ExecMacro (vcMacro, lnNoRun)

vcMacro – Macro-language script to be executed

lnNoRun – If .T. the downloaded file will not be run

Macro language commands:

"D:" URL

URL from where the file will be downloaded. This command automatically runs the downloaded file, if lnNoRun parameter is .F.

"C:" nTimeOut; lPartial; lTopLevelForm; lResultOnError; SearchedString

Waits while the window appears

nTimeOut - timeout in seconds - if expected form does not open within it, timeout error is generated

lPartial – determines whether the class looks for partial or entire string in form caption

lTopLevelForm - to check only top level forms. The string is searched only in caption of the form that is on top

lResultOnError – Returned value if form is not found within given timeout period. If form existence is significant for further execution, then when timeout occurs execution should be stopped. In this case the value of lResultOnError has to be .F. If the execution of the macro can continue regardless of the timeout, the passed value will be .T.

SearchedString –Srting that will be searched in form's caption

"W:" nTimeOut; lPartial; lTopLevelForm; lRezultByError; SearchedString

Waits until the window containing searched string is closed

nTimeOut - timeout in seconds - if expected form does not close within it, a timeout error is generated

lPartial – determines whether the class looks for partial or entire string in form caption

lTopLevelForm - to check only top level forms. The string is searched only in caption of the form that is on top

lResultOnError – Returned value if form is not closed within given timeout period. If form closing is significant for further execution, then when timeout occurs execution should be stopped. In this case the value of lResultOnError has to be .F. If the execution of the macro can continue regardless of the timeout, the passed value will be .T.

SearchedString – String that will be searched in form's caption

"X:"

Closes the top level window. You need to ensure in advance that it is the window that has to be closed using "C:" command

"K:" nKeyCode1; nKeyCode2; …

Puts listed characters into the keyboard buffer

"U:" URL

URL from where a file will be downloaded. The downloaded file will not be run, regardless of the lnNoRun parameter's value

Example:

How the Install_GS script works:
D: ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/AFPL/gs800/gs800w32.exe
Downloads the file gs800w32.exe and runs it.

C: 30; .F.; .F.; .F.; WinZip Self-Extractor - gs800w32.exe
Waits until window with caption "WinZip Self-Extractor - gs800w32.exe" appears.

K: 43
Sends "Enter" to the active window – this will start the files extraction.

C: 60; .F.; .F.; .F.; AFPL Ghostscript Setup
Waits until the window with the caption "AFPL Ghostscript Setup" appears.
K: 43
Sends "Enter" to the active window – this will start the GhostScript installation.

W: 240; .F.; .F.; .F.; AFPL Ghostscript Setup Log
Waits while the window with the caption "AFPL Ghostscript Setup Log" is open. This windows shows the installation process and the script must wait until this process finishes.

C: 30; .T.; .T.; .T.; Ghostscript
Waits until the window with the caption "Ghostscript" appears. This window shows the Setup succesful message.
X:
Closes the last window.

With this the GS installation is completed.

cArchive Class

This class is designated to archive and restore user's data. The archive's name is constructed using the name of the data folder and the current date and time. For example if your data folder is named "Data" and you are creating the archive on 10th of September, 2003 the name of the created archive will be Data20030910.ZIP.

Properties:

OverrideFile – determines what action will be performed if a file with same name already exists

0 – cancel operation if file exist

1 – adds new files and replaces existed in archive file (when creating archive)
do not overwrite existing file (when restoring from archive)

2 – overwrite existing zip archive (when creating archive)
overwrite existing file with the file from archive (when restoring from archive)

OperationSuccessfully – holds the result from the last operation

.T. – if operation completed successfully,
.F. - if operation did not completed successfully

Methods:

CreateArchive **(**lcFileLocation, lcMask, lcArchFilePathName)

Return value: .T. – if operation completed successfully, .F. - if operation completed not successfully

lcFileLocation – full path to the directory whose contents will be archiving

lcMask – file mask (example "*.DBF;*.FPT")

lcArchFilePathName – archive name with full path will be created

ZipProgress (*tcCurrentOperatedFile, nState, nAllFilesSize, nZIPedFilesSize, nArchiveCurrentSize)*
the method runs as callback from CreateZipArchive function (in VFX.FLL)

Parameters:
tcCurrentOperatedFile – Name of file being added to archive

nState **–** current operation
1 – file exist
2 – start adding file to archive
3 – successfully added file to archive
4 – cannot add file to archive
5 – archiving operation completed successfully
6 – archiving operation not completely successfully
7 – no files to archive

nAllFilesSize – size of all files which will be archived

nZIPedFilesSize – size of files added to archive so far

nArchiveCurrentSize – the size of archive so far

Return value:
0 – cancels archiving operation
1 – continue adding files to existing archive and overwrite existing files in archive
2 – overwrite existing archive file

EextractFromArchive(lcArchFileForExtract, lcPathForExtract)
Parameters:
lcArchFileForExtract – full Zip file path name to be extracted
lcPathForExtract – destination folder where to extract files

UnZipProgress (tcCurrentOperatedFile, nState, nArchiveFilesSize, nUnZIPedFilesSize)
the method runs as callback from ExtractZipArchive function (in VFX.FLL)
Parameters:
tcCurrentOperatedFile – name of currently extracted file from archive.

nState – current operation

1 – file already exist;
2 – start extracting file;
3 – end extracting file;
4 – cannot extracting file;
5 – extracting form archive completely successfully;
6 – extracting form archive completely not successfully.

nArchiveFilesSize – size of archive
nUnZIPedFilesSize – size of part of archive already extracting

Return value:

0 – cancel entire operation;
1 – do not extract current file;
2 – overwrite existing file.

