

*The extensive Application  
Development Framework that  
makes Microsoft Visual FoxPro  
Development easy!*

# VISUAL EXTEND 9.5



*English User Manual*

dFPUG c/o ISYS GmbH

Uwe Habermann, Venelina Jordanova

## Copyright

Visual Extend is a product from dFPUG c/o ISYS GmbH. Any reuse of VFX related material needs the written permission of dFPUG c/o ISYS GmbH; also VFX related publications must have the copyright notice of dFPUG c/o ISYS GmbH.

## Note

**This user manual has identical content as the user manual for Visual Extend 9.0. All differences between the versions VFX 9.0 and VFX 9.5 as well as new features of VFX 9.5 are described in the document *VFX95WhatIsNew*.**

<b>1. INTRODUCTION</b> .....	<b>7</b>
1.1. BASED ON VISUAL FOXPRO 9.0.....	7
1.2. THE COMBINATION THAT MAKES IT: ALL IN ONE.....	7
1.3. EVEN MORE PRODUCTIVE WITH NEW BUILDERS IN VISUAL EXTEND 9.5 !.....	8
<b>2. QUICK OVERVIEW</b> .....	<b>10</b>
2.1. INTRODUCTION.....	10
2.1.1. <i>Installation</i> .....	10
2.1.2. <i>VFX Task Pane</i> .....	10
2.1.3. <i>VFX - Application Wizard</i> .....	11
2.2. FUNCTIONALITY OF THE NEW APPLICATION.....	12
2.2.1. <i>Usage</i> .....	12
2.2.2. <i>Standard toolbar</i> .....	12
2.2.3. <i>Open-Dialog</i> .....	12
2.2.4. <i>Forms</i> .....	13
2.2.5. <i>User's management</i> .....	13
2.2.6. <i>Error tracking</i> .....	14
2.2.7. <i>Database Tools</i> .....	14
2.2.8. <i>About Dialog</i> .....	14
2.3. CREATING A FORM WITH THE VFX – FORM WIZARD.....	15
2.4. VFX – DATA ENVIRONMENT BUILDER.....	15
2.5. THE VFX - FORM BUILDER.....	16
2.6. THE VFX - CGRID BUILDER.....	16
2.7. TEST.....	16
<b>3. INTRODUCTION</b> .....	<b>17</b>
3.1. OVERVIEW.....	17
3.2. SPECIFICS OF APPLICATIONS CREATED USING VISUAL EXTEND.....	17
3.3. KEY FEATURES FOR DEVELOPERS.....	18
<b>4. INCLUDED TOOLS</b> .....	<b>21</b>
4.1. VFX – CLASS LIBRARIES.....	21
4.2. VFX – WIZARDS AND BUILDERS.....	21
4.3. VFX DEVELOPER PRODUCTIVITY TOOLS.....	22
4.4. OTHER DEVELOPER TOOLS.....	22
4.5. VFX 9.5 TASK PANE.....	22
<b>5. INSTALLATION</b> .....	<b>24</b>
5.1. HARDWARE- AND SOFTWARE- REQUIREMENTS.....	24
5.2. THE VFX INSTALLATION.....	24
5.3. REGISTRATION AND ACTIVATION OF VFX 9.5.....	25
5.4. SETUP THE VISUAL FOXPRO ENVIRONMENT FOR VFX.....	25
<b>6. GENERATE A NEW APPLICATION USING THE VFX - APPLICATION WIZARD</b> .....	<b>27</b>
6.1. OBJECTIVE.....	27
6.2. PREPARATION.....	27
6.3. THE VFX - APPLICATION WIZARD.....	27
6.4. GENERATE THE PROJECT.....	31
<b>7. DISCUSSION OF THE GENERATED VFX - APPLICATION</b> .....	<b>32</b>
7.1. OFFICE-COMPATIBLE USER INTERFACE.....	32
7.1.1. <i>File Menu</i> .....	32
7.1.2. <i>Edit Menu</i> .....	33
7.1.3. <i>View Menu</i> .....	33
7.1.4. <i>Favorites Menu</i> .....	34
7.1.5. <i>Tools Menu</i> .....	34

7.1.6.	<i>Windows Menu</i> .....	34
7.1.7.	<i>Help Menu</i> .....	35
7.1.8.	<i>Standard Office-Like Toolbar</i> .....	35
7.1.9.	<i>Final words about Office - Compatibility</i> .....	36
7.2.	DATABASE TOOLS.....	37
7.3.	USER LIST.....	37
7.3.1.	<i>Currently logged users</i> .....	39
7.4.	USER GROUPS.....	39
7.5.	ERROR TRACKING.....	42
7.6.	ERROR HANDLING.....	42
7.7.	SYSTEM LOCKS.....	42
7.8.	OPTIONS.....	43
7.9.	ABOUT DIALOG.....	44
<b>8.</b>	<b>THE VFX BUILDERS</b> .....	<b>45</b>
8.1.	VFX - APPLICATION BUILDER.....	45
8.2.	VFX FORM WIZARD.....	54
8.3.	VFX - FORM BUILDER.....	54
8.4.	VFX - DATAENVIRONMENT BUILDER.....	54
8.5.	VFX - CDATAFORMPAGE BUILDER.....	56
8.5.1.	<i>Edit Pages</i> .....	57
8.5.2.	<i>Grid Page</i> .....	60
8.5.3.	<i>Form Options</i> .....	61
8.5.4.	<i>View Parameters</i> .....	63
8.5.5.	<i>Linked Tables</i> .....	64
8.5.6.	<i>Required Fields</i> .....	65
8.5.7.	<i>Report</i> .....	66
8.6.	VFX - CTABLEFORM BUILDER.....	68
8.7.	VFX - CONETOMANY BUILDER.....	69
8.8.	VFX - CONETOMANYPAGEFRAME BUILDER.....	74
8.9.	VFX - CTREEVIEWFORM BUILDER.....	75
8.9.1.	<i>Databinding of the TreeView control</i> .....	76
8.9.2.	<i>Layout settings of the TreeView Control</i> .....	77
8.10.	VFX - CTREEVIEWONETOMANY BUILDER.....	77
8.10.1.	<i>Databinding of the TreeView control</i> .....	78
8.10.2.	<i>Layout-Settings of the TreeView Control</i> .....	78
8.11.	ENHANCEMENTS IN ONETOMANY-FORMS.....	79
8.12.	VFX - CGRID BUILDER.....	79
8.13.	VFX - CCHILDGRID BUILDER.....	80
8.14.	VFX - CPICKFIELD BUILDER.....	82
8.15.	VFX - CPICKALTERNATE BUILDER.....	86
8.16.	VFX - CPICKTEXTBOX BUILDER.....	88
8.17.	VFX - COMBO PICK LIST BUILDER.....	89
8.17.1.	<i>Pick list maintenance form</i> .....	91
8.17.2.	<i>The class CComboPicklist</i> .....	91
8.18.	VFX - PARENT/CHILD BUILDER.....	92
8.18.1.	<i>Preparing the Parent-Form</i> .....	92
8.18.2.	<i>Preparing the Child-Form</i> .....	93
8.18.3.	<i>Settings in VFX - Parent/Child Builder</i> .....	95
8.19.	VFX - DOCUMENT MANAGEMENT BUILDER.....	96
8.20.	VFX - MESSAGEBOX BUILDER.....	96
8.21.	VFX - MESSAGE EDITOR.....	98
8.22.	VFX - CLASS SWITCHER.....	99
8.23.	VFX - PROJECT PROPERTIES.....	99
8.24.	VFX - HELP WIZARD.....	101
8.25.	VFX - PROJECT UPDATE WIZARD.....	101
8.26.	PDM - PROJECT DOCUMENTING.....	102

<b>9.</b>	<b>THE VFX – MENU DESIGNER .....</b>	<b>103</b>
<b>10.</b>	<b>FEATURES OF THE CREATED FORM.....</b>	<b>106</b>
10.1.	CDataFormPage FORM USER INTERFACE .....	106
10.2.	THE VFX POWER GRID .....	107
10.3.	FORM BASED ON THE CLASS CTableForm .....	108
10.4.	FORM BASED ON THE CLASS COneToManyForm .....	109
10.5.	PRINTING.....	109
10.6.	SENDING E-MAIL.....	111
10.7.	SENDING FAX .....	112
10.8.	FILTERING .....	114
10.9.	LAYOUT .....	115
10.10.	DOCKABLE FORMS .....	116
10.11.	VFP TOOLBOX FOR END-USERS.....	116
10.12.	TREEVIEW .....	117
10.13.	DOCUMENT MANAGEMENT WITH THE CLASS CDOCUMENTMANAGEMENT.....	118
10.14.	ABOUT DIALOG .....	118
10.15.	FURTHER END-USER FEATURES IN VFX 9.5.....	119
<b>11.</b>	<b>DATA HANDLING .....</b>	<b>120</b>
11.1.	DATA HANDLING CONCEPTION .....	120
11.2.	CONCEPT OF NEW APPLICATIONS.....	120
11.3.	VFX - CURSORADAPTER WIZARD.....	121
11.3.1.	<i>Choosing the Datasource .....</i>	<i>121</i>
11.3.2.	<i>Choosing the underlying class and class library.....</i>	<i>121</i>
11.3.3.	<i>Choosing Tables.....</i>	<i>123</i>
11.4.	DATA ACCESS WITH CURSORADAPTER .....	123
11.4.1.	<i>The class CBaseDataAccess.....</i>	<i>124</i>
11.5.	MANAGING DATA ACCESS USING THE FILE CONFIG.VFX .....	124
11.6.	SWITCHING BETWEEN DBC AND SQL SERVER.....	126
11.7.	FORMS BASED ON VIEWS.....	126
11.8.	MULTI-CLIENT-SUPPORT.....	127
11.9.	UPDATING THE USER'S DATABASE.....	128
11.9.1.	<i>VFP-Databases usage .....</i>	<i>128</i>
11.9.2.	<i>SQL Server-Database usage.....</i>	<i>128</i>
11.10.	INDEX FILES.....	129
<b>12.</b>	<b>APPLICATION PROTECTION USING ACTIVATION KEY.....</b>	<b>130</b>
12.1.	LIST OF USED TERMS.....	130
12.2.	HOW IT WORKS .....	130
12.3.	DEFINING ACTIVATION RULES .....	132
12.4.	GENERATING AN ACTIVATION KEY.....	135
12.5.	CVFXACTIVATION CLASS PROPERTIES .....	137
<b>13.</b>	<b>CREATING MULTILINGUAL APPLICATIONS USING VFX .....</b>	<b>138</b>
13.1.	LOCALIZING AT DESIGN TIME .....	138
13.2.	RUN-TIME LOCALIZATION .....	139
13.3.	VFX – LANGSETUP BUILDER.....	139
<b>14.</b>	<b>VFX.FLL.....</b>	<b>142</b>
14.1.	PRODUCT ACTIVATION .....	142
14.2.	DATA BACKUP AND ARCHIVING .....	142
14.3.	SQL SERVER .....	144
14.4.	INTERNET, E-MAIL AND SUPPORT FUNCTIONS .....	144
<b>15.</b>	<b>VFX – AFP WIZARD.....</b>	<b>147</b>
15.1.	VFXAFPMETA.DBF DESCRIPTION .....	148

<b>16.</b>	<b>ADVANCED DEVELOPMENT TECHNIQUES.....</b>	<b>150</b>
16.1.	ADDING A FORM IN THE OPEN-DIALOG.....	150
16.2.	SYSTEM SETTINGS IN OPTIONS DIALOG .....	151
16.3.	ACTIVE DESKTOP.....	151
16.4.	MORE FUNCTIONS.....	152
16.5.	MOVER-DIALOG .....	152
16.6.	OLE-CLASSES .....	153
16.7.	DEBUG-MODE .....	153
16.8.	DELAYED INSTANTIATION .....	153
16.9.	IMPORTANT VFX – METHODS .....	154
16.9.1.	<i>Form methods</i> .....	154
16.9.2.	<i>Methods of the Application object</i> .....	155
16.10.	VFX PRIMARY KEY GENERATION .....	155
16.11.	DATA MANIPULATION TRACKING.....	156
16.12.	ASKFORM .....	156
16.13.	PROGRESS BAR.....	157
16.14.	DATE SELECTION .....	157
16.14.1.	<i>cPickDate class</i> .....	157
16.14.2.	<i>cDatetime class</i> .....	158
16.15.	REPORT SELECTION .....	158
16.16.	THE MICROSOFT AGENTS.....	158
16.17.	THE VFX – RESOURCE TABLE.....	159
16.18.	THE INCLUDE FILES.....	159
16.19.	OLE DRAG & DROP .....	160
16.20.	HOOKS .....	160
16.21.	BUSINESS GRAPH .....	161
16.21.1.	<i>Example</i> .....	162
16.22.	TOOLBARS.....	163
16.22.1.	<i>Use the Main Toolbar you like</i> .....	163
16.22.2.	<i>Adding a toolbar to a Form</i> .....	165
16.23.	CWIZARD-CLASS.....	166
16.24.	CDOWNLOAD CLASS .....	166
16.24.1.	<i>Macro language commands</i> .....	167
16.24.2.	<i>Example</i> .....	168
16.25.	CCREATEPDF CLASS.....	168
16.26.	CEMAIL CLASS.....	169
16.27.	CARCHIVE CLASS.....	170
16.28.	APPLICATION UPDATE .....	171
16.29.	VFP TOOLBOX FOR DEVELOPERS .....	172
16.30.	FURTHER DEVELOPMENT WITH VFP.....	172
16.31.	ERROR HANDLING .....	172
16.32.	TROUBLESHOOTING GUIDE.....	172
16.33.	FURTHER ENHANCEMENTS FOR DEVELOPERS.....	173
<b>17.</b>	<b>REMOTE SUPPORT .....</b>	<b>175</b>
17.1.	HOW DOES THE REMOTE CONTROL WORK? .....	175
17.2.	PREREQUISITIES .....	175
17.3.	SUBDOMAIN REGISTRATION .....	175
17.4.	THE REMOTE SUPPORT APPLICATION RADMIN .....	176
17.5.	THE REMOTE CONTROL FROM THE POINT SUPPORTERS .....	176
<b>18.</b>	<b>DOCUMENTATION.....</b>	<b>178</b>
18.1.	SUPPORT.....	178
<b>19.</b>	<b>SUMMARY .....</b>	<b>179</b>
19.1.	YOUR FEEDBACK IS IMPORTANT FOR US!.....	179

## 1. Introduction

Rainer Becker

Welcome to the new version, 9.5, of Visual Extend, which we are particularly proud of! This one is the biggest update produced for this well-known framework so far. That marketing phrase of course begins to become a little boring with its repetitions both for Visual FoxPro and Visual Extend; but nevertheless the statement actually hits the nail on the head in many areas of both products! Let us start with Visual FoxPro 9.0:

### 1.1. Based on Visual FoxPro 9.0

Visual Extend 9.5 is based on Visual FoxPro 9.0. Both new versions are available for purchase since the beginning of 2005. Besides the fact that Visual Extend 9.5 needs Visual FoxPro 9.0 as a prerequisite, there are many more reasons to take a closer look at the newest version of Visual FoxPro, and to buy it. Among other features, Visual FoxPro 9.0 provides:

- Substantial extensions in the area of the database engine, especially as regards SQL syntax and the fall of many of Visual FoxPro's limitations.
- A feature much demanded and yearned for in many years, particularly in countries of German language, has finally arrived: The totally new-built report designer and a fundamental reworking of the report execution, with convincing results.
- Diverse improvements to the user interface such as Docking/Anchoring for forms, enhanced support of graphics, Autotext, and many others – and Format Z is back, too.

And a lot of details have been fixed, improved and extended in the new version. A nice addendum we find in a new little property for grids:

**Tip: Rushmore Optimization in Grids**

A new property 'optimize' is available for grids, for the first time providing the long-awaited Rushmore Optimization for the presentation of tables. Now the grid is not slower than a BROWSE command any more.

PS: If you ever happened to be obliged to use a filtered table within a grid, set this new property to .T. (the default value is of course .F.).

The actual list of improvements we naturally do not want to reproduce here fully in print, but you can believe the final version of Visual FoxPro 9.0 to be considerably different from the Public Beta that has been available for a long time, and it has become quite a bit larger!

### 1.2. The Combination that makes it: All in One

As an object-oriented development environment and as a relational database system, Visual FoxPro 9.0 has become even more attractive for application development. Now the framework Visual Extend enriches Visual FoxPro's toolset by those components essential for the quick development of applications known as 'RAD'.

The first way this is being achieved consists in the supply of a large application frame with many important standard functions for your application like e.g.

- Administration of users, groups, access rights
- Data backup and restore
- Database maintenance and repair
- Error log; locking, user, and changes log
- Favorites, Customize and Options, Info box
- Sorting and Searching in the grid, Filtering via dialogue
- Report output incl. output as PDF/Fax etc.

PS: We underlined only the end user features new and for the first time available in Visual Extend 9.5. Everything else – and a lot more – has already been available in VFX for a long time.

*Read more about the numerous functions of the generated application which are prepared and ready to use for you in chapter 3.2. 'Properties of applications generated with Visual Extend'.*

The second way consists in the supply of a comparatively small set of base classes, mainly in the areas of forms, grids, and lookups of various flavors. And the appropriate huge builders come with it, working together like the tools of a multi-function Swiss army knife, enabling the developer to configure these classes fast.

*Read more about the many classes prepared for you and about the related builders in chapter 3.3 'Features for developers'.*

The package is supplemented and rounded off by administrative functions for software developers and for small and medium sized software companies, for example

- Database and application updating
- Activation key and version update for modules
- Support of remote administration (new)

And then there is our new Web Service for your simplified registration of Visual Extend plus the requesting of replacement keys, and... but we will not treat the entire manual in the introduction. Let us only consider one subject of prominent importance to Visual Extend in a little more detail, the subject of Builders:

### **1.3. Even more productive with new Builders in Visual Extend 9.5 !**

Provided you are already working with Visual Extend, you will find out in almost every item of the following list of functions how it will facilitate your daily work! If you are not yet using Visual Extend, you will at least realize how big the current update really is. Please read:

- All properties of the application object can be called on in the extended options of the [Application Wizard](#) – and later they can be changed as well in the [Application Builder](#)!
- In the [Project Properties](#) you can define the selectable classes for all Builders, and you can also define them as default and as AutoComplete in one go
- The [Project Toolbox](#) offers you all project-specific classes in an overview and for direct drag & drop or (see right mouse button) for direct instantiating
- The [Project Documenting Wizard](#) gives you an interface to a special VFX version of PDM for the documentation of your application
- The [Project Update Wizard](#) permits the semi-automatic instead of the manual updating of existing projects to new versions and to new builds of Visual Extend
- The [Data Environment Builder](#) (integrated with the Form Wizard/Builder) makes the visual assembly of the data environment possible, including the integration of the CA Builder
- All the extended [Form Builders](#) have tabs for view parameters (including input fields and requery buttons), linked tables, required fields, and additional columns for the report display
- The [Parent/Child Builder](#) enables you to visually define all dependent child forms instead of the manual defining in the 'onMore' method
- In the [Language Setup Builder](#) you can activate the localization/ translation of the user interface at runtime so that the users can choose of their own ...
- In The [Customer list](#) the developer may not only generate activation keys, but also administer all the related customer data
- In the [Update management](#), define new versions and grant the customers the matching download permissions
- In the [Configuration management](#) you can now store as many definitions as you please, put all VFX tables onto the backend server, and add your own columns to the definition which are stored encrypted, too
- The [CursorAdapter Wizard](#) automatically generates cursor adaptor classes for all tables within a database container in a library you choose
- The [AuditTrigger Wizard](#) automatically generates all triggers for the audit trail for single tables or all tables of a database container for tracking and auditing purposes



- In the System object you can define and manage the download scripts for Ghostscript, Acrobat Reader, OutlookYesNo, as well as Update, Backup, DUN and DynDNS via a definitions mask
- Place a DocumentManagement container on an empty tab and define the document linkage to the current record with the Document Management Builder – thus all document links are in one central table
- Place a cBusinessGraph container onto an empty tab, and... hold on, this Builder is regrettably just not finished yet <bg>.
- Place a cComboPicklist on your edit page and use the ComboPickList Builder to define and state the selectable values. And:
- Edit the values in the respective maintenance form, and use the definition in the next form again by choosing from the Combobox overview!
- Or use a cTextCalculator class, a cTexteMail, cTextHyperlink, cLinkTextbox, or a cTextTAPI class – you won't even need a Builder for this...

We underlined **ONLY** the new or substantially extended builders and system functions from the VFX menu. Therefore we say:

## **Visual Extend 9.5 – More productive than ever!**

And we suppose that you can agree with us on this statement straight away.  
Now, have fun with our manual of the newest version!

## 2. Quick Overview

### 2.1. Introduction

For many years, Visual Extend belongs to the most efficient add-in products of Visual FoxPro. With Visual Extend (abbreviated in the following text with VFX), it is possible to create the skeleton for a fully functional Visual FoxPro application within a few minutes. When a database for the developed application is already available, it is easy within shortest time to create data manipulation forms with the builders of VFX. We will learn the most important features of VFX, which we will walk through during steps while creating an application.

Visual Extend requires a Visual FoxPro version with at least same version number as Visual Extend that you have. To use Visual Extend 9.5, you need Visual FoxPro 9.0.

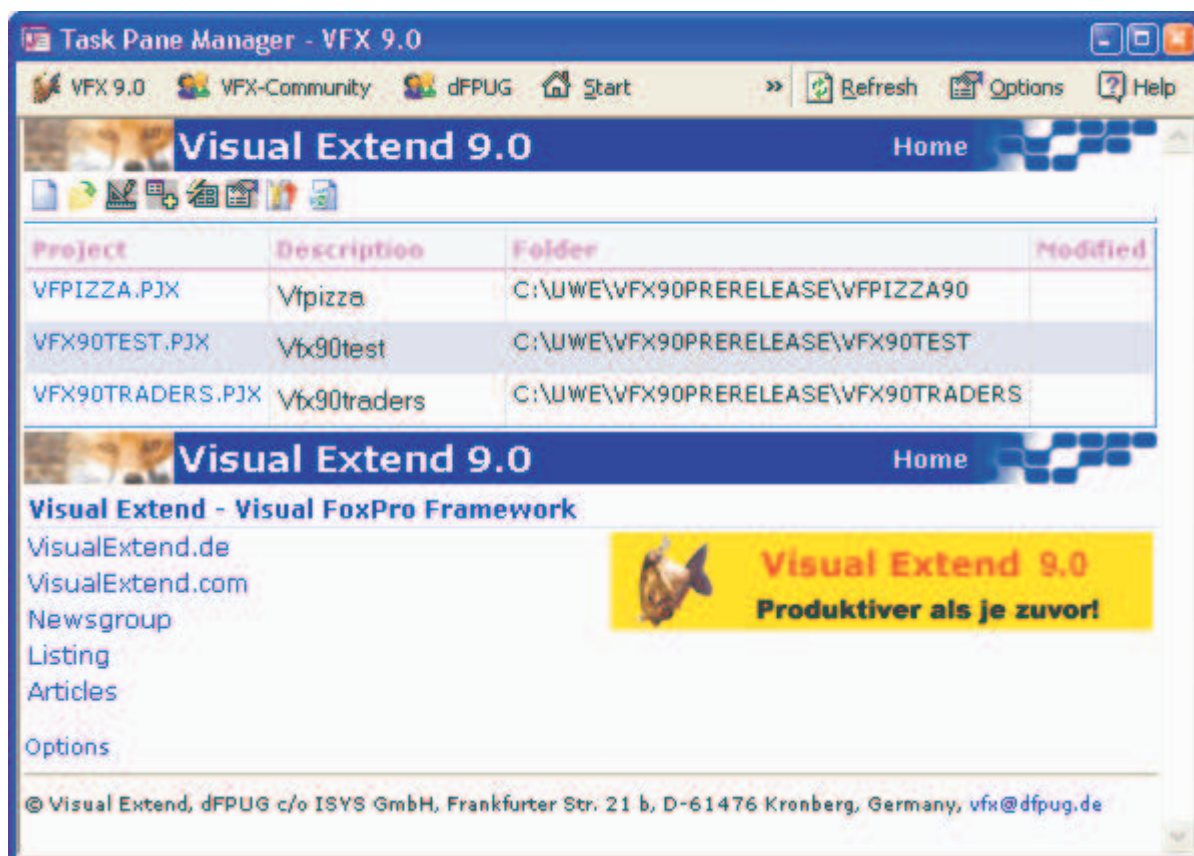
#### 2.1.1. Installation

After the installation of VFX, it is helpful to integrate the VFX Menu into the standard Visual FoxPro Menu. For this purpose in Config.fpw file should be inserted a line:

```
Command = DO <VFX - Installation path>\builder\vfxmlnu.app
```

#### 2.1.2. VFX Task Pane

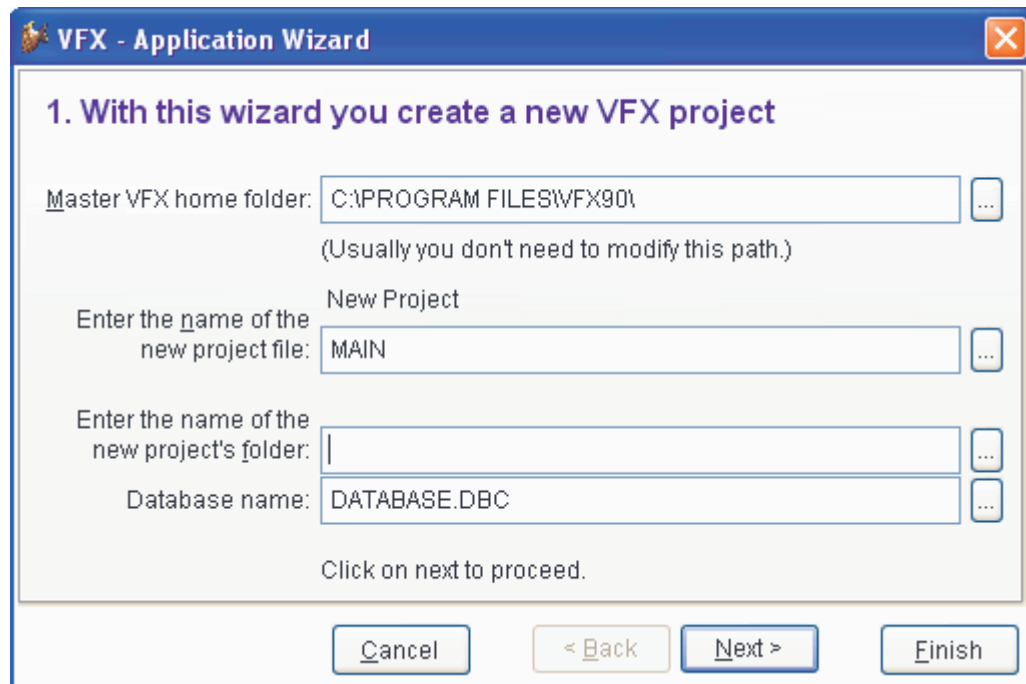
At first start of VFP after VFX 9.5 installation, VFX 9.5 Task Pane will be automatically integrated into Visual FoxPro Task Pane.



In the VFX 8.0 Task Pane is placed a useful tool - the Application Manager. Information about all VFP projects is stored in a table. Through the VFX - Application Manager can be opened a project. The path is set automatically into the project folder. In addition through VFX - Application Manager can be performed "Rebuild all". By this, the project will be completely compiled. Changes in Include files are taken into account.

### 2.1.3. VFX - Application Wizard

A new application will be created with the VFX - Application Wizard.



**VFX - Application Wizard**

#### 1. With this wizard you create a new VFX project

Master VFX home folder: C:\PROGRAM FILES\VFX90\ ...  
(Usually you don't need to modify this path.)

New Project

Enter the name of the new project file: MAIN ...

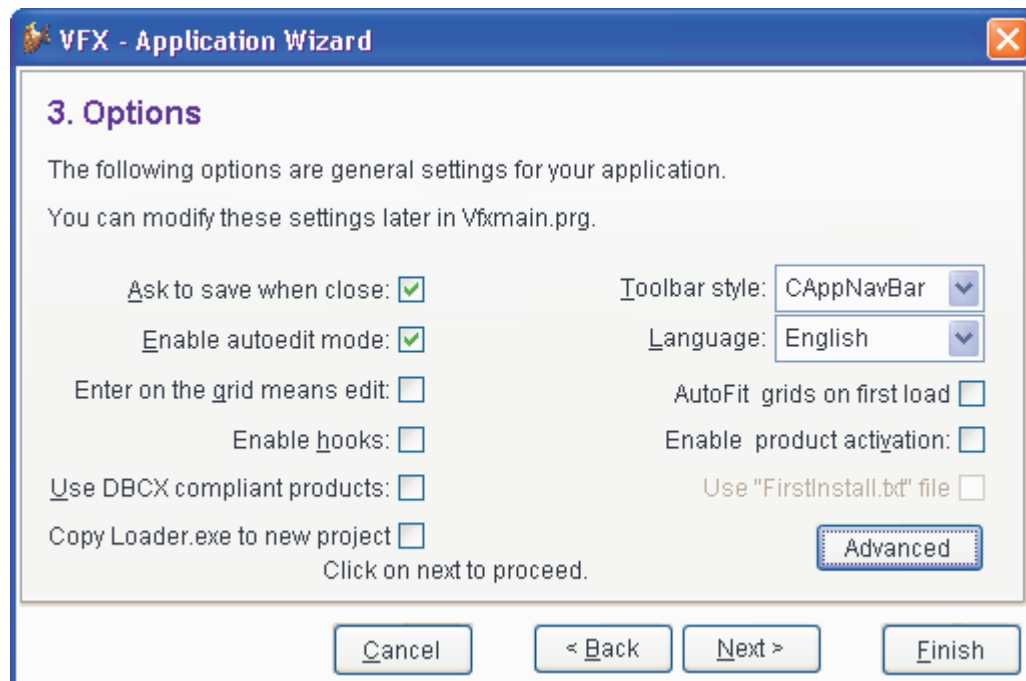
Enter the name of the new project's folder: | ...

Database name: DATABASE.DBC ...

Click on next to proceed.

Cancel < Back Next > Finish

At first run of the wizard, as language for the created application by default will be suggested the language of the used Visual FoxPro version. On every subsequent run will be proposed the last used language. After the "Finish" – button is pressed, the files from the empty VFX Sample application will be copied into the newly created project folder and compiled afterwards.



**VFX - Application Wizard**

#### 3. Options

The following options are general settings for your application.  
You can modify these settings later in Vfxmain.prg.

Ask to save when close: <input checked="" type="checkbox"/>	Toolbar style: CAppNavBar ▾
Enable autoedit mode: <input checked="" type="checkbox"/>	Language: English ▾
Enter on the grid means edit: <input type="checkbox"/>	AutoFit grids on first load <input type="checkbox"/>
Enable hooks: <input type="checkbox"/>	Enable product activation: <input type="checkbox"/>
Use DBCX compliant products: <input type="checkbox"/>	Use "FirstInstall.bt" file <input type="checkbox"/>
Copy Loader.exe to new project <input type="checkbox"/>	Advanced

Click on next to proceed.

Cancel < Back Next > Finish

## 2.2. Functionality of the new Application

The application created with the Application Wizard can be tested immediately. For this purpose the main program *Vfxmain.prg* can be started directly from the project manager. Alternatively also an App or an Exe file can be create and tested. However, usually this is not necessary during the development.

The application starts with a Splashscreen. As picture for the Splashscreen is used a png-file, which can be easily edited or exchanged by the developer. It is possible to suppress the Splashscreen. After displaying the Splashscreen, the main window is created and login form is invoked. By default each user of a VFX application must be logged in with user name and password. It is possible to skip the login form and to log the user automatically using the Windows login name.

### 2.2.1. Usage

After the login, the VFX application is used similarly like Office applications. Users, who are familiar with the usage of Word or Excel, can practically immediately work productively with a VFX application.

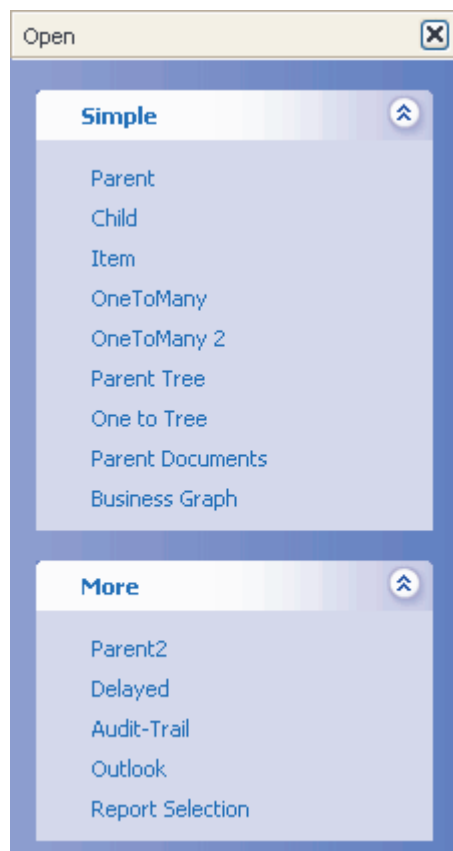
### 2.2.2. Standard toolbar



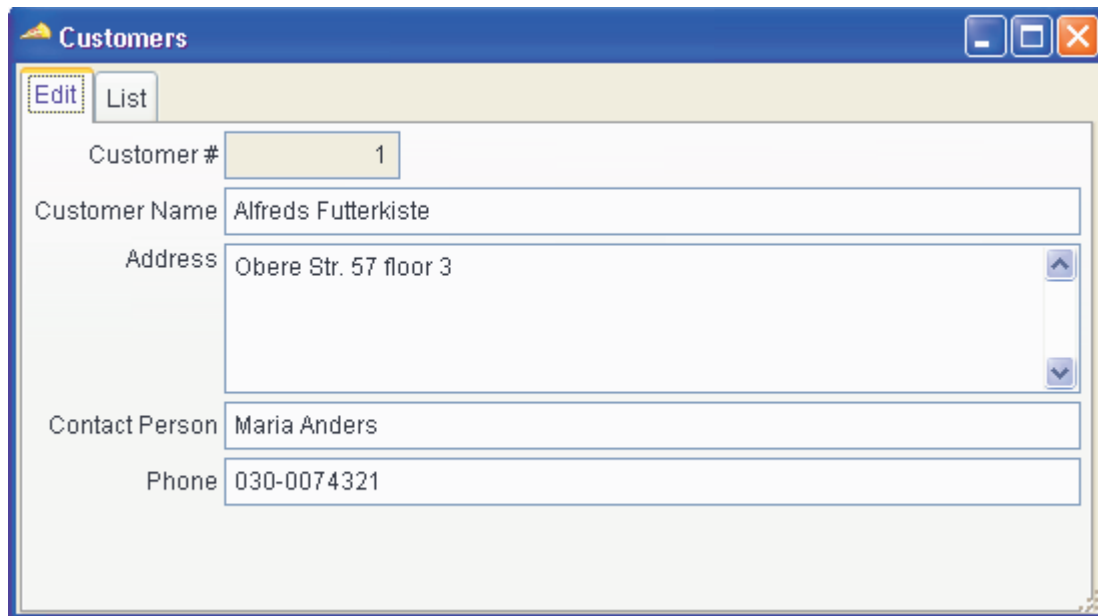
Many of the buttons on the toolbar are identical in their function with those from MS Office products.

### 2.2.3. Open-Dialog

By default, forms are started using the Open-Dialog. The Open-Dialog appears in Windows XP layout. The information about the forms that are shown in Open-Dialog is located in the table *Vfxfopen.dbf*.



## 2.2.4. Forms



If for a form the property *lAutoedit* is set to *true* (that is the default value), all control members on the form are always active. The user can select an element with the mouse or the keyboard and can immediately start editing data. As soon as data are interactively changed, the form switches automatically into Edit mode.

On the list Page in VFX forms is placed a Grid. By default in all columns of the Grid can be searched incremental. For this purpose, just set the focus into the desired column. With the first typed letter or digit, the sorting sequence will be changed on this column. If necessary, a temporary index is created automatically. The column header is marked by up or down arrow, similar to the Windows Explorer.

By default the size of VFX forms can be changed by the user at run-time. Thereby the size of all controls changes proportionally. Within Grids the size of the control is not changed by default. When a form is increased, more lines and columns become visible in the Grid.

All settings on forms will be saved on a per user basis. If the user opens the form again, the form will appear at the same position of the screen and with the same size, as it was last closed. In addition, the settings of the Grids (column width, column sequence and sort order) are saved.

Usually VFX forms have a private data session and can be opened several times without problems. Through a property of the form (*lMultiinstance*) multiple instances can be disabled.

## 2.2.5. User's management

In VFX is included a user administration. Here are placed a form for editing the user data, a form for defining the user rights, a management of user's groups as well as a login screen.

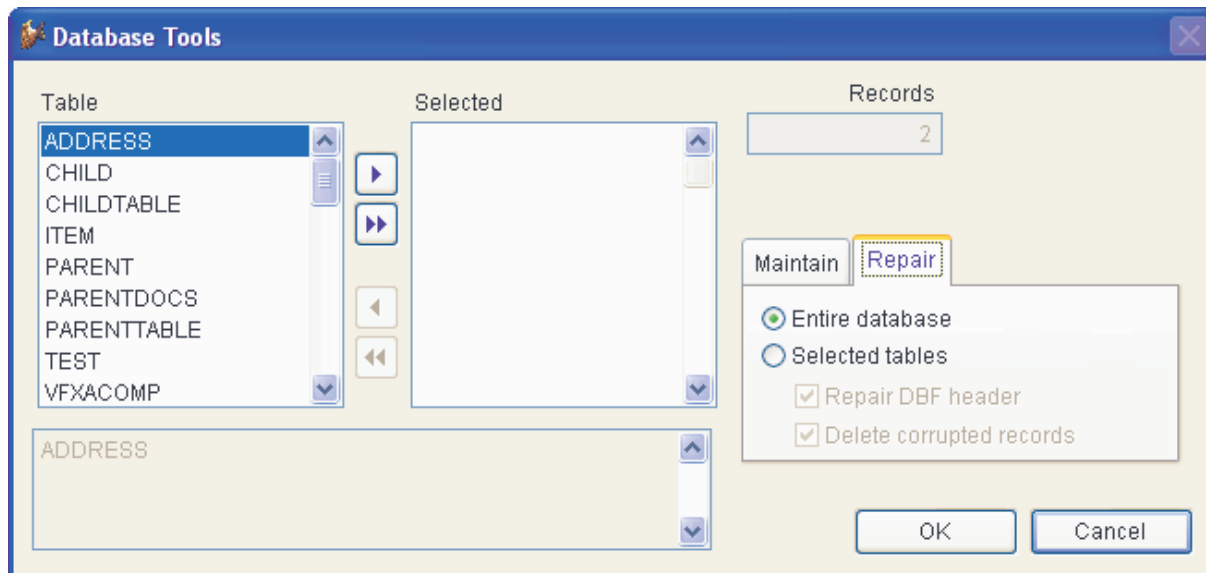
After the successful login of the user, is created a global object named *goUser*. For all fields of the current user data record (from the table *Vfxusr.dbf*), corresponding to the currently logged user, are added properties of the object *goUser*. The name of the property correspondent to the field name in the table *Vfxusr.dbf*. It is possible in each place in the program to check the value of properties of this global object, in order to decide whether a user may perform a certain action. So can, for example, be restricted the selection of a menu option, which opens a form or disable the editing of a field on a data manipulation form.

## 2.2.6. Error tracking

When a run-time error is raised, the error is displayed in a MessageBox. In addition, the error is logged in a table. At this point are saved the name of the current user, date, time, and the status of all opened tables as well as the list of memory variables. Further application's behavior when a run-time error is raised can be set through the properties of the application object.

## 2.2.7. Database Tools

Under the menu option Tools – Database... is invoked a form with a Mover Dialog.



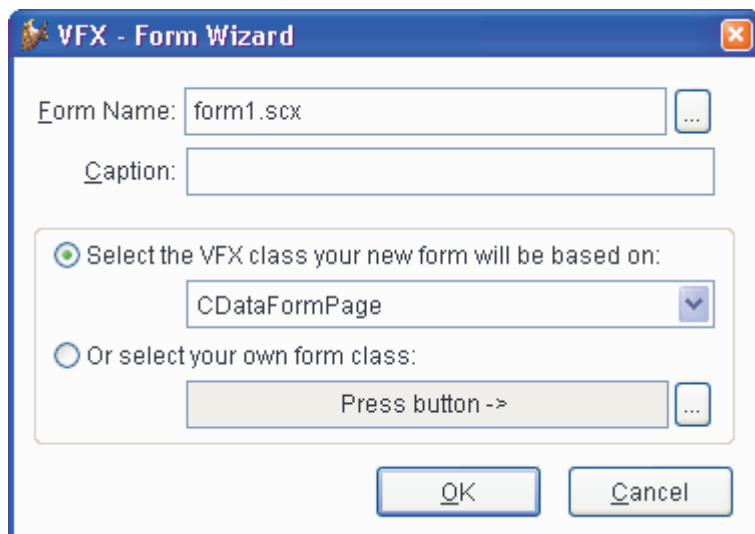
Here tables can be packed or reindexed.

## 2.2.8. About Dialog

A standard About Dialog is included in all VFX applications. The displayed values come from an Include file, which was created when the project was generated.

### 2.3. Creating a Form with the VFX – Form Wizard

With the help of the VFX – Form Wizard a new form, based of a VFX Form class, can be created and included into the project. The most often used form class is the class *CDataFormPage*.



### 2.4. VFX – Data Environment Builder

At next step, in every VFX – Form Builder is edited the data environment. The tables or views, used by the form are inserted into the dataenvironment.

In dataenvironment can be added tables, views or existing CursorAdapter classes and can be created new CursorAdapter classes. By clicking on *Add* button you can add existing tables or views in the dataenvironment. The VFP Add Table or View dialog is open. When a cursor in the dataenvironment is based on a table, in the column *Order* can be chosen an index tag of the table.

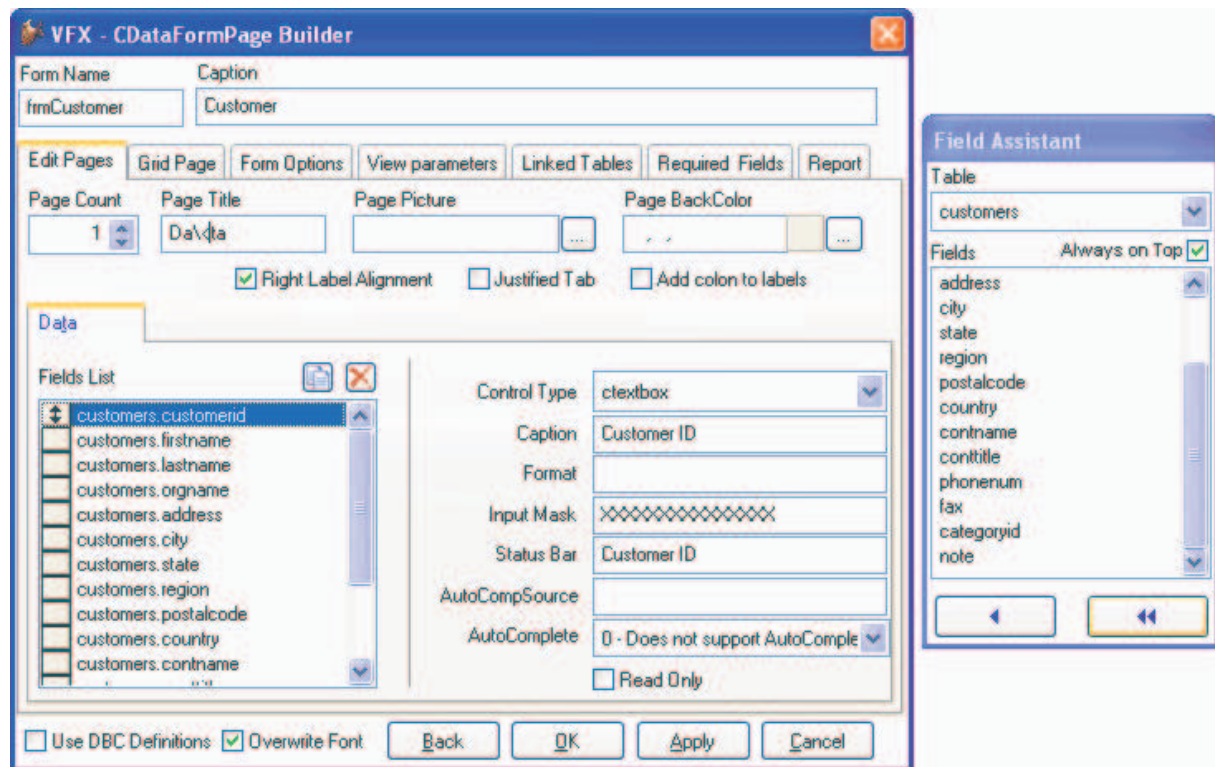
For a simple form for editing data from a table, it is enough to add this table into the dataenvironment. Afterwards, using VFX - Form Builder, to the form can be added controls.

The VFX – Form Builder reads the data environment and allows controls generation, based on the fields of the tables. At run-time dataenvironment will be checked, to determine whether a *Tableupdate* and/or a *Tablerevert* must be performed for tables.

## 2.5. The VFX - Form Builder

With this builder are generated the necessary controls for the form. Thereby, can be selected base VFX class for each control as well as some properties can be set.

Along with creation of a form, in the table *Vfxfopen.dbf* automatically will be inserted a row, so that the form can be started using the Open Dialog.



The VFX – Form builder is fully reentrant. This means, that the builder can be arbitrarily often called, when the form’s settings has to be changed. It is also possible the form is to be edited on by hand with VFP and afterwards to be edited again with the Form Builder, without losing or overriding the properties.

## 2.6. The VFX - CGrid Builder

When it is needed changes to be made for the Grid only, it is not necessary to use the Form Builder. The properties of the Grids can be changed with the VFX - Grid Builder. As all VFX builders, the Grid Builder is reentrant too.

## 2.7. Test

The form can be started and tested directly from the form designer or from the project manager. In the init method of all VFX forms it is checked whether the application object exists. In case when it is missing, the form was started directly from the project manager and VFX creates the environment independently, in order to let the form run fully functionally. Also the main toolbar is instantiated and can be used for the operation of the form.

Of course, it is also possible the project to be started by the main program *Vfxmain.prg*. The form can be started then from the Open Dialog.



## 3. Introduction

### 3.1. Overview

Visual Extend requires a Visual FoxPro version with at least same version number as Visual Extend that you have. To use Visual Extend 9.5, is needed Visual FoxPro 9.0.

Visual Extend 9.5 holds out a comprehensive development environment for software developers working with Microsoft Visual FoxPro 9.0. Visual Extend includes Builders which assist the Software Developer in their daily work and dramatically speed up the software development process without sacrificing any of the Visual FoxPro features. With Visual Extend, Visual FoxPro becomes a real Rapid Application Development Tool for both Desktop and Client Server Database Application Development.

Visual FoxPro is an outstanding Software Development Environment. Thanks to its Object Orientation and OLE Capabilities, the Software Developer's dream of easy code reuse of either personally developed, or third party modules, becomes true. However, starting to develop your own Software Development Environment in Visual FoxPro from scratch is a major undertaking. Not only because it's difficult to develop a solid Class Library as a Foundation Class for all Applications. It is also rather time consuming to use these Classes and manually fill in the right Properties and Methods in the Property Sheet while developing new Applications all over again.

Visual Extend for Visual FoxPro fills exactly this gap by bringing a complete Application Development Framework to the Visual FoxPro Software Developer Community. Thanks to the thought-out, modular design of Visual Extend, every Software Developer can decide whether to use all of the Visual Extend Application Development Framework or to take only some parts of it for creating their own application. The Object Orientation of Visual Extend allows the Developers to subclass existing Visual Extend classes and to customize and enhance further the development environment according their specific needs.

Visual Extend is not just a set of Foundation Class Library. It's much more. Visual Extend provides the Software Developer with a powerful Foundation Class Library with equally powerful Builders for a maximum of productivity gain. Visual Extend includes the following components:

- Modular, Microsoft Compliant Visual Extend Foundation Class Library with extensive Application Development Support
- Visual Extend Wizards and fully reentrant Builders for Application, Form, Grid, Childgrid, Picklist, PickTextbox and OneToMany Forms and much more other.
- Other Visual Extend Developer Productivity Tools like the Developer Menu, VFX Task Pane, VFX – Base Class Switcher and Visual Object Name Picker

### 3.2. Specifics of Applications created using Visual Extend

Applications which have been developed using Visual FoxPro together with the Application Development Framework Visual Extend will have the following characteristics:

- Ready for Office Compatible Certification
- Standard Toolbar and including optional individual Toolbars for any Form
- Using XP-Themes in all controls.
- Hot Tracking for buttons in Toolbar.
- Icons in Menus.
- Navigation, Search, New, Copy, Edit, Delete optional on Form or on Main Toolbar.
- Multi-instance of Forms.
- Recently used form list in File Menu, actually open forms in Window Menu.
- Incremental Search including Autosort in all VFX Grids.
- Alternate Sort by Doubleclicking on any Column Header in any VFX Grid.
- Indicate actual sort in column header optionally using colors.
- Auto Save and Restore of Size and Position of any Form.
- Auto Save of all Grid Layout Changes including the Current Sort.

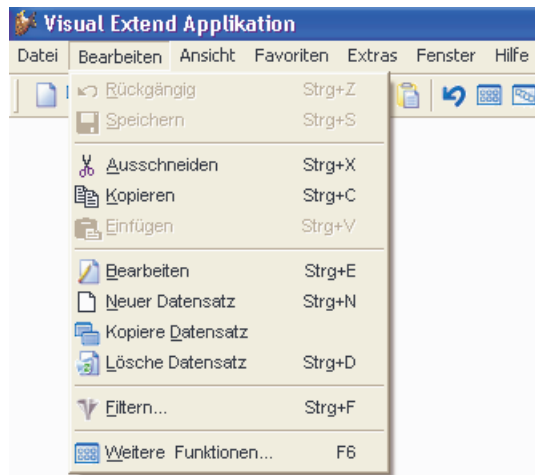
- Picklist Control with Auto-validation and optional Data Fetching.
- Picklist Form with Incremental Search, Auto-sort, Alternate Sort by Doubleclicking on any Column Header as well as Maintenance and Insert features.
- Auto-save and Restore of Size and Position of Picklist Forms including any Picklist Grid Layout Changes.
- Powerful Picklist Object within Childgrid.
- User Access Management including Password Encryption.
- Auto use of Network Logon Names including Autologon Feature.
- User Security including Form Level Security View, Edit, Insert and Delete restrictions.
- Database Tools for Packing, Reindexing and Repairing of local Tables.
- Complete run-time Error Tracking System.
- About Dialog
- User friendly Mover Dialogs for easy selection of multiple elements
- Automatic Synchronization with Windows System colors
- Favorites Menu
- XP-Style open dialog.
- Optional Active Desktop Single-Click User Interface.
- Auto Report feature for automatic creation of printed reports based on data in a grid.
- Report Selection and manipulation interface.
- Multi Data support including run-time switch between different databases.
- Automated Client Site Update for tables' structure updates for VFP- and SQL Server Databases.
- Optional Audit Trail Feature for data manipulation tracking.
- Optional Microsoft Agent assisted user interface.
- Automatic Printscreen feature.
- Possibility to create multilingual Applications.