VFX Menu Designer

VFX Menu Designer (VMD) is a tool for quick designing menus, giving you a visual image of how your menu will look like. It makes menu creation easier by handy user interface and quick property settings.

VMD can open existing Visual FoxPro menu files (.mnx) and to convert them in VMX format. You can create your menus multi-language by choosing correspondent check box in the create menu dialog or later - by using “Create menu to Multilanguage” button in the toolbar or the same bar in the menu.

To open previously created VMD structure select either “Open menu” button in the toolbar or *Menu ->Delete* from VMD main menu. You can also use the Shortcut CTRL+DEL In the dialog screen “Open file”, choose the desired file name from the list and press OK button. The menu is loaded in VMD Design Panel.

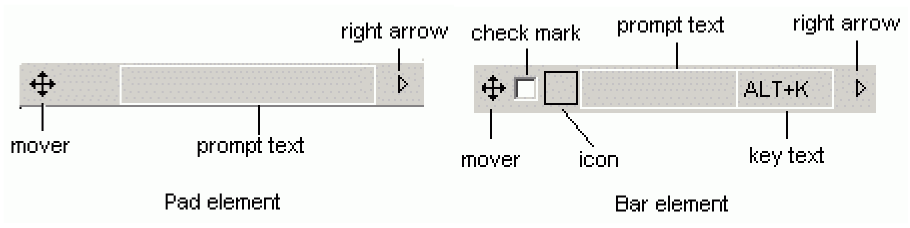
You could edit and modify the opened menu, by adding or removing bars, pads and popups and to change their settings.

To open a standard Visual FoxPro menu in the “Open menu” dialog window select *Files of type*: Menus (*.mnx) and then choose the desired .mnx file.

You can add new pads by clicking on the right arrow of the last pad in menu bar. You can add new bar in the popup by clicking on the arrow at the bottom of the popup (when you do not have any bars in the popup, you can see only this arrow under the pad element). Clicking the right arrow of a bar generates a submenu (sub-tree) popup for the selected bar.

After a new element is created, its properties can be set in the Properties Panel.

A bar or pad could be deleted when element has the focus. Use *Menu ->Delete* from VMD main menu or press CTRL+DEL or press "Delete element" button in the Toolbar.
Design panel allows you to set some of the elements’ properties visually:



- *Prompt text*
Prompt text could be set for each element directly in design panel when the element has the focus. In the settings panel "Prompt text" box is used only for information which is the active element in the design panel.

- *Setting key text*
The key text shows the user text indicating access key or key combination for the element. It should correspond to the access key or key combination and is displayed to help users choosing right keys.

- *Setting check mark*
If check mark is set for a particular bar, this bar will be checked when the menu is loaded. In addition to this using AutoMark property of this bar allows you to write specific code for execution when bar automatically changes its state to marked or unmarked. The code can be entered by clicking buttons "ActionOnMark" and "ActionOnUnmark. Standard Visual FoxPro editor opens and allows you specifying code for each action. If you do not want VMD to add code for automatically alternative changing of the elements state (marked/unmarked), simply set AutoMark property to OFF.

- *Setting picture (icon)*
Each bar in menu or shortcut menu can have an icon (or a picture). This picture could be selected from a Visual FoxPro integrated resources or another picture file. For selecting picture, there is a square with black bounds in each bar and clicking in it "Get picture from" dialog.
You have two options – to selecting icon from a picture file or selecting icon from VFP resources.
If at the same time an icon and a check mark are set for a bar, the icon itself behaves as a check mark. When bar is checked the icon becomes sunken, and when bar is unchecked – the icon becomes normal.

Position of every element in the structure can be changed with drag and drop operation. For this operation is used "4way" button ✥at the left of all pads and bars. Moving the elements is restricted in some cases: a pad can not become bar and a bar cannot become pad. In addition, a bar cannot be moved in its child popup.

In the Properties panel you can change all other properties of the elements: font, foreground and background colors, status bar messages, constant that will be used for multi-language menus, element behavior, event actions etc. When an element of the structure has focus, it is active and its properties could be changed. By default, newly created child elements receive properties of their parent.

You can set the code that is executed when the user clicks on pad or bar, by using "ActionOnSelect" button" and "SkipFor" button allows you to enter code for SKIP FOR clause.

When you check AutoMark checkbox, you can specify "Action On Mark" and "Action On Unmark" code, which will be executed every time, when the bar changes its state.

Along with this you can define the sequence for executing the code assigned to that bar. You may want the Action On Mark/Unmark code to be executed first and then the Action On Select code. Then “Before Action On Select” option has to be chosen. If you want first to execute the Action On Select code and after it Action On Mark/Unmark code to be executed, choose
“After Action On Select” option

For each element you can set Font properties.
With checkboxes Bold, Italic, Underline or Strikethrough element font can be modified. A string, representing checked properties is shown in the rightmost textbox (B - **Bold**, I - *Italic*, U - Underline and "-" - ~~Strikethrough~~).

Font properties can be changed by clicking “Font” button. Standard font dialog window appears. In this dialog, you can select font name and font size for the element. Selected values are shown in the middle two of the textboxes. All these textboxes only show selected values and do not allow changing values. Values can be changed only by “Font” button or by using Checkboxes. Standard font dialog screen also allows change of **Bold**, *Italic*, Underline and ~~Strikethrough~~ properties.

At any time you can preview your menu using Preview button in the toolbar or *Menu ->Preview* from VMD main menu.