### **3.3. Key Features for developers**

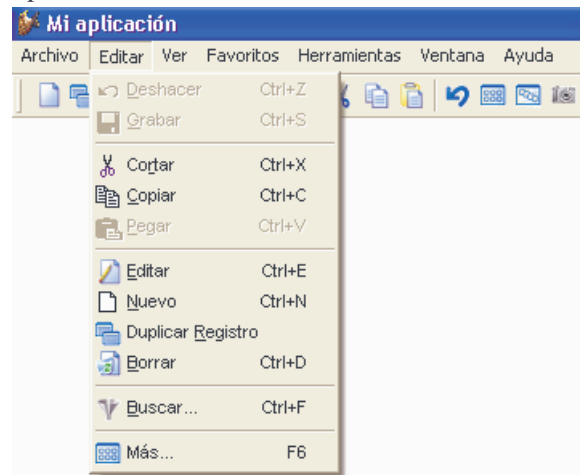
Software Developers using Visual Extend will appreciate the following features:

- Application Wizard for the automatic generation of new Applications in the language of your choice. After just a few seconds, your distribution-ready Visual FoxPro Application is prepared!
- Fully reentrance of all VFX Builders (Form Builder, OneToMany Form Builder, Table Form Builder, Grid Builder, Child Grid Builder, and PickTextbox Builder) which makes it easy to make changes on already created forms using the VFX Builders!
- Use the Visual FoxPro Environment whenever you want without losing the reentrance feature of the VFX Builders as long as you add/remove all controls using the VFX builders!
- Builders for Standard Forms including Parent/Child Form technique (Calling and Called By).
- Builder for Power Grids.
- Builder for all your Picklist needs.
- Builders for classical as well as advanced OneToMany Forms including Pageframe for the Parent and another Pageframe for multiple Childs tables all on one Form.
- All Builders get the Field Descriptions and other properties automatically from the Dataenvironment.
- Form Builders will automatically size any Textbox Controls according underlying Field Length.
- Use the VFX – Form Builders with own VFX based form and control classes.
- Run forms directly from form designer.
- Toolbar Navigation or Navigation Buttons on form as well as Buttonbar into a Form.
- Messagebox Builder.
- Task Pane Applications Manager.
- Easy subclassing of the application class and setup of the environment class.
- Easy setup of application specific main toolbars.
- Linked Parent/Child forms techniques.
- The complete Application Development Framework covers already all user interface elements in Bulgarian, Czech, Dutch, English, French, Finnish, German, Greek, Italian, Portuguese, Russian and Spanish. Start a new Application in the language of your choice without need of translating a single word of the Visual Extend Application Development Framework

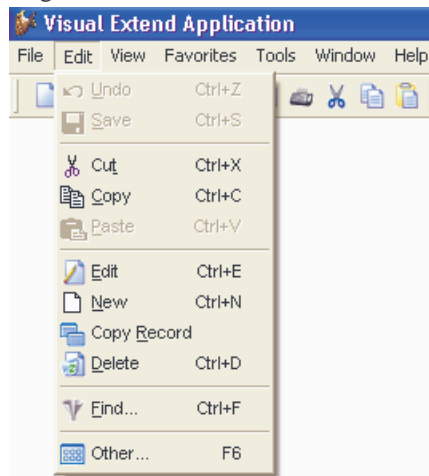
German



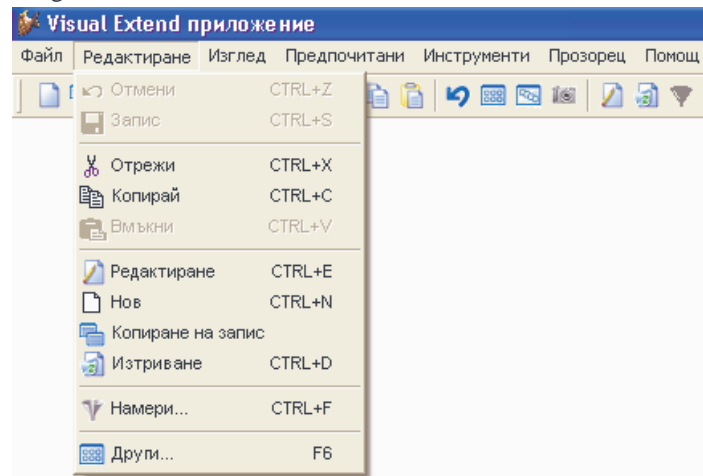
Spanish



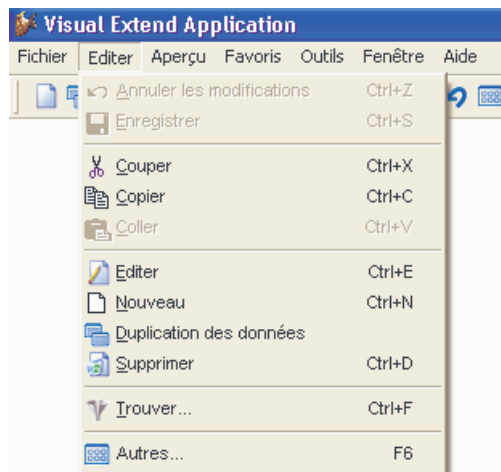
English



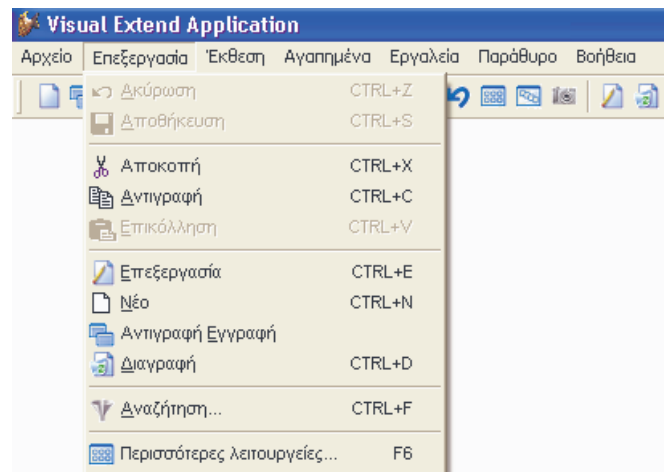
Bulgarian



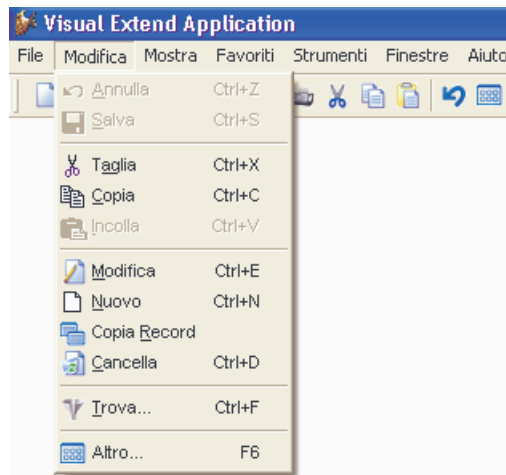
French



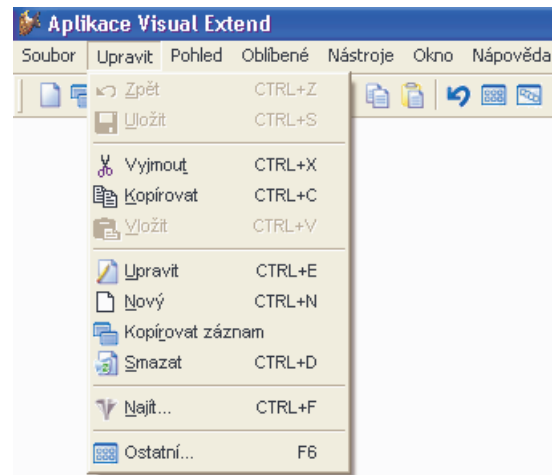
Greek



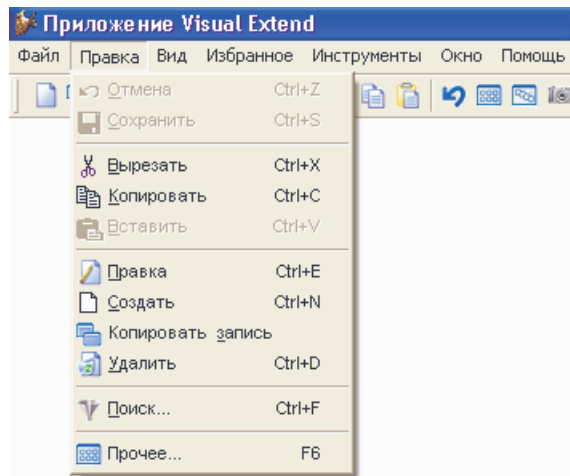
Italian



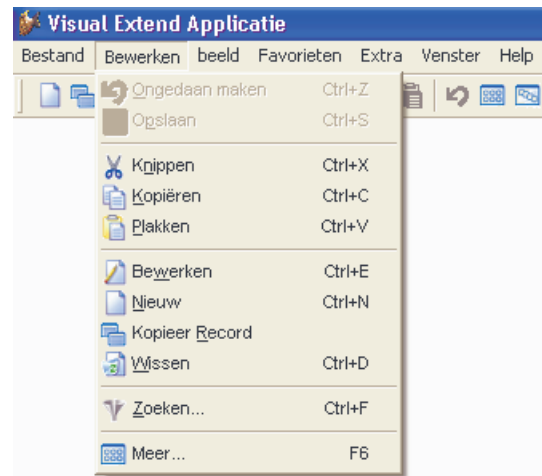
Czech



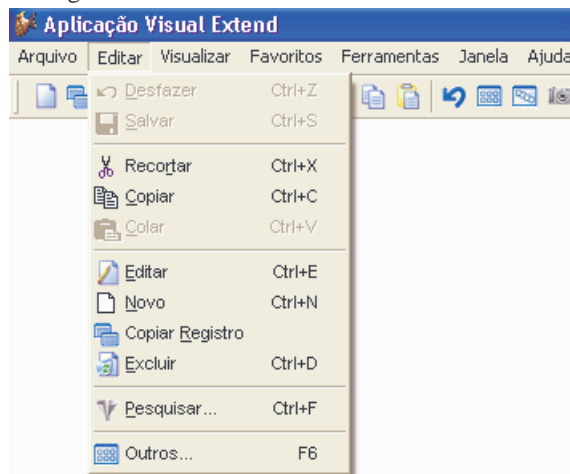
Russian



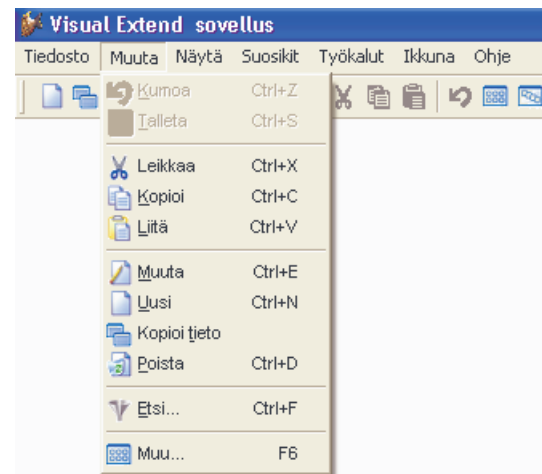
Dutch



Portuguese



Finnish



VFX helps you to create your Visual FoxPro Applications in a higher quality in much less time and therefore dramatically increases your development productivity. And all this without loosing any of the Visual FoxPro Features you like and use. Be more productive than ever before with Visual Extend for Visual FoxPro!

## 4. Included Tools

### 4.1. VFX – Class Libraries

You will find the class library files in the subfolder \VFX90\LIB. For a detailed description of all class library files including their classes, properties and methods, please refer to the VFX technical reference online help file. The technical reference is a Windows help file.

### 4.2. VFX – Wizards and Builders

All VFX Wizards and Builders are located in the \VFX90\BUILDER folder:

Builder	File	Description
<b>VFX Menu</b>	VFXMNU.APP	Adds a special menu point in your Visual FoxPro menu. From this menu you can call the VFX Application Wizard and other VFX wizards. <b>Tip:</b> If you have followed the installation instructions, this menu is automatically loaded when you start VFP.
<b>VFX – Wizards and Builders</b>	VFXBLDR.APP	The following VFX Wizards and Builders will help you to create professional Visual FoxPro Applications in record- breaking time: <ul style="list-style-type: none"> <li>√ Application Wizard for the generation of a new Application</li> <li>√ Form Wizard for the generation of a new Form</li> <li>√ Form Builder (including multi page forms, <b>reentrant</b>)</li> <li>√ Grid Builder (<b>reentrant</b>)</li> <li>√ Picklist Builder (<b>reentrant</b>)</li> <li>√ OneToMany Builder (including multi page for parent and multi page for multiple child tables, <b>reentrant</b>)</li> <li>√ ChildGrid Builder (<b>reentrant</b>)</li> <li>√ Picklist Builder for Picklists within child grids (<b>reentrant</b>)</li> </ul> <p><b>How to start:</b> If you followed the installation instructions you can call the VFX builders using the right mouse anytime when the corresponding object has been selected.</p>
<b>VFX LangSetup Builder</b>	LANGBLDR.APP	Automates the generation of the code for the VFX LangSetup method. Great help for creation of a multilingual application.  <b>How to start:</b> Either from the VFX Menu by selecting Form/LangSetup Builder or by starting LANGBLDR.APP.
<b>VFX Messagebox Builder</b>	MSGBLDR.APP	Automates the generation of messagebox dialogs and the associated constants for the include files.  <b>How to start:</b> Either from the VFX Menu by selecting Form/Messagebox Builder or by starting MSGBLDR.APP.
<b>VFX Message Editor</b>	MSGEDIT.APP	Automates the localization of messages and other captions, as well as the generation of the associated include files.  <b>How to start:</b> Either from the VFX Menu by selecting Form/Message Editor or by starting MSGEDIT.APP.
<b>VFX Menu Designer</b>	VMD.APP	Creates professional menus, using all options available in VFP. Using the visual VFX Menu designer it is possible to key up your menu much more detailed than using VFX Menu designer.  <b>How to start:</b> The VFX Menu designer is started when you open a menu for edition in VFP Project manager.
<b>VFX – AFP Wizard</b>	VFXAFPWIZARD.APP	Creates Internet applications with forms with same appearance and same functionality corresponding to the forms of your VFX application.  <b>How to start:</b> From the VFX Menu by selecting AFP/VFX – AFP Wizard.
<b>Project Documenting</b>	PDM.EXE	The Project Documenting wizard generates an extensive technical documentation for your VFX project in HTML format.  <b>How to start:</b> From the VFX Menu by selecting Project/Project Documenting

All VFX Form, Grid and Picklist Builders are fully reentrant! This means that during the Development cycle, you can call them as many times as needed without losing any of the settings, which have already been defined. In addition, changes done after the original generation of your forms within Visual FoxPro will be read by the Builders next time you call them.

Because of the very open approach of the VFX builders, advanced users might find it useful that the code the builders use is located in a dbf file *VFX90\LIB\BUILDER\VFXC CODE.DBF*. That makes it very easy if you want the builders to use your custom code. **Warning:** making changes to this code table requires advanced knowledge of VFX.

---

**NOTE:** Make sure to use the VFX – Form Builder as long as possible for adding and removing of any controls (defined through the selected fields). This allows you to profit the most from the high productivity the builders provide!

---

### 4.3. VFX Developer Productivity Tools

To make your VFX Development life even easier, VFX includes some very useful power tools:

Tool	File	Description
<i>VFX Task Pane</i>	VFXTASKPANE.XML	The VFX Task Pane allows easy switching from one project to another. The table which stores the actual references to your projects is called VFXAPP.DBF/CDX/FPT. This table is located in the folder <i>C:\Documents and Settings\All Users\Application Data\FPUG\Visual Extend\9.5</i> .
<i>VFX Class Switcher</i>	< in VFXBLDR, called from the VFX Menu>	Change the class of all of your forms within the current folder\lib and current folder \form. Allows easy switching from version without navigation buttons on the form (i.e.: CDataFormPage) to the one with navigation buttons (like i.e.: CDataFormPage). You can use the class switcher also to switch the class of a selected form control.
<i>VFX Object Name Picker</i>	< in VFXBLDR, called from the VFX Menu>	Puts the complete reference of the currently selected control into the clipboard. Sometimes very usefully since more visual than the VFP Object List called with the right mouse while in a code window.

### 4.4. Other Developer Tools

In addition to the well-known Builders, existing in former VFX versions, in VFX 9.5 are available new Power Builders for following Classes:

*CTreeViewForm*

*CTreeViewOneToMany*

*CPickAlternate*

*CPickAlterTextbox*

For further usage on disposal are new and enhanced wizards:

Define Activation Rules – Sets System specific parameters, which will be used for Product activation, as well as possible User rights.

Create Activation Key – Creates an Activation key based on a customers Installation key

Customer List – Managing customers data and customers activation key.

Manage Application Updates – Managing online product actualization.

Metadata Wizard – For creation and actualization SQL Server-Databases at customer side.

Manage Config.vfx – Managing data access.

Cursor Adapter Wizard – Automatically generates of CursorAdapter classes for all tables of a database.

Audit Trigger Wizard – Creates triggers for auditing changes for selected tables.

Manage Vfxsys.dbf – Managing the table Vfxsys.dbf with partial encrypted content.

VFX –AFP Wizard – Generates AFP-Web pages from VFX – Forms.

Update Project Wizard – Actualizing existing VFX projects to the actual build or the actual version.

Project Documenting – Generates technical documentation in HTML-Format.

Project Toolbox – Adds classes of the active project to VFP Toolbox.

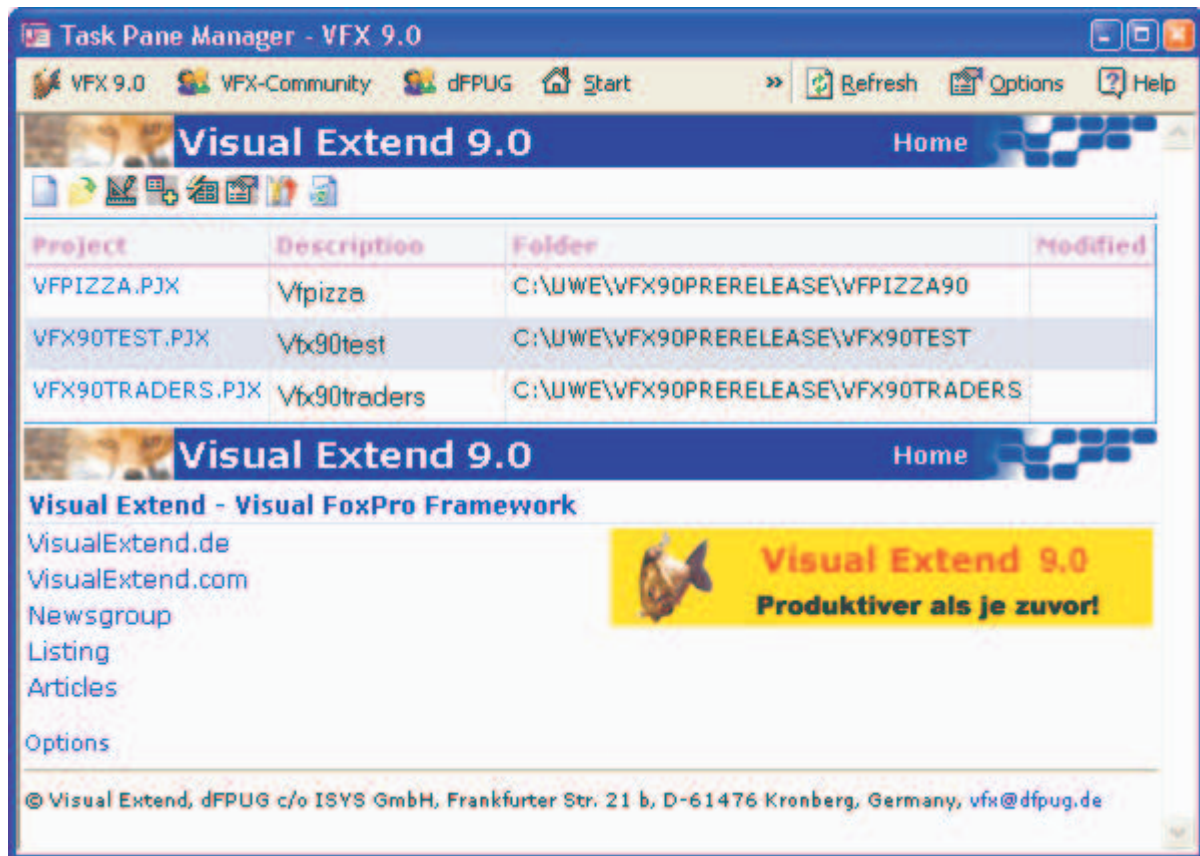
Parent/Child Builder – Managing relations between Parent- and Child- forms.

VMD (Visual Extend Menu Designer)

### 4.5. VFX 9.5 Task Pane


The VFX – Application Manager is integrated into the VFP Task Pane.





The following functions are available through buttons on the toolbar:

- New Project* Starts the VFX- Application Wizard.
  - Open Project* Opens a VFP-Project und sets current path to the project folder
  - Modify Project* Opens the project, selected in the VFX 9.5 Task Pane and sets current path to the project folder.
  - Add Project* Adds an existing VFP-Project to the VFX 9.5 Task Pane.
  - Rebuild* Rebuilds all files in the project, selected in the VFX 9.5 Task Pane. After rebuild, the Project will be opened for edition.
  - Properties* Opens the VFX – Project Properties for the project, selected in the VFX 9.5 Task Pane.
  - Project Backup* Creates a Zip-file for selected Project.
  - Delete* Deletes the project, which is selected in the VFX 9.5 Task Pane.
  - Delete* Removes the selected Project from VFX 9.5 Task Pane.
- There are two new features in the VFX 9.5 Task Pane.

With a single mouse click can be created a backup copy of a project in a ZIP file. By clicking on  icon the backup will be started. If the project is open at this moment, it will be closed when backup start.

## 5. Installation

### 5.1. Hardware- and Software- Requirements

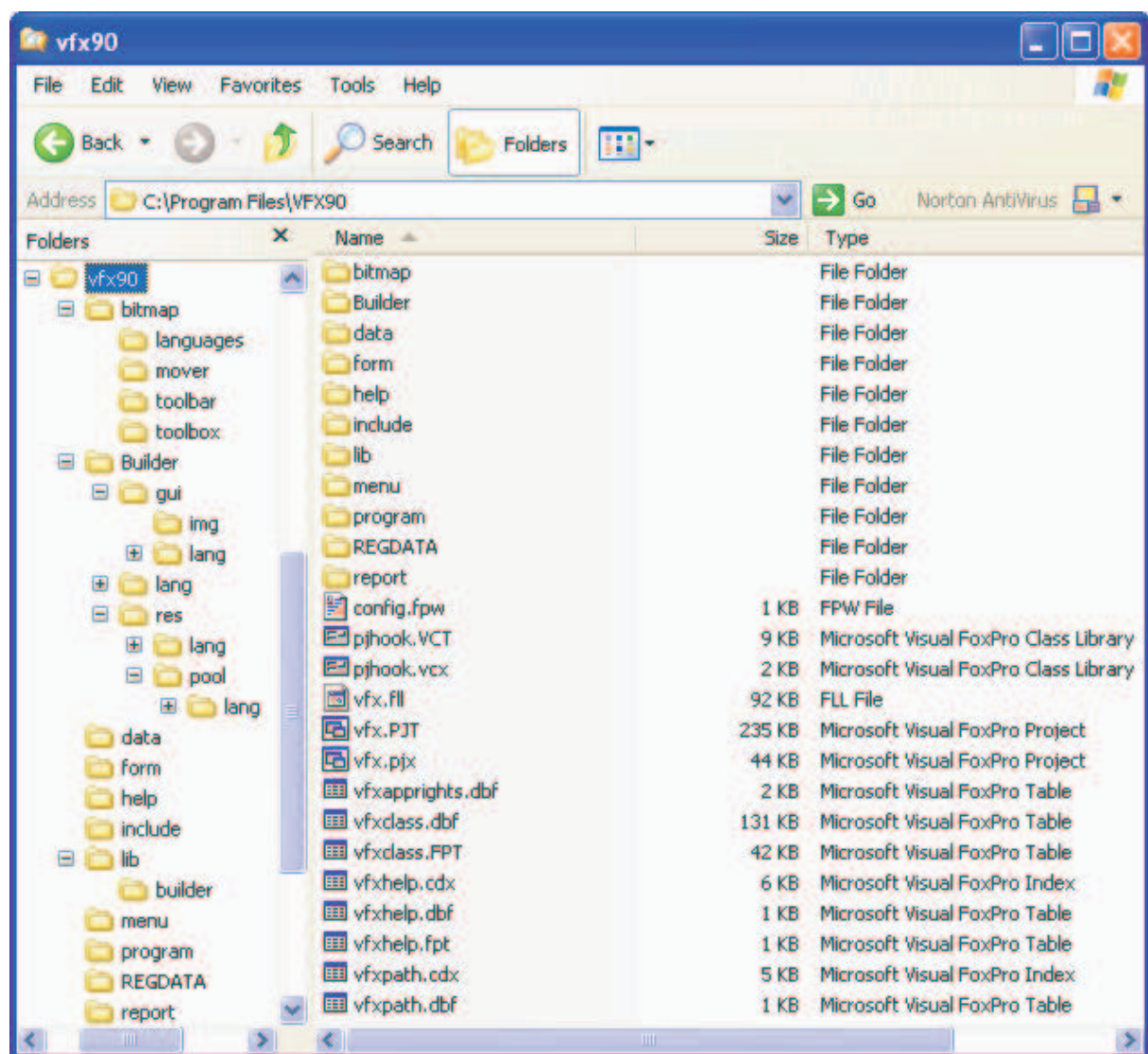
Since VFX is an extension to the Microsoft Visual FoxPro 9.0 Development System, you need a Hard- and Software Environment under which you can run Visual FoxPro 9.0. Please refer to the Visual FoxPro Hardware and Software Requirements for further information.

### 5.2. The VFX Installation

Run the setup program called *VFX90Setup.exe* and follow the instructions on the screen.

**Make sure to install VFX 9.5 with the installation program we provide into a new folder. Do not install VFX 9.5 in the same folder as earlier VFX versions!**

After the installation of VFX, you will have the following folder structure in VFX folder:



The VFX folder serves as a central storage for all VFX components and is a base for all Projects, created with VFX - Application wizard, as described later in this document.



---

**IMPORTANT:** Do not work directly in this project. It is NOT foreseen for direct use. Use the VFX - Application wizard in order to create a new project.

---

### 5.3. Registration and activation of VFX 9.5

As previous VFX versions, VFX 9.5 is secured with product activation. The VFX 9.5 activation is accomplished through a Web service. The advantage is that the activation key is sent directly to the developer's PC and manual operations for entering the key are avoided.

VFX 9.5 has a software copy protection. After VFX 9.5 installation, at first start of one of VFX builders or the VFX menu, a register dialog is invoked. Please, fill-in all required fields and click on the button "Register Online". Through the internet your personal data will be passed to a web service of VFX – Internet-Registration-Server. You will receive back from the web service an activation key, which will be saved on the hard disk of your computer. The activation key is valid for 30 days. During this time period you can test full VFX functionally.

In case the activation is not possible through the web service, you can order an activation key on the website

<http://www.visualextend.com>

Then you will receive the activation key, sent by e-mail.

When you use VFX 9.5 with a 30 days trial key, the remaining number of days will be shown in a dialog. By clicking on *Buy VFX* button, the Visual Extent website will be open and a license can be obtained online. After payment processing, you will receive an unlimited activation key per e-mail.

Note that you cannot copy the VFX installation from one PC to another without requesting a new Activation Key. Your Registration Key is based on your PC and is unique. Every VFX user will have a distinct unique Registration Key and therefore needs to register online on our web site, to get the Activation Key and to be able to work with the VFX Builders.

We hope that you appreciate the new Software based approach and welcome you to the next generation of VFX. The very best VFX ever!

### 5.4. Setup the Visual FoxPro Environment for VFX

You must have Microsoft Visual FoxPro 9.5 properly installed, before you can start working with VFX 9.5.

Next, you should ensure that the VFX 9.5-Menu is automatically loaded each time, when you start your Visual FoxPro 9.0. Start the application *Vfxmnu.app* directly from Windows Explorer or from VFP command window. The application *Vfxmnu.app* is placed in *Builder* subfolder of your VFX installation.

We suggest the following way:

Add this line in the file *CONFIG.FPW* in the VFP 9.0-folder:

---

**NOTE:** If you do not have a file named CONFIG.FPW just create an empty one with Notepad.

---

```
command = do (HOME() + "vfx.prg")
```

This row instructs VFP that the program VFX.PRG should be run when VFP is started. In the file VFX.PRG (create this file with Notepad and save it in VFP folder too) add the following row:

```
do C:\Program Files\VFX90\Builder\vfxmnu.app
```

We assume here that VFX is installed in folder C:\Program Files\VFX90\Builder. Adapt the path accordingly if necessary.

At first start of VFX – Menu, the following settings will be made automatically:

- **Builder:** points to the VFX – Application Wizard named VFXBLDR.APP in folder `\VFX90\BUILDER`.
- **Searchpath:** `\VFX90\BUILDER` will be added to the Searchpath.

At first start of VFP after VFX 9.0 installation, the VFX 9.5 Task Pane will be incorporated into VFP Task Pane automatically.

**Very Important: Make sure that your current folder is always in the folder of your application!** Use the command `cd ?` in the command window to quickly verify where you are and change on the fly. Even better: use the VFX Task Pane for easy project switching without the need to manually change any folders. If you are in a wrong folder, Visual FoxPro might use other Include Files or Class Libraries than the ones you expect.

**The best tool to switch from one project to another is to use the VFX 9.5 Task Pane.** You can open the Task Pane under the menu *Tools/Task Pane*. We recommend you to leave Task Pane to be started by VFP automatically. To accomplish this, choose the option “Open the Task Pane Manager when Visual FoxPro starts” in Task Pane Manager.

## 6. Generate a New Application using the VFX - Application Wizard

### 6.1. Objective

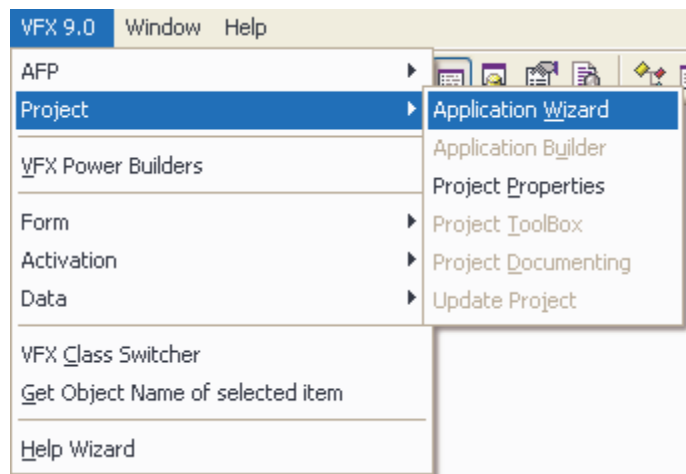
If you want to start a new project, you could set up the whole directory structure manually, copy all the needed support files, like the class library, some standard forms, some configuration files, bitmaps, and so on. That is where the VFX Application Wizard pays off: it does the whole set up of a new project automatically in the language of your choice. It also sets the most important application class properties and defines the most important include file constants to reduce to minimum your manual work.

### 6.2. Preparation

Close all forms and make sure you do not have open any class libraries of the VFX project. The best is to quit Visual FoxPro and restart it before you run the application Wizard.

### 6.3. The VFX - Application Wizard

Select the option *Project, Application Wizard* in the VFX 9.5-Menu.



Or start the *Application Wizard* from the VFX Task Pane by clicking on the icon link.



The VFX – Application Wizard appears:

**1. With this wizard you create a new VFX project**

Master VFX home folder: C:\PROGRAM FILES\WFX90\  
(Usually you don't need to modify this path.)

Enter the name of the new project file: MAIN

Enter the name of the new project's folder: |

Database name: DATABASE.DBC

Click on next to proceed.

Cancel < Back Next > Finish

The settings that are made in the VFX - Application Wizard are saved for usage in subsequent projects.

Enter the following information before you start to build your new application.

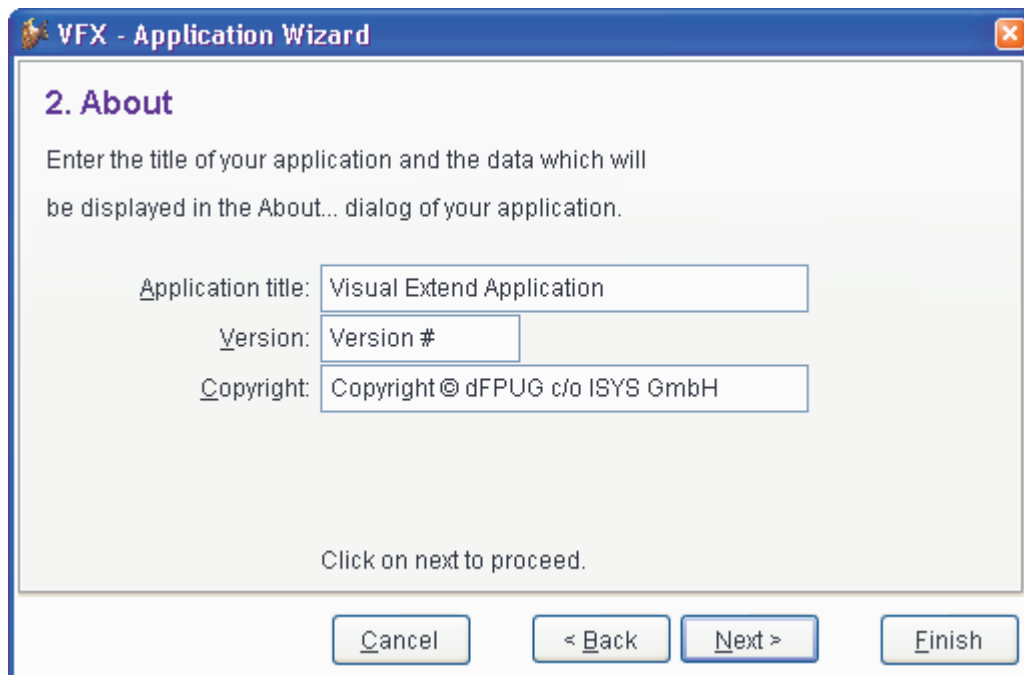
**Master VFX home folder:** Locate or type the VFX home directory where all VFX support files are located. Usually the default value is correct and you do not need to make any changes.

**Enter the name of the new project file:** Enter the Name for your new project file. Don't include any path and file extensions, just type the name of your Project.

**Enter the name of the new project's folder.** Locate or type the directory for your new project. If the directory does not exist, the VFX Application Wizard will create it for you. The standard path in which new projects are created is "My Documents\VFX Projects\". If another path is chosen for generated project, all subsequent projects are stored under this path too. By default is generated a project folder named *VFX APPLICATION*, followed by a sequential number.

**Database Name.** Enter the Name for your new Database Container (DBC). Type the name of your Database Container, without any path and file extensions. When your application will access a remote data source and will use only CursorAdapter for data access, you can leave this field empty.

On the page named *2.About* enter the following:



**VFX - Application Wizard**

## 2. About

Enter the title of your application and the data which will be displayed in the About... dialog of your application.

Application title: Visual Extend Application

Version: Version #

Copyright: Copyright © dFPUG c/o ISYS GmbH

Click on next to proceed.

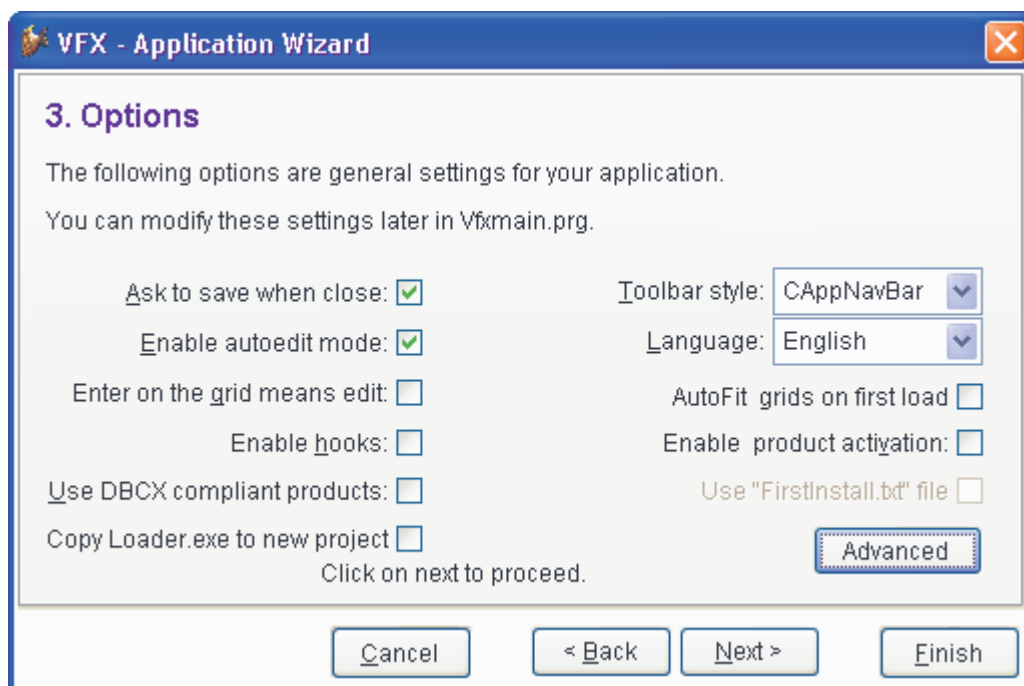
Cancel < Back Next > Finish

**Application title:** Enter the Caption for your Main Application Window. This caption will set the constant CAP\_APPLICATION\_TITLE in the include file USERTXT.H for you.

**Version:** Enter the Version Number used in the about dialog of your Application. This will set the constant CAP\_LBLVERSION in the include file USERTXT.H for you.

**Copyright:** Enter the Copyright Information used in the about dialog of your Application. This will set the constant CAP\_LBLCOPYRIGHTINFORMATION in the include file USERTXT.H for you.

On the page named *3.Options* you can set following options:



**VFX - Application Wizard**

## 3. Options

The following options are general settings for your application.  
You can modify these settings later in Vfxmain.prg.

Ask to save when close:       Toolbar style: CAppNavBar

Enable autoedit mode:       Language: English

Enter on the grid means edit:       AutoFit grids on first load

Enable hooks:       Enable product activation:

Use DBCX compliant products:       Use "FirstInstall.txt" file

Copy Loader.exe to new project       **Advanced**

Click on next to proceed.

Cancel < Back Next > Finish

**Ask to save when close:** Checking this option sets the Application Class property *nAsktoSave* to 1, which defines how VFX behaves when a user closes a form or moves to another record after having made changes to the current record.

**Enable autoedit mode.** Checking this option sets the Application Class property *nAutoEditMode* to 1, which defines that the user can start anytime to make changes without the need to change to the edit mode before any editing can occur.

**Enter on the grid means edit:** Checking this option sets the Application Class property *nEnterIsEditInGrid* to 1, which defines that the Enter key while in the data grid changes to the Edit mode.

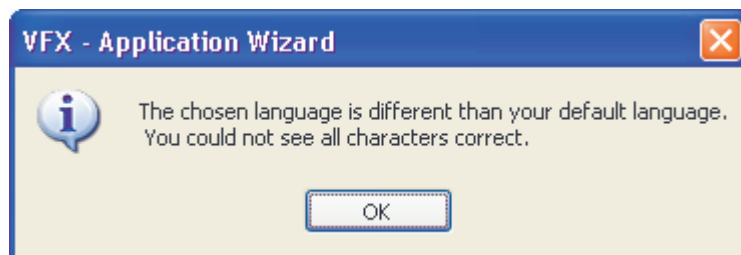
**Enable hooks:** Checking this option sets the Application Class property *nEnableHook* to 1, which defines that the hooks should be activated.

**Use DBCX compliant products:** Select this option when the Stonefield Database Toolkit will be used with the generated VFX application.

**Copy Loader.exe to new project:** Loader.exe will be used for application update on customer side through internet. Choose this option when you would like to customize the Loader project especially for your application.

**Toolbar style:** Select which toolbar Style Class you want to use. *CAppNavBar* includes the record navigation and other editing related buttons in the main toolbar. *CAppToolBar* does not include buttons for record navigation and not all editing buttons.

**Language:** Select the desired language for your new project. On selection of a language for generated application, VFX checks the current Unicode settings of the operation system. When characters of chosen language cannot be displayed correctly with current settings, a warning message is displayed.




**AutoFit grids on first load:** When this option is selected, the value of the property *nUseAutofit* of the Application object is set to 1. This means that by initializing of grids the *Autofit()* event is called.

**Enable product activation:** Checking this option sets the Application Class property *lUseActivation* to .T., which defines that the application will require product activation.

**Use "Firstinstall.txt" file:** Checking this option sets the Application Class property *lActivationType* to .T., which defines that the product activation will require the file "Firstinstall.txt". This will improve your application protection.

**Advanced:** This button opens the VFX – Application Builder, which offers a huge number of further setting of the application object. In the lower part of this dialog is shown a help text area with explanations for the selected property.


On the page 4. *Author*, you can enter your personal data to document new project.



**VFX - Application Wizard**

### 4. Author

The following information are stored in the project info.



Author:

Company:

Address:

City:  State:

Country:  Postal Code:

Click on finish to generate your project.

This Information will be written into the new generated project.

#### **6.4. Generate the project**

Select *Finish* and the VFX Application Wizard will create a new project according the parameters you selected. During this process the sample application from the VFX – Install folder will be copied into the new project folder. Also include files, correspondent to the chosen language, will be copied. Finally, will the entire project be compiled, so the include files will be included into the application. A final message shows that your new application has been successfully prepared.

---

**NOTE:** Since you may want to start working on your new project immediately, the VFX Application Wizard has automatically set the default directory to the home directory of your new project. To start the application from the project manager, locate the main program *VFXMAIN.PRG* and select “Run”.

---

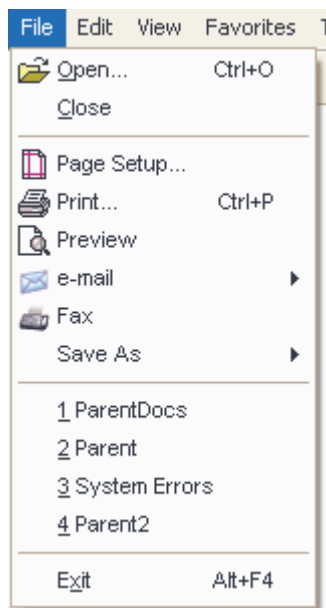
## 7. Discussion of the Generated VFX - Application

After a successful application generation using the VFX Application Wizard, you have a running Application with everything a new Application needs from the beginning. Starting from the Menu, the Standard Toolbar, User Management, System Options, Database Tools, System Error Tracking, System Locking Tools, as well as an About Dialog.

### 7.1. Office-Compatible User Interface

VFX creates applications which are ready to pass the *Office Compatible* Certification.

#### 7.1.1. File Menu



The complexity of the menus is reduced using the standard *File/Open* dialog. The user will open a form always through a common *Open Dialog* for which VFX makes a suggestion for the layout. By default the Open Dialog is displayed in Windows-XP style docked to the left of the screen.

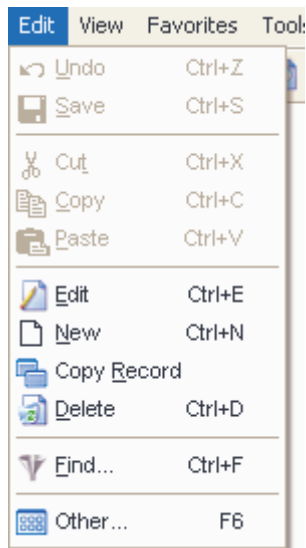
This default can be changed at any time by the developer to meet the application specific needs.

Following *Office-Compatible* Standards, VFX applications have a user-specific *Recently Used File List* where the last selections appear and are only one click away from being selected again. The number of entries in the list is individual customizable for every user in User management dialog

The *File/Exit* menu option conforms to the *Office Compatible* principles, too.



### 7.1.2. Edit Menu

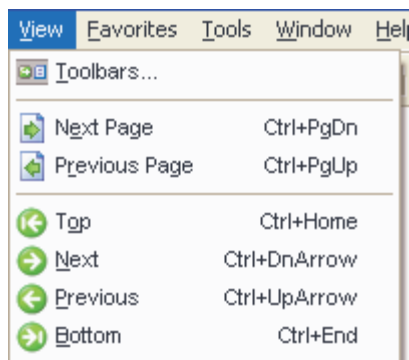


Here are placed all *Data Manipulation Functions*, which apply to the currently selected record as well as the possibility to call the *Find* and *Other Function* forms. Some menu options might be disabled, depending on the mode of the active form which can be either in

- Edit mode (`oForm.nFormStatus = 1`),
- Insert mode (`oForm.nFormStatus = 2`) or
- View Mode (`oForm.nFormStatus = 0`)

For detailed information regarding this functionality, please refer to the chapter *Discussion of the VFX Data Manipulation Form* later in this document.

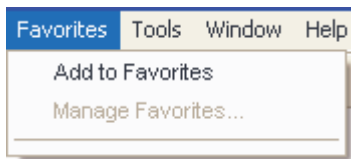
### 7.1.3. View Menu



Here you can customize your *Toolbar* as well as toggle the pages in a multi tab page dialog or simply *navigate* through the current set of records in a data manipulation form.

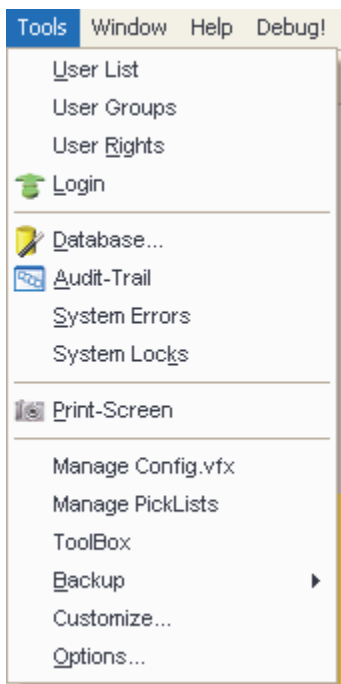
For detailed information, please refer to the *Discussion of the VFX Data Manipulation Form* chapter later in this Document.

### 7.1.4. Favorites Menu



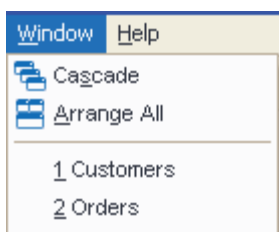
This is the VFX Favorites menu. The first option is to add the currently selected record to the favorite menu. The second is to manage the favorites. At the bottom all currently available favorites grouped by form are displayed as additional menu options at runtime.

### 7.1.5. Tools Menu



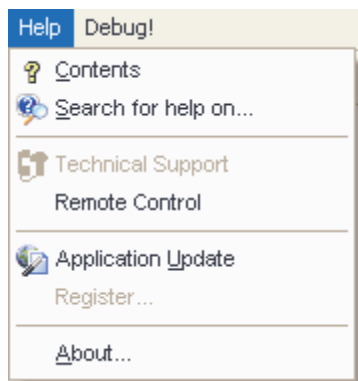
For further detailed information regarding the features described above, please refer to the chapter *User List*, *User Rights*, *Login*, *Database Tools*, *Audit-Trail*, *System Errors*, *System Locks*, *Print-Screen* and *Options Dialog* later in this documentation.

### 7.1.6. Windows Menu



If you have multiple windows open, you will see their form captions in the *Windows menu*.

### 7.1.7. Help Menu



The help-menu lets you search the help index of the help file.

### 7.1.8. Standard Office-Like Toolbar

VFX Applications have a standard toolbar on which you can easily put your own, application-specific toolbar buttons. This way, the users will have user-friendly way to access the functionality that your application offers. The VFX toolbar is displayed in „Hot Tracking“ Layout.



<i>New Record(Ctrl+N)</i>	Inserts a new data record
<i>Copy Record</i>	Currently selected data row will be copied into a new data row
<i>Open (Ctrl+O)</i>	Opens the Open-Dialog at the left side of the screen
<i>Save Record(Ctrl+S)</i>	Saves changes in the active form
<i>E-Mail</i>	Sends an E-Mail containing the report from the active form as attachment.
<i>Print (Ctrl+P)</i>	Prints a report or a list from the active form
<i>Print Preview</i>	Shows print preview for a report or a list from the active form
<i>Fax</i>	Sends a fax from the report output of the active form.
<i>Cut (Ctrl+X)</i>	Removes the selection and puts it into the clipboard
<i>Copy (Ctrl+C)</i>	Copies the selection into the clipboard
<i>Paste (Ctrl+V)</i>	Inserts the content of the Clipboard
<i>Undo (Ctrl+Z)</i>	Reverts the changes made in the active form
<i>More Function...(F6)</i>	Opens the window with More functions for the Active form
<i>Audit-Trail</i>	Opens the Audit-Trail form for the current data record in the active form
<i>Screenshot</i>	Prints the screen content.

---

<i>Edit Record</i> (Ctrl+E)	Switches the form into Edit mode
<i>Delete Record</i> (Ctrl+D)	Deletes the current record in the active form
<i>Search Record...</i> (Ctrl+F)	Filters the data in the active form according given criteria
<i>First Record</i> (Ctrl+Home)	Moves the record pointer to the first record of current table or view
<i>Previous Record</i> (Ctrl+Up Arrow)	Moves the record pointer to the previous record of current table or view
<i>Next Record</i> (Ctrl+Down Arrow)	Moves the record pointer to the next record of current table or view
<i>Last Record</i> (Ctrl+End)	Moves the record pointer to the last record of current table or view
<i>User</i>	Example for an application specific button
<i>Refresh</i>	Refreshes the view for active form, based on entered parameters for data selection.
<i>Help (F1)</i>	Calls the context sensitive Help
<i>Login...</i>	Allows another user to login while the application is running
<i>Close (ESC)</i>	Closes active form

Besides the standard toolbar, VFX also offers the possibility to define toolbars which are associated to particular forms. All you have to do is set up the toolbar and set the VFX form property *cToolbarClass* to the name of the desired toolbar. VFX handles the rest for you automatically.

---

**NOTE:** For a detailed technical description about the usage of form-specific toolbars, please refer to the separate VFX Technical Documentation.

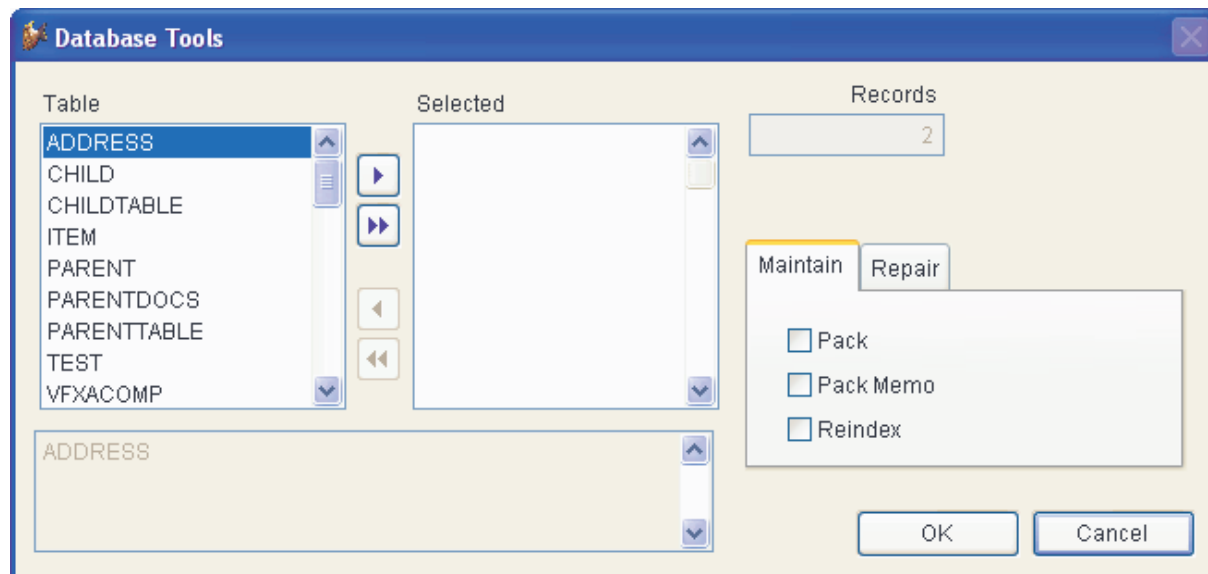
---

### 7.1.9. Final words about Office - Compatibility

Depending on the type of your application, the level of Office Compatibility might differ from what is suggested here. Look at the VFX menu as one alternative which will cover most, but definitely not all, possible applications with their special needs. It's definitely worth investing some time to prepare the right decision about the menu and toolbar user interface you plan to use in your application.

## 7.2. Database Tools

By selecting the Menu Option *Tools Database*, you will see the following dialog:



In this dialog you see a list of tables, used in your application. In a user-friendly *VFX Mover Dialog*, you can select the tables you want to process.

You can select from the following options:

- Pack
- Pack Memo
- Reindex

Select *OK* to run the desired database maintenance action for the selected tables.

---

**NOTE:** If you like the above Mover Dialog, you will be happy to hear there is a VFX Mover Class which allows you to easily integrate Mover Dialogs into your own applications!

---

In addition to the database maintenance, VFX 9.5 includes a new tool for repairing corrupted data. In the *Database tools* dialog form, now is available additional option- *Repair*

Optionally can be repaired either the entire database on only selected tables. When only the selected tables will be repaired, additionally can be specified to repair only table headers or to delete corrupted records.

For repairing DBC container and DBF headers, an empty database with same structure as the operational one is used. When the executable file is built, the structure of database is included in it using a generated program file, created using *Gendbc.prg*. Later at run-time an empty database is created using this program and is used for automatically repair of database container and DBF headers of the operational database.

If *Delete corrupted records* option is chosen, all records with empty or duplicated primary key values will be deleted from tables.

## 7.3. User List

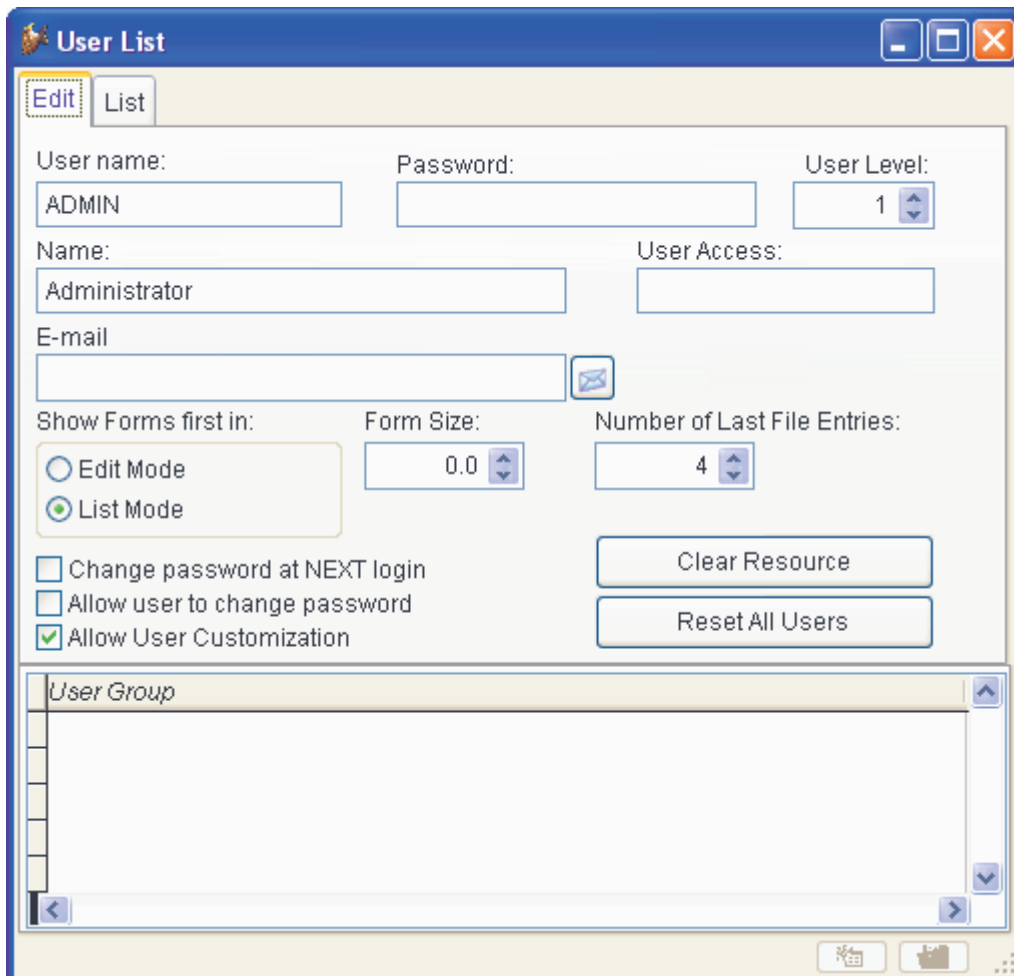
In every multi-user application, you must have a user list. First of all you need to define who has access to your application, which is the username and password and what the security level per user is.

The table in which user-specific information is stored, is the free table *VFXUSR.DBF/CDX*. If you want to take advantage of features like long field names, you have to add this table into the database container.

Users can delete their own records in VFX resource table, if they want to start over with new settings, or if they are switching from a larger display resolution to a smaller, or simply if they aren't satisfied anymore with their user preferences for forms, grids, sort orders and Picklists. To clear the VFX resource table, click on the command button called *Clear resource*.

VFX 9.5 provides new advanced features for managing application users. Now the administrator can reset the resources for all users with the button "*Reset all users*".

For every user the administrator can specify that the password must be changed with the next login. The administrator can also setup that a particular user is not allowed to change his password.



It is also possible now, to give users better flexibility in setting environment. By setting to `.T.` the property `!AllowUserCustomization` of `goProgram` object, the developer can allow end-users to define their own global environment.

```
goProgram.!AllowUserCustomization=.T.
```

When this property is set to `.T.` the end-user administrator can make global customization and in turn, enable or disable customization for all other users. The checkbox "*Allow User Customization*" is used to control whether other users are allowed make their own environment setting. If this property is set to `.F.`, the checkbox "*Allow User Customization*" is not visible and applications environment cannot be customized.

If the end-user administrator disables user customization option, the settings defined under Administrator's account become global settings for all users of the application.

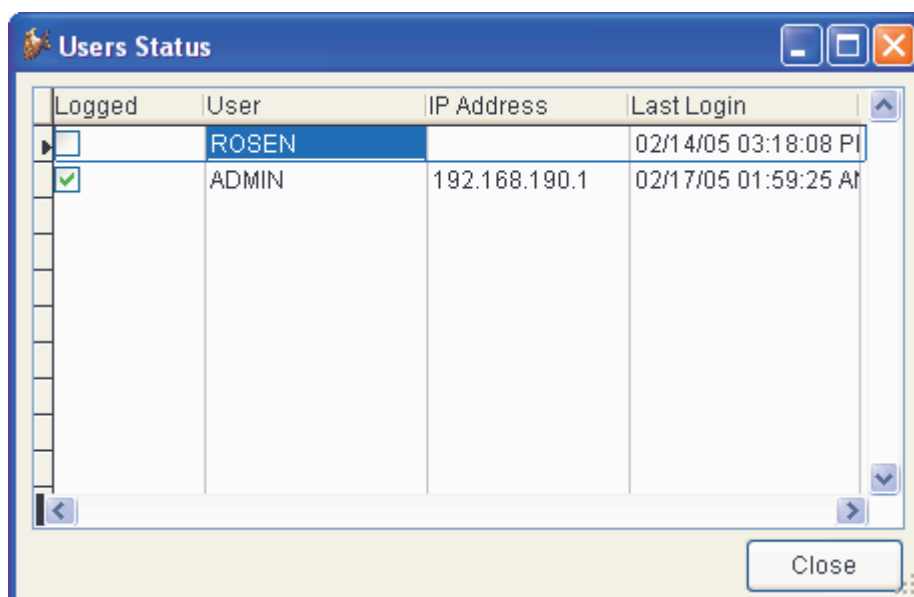
### 7.3.1. Currently logged users

VFX 9.5 keeps track of logged users in a table with possibility to block users from login more than once at a time. The property of goProgram object, *lAllowMultipleLogin* controls the behavior of the application when at login time appears that the user is already logged in the system. When this property has a value *.T.*, users can login more than once at a time. Default value is *.F.*

```
goProgram.lAllowMultipleLogin=.T.
```

For every user is kept the IP address of the workstation where he logged in. After user logs out, IP address field is cleared.

Users with administrator rights can see currently logged under menu *Tools/ Users Status*. IP address and login time are displayed. The column last login time always shows the last date and time when user had logged in, even if the user is not currently logged in.



Depending of where VFX system tables reside, this feature uses different approach to identify if user is really active now or for some reasons his connection had broken and login attempt is to re-enter the system, but not to login simultaneously twice.

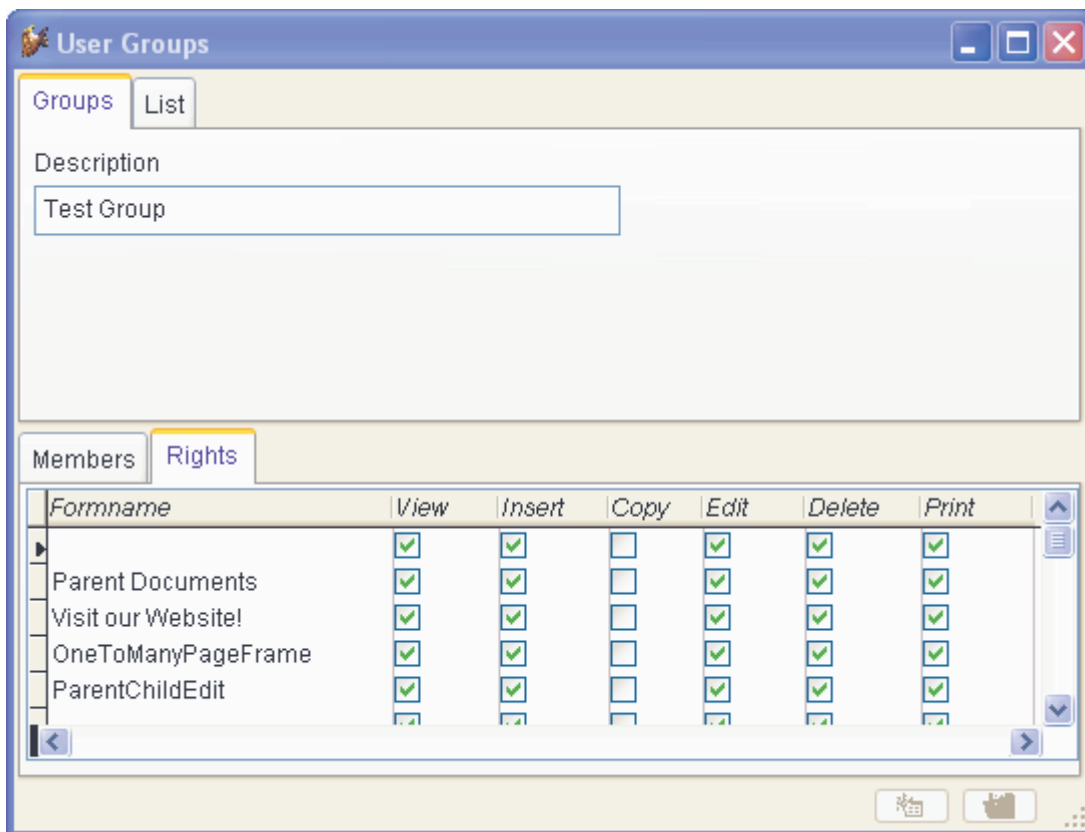
When DBF tables are used, the record for the user is locked all the time while he is logged in the system. In case of broken connection or another emergency exit, record lock is automatically released and when user attempt to login next time this action is not threat as multiple login.

When VFX tables reside in a SQL Server database, the system process Id is used to identify logged user. The active *SPID* is kept in *vfxusr* table. On a login attempt is checked if the user is already logged in and if that process is still active. In case of active process and if multiple login is disabled by the developer, user is not allowed to log in the application.

### 7.4. User Groups

Along with former possibilities to define user rights, VFX 9.5 offers a new way to manage user access to application's modules using User Groups. A particular user can be member of one or more user groups. For user groups can be defined access rights. When the user is member of more than group rights which every one of these groups permits are combined and the user are granted rights of all user groups that he is member of.

Users with administrator rights (userlevel 1) can define user groups and specify access rights to every user group for every form in the application. User rights are defined for all forms, included in *Vfxfopen* table.



At run time is created a global object *goUserRights*. This object contains child objects for every form of the application. Names of these objects correspond to the form name. Every child object has properties corresponding to user rights for the particular form *deletepermit*, *editpermit*, *newpermit*, *printpermit* and *viewpermit*.

At run-time the properties of the object *goUserRights* and its members will look in this way:

Name	Value	Type
[-] goUserRights	(Object)	O
[-] frminvoices	(Object)	O
deletepermit	.F.	L
editpermit	.T.	L
newpermit	.T.	L
printpermit	.T.	L
viewpermit	.T.	L
[-] frmorders	(Object)	O
deletepermit	.T.	L
editpermit	.T.	L
newpermit	.T.	L
printpermit	.T.	L
viewpermit	.T.	L

If a particular user is not a member of any user group, access rights are defined by the user’s security level as in former VFX versions.



Title	Form	Viewlevel	Insertlevel	Copylevel	Editlevel	Deletelevel	Printlevel
Parent	parent	1	0		0	0	
Child	child						
Item	item						
OneToMany	oneto						
OneToMany 2	oneto2						
Parent Tree	parenttree						
One to Tree	onetotree						
Parent Documents	parentdocs	0	0		0	0	
Business Graph	BUSINESSGR	6	4		3	2	
Parent2	parent2						
Delayed	delayed						
Audit-Trail	audit						

The administrator has the user security level 1 and thus all rights. A user, who has the user security level 99, has the few rights. In the form user rights can for each form can be specified which user security level is necessary in order user to be allowed to run the form, to insert new data records, to edit existing data records and to delete data records.

---

**NOTE:** Users cannot view or alter other user accounts with a higher security level than their own. Security levels starts with 1 (Administrator) and ends with 99 (lowest security level). Additionally, you can define an access string for further customization of your security needs. For additional security issues, especially all the VFX form security features, please refer to the VFX Technical Documentation.

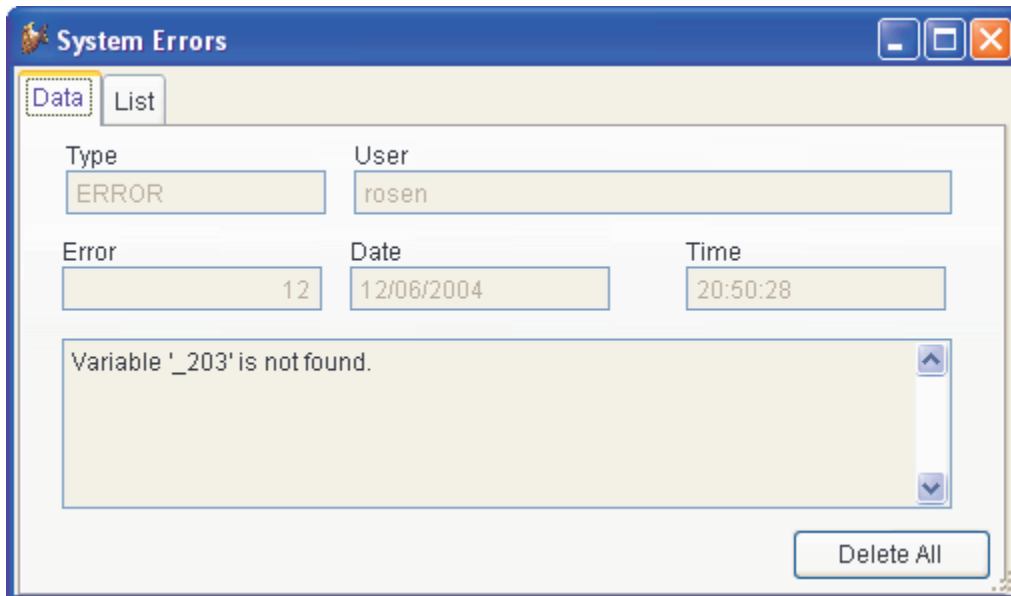
---

When a particular user does not have rights to run a form, the concerned form will not be initialized. If user rights are not defined in the user rights dialog, the properties values *lCanInsert*, *lCanCopy*, *lCanEdit* and *lCanDelete*, that were set within the VFX - form Wizard are valid.

## 7.5. Error tracking

VFX tracks all runtime errors automatically. The error log file, in which all the runtime errors will be stored, is the free table *Vfxlog.dbf/cdx*.

The data manipulation form based on the class *CDataFormPage*, will be prepared automatically from the VFX Application Wizard.



The administrator can clear this list by selecting the command button called *Delete All*.

---

**NOTE:** For additional information, please refer to the VFX Technical Documentation.

---

## 7.6. Error handling

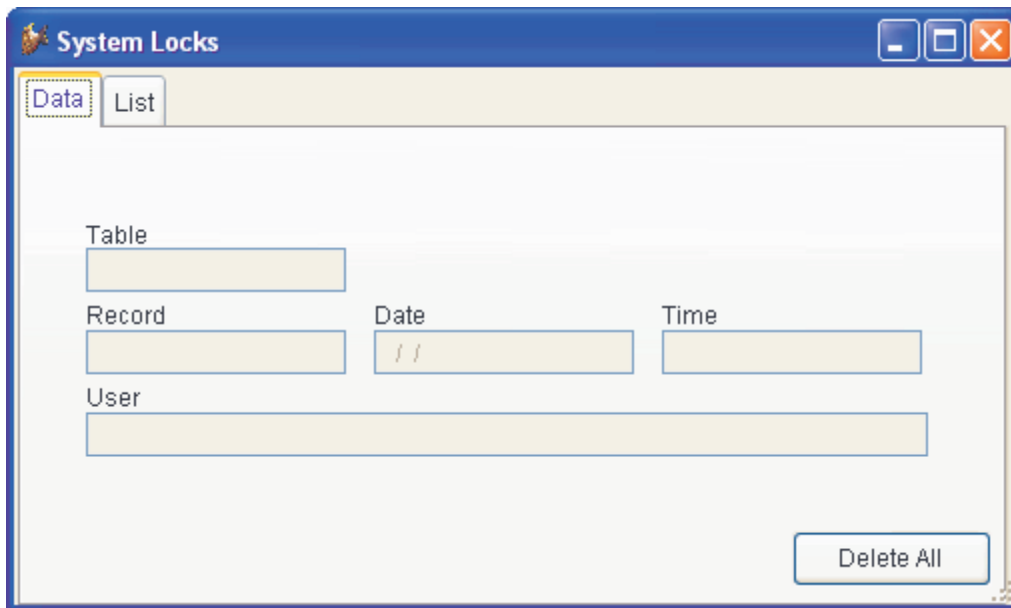
In VFX 9.5 is implemented an extended error handling. In addition to creating an error log, it is now possible for end user to send an error report e-mail to the developer. When in the error message dialog user selects Abort, the error report dialog is invoked informing the user what is included in generated report. Sending an error report to the developer is the fastest way to localize and eliminate problems in the application. The e-mail is sent to the e-mail address, set in the property of *goProgram.cSupportEmail*. The value of this property can be set using VFX –Application Builder.

## 7.7. System Locks

In heavily used multi-user applications, a message like “*Record is in use by another user*” might simply not be enough. VFX offers a System Locks table for such cases. In the table will be saved exactly which user, which data record and since when has been locked. (You can use the Functions *XLock()* and *XUnlock()*, described in the Technical Reference under *Functions*)

The System Locks table in which all the system locks will be stored, using standard VFX Function calls, is the free table *Vfxlock.dbf/cdx*.

The data manipulation form based on the class *CDataFormPage*, will be prepared automatically from the VFX Application Wizard.



The administrator can clear this list by clicking on the command button called *Delete All*.

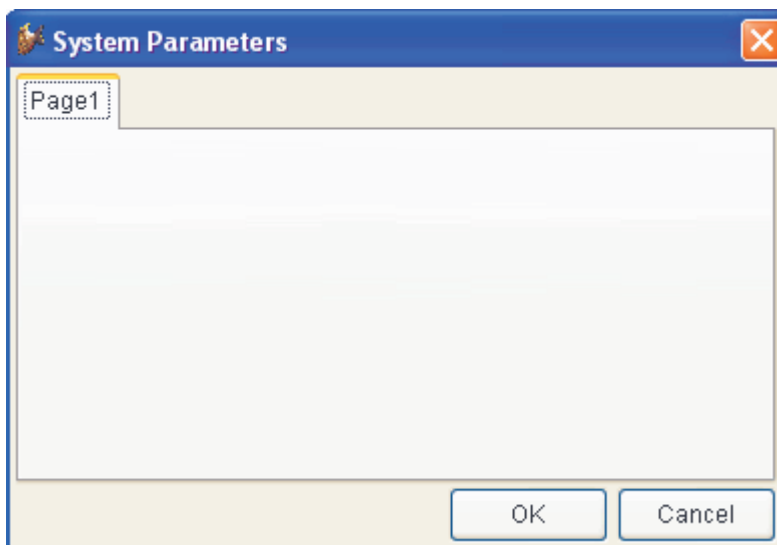
---

**NOTE:** For additional information, please refer to the VFX Technical Documentation.

---

## 7.8. Options

VFX offers a table called *Vfxsys.dbf* which stores application-specific settings.



The above form is just an example of an application specific System Options Dialog.

VFX Application Wizard creates a form *Vfxsys.scx*, which is ready to use. This form inherits the class *CSystemDialog*. All you have to do is to create desired fields in the *Vfxsys.dbf* table and to place the corresponding controls on the above dialog with the Control Source Property pointing to a property of the object *goSystem*.

VFX creates for every field in the *Vfxsys.dbf* table, a property of the object *goSystem* and handles automatically the save & restore of these values.

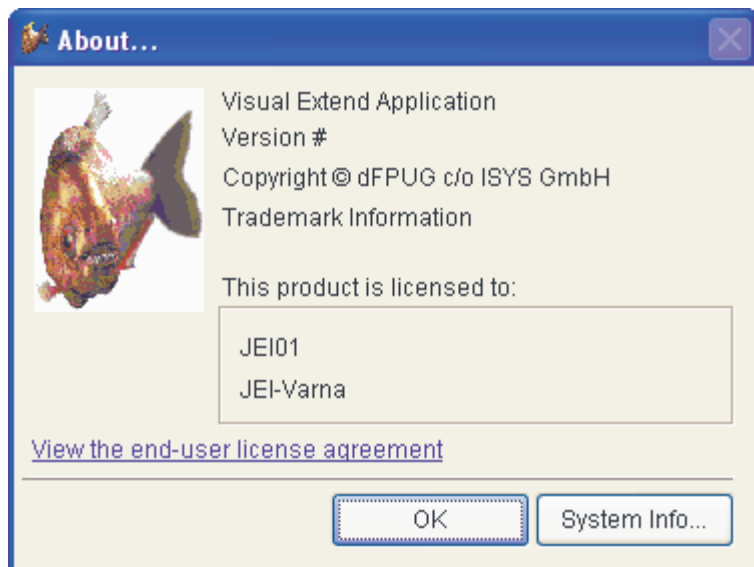
Assume you have a field called *Test* in the *Vfxsys.dbf* table. In that case will be created a property named *Test*, which will pick up the value from the field *Test* from table *Vfxsys.dbf*. In case of changing value of this property by closing of Options dialog, this value will be saved back into the field *Test* in *Vfxsys.dbf*.

In this way, it is easy to store and retrieve application-specific settings. Try it out!

## 7.9. About Dialog

VFX Application Wizard creates an about dialog which inherits the class *cAboutDialog*.

Select the about dialog under the menu option *Help About*.



To customize this About Dialog, VFX offers you the possibility to make the changes in the Include File *Usertxt.h*:

```
...
#define CAP_APPLICATION_TITLE "VFX 9.50 Build 0000 Test Application"
#define CAP_LBLCOPYRIGHTINFORMATION "Copyright © dFPUG c/o ISYS GmbH"
#define CAP_LBLTHISPRODUCTISLICENSEDTO "This product is licensed to:"
#define CAP_LBLTRADEMARKINFORMATION "Trademark Information"
#define CAP_LBLVERSION "Version "
#define CAP_LBLYOURAPPLICATIONNAME "VFX Test Application"
...
```

**NOTE:** When you make changes in this include file, you must open and save or compile the form *vfxabout.scx* before starting your app, otherwise the changes in the include file might not go through and you still see the old text.

## 8. The VFX Builders

The VFX Builders help the developers in creating and editing forms, grids and pick fields.

Defining a form can be time consuming, especially if you have many forms with many Fields to be displayed. Putting 20 Fields on a form forces you to put 40 objects, the *Textbox*, or any other control, plus normally a *Label*. If you use a Foundation Class Library, you have to customize your Toolbar, or drag the desired control from the class library and drop it on the form. With the Visual Extend Form Builders this task has become very quick and simple.

**Another great benefit of the VFX – Form Builders is that they are fully reentrant.** This means that you can use them to populate changes you made in your database container automatically by just reapplying the builder and check the Use DBC Definition option. Also adding pages to the pageframe or changing the grid columns is very easy with the reentrance feature of the VFX – Form Builders.

Please refer to the Chapter *Form User Interface* later in this document, to get an idea about the User Interface of the standard data manipulation forms created using VFX..

First of all, you need to setup the database container for your application. Create your tables, fields and index keys.

---

**NOTE:** If you put the information for the Field Captions, Format, Input Mask and Display Class Library in the database container, these captions will automatically be used by the VFX form and grid builders.

---

As we have already seen, new project is generated with VFX – Application Wizard. This wizard can be started by clicking on the link icon in VFX Task Pane or from VFX 9.5 Menu

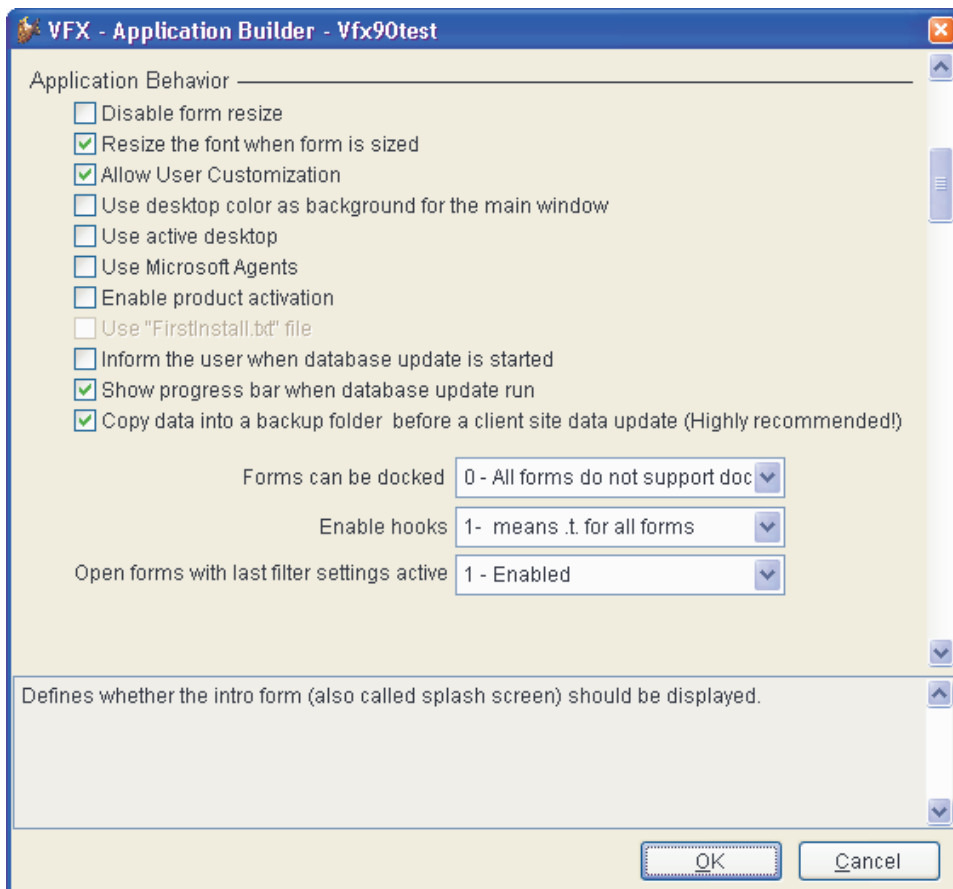
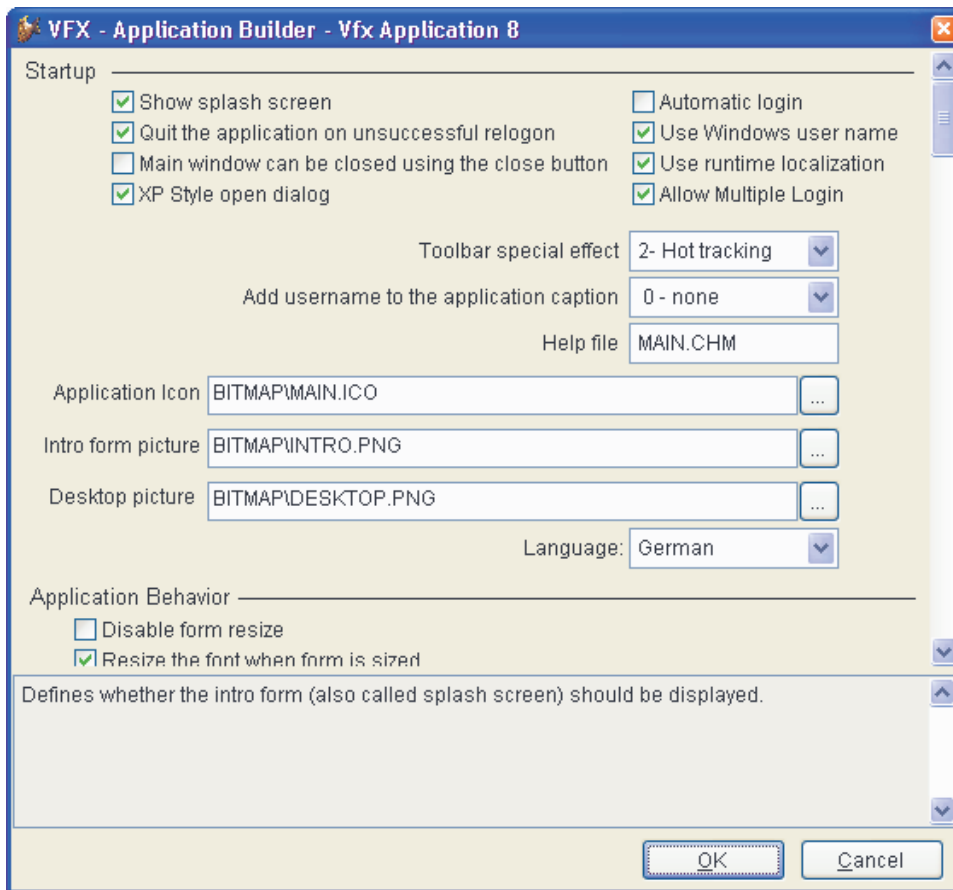
### 8.1. VFX - Application Builder

This dialog can be invoked at any time from the menu option *Project, Application Builder*, to edit settings of the application object.

---

**NOTE:** Changes that are made using VFX - Application Builder will be kept and used for next new generated project.

---



**VFX - Application Builder - Vfx90test**

Error handling

Error processing 1- show error message

Log error details 1 - Write only Call stack

Edit

Show century in date fields Null is valid value 0 - Use Contr

Century for rollover 20 Always ask prior any save operation 1 - Enabled

Year for rollover 49 Hide controls when table is empty 0 - Nothing

Date format GERMAN Autoedit mode 1- Force to .t

Hide list page while editing 0 means "use form property Idonthidelistpage"

Move the focus to the next object on Enter key for cCheckBox

Refresh all pages before the form valid event on Save

Allow to delete child data even if the deletion of parent records is not allowed

User is allowed to send BCC E-Mail

Name of the field in any table to be automatically used to store the "user" who inserted this record ins\_usr

Defines whether the intro form (also called splash screen) should be displayed.

OK Cancel

**VFX - Application Builder - Vfx90test**

Name of the field in any table to be automatically used to store the "user" who last modified this record edt\_usr

Name of the field in any table to be automatically used to store the "date" when this record has been inserted ins\_date

Name of the field in any table to be automatically used to store the "last edit" date edt\_date

Name of the field in any table to be automatically used to store the "time" when this record has been inserted INS\_TIME

Name of the field in any table to be automatically used to store the "last edit" time EDT\_TIME

Specifies the source table name for Auto Complete data. vfxacomp.dbf

OLE Drag&Drop

Enable OLE drag from pages of pageframes

OLE drop operation switches the form into editmode

Initialize OLE drag in any control

Disabled (Default).  Enabled  Pass to Container

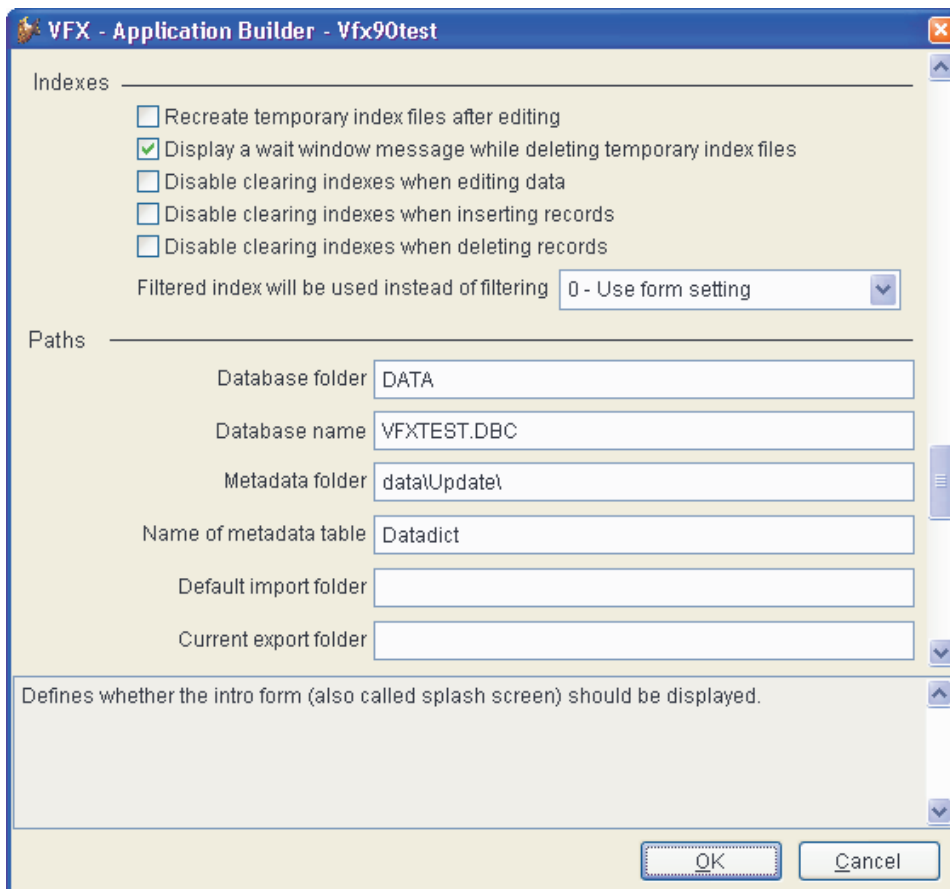
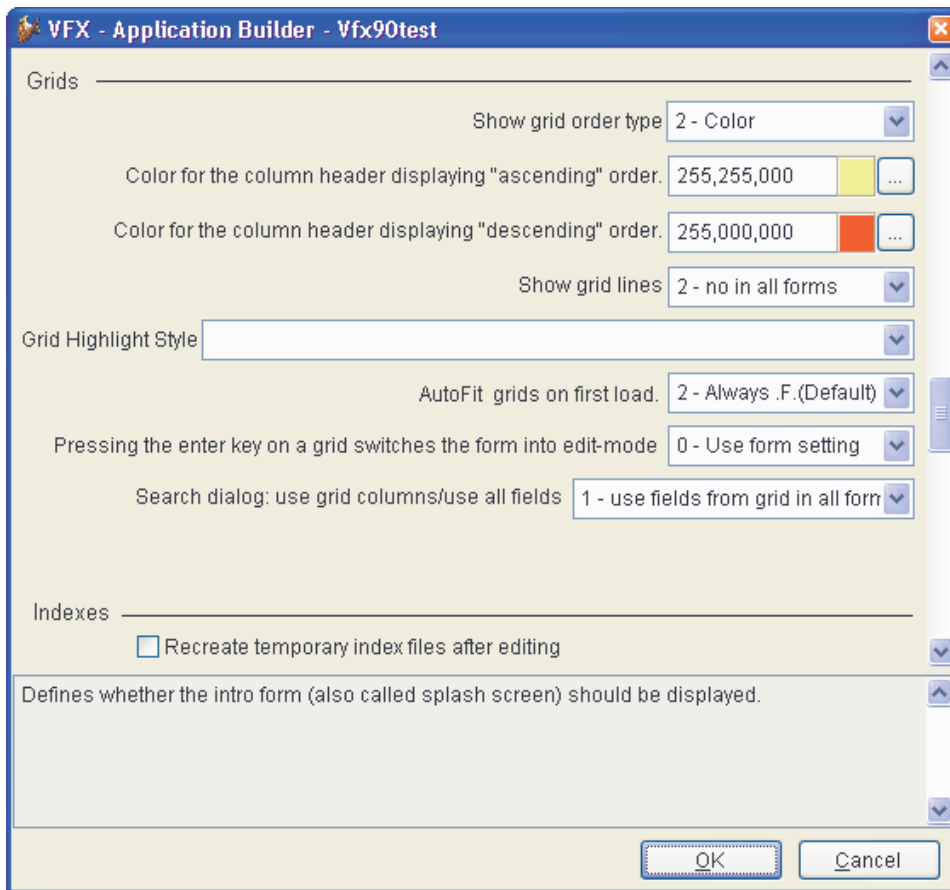
Disabled (Default).  Enabled  Pass to Container

Disabled (Default).  Enabled  Pass to Container

Defines whether the intro form (also called splash screen) should be displayed.

OK Cancel





**VFX - Application Builder - Vfx90test**

Default import folder

Current export folder

Path to the external report files (\*.vfx)

Save Export files folder per user

Misc

Name of Postscript printer to be installed when necessary

Name of Fax printer driver to be used when sending fax reports

URL to be used when check for internet connection existence

Password to be used for encrypting config.vfx file

Support URL

Support e-mail

Defines whether the intro form (also called splash screen) should be displayed.

**VFX - Application Builder - Vfx90test**

Password to be used for encrypting config.vfx file

Support URL

Support e-mail

Author

Author:

Company:

Address:

City:

State:

PostalCode:

Country

Support E-mail Address

The class *cApplication* is the base class of the Application object. The properties and Methods of the Application objects are globally accessible for use in the entire application.

The class *cApplication* is programmatically instanced from the visual class *cFoxAppl* from the class library *Appl.vcx*. Settings, keyed up using VFX – Application Builder, are saved in this class. In case of need, methods can be overwritten or changed here, too. The properties of the Application object that are, in particular, important for the control of the application must be defined here.

*cAscOrderRGB* – RGB-Value of the color of the Grid column header, which will be used to indicate ascending sort order in a Grid Column. By default this value "*RGB(255,255,0)*".

*cDataDir* – The path, where the database is placed. By default this path is loaded from the constant *datapath\_loc* defined in the include file *Userdef.h*. Leave this property empty, when you want to use Multi-Client-Database feature of the VFX.

*cDateFormat* – The date format, which will be used by default. The value of this property will be passed as a parameter to the command SET DATE. Normally, the value of this property is set in the method *SetLangID* of the Application object according to the chosen language.

*cDescOrderRGB* – RGB-Value of the color of the Grid column header, which will be used to indicate descending sort order in a Grid Column. By default this value "*RGB(255,255,0)*".

*cEdt\_Date* – The name of a field in any table. If a data record is saved and a field with this name exists in the table, the date and if necessary the time of the operation are stored here. The type of the field can be *Date* or *Datetime*. The default value is a field named *edt\_date*.

*cEdt\_Usr* – The name of a field in any table. If a data record is saved and in the table exists a field with this name, the of the user, that made changes in the database, is stored here. The type of the field must be of *Character* type. The default value is a field named *edt\_usr*

*cExcludeFiles* – Here one can enter a comma separated list contained file names. The files specified here do not appear in the database maintenance dialog and are excluded from data base maintaining. By default this value is  
*"DBCXREG.DBF;CDBKMETA.DBF;SDTMETA.DBF;SDTUSER.DBF;COREMETA.DBF"*

*cHelpFile* – The name of the help file that will be opened when the key F1 is pressed. The file extension (*chm* or *hlp*) should also be given. The default value is the name of the name of the project with the extension *chm*.

*cIns\_Date* – The name of a field in any table. If a new data record is saved and in the table exists a field with this name, the date and if necessary the time of the operation are stored here. The type of the field can be *Date* or *Datetime*. The default value is a field named *edt\_date*.

*cIns\_Usr* – The name of a field in any table. If a new data record is saved and in the table exists a field with this name, the of the user, that made changes in the database, is stored here. The type of the field must be of *Character* type. The default value is a field named *edt\_usr*.

*cIntroBitmap* – The name of a graphical file that will be displayed as splashscreen. All graphical formats, supported by VFP can be used, for example *bmp*, *jpg*, *gif* or *png*. The default value is *Bitmap\Intro.png* and is read from the include file *Userdef.h*.

*cIntroForm* – The name of a form class, the will display the splashscreen. Change of this value is necessary only if a Splashscreen with special properties will be used. The default value is *cSplashDialog*.

*cLoginForm* – The name of a form file that contains the login dialog. Change of this value is necessary only if the user administration of VFX will not be used is and for the user administration is used your own procedure. The default value is *Vfxlogin.scx*.

*cMainDatabase* – The name of the database. The value is read from the *database\_loc* constant from the include file *Userdef.h*. The default value will be set by the VFX – Application Wizard, while creation of the project.

*cMainForm* – The name of a form that must be invoked when the application starts, after the user login. The default value is an empty string.

*cMainIcon* – The icon of the application. By default this icon is used in all forms in the application. The default value is *Bitmap\Main.ico* and will be read from the constant *mainicon\_loc* from the include file *Userdef.h*.

*cMainTitle* – The name of the application, displayed in the title bar of the application. The name of the application can be also defined in the command *CREATEOBJECT(„capplication“, <Application Name>)* by passing the name as second parameter. In this case the value of the property will be overwritten.

*cMainToolbar* – The name of the standard toolbar. The default value will be set with the VFX – Application Wizard while creating the project, VFX supplies two toolbar classes to be used. The class *cAppToolbar* does not contain navigation buttons for moving the table record pointer in the form. The class *cAppNavBar* contains navigation buttons for moving the table record pointer in the form.

*cVfxpath* – In this property can be entered the name of the table, where is contained the information for paths for the databases used by the application. The default value is *Vfxpath.dbf*.

*FileMnuName* – In this property will be entered the name of the menu pad „File“. The name might not correspondent to the displayed caption. The name of the pad *File* that is set in the *Vfxmenu.vmx* will be used. The name must be known from the Application object, as far as at run-time to this menu pad are added entries, correspondent to the latest used forms.

*FileMnuOffset* – This is the number of the entry in the menu pad „File“, that will be used for the first entry added for the latest used form. When new entries are added, in the menu File, the value of this property must be changed accordingly.

*lAllowDeleteChildData* – When the value if this property is set to *.T.*, the user that cannot delete any records in the OneToMany-Form, can however delete child data records, When this value is set to *.F.*, cannot be deleted any child data records.

*lAutoLogin* – When the value if this property is set to *.T.*, the users, that is registered in user administration, will be automatically logged in on the application start, without a requirements to enter a password. The default value of this property is *.F.*

*lCentury* – When this property value is set to *.T.*, all date fields in the application will display the year using 4 digits. The default value is *.F.*, Year will be displayed within two digits.

*lDisableFormResize* – When this property is set to *.T.*, it is not possible to change the size of the forms. The default value is *.F.*, the size of the forms can be changes by the user.

*lNoClearIdxOnDelete* – By default VFX deletes temporarily created index files, when a data record is deleted. Set this property value to *.T.* if temporarily created index files should not be deleted in this case. Please, note that temporarily index files should not stay opened when a transaction is started. The default value is *.F.*

*lNoClearIdxOnEdit* – By default VFX deletes temporarily created index files, when a data record is edited. Set this property value to *.T.* if temporarily created index files should not be deleted in this case. Please, note that temporarily index files should not stay opened when a transaction is started. The default value is *.F.*

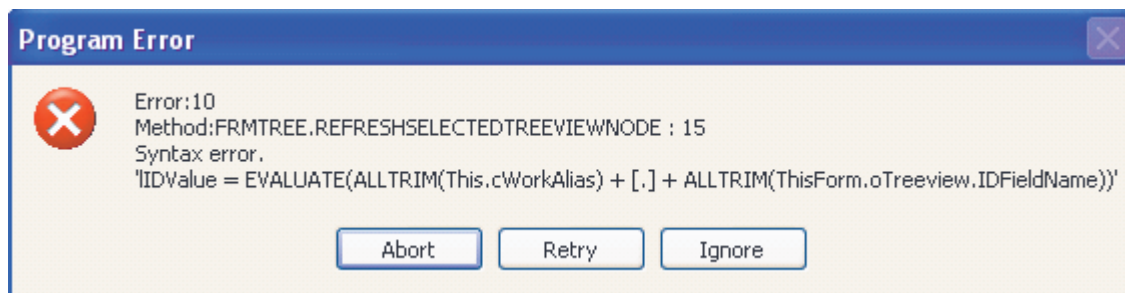
*INoClearIdxOnInsert* – By default VFX deletes temporarily created index files, when a new data record is inserted. Set this property value to *.T.* if temporarily created index files should not be deleted in this case. Please, note that temporarily index files should not stay opened when a transaction is started. The default value is *.F.*

*IRelogonQuit* – Controls the behavior of the application when an user tried to be changed, while the application runs and the login is denied. When the value of this property was set to *.T.*, the application will be terminated. If the value of this property was set to *.F.*, the last logged user stays logged in.

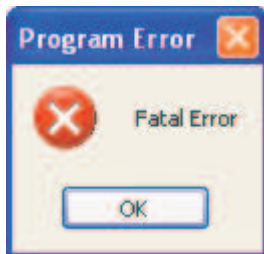
*IRemakeIdxAfterClear* – When the value of this property was set to *.T.*, the temporarily created index files will be automatically recreated after the save procedure finishes. Compare also with the properties *INoClearIdxOnDelete*, *INoClearIdxOnEdit*, *INoClearIdxOnInsert*. The default value is *.F.*

*nAppOnErrorBehavior* – This property controls the behavior of the application in case of error

- 0 – Run-time error will be ignored
- 1 – An error message will be displayed (Default).



- 2 – After a message is displayed, the execution of the application will be terminated.



*ErrorDetailLevel* – This property controls what information in case of error, will be tracked in the table *Vfxlog.dbf*

- 0 – Error message only, without any information concerning the call stack.
- 1 – The error message and the call stack information (Default).
- 2 – Complete, detailed error information.

*PSPrinterToInstall* – This property specifies the name of the standard Postscript Printer driver which will be automatically installed, when a Postscript Printer driver is not installed, and the application needs a Postscript Printer driver to create a PDF file. Default value "HP DeskJet 1200C/PS"

*cConnectionCheckURL* – This 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. If necessary, GhostScript will be downloaded and automatically installed. GhostScript is used to transform Postscript file into PDF file. If neither an internet connection nor a dial-up connection exists, a dial-up connection will be established from VFX. All properties of the dial-up connection can be predefined by the developer. If necessary, user will be asked to change the phone number to be dialed, user name and password to access the network resources.

*IUseActivation* – Through this property the product activation can be switched on or off. The property can be set in the VFX Application Wizard when a new project is created. Later its value can be changed in the class *cFoxAppl* in the class library *Appl.vcx*. The default value is *.F.*, the product activation will not be implemented.

*IActivationType* – When this property is set to *.T.*, the class *cVFXActivation* class will check existence of the *FirstInstall.txt* file when the application starts. This property can be set in the VFX Application Wizard when a new project is created. Later its value can be changed in the class *cFoxAppl* in the class library *Appl.vcx*. The default value is *.F.*, the existence of the file *FirstInstall.txt* will not be checked.

*cConfigPassword* – Password used for encryption of *Config.vfx* file. This password is used for security purposes, so the connection information to the datasource such as server name, user name and password being kept encrypted.

*cFaxPrinterName* – The name of Fax printer driver to be used when sending reports by fax. When this property is empty, VFX tries to find a printer driver with name, containing “Fax”. With higher priority are used printer drivers Winfax and Fritz!fax.

*cMetadataTableName* – Name of table where metadata definitions are kept. This table is needed for actualization a SQL Server database. The default value is *Datadict*.

*IAllowMultipleLogin* – When this property is set to *.T.*, application allows same user to log in simultaneously from different workstation. The default value is *.T.*

*IAllowUserCustomization* – When this property is set to *.T.*, end-users are allowed to set their own environment preferences. The default value is *.T.*

*IInformUserForUpdate* – When this property is set to *.T.*, user will be notified with a message when database update is started. The default value is *.F.*

*ISaveExportPathPerUser* – When this property is set to *.T.*, the export folder chosen by every user for PDF, BMP, HTM- und TIFF files is kept in the resource file per user. The default value is *.T.*

*IShowProgressOnUpdate* – When this property is set to *.T.*, a progress bar form will be displayed while the database update runs. The default value is *.T.*

*IUseBCCRecipients* – When this property is set to *.T.*, a BCC input textbox in E-Mail details form is visible. The default value is *.T.* This feature cannot be used with all e-mail clients. BCC option is available when the e-mail client is Outlook Express, but it is not available when the e-mail client is Outlook.

*nDockable* – Defines if forms can be docked.

- 1 - Use form settings,
  - 0 - All forms do not support docking.,
  - 1 - All forms support docking and are dockable,
  - 2 - All forms support docking, but are not dockable. In this case forms can only be docked one into another.
- Modal forms cannot be docked.

*nHighLightStyle* – This property is used to set globally *HighlightStyle* style for *cGrid* class.

*nIndexInsteadOfFilter* – Defines if filtered index will be used instead of filtering

- 0 - Form settings will be used,
- 1 - Force to *.T* – always use filtered indexes,
- 2 - Force to *.F*. – always use filtering.

*nNullValid* – This property defines if *Null* is valid value for Pick classes.

- 0 - Use control settings,

- 1 – Null value is allowed,
- 2 - Null value is not allowed.

*nSearchOnInit* – This property is used in conjunction with *lSearchOnInit* property of *cDataForm* class. It specifies whether a form should come up with the last search performed by user.

## **8.2. VFX Form Wizard**

As in former VFX versions, the VFX – Form Wizard should be used to create a new form. The user interface of the VFX – Form Wizard was already discussed in the chapter *Quick overview*. As an extension to the behavior of VFX 8.0 Builders, now after creating a form in VFP Form designer, the VFX – Form Builder will be automatically started. The VFX – Form Builder includes the new VFX – Data Environment Builder. The developer is led by the builder step by step, starting with the choice of an appropriate form class up to the running form.

## **8.3. VFX - Form Builder**

The VFX – Form Builders support all new form's properties of VFX 9.5. The Form Builder of VFX 9.5 were wholly remade and large number of features were included. Additionally, now it is possible to key up functionality that were only manually editable in VFP before. On new pages of the Form builder can be defined view parameters, related tables, required fields and fields to be used in reports.

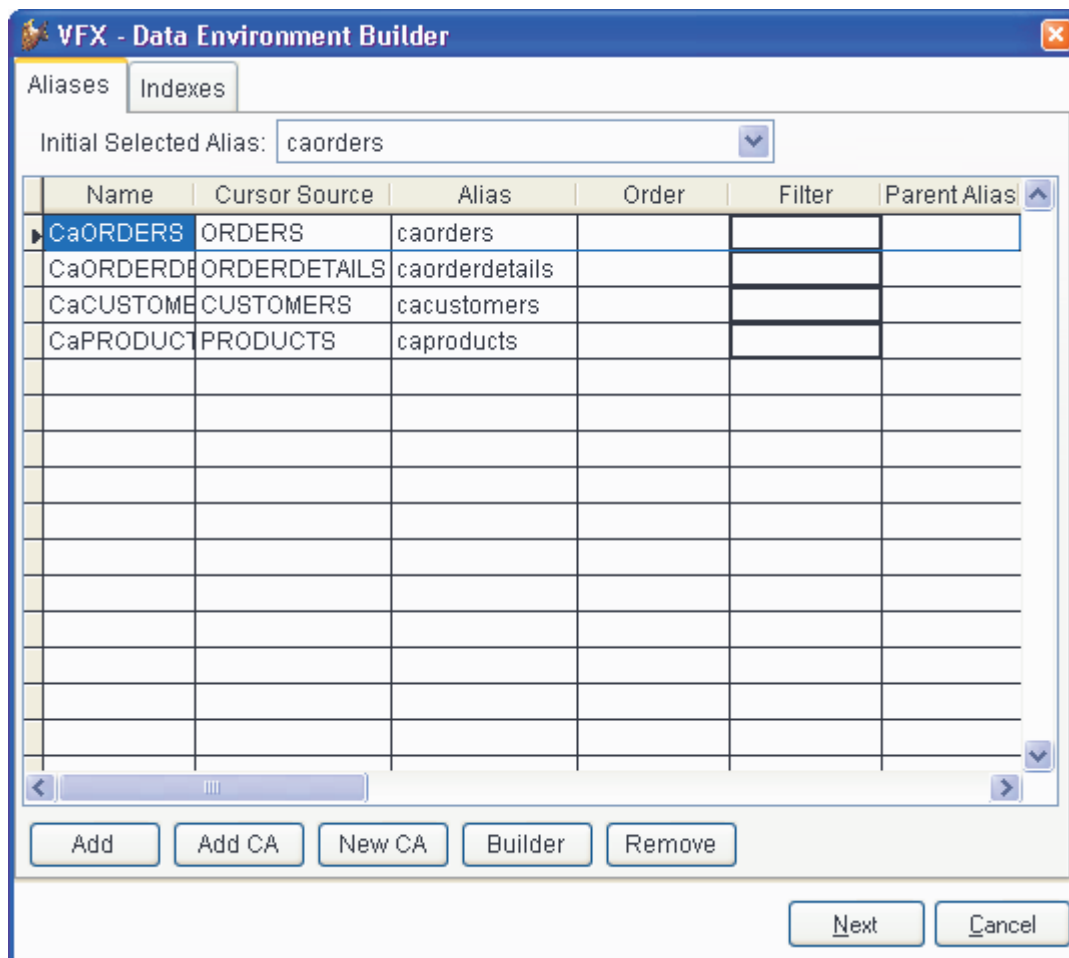
The greatest new feature here is the first step of the Form Builders dialog, which shows a Data Environment Builder.

## **8.4. VFX - Dataenvironment Builder**

The VFX - Form builder now give developers opportunity to set data environment along with building form layout.

Tables, views or predefined CursorAdapter classes can be added to dataenvironment and new CursorAdapter classes can be created, too. It is possible to set order and filter of data, to define used indexes and to define relationship between objects in data environment.

On page *Aliases* can be added tables, existing CursorAdapter classes as well as creating new CursorAdapter objects.



Clicking on *Add* button will add existing table or view to data environment. The VFP “Add table or view” dialog is invoked for choosing a table or a view. When the cursor is based on a table, or an index is defined for CursorAdapter cursor, in the column *Order* you can choose from index tags of the table or defined indexes for the CursorAdapter cursor.

By clicking the button *Add CA*, a CursorAdapter object based on existing class can be added. Such class can be, for instance, a class for table in your database created with VFX CursorAdapter Wizard. In the Open dialog you have to select needed class library and class.

The button *New CA* creates a new object based on *cAppDataAccess* class and starts the VFP CursorAdapter builder for keying up its settings.

If the cursor in data environment is based on a table, in *Order* column you can choose from existing index tags that the table has already created. If this is a cursor based on CursorAdapter class, on *Indexes* page you can define index expressions. Indexes will be created at run-time, after the cursor is created in its *AfterCursosFill* event. To fill the *Order* column in page *Aliases* you can choose a value among this list of created indexes.

The names and alias names for cursors can be freely changed in the data environment.

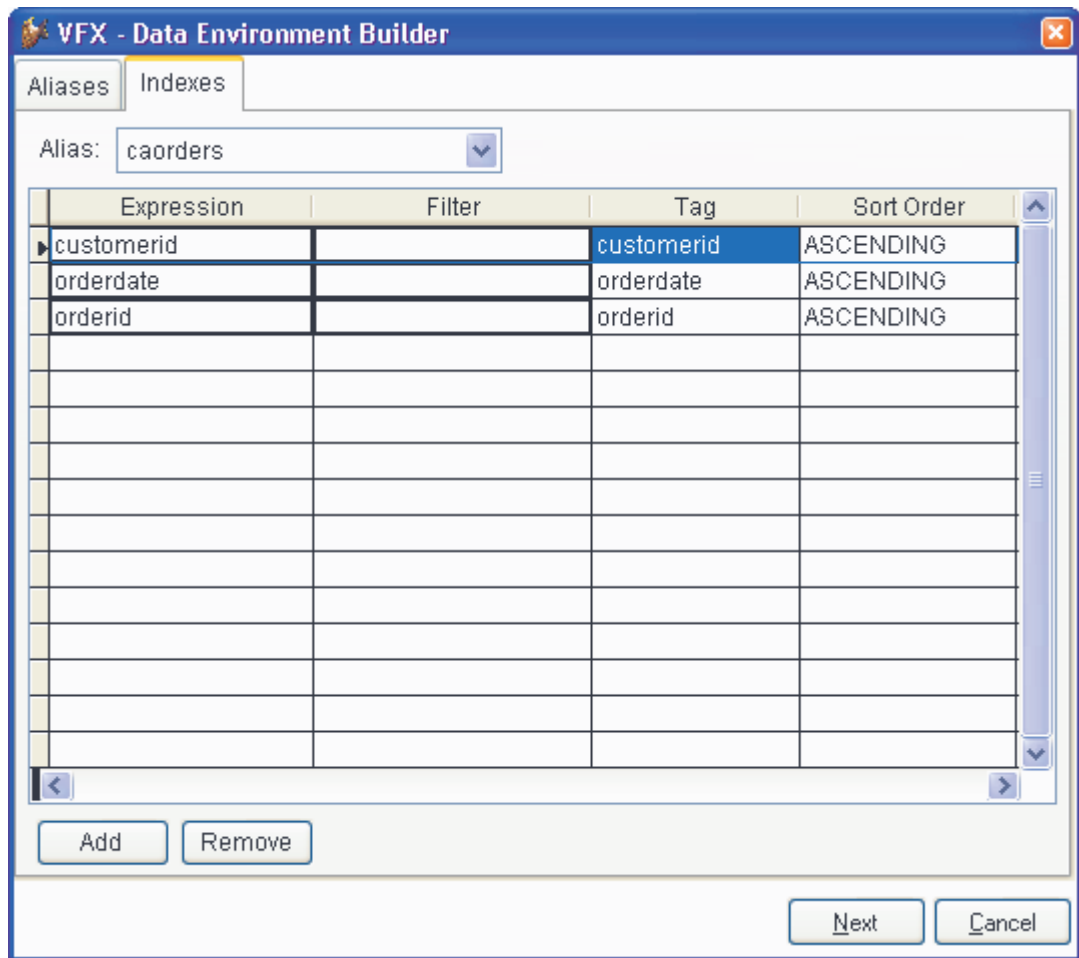
In the column *Filter* can be entered a valid logical expression to be applied as filter expression. It corresponds to the *Filter* property of the Cursor object.

The columns *Parent Alias* and *Rel.Expression* give you the chance to build relationship between cursors in the dataenvironment. After selecting the alias in *Parent Alias* column, you can choose the field which will



maintain the relationship from the drop-down list in *Rel.Expression* column or enter your own relational expression.

Defined relations are maintained by *oRelationMgr* object – an instance of the class *cRelationMgr*.



For CursorAdapter objects the only way to build relationship is to create temporary indexes at run time. Developer can define necessary indexes on *Indexes* page. VFX framework takes care to create necessary index files at run time and to establish relationships which are defined on *Aliases* page.

For Cursor objects, based on a table or view, indexes are created in advance and here, in builder, they are displayed read-only. Here, can be managed only indexes that will be created for CursorAdapter objects at run-time.

For every defined index must be entered values for *Tag*, *Expression* and *Sort order*. If a filtered index need to be made, in the column *Filter* is entered the filter expression.

Clicking on the button *Next* brings the next step of the VFX – Form Builder

### 8.5. VFX - CDataFormPage Builder

To call the VFX – Form Builder, put the mouse on the white Background of the form, click the right mouse button and select builder.

The VFX – Form Builder loads and shows an user friendly Dialog:

### 8.5.1. Edit Pages

In VFX – Form Builder, on page *Edit Pages*, can be keyed up all new form's properties of VFX 9.5 like Background picture or Back color for pages in a pageframe control, linked tables and required fields as well as the properties related to AutoComplete.

When the checkbox *Add colon to labels* is marked a colon char will be appended at the end of all labels.

**Form Name.** Enter the name of the new Form. VFX – Form Builder assigns a default form name following the common naming conventions, beginning with *frm*. Of course you can give your form any name, but we recommend that you follow the common naming conventions.

**Caption.** Enter the caption for your form. While you type the caption, you'll see it displayed in the form builder's caption. If your form has a variable caption, depending how the Form will be called, don't worry too much about this caption, just use a more or less descriptive caption in that case.

**Page Count.** Enter how many Edit Pages your form will have. For some forms, one edit page will be enough. If you have more fields, you might want to spread them over multiple pages. Depending on the number of pages you select, you will see in the tab dialog on the form builder, a Dialog simulating these Pages. If you setup two Edit Pages, two tabs will appear, if you select three, you will see three and so on.

**Page Title.** Enter the caption for the edit page you currently selected. If you want to enter the caption for the second Page, you click the second page tab and you can enter the caption for it. VFX – Form Builder will instantly reflect your entry on the tab captions of the corresponding page.

**Justified Tab.** Check this option, if your tabs should be justified, otherwise they will be variable in the length and will not fill the whole width of your form.

For every page defined through the Page Count option, you can select the following options:

**Fields Selected.** Here you see all fields you selected for the currently selected edit page. To select fields, use the *Field Assistant* Window, which is a separate form that offers all fields currently available in the data environment.

**Control Type.** Define which control type you want to use for every selected field. All control types defined in VFX are available for usage.

---

**NOTE:** To use your own classes, make sure to add them field by field in the DBC as display class library!

---

**Caption.** Caption for the selected field. The default will be read from the database container.

**Format.** Format property for the selected field. The default will be read from the database container.

**Input Mask.** Input Mask property for the selected field. The default will be read from the database container.


**Status Bar.** Status Bar Message used for this field. The default will be read from the database container (property comment resp., if empty, the caption).

**Read only.** If a control will be used for display information only, check this checkbox.

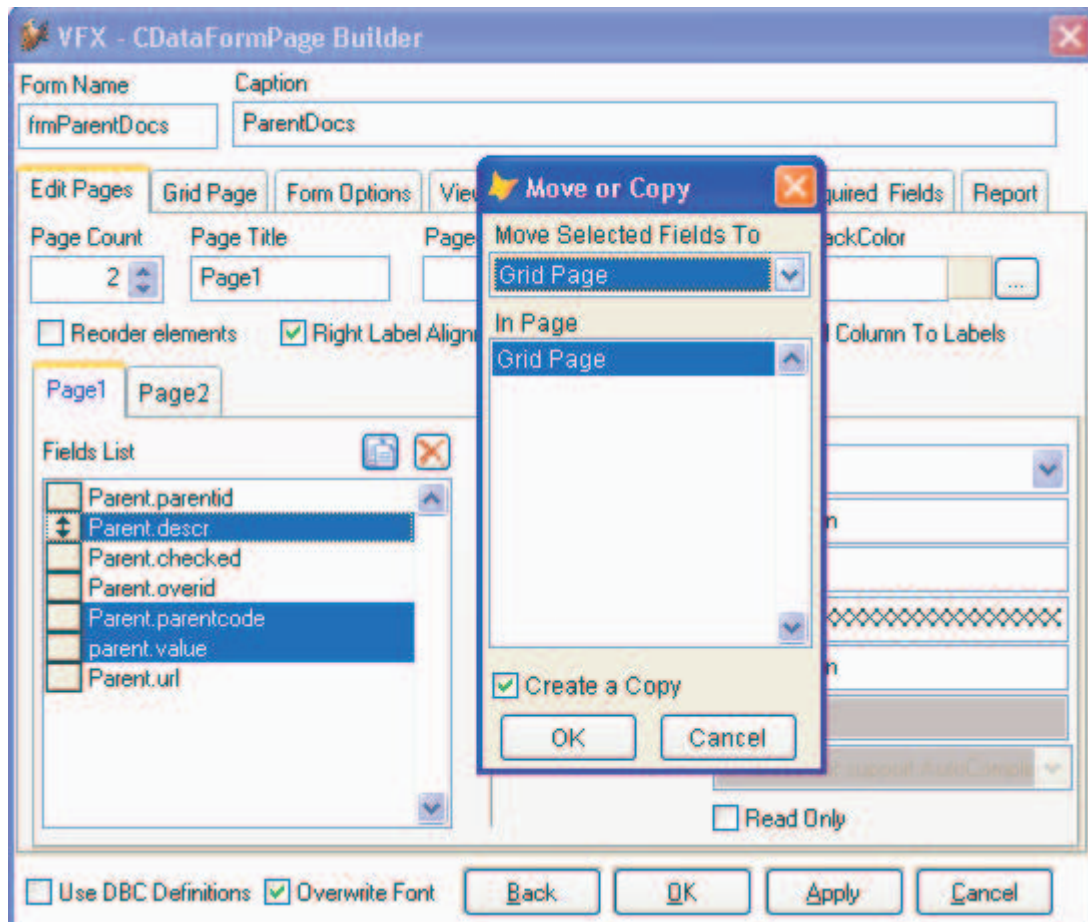
**AutoCompSource.** Name of table that will be used for VFP AutoComplete function. It is not required to deploy AutoComplete table with the developed application. This table will be created by VFP automatically when necessary.

**AutoComplete.** The value for AutoComplete property. The AutoComplete function is available for textbox controls only.

**Read only.** When a particular control will be used only for displaying information, mark this checkbox.

When redesigning existing forms, the new button  *Move or Copy Fields* is very useful. It would help you a lot if a page has to be moved or list was built on the wrong page by accident or even if grid list is similar to edit field list or vice versa. All the settings would not have to be keyed up again which is very important in case of mistakes or maintenance.

In Fields list can be selected all desired fields In the dialog *Move or Copy Fields* should be selected destination page. Type of the destination page is chosen From the *Move Selected Fields To* drop-down list. Destination page type can be Edit page, Grid page or Report fields page. After you select type of destination page, the list *In Page* shows all existing pages of chosen type.



When it is necessary not to move selected controls, but just to create a copy of them using all currently set properties, you should mark the *Create a Copy* check box. In this case controls remain on their original place and a copy of each control is created on the destination page.

## 8.5.2. Grid Page

The screenshot shows the 'VFX - CDataFormPage Builder' dialog box with the 'Grid Page' tab selected. The 'Form Name' is 'frmParentDocs' and the 'Caption' is 'ParentDocs'. The 'Grid Page Title' is 'List' and the 'Grid Class' is 'cgrid'. The 'Grid Page Picture' field is empty, and the 'Use Grid Page' checkbox is checked. The 'Grid Page BackColor' is set to a light yellow color. The 'Fields Selected' list contains the following fields: parent.parentid, parent.descr, parent.date, parent.checked, parent.value, parent.ins\_date, parent.ins\_usr, parent.edt\_date, parent.edt\_usr, parent.overid, and parent.parentcode. The 'Control Type' is 'textbox', the 'Header' is 'Parent ID', the 'Control Source' is 'parent.parentid', and the 'Output Mask' is '999999999'. The 'Read Only' and 'Incremental Search' checkboxes are checked. At the bottom, there are buttons for 'Back', 'OK', 'Apply', and 'Cancel', and checkboxes for 'Use DBC Definitions' (unchecked) and 'Overwrite Font' (checked).

The following options are available on the page *Grid Page*:

**Use Grid Page.** Check this checkbox if you want a Search grid page on your form.

**Grid Page Title.** Enter the caption for the last page in your form which normally will be a grid to display all records from within your table or view.

**Grid Class.** Select the grid class you want to use or use the default, which is the *CGrid* Class.

**Fields Selected.** Here you see all fields you selected for the grid. To select fields, use the *Field Assistant* Window, which is a separate form which offers all fields currently available in the data environment.

**Calculated Fields.**  Click this button to add whatever calculated field you want.

**Control Type.** Define for all selected fields which control type you want to use.

**Header.** Captions for the column headers of your grid. VFX – Form Builder will automatically take the captions from your database container.

**Output Mask.** VFX – Form Builder takes the input mask from the length of the field. You can change the input mask to accommodate your particular needs.

**Read only.** When a particular control will be used only for displaying information, mark this checkbox.

**Incremental Search.** Mark this checkbox, if you want to make available the incremental search feature for the selected column. Note that VFX creates a temporary IDX index file, if there is no index tag available for this

column. (with the *CGrid* property *nMaxRec* you can define, when you want to have a message to pop up before a temporary index will be generated)

### 8.5.3. Form Options

The following options are available on the page *Form Options*:

The screenshot shows the 'VFX - CDataFormPage Builder' dialog box with the 'Form Options' tab selected. The 'Form Name' is 'frmParentDocs' and the 'Caption' is 'ParentDocs'. The 'Form Options' tab contains the following settings:

- Report Name: (empty text box)
- Font: Arial,9,N
- Is Child Form
- Has More Functions
- Has Linked Child Form
- Auto Sync. Child Form
- Put In Last File Menu
- Put In Window Menu
- Can Edit
- Can Insert
- Can Copy
- Can Delete
- Multi Instance
- Close with ESC Key
- Save/Restore Positions
- Add SpeedBar Control

At the bottom, there are checkboxes for  Use DBC Definitions and  Overwrite Font, along with 'Back', 'OK', 'Apply', and 'Cancel' buttons.

**Report Name.** Here you can select a report name. Whenever the user selects *print* or *preview*, this report will be selected and printed respectively previewed. All this without the need to write code in the method onPrint. When this property is left empty, VFX searches for a report that has same name as the form.

**Is Child Form.** If the form you are currently creating will be called from another form, this form acts as a child form.

---

**NOTE:** Please don't mix this up with the later described One-To-Many Form where you can have the parent and the child table processed **on the same form**. Here we are talking about this scenario: Form1 -> calls Form2, whereas Form1 could be the parent form and Form2 could be the child form and in Form2 you would see only those records which match a certain criteria, which might be the link to the parent table in Form1.

---

If, for example, you have a Form in which you want to provide the ability to process the orders of a customer, check this checkbox and have VFX – Form Builder automatically have set up the form as a child form. This will automatically put the needed lines of code in the *Init Event* of the form. All you have to do is review this init code and adapt it to your specific needs.

For further details please refer to the topic *VFX – Parent/Child Builder* later in this document.

---

**NOTE:** Although, if you have a form which will act as both, a Child Form as well as a Normal Form, you should set it up as a Child Form marking *Is Child Form*. There is no need to setup two Forms for, i.e. Orders. With one form you can perfectly handle the scenario All Orders, as well as Orders for a particular Customer.

---

**Has More Functions.** If the form you are currently creating will call other forms or actions, you have to check this checkbox. This will automatically generate the needed code in the *OnMore()* method of your form. You simply have to review the *OnMore()* code and adapt it to your specific needs. Normally you will have a couple of actions which will be presented in a form and the user selects the desired option.

**Has Linked Child Form.** If the form you are currently creating will have child forms which should be dynamically linked to this parent form, check this option. This will automatically generate the form method *OnSetChilddata()* which will be called automatically for every child form available.

**Autosynch Child Form.** This sets the form property *IAutosynchChildform* which defines, whether you want the linked child forms synchronize automatically whenever the parent record changes or only when the user activates the child form. If this option is not marked, the Child-form will be refreshed, when the user activates it.

**Put in Last File Menu.** This sets the form property *IPutinLastFile* which defines whether you want to put the form caption in the recently used file list in the *File* menu.

**Put in Window Menu.** This sets the form property *IPutinWindowmenu* which defines whether you want to put the form while running in the *Windows* menu. See also the application class property *nWinMnuCount* and the application class method *RefreshWindowMenu()*.

**Can Edit.** This sets the form property *ICanEdit* which defines whether the user can edit records in the current form or not.

**Can Insert.** This sets the form property *ICanInsert* which defines whether the user can insert records in the current form or not.

**Can Copy.** This sets the form property *ICanCopy* which defines whether the user can copy records in the current form or not.

**Can Delete.** This sets the form property *ICanDelete* which defines whether the user can delete records in the current form or not.

**Multi Instance.** This sets the form property *IMultiInstance*. By default all of the forms you create with VFX can be called multiple times (so called multi instantiation). This is a great feature, all you have to remember when use multi-instantiation is to set the form to private data session which is the default of all VFX forms.

However sometimes it is convenient to disable the feature of multi-instantiation. That's why we created the property *IMultiInstance*. Just set this property to .F. and the form can only be called once.

**Close with ESC key.** This sets the form property *ICloseOnEsc* which defines whether the user can close a form using the escape key or not.

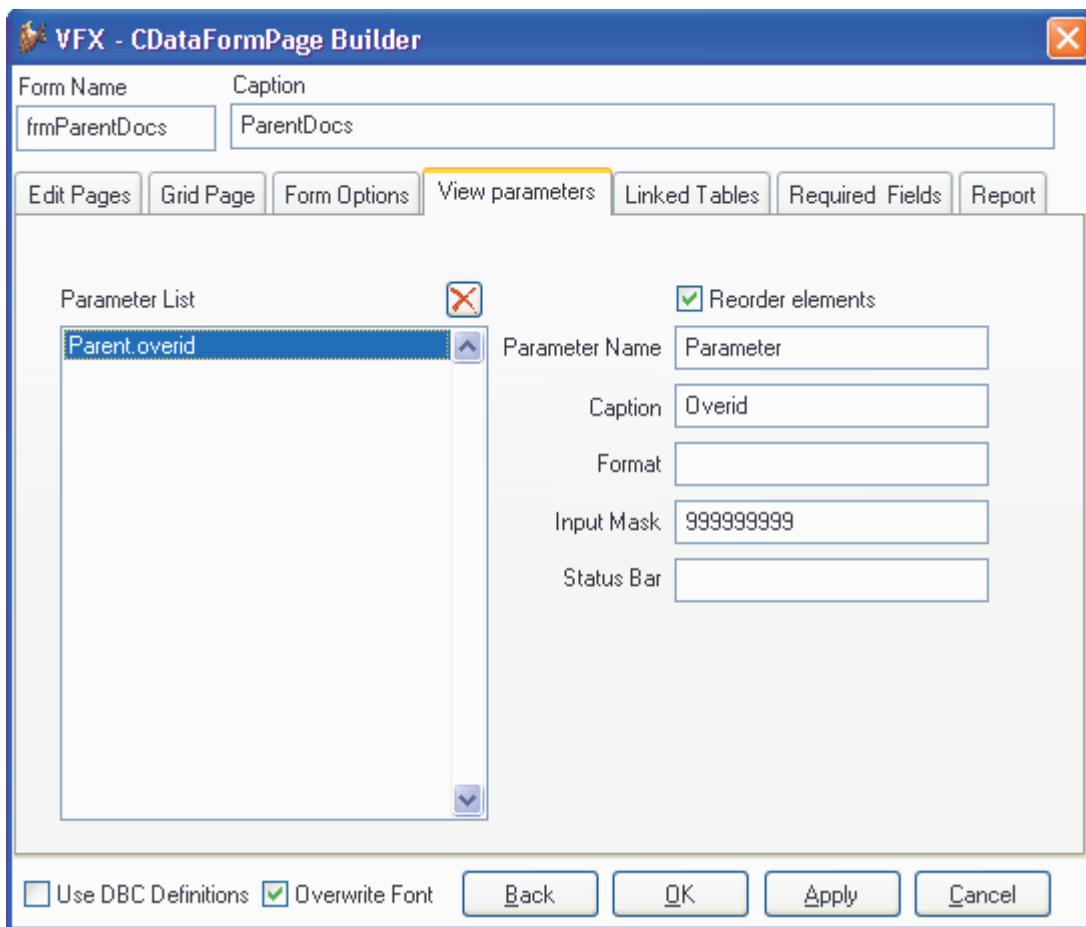
**Save/Restore positions.** This sets the form property *ISavePosition* which defines whether you want that all the position and other form settings are stored within the VFX Resource File.

**Add Speedbar Control.** This adds a form toolbar, similar to this:



Additionally there are four new pages in VFX – Form Builder for keying up the new properties of VFX form classes.

### 8.5.4. View Parameters



On page *View Parameters* can be defined controls for entering view parameters. Similar to *cAskViewArg* form class end-users can fill in values at run-time. The button *Requery* in the standard toolbar is used to refresh the view. This approach avoids invoking of another forms for parameters input.

Input controls for view parameters are placed at upper part of the form, above the data pageframe. They are always visible and user can change arguments and invoke data reloading. For these controls a parameter name is specified instead of *ControlSource* property



### 8.5.5. Linked Tables

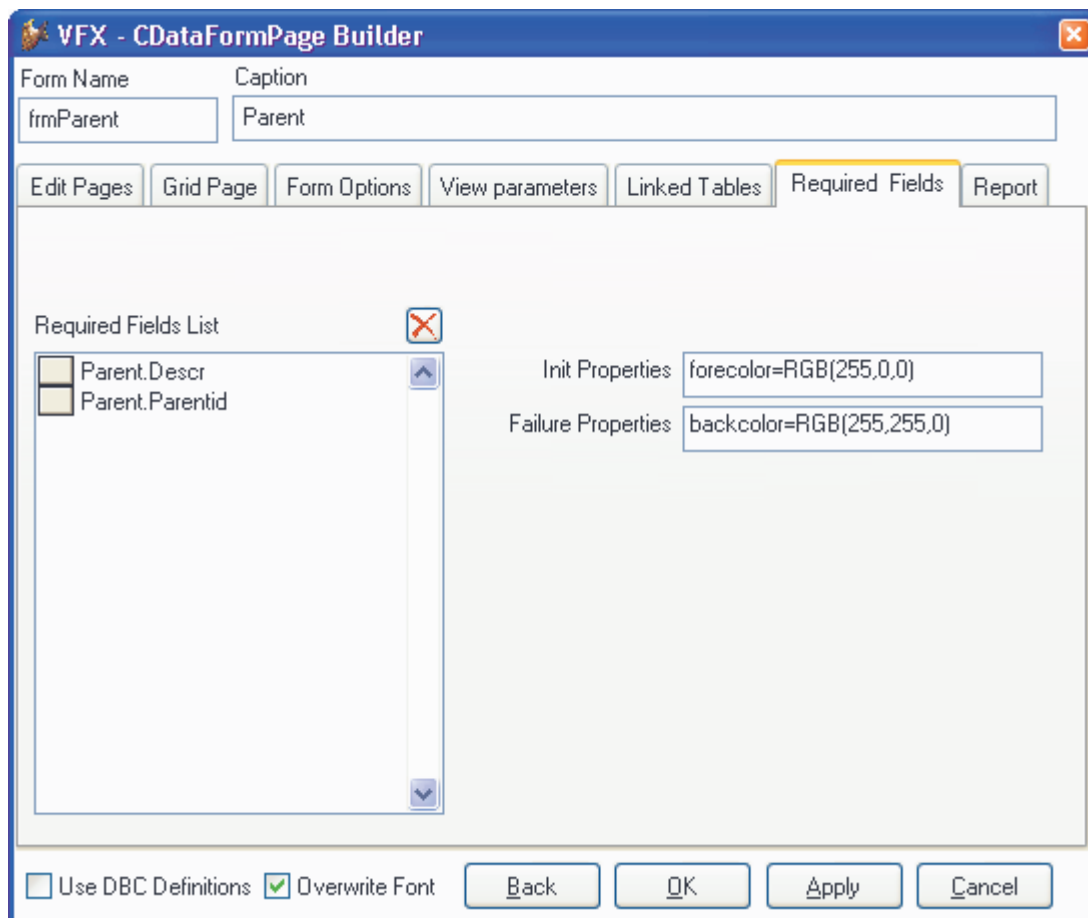
The screenshot shows the 'VFX - CDataFormPage Builder' dialog box with the 'Linked Tables' tab selected. The 'Form Name' is 'frmParentDocs' and the 'Caption' is 'ParentDocs'. The 'Linked Tables' tab is highlighted with a dashed border. Below the tabs, there are two dropdown menus: 'Master Table' set to 'Parent' and 'ID Filed' set to 'Parentid'. To the right is a 'Parameter List' area with a scroll bar. At the bottom, there are checkboxes for 'Use DBC Definitions' (unchecked) and 'Overwrite Font' (checked), along with 'Back', 'OK', 'Apply', and 'Cancel' buttons.

VFX 9.5 forms allow you to define additional list of tables which are 1:1 related to the main table. This feature gives the developers good support for more complex database design without need to write additional code to maintenance referential integrity. Having described the relational condition, VFX takes care to keep data consistent.

It is not required related tables to have same primary key name as the parent table, however it is good design to keep them same. This key is automatically filled in insert operations after inserting new record in parent table. Related records are deleted when delete operation is performed on parent table.

On *Linked tables* page you have to select parent table and its primary key field. In the Parameter List you can add fields from related table which maintain referential integrity. In the parameter list are listed alias and related field. It is allowed to choose only one related field per table. When a second field from a table that is already in the list is chosen, it overwrites the field that is already placed in the list.

## 8.5.6. Required Fields



The new form's properties *cRequiredFields*, *cRequiredFieldInitProps*, *cRequiredFieldFailureProps* and *cRequiredFieldFailureForm* are designated to avoid NULL or empty values being saved in database.

In the listbox *Required Fields List* can be added any desired fields from the *Field Assistant* Window. At initiating of form control source of all controls will be checked against required fields list. All controls with a controlsourc corresponding to the list, will be treated as required fields.

Form Builder stores the list of required fields in property *cRequiredFields* of the form.

In *Init Properties* textbox can be entered a semicolon separated list of properties and corresponding values for every property in following format:

```
PropertyName = cExpression [; PropertyName = cExpression ]
```

At initiating time, for all required fields this settings will be applied. In case of sample screenshot above for all controls which have required fields as control source, *Forecolor* property will be set to red, when controls are initiated.

If more than one properties need to be set, separate the list with semicolon character. For example, if you'd like to show required fields initially with bold font, in red fore color and light yellow background you need to write following expression in *Init Properties* textbox:

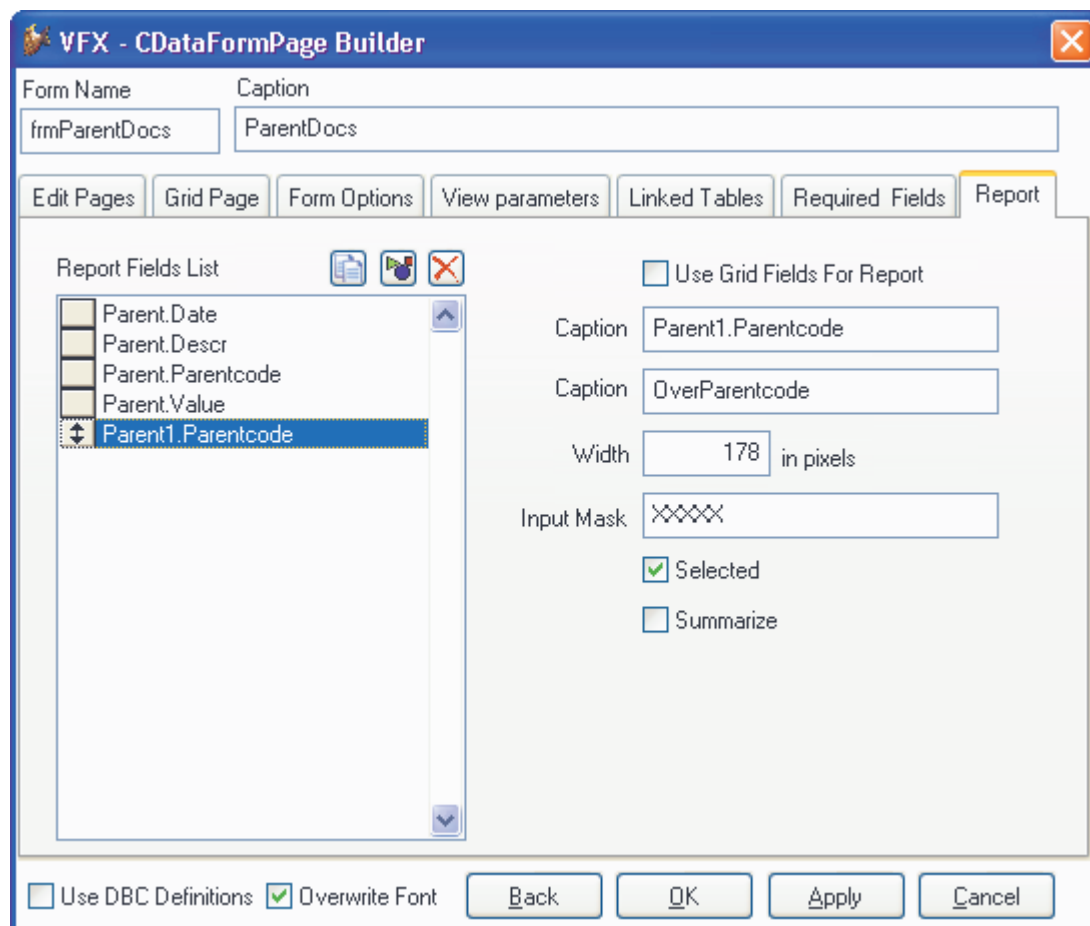
```
FontBold = .T.; ForeColor = RGB(255,0,0); BackColor = RGB(255,255,196)
```

This is very good way to show the end users which fields must be filled. The value of textbox *Init Properties* will be stored into *cRequiredFieldInitProps* property of the form.

When data is saved, values entered in all the required fields are checked for valid input. If a missing value is encountered, the properties of corresponding control will be changed according expression entered in *Failure Properties* textbox. This expression is constructed in same way as the expression for *Init Properties*. The value of *Failure Properties* textbox is saved in property *cRequiredFieldFailureProps* of the form.

As long as not all required fields is given a value, data entered in the form will not be saved.

### 8.5.7. Report



Often it is necessary to include in printed report some fields that are not needed in the grid field list. In other cases you may need to include in grid page fields, which do not need to be printed. *Report* page allows you to select fields that will be listed in on *Advanced* page of Grid report dialog. On *Advanced* page of Grid report dialog user can additionally specify which fields should be included in the report and which fields need to be summarized. You can predefine this selection here in *Report fields* page.

For every field should be entered width, input mask and a header text.

If former VFX behavior should be kept and use in reports same fields, which are included in the search grid, just mark the *Use Grid Fields for Report* checkbox.

**OK.** Click this command button, if you want to generate your form. This will take some seconds and the result is a Form, on which you have the desired amount of edit pages with the selected fields on each page. If you selected more fields than would fit on a page, two columns will be created.

The form build process can be run more than once, this feature is called reentrance.

---

**NOTE:** The reentrance feature is only available at 100% for forms created with the VFX 8.0 form builder. For a maximum of reentrance guarantee it's best to use the Form Builder whenever you want to add a new field to your form.

---

Another great advantage of the reentrance is the fact that you can reapply changes you made within the database container (i.e. captions, format or input mask options) by just calling the form builder again and selecting the checkbox *Use DBC Definitions*.

Start your application, locate in Open dialog your newly created form and start it with a mouse click. Test the form and evaluate, where it can be enhanced.

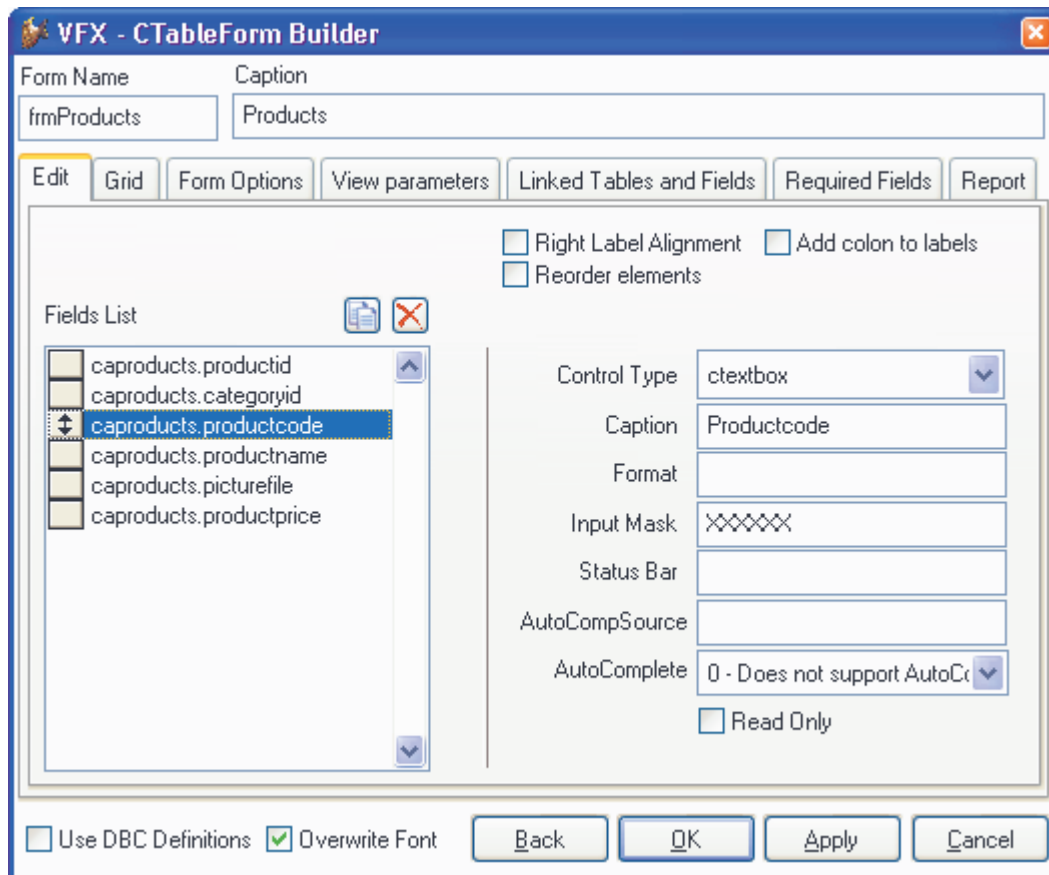
To become more familiar with the VFX – Form Builder, generate some simple forms and try to increase the complexity by generating forms that call other forms, as well as forms that will be called from other forms.

Once you are familiar with the standard VFX data manipulation form, you might want to take the step to the development of *One-To-Many* data manipulation form.

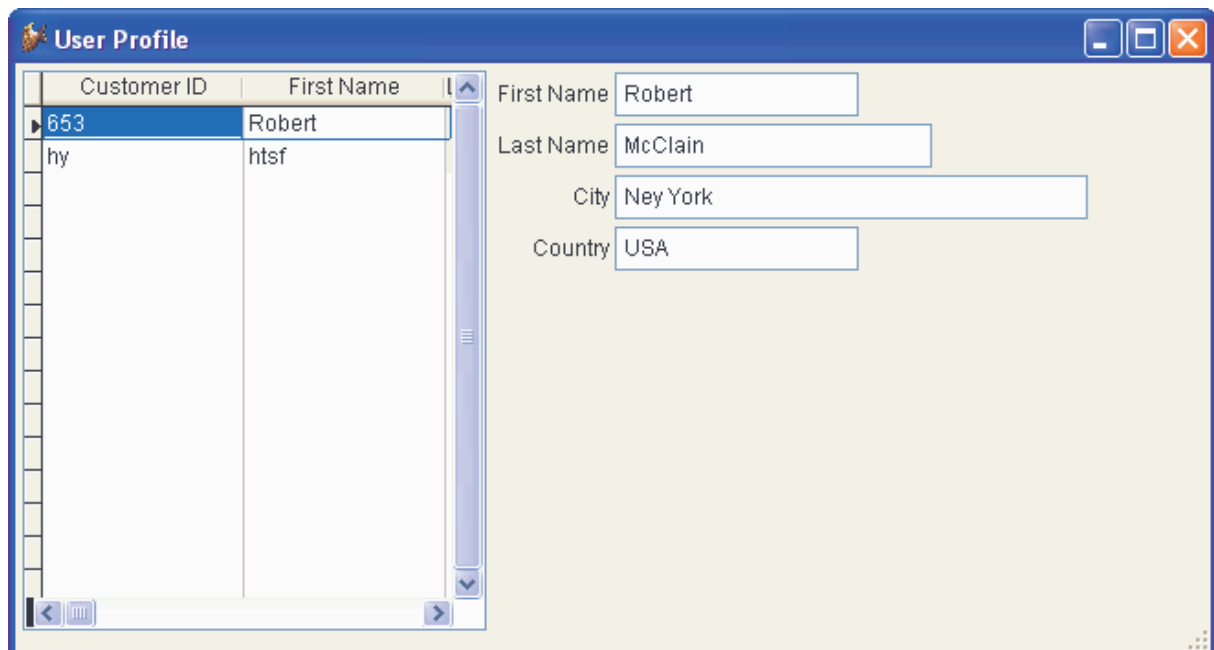
**Apply.** Does the same as OK but does not quit the form builder dialog.

**Cancel.** Cancels the VFX – Form Builder process. Any selections and entries will be lost.

### 8.6. VFX – CTableForm Builder



Another type of form is the *cTableForm*. With this form the List-Grid and the controls are placed one next to other or among themselves. Therefore, it is suitable in particular for forms with only few input fields. Here is a sample of a form based on the class *cTableForm*:



## 8.7. VFX – *COneToMany Builder*

The One-To-Many form is an evolution of the standard VFX data manipulation form. This means that you can have, on one single form, a full featured standard data manipulation form functionality, together with a grid showing the child records for the currently selected parent record. VFX allows you also to have multiple children to one Parent in a tab dialog. If you have many input Fields for a child table, you can also have a multi-page tab dialog for the child data. This allows you to cover a lot of scenarios without the need of real programming. All you need to understand if you create one-to-many forms is the database design and through which Fields the Parent and Child Tables are linked. Let's look at a simple example:

As described earlier in this document, you have to setup the database container of your application. Define your tables, fields and indexes as well as the field captions. This allows the VFX builders to use this information, so you don't have to retype the same captions again.

In order to create a one-to-many form, you must be familiar with the basics of database design and especially, with the one-to-many relations where you have a given parent record and multiple child records. A good example for a parent child relation is the Order (Parent table) and Items (Child table) situation, of any order and invoicing system.

If you don't want to assure the referential integrity (RI) manually using VFX methods like *OnPostDelete()*, it's a good practice, to generate the RI code in the database container before you generate one-to-many forms. If you don't do this, you will have to write manually the code for the deletion of parent records which have child records and in the case where you allow the change of key fields, even the update code as well.

Start the VFX Form wizard from VFX menu and create a form based on *cOneToMany* class.

Setup the data environment of the form you want to create. The VFX – Form Builder automatically picks up this information while creating the one-to-many form.

The VFX OneToMany Form Builder assists you in generating sophisticated one-to-many forms with almost no coding. If you setup the one-to-many relation from the parent table to the child table (in the case of multiple child tables which all depend from the same parent table you might have more than one child table, each linked with the parent table through a relation) you can generate one-to-many forms as easy as standard VFX data manipulation forms.

---

**IMPORTANT:** Remember to define the *InitialSelectedAlias* property, and the one-to-many relation from the parent to the child in the data environment, otherwise your form may not run as you expect!

---

The VFX One-To-Many Form Builder has an intuitive user interface.

The screenshot shows the VFX - COneToMany Builder dialog box. The 'Form Name' is 'frmOrders', the 'Caption' is 'Orders', and the 'Master Table' is 'caorders'. The 'Form Options' section is active, showing 'Page Count' as 1, 'Page Title' as 'Page1', and 'Right Label Alignment' checked. The 'Fields List' on the left shows a list of fields from the 'caorders' table, with 'caorders.orderid' selected. The 'Control Type' is set to 'ctextbox', and the 'Caption' is 'orderid'. The 'Input Mask' is '999999999'. The 'Read Only' checkbox is checked. At the bottom, there are checkboxes for 'Use DBC Definitions' (unchecked) and 'Overwrite Font' (checked), and buttons for 'Back', 'OK', 'Apply', and 'Cancel'.

First, you select the following options:

**Form Name.** Enter the name of the new Form. VFX – Form Builder assigns a default form name following the common naming conventions, beginning with *frm*. Of course you can give your form any name, but we recommend that you follow the common naming conventions.

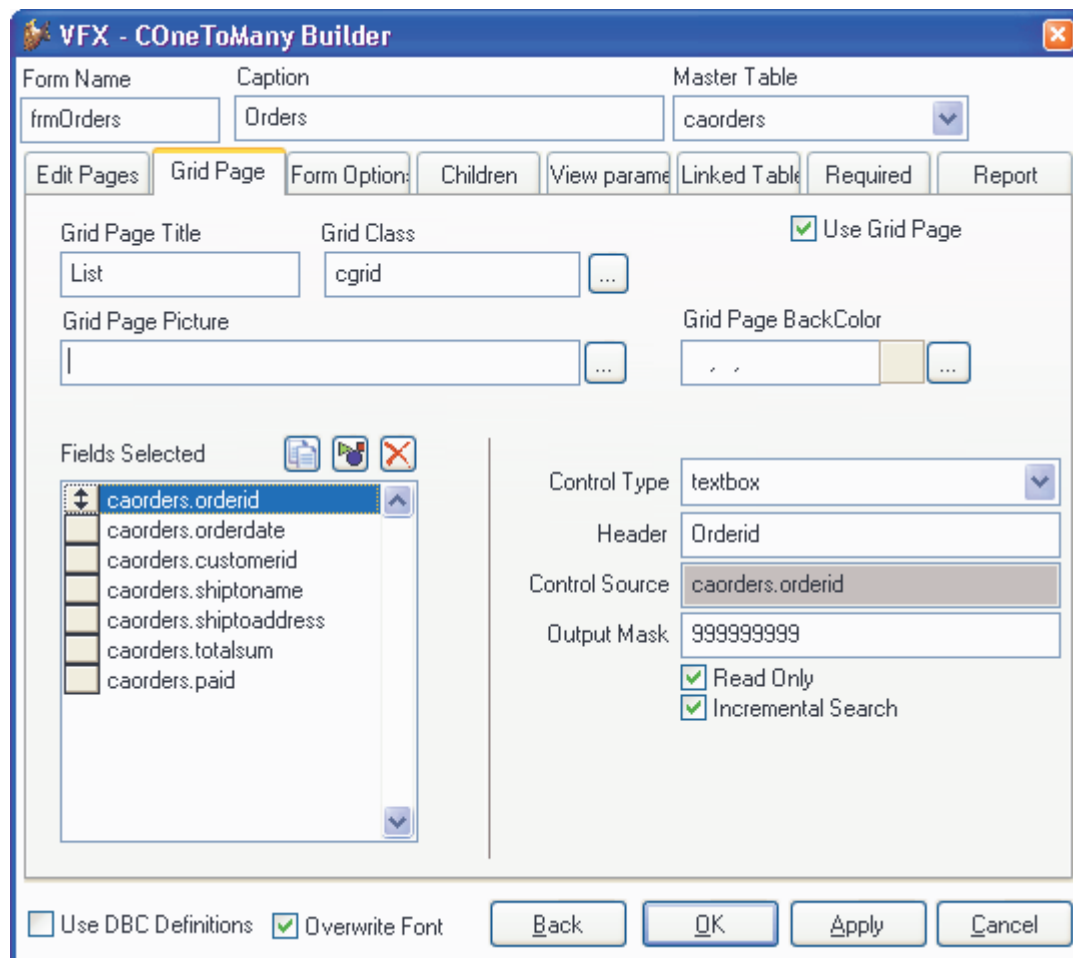
**Caption.** Enter the caption for your form. While you type the caption, you'll see it displayed in the form builder's caption.

**Master Table.** Table name of the parent table or view.

Then you have a page frame with the pages *Edit Pages*, *Parent Grid Page*, *Form Options* and *Children* which are explained here:

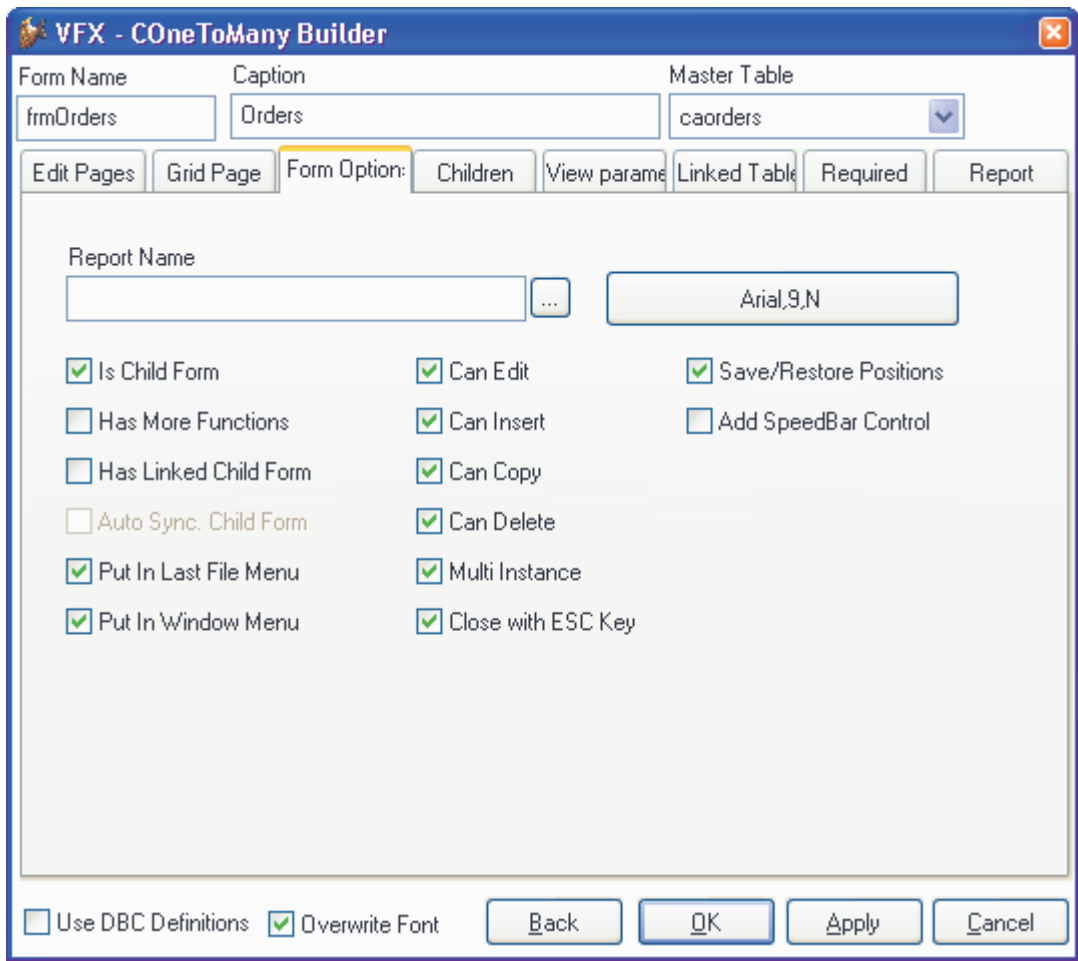
On the page named *Edit Pages* you see the same user interface like in the VFX CDataFormPage builder, described earlier in this documentation. Here you define the pages for editing the parent table.

On the page named *Grid Page* you see the same user interface like in the VFX CDataFormPage builder, described earlier in this documentation. Here you define the Grid Page of the parent table:





On the page named *Form Options* you see the same user interface like in the VFX – Form Builder, described earlier in this documentation. Here you define the Options for the OneToMany Form:



On the page named *Children* you define how your child grid(s) or pages containing child data, will look like:

**Page Count.** Enter how many Child Grid Pages your form will have. For most of the OneToMany Forms, one child grid page will be enough, if you have more child tables, you might want to spread them over multiple pages. Depending on the number of pages you select, you will see in the tab dialog on the form builder, a Dialog simulating these Pages. If you setup two Edit Pages, two tabs will appear, if you select three, you will see three and so on.

**Page Title.** Enter the caption for the child grid page you currently selected. If you want to enter the caption for the second Page, you click the second page tab and you can enter the caption for it. VFX – Form Builder will instantly reflect your entry on the tab captions of the corresponding page.

**Child Table.** Select the record source for your child grid. Attention: This is very important to set, if you don't set this property, your form will not work correctly.

**Justified Tab.** Mark this option, if your tabs should be justified, otherwise they will be variable in the width and will not fill out the pageframe width.

**Inplace Editing.** Set this option, if you want to enter data in your child grid, which is usually what you want.

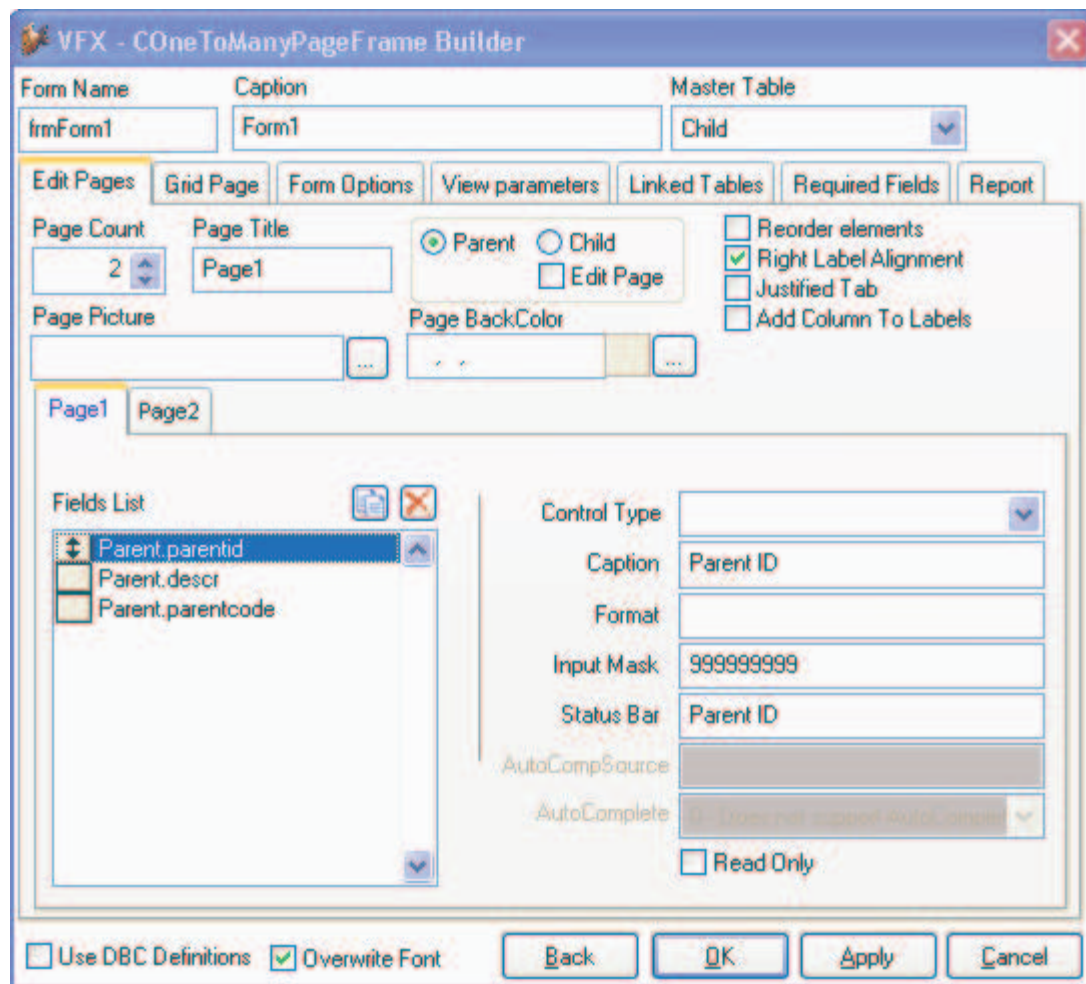
**^Ins+^Canc.** Check this option, if you want to have the possibility to add new records with *Ctrl+Ins* and to delete records with *Ctrl+Del* from the child grid.

The other options are the same like in the VFX CDataFormPage builder on the grid page.

### 8.8. VFX – COneToManyPageFrame Builder

The new class *cOneToManyPageframe* gives the developers the chance to place on different pages of one Pageframe both parent and child data. This class combines features of *cDataFormPage* and *cOneToMany* form classes.

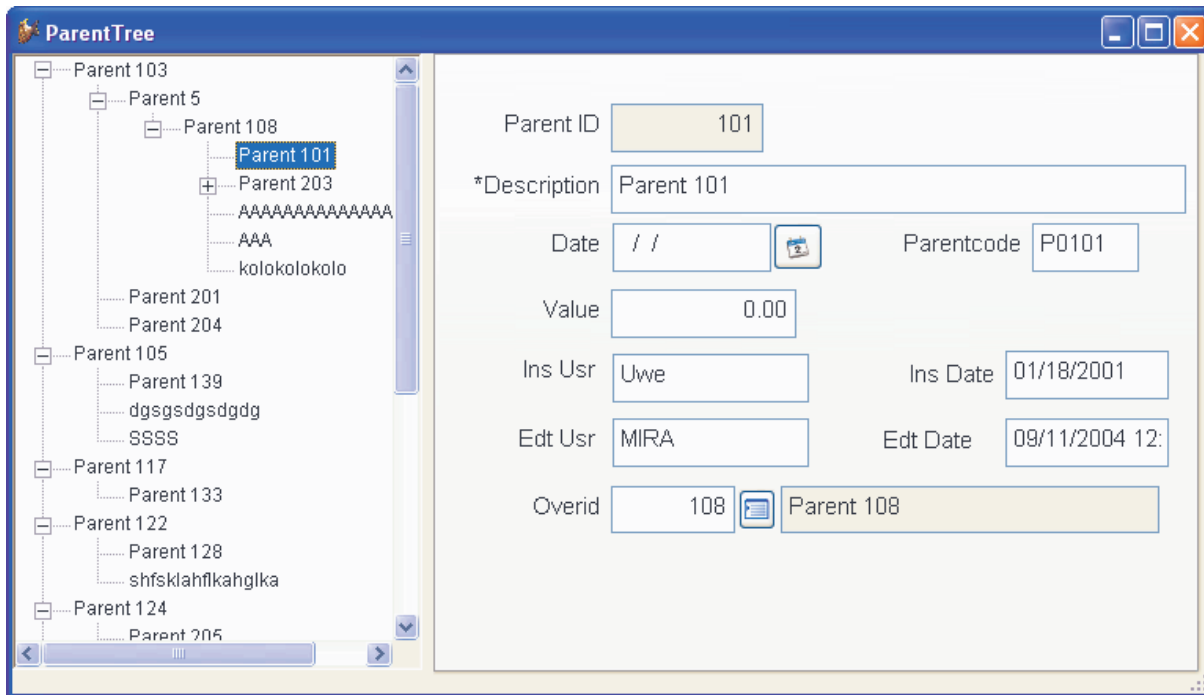
When active page of the pageframe control is of type “Parent”, navigation buttons work for the alias *cWorkAlias*. If the active page is of “Child” type, navigation is applied on the child table. On Child pages you can place either child grid or edit controls.



In addition to all other setting which the developer keys up in forms builders for other form classes, in *cOneToManyPageframe* form builder it is necessary to define for every page if it will be of Parent type and will show data from parent table or it will be of Child type and will display data from child table(s). If the page is of Child type, additionally should be specified if it is an edit page or a grid page.

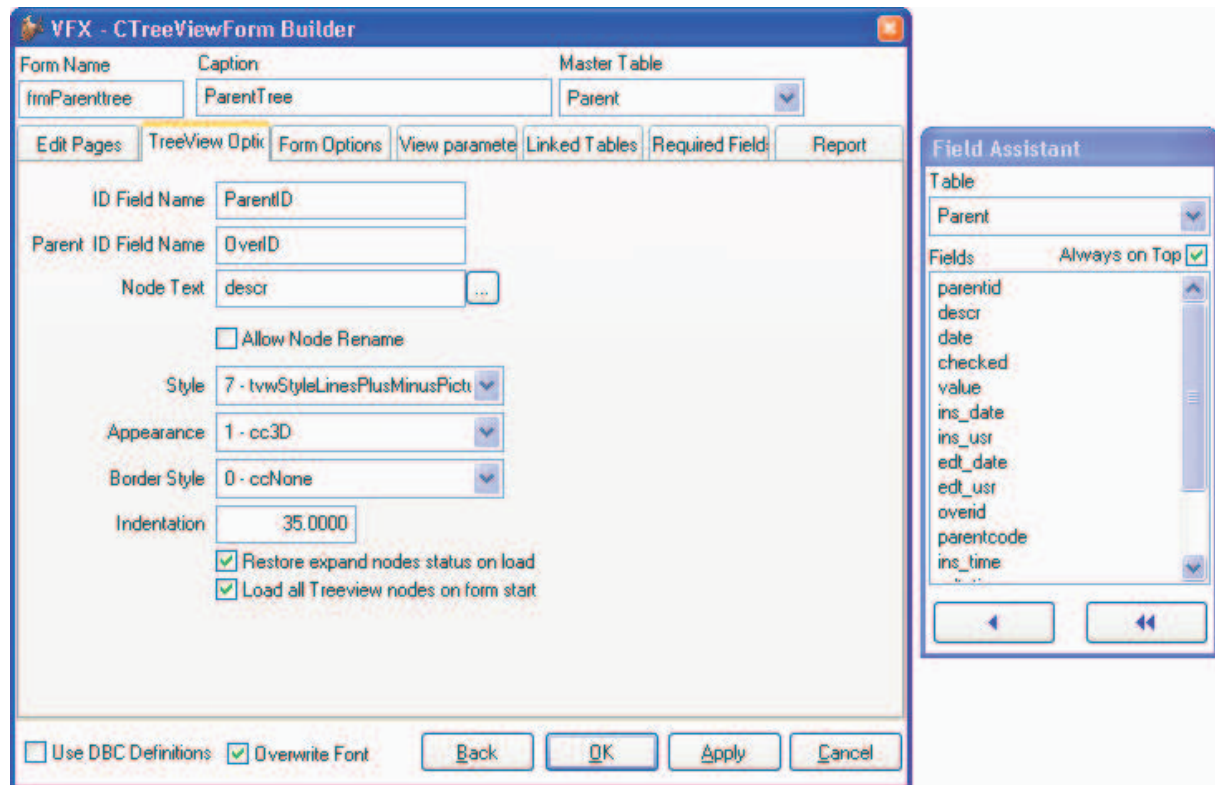
### 8.9. VFX – CTreeViewForm Builder

Main purpose of this class is to represent the data contained in a particular table (Parent table) in a tree structure. The tree-view structure gives the end-user complete overview of the hierarchical data relations contained in the table. Here is an example:



The class is based on *cDataFormPage* (*Vfxform.vcx*) and contains a *TreeView* control - *cTreeView* (*Vfxappl.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.

With the VFX CTreeViewForm builder you can rapidly create and key up a form based on *cTreeViewForm* class and set all necessary properties.



The builder works similar to the VFX - CDataFormPage builder. You can make your setting in the *EditPage* and *FormOptions* pages in the same way, as in VFX – CdataFormPage builder. In addition to this you can make your setting for the TreeView control in the page *TreeViewOptions*.

There are two types of settings for the TreeView control that you need to set.

### 8.9.1. Databinding of the TreeView control

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

*ParentIDFieldName* - This property holds the name of the field, where will be stored the Primary key value of the parent data 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 value will be used as NodeText in the tree structure. When a field name is used, the developer can allow the end-user to edit the description text directly into the TreeView control. This also depends of the *AllowNodeRename* property. If *AllowNodeRename* is set to .T., the user can edit labels in the TreeView control and this will automatically update the corresponding data field in the main table.

*AllowNodeRename* - This property defines if the user is allowed to edit description text in the TreeView control. Editing the description in the TreeView control is allowed only when the Node Text is based on single table field. The content of the corresponding data field, will be automatically be updated according new description entered in the TreeView control.

Other properties of cTtreeview class:

*ILoadAllTreeviewNodes* – if this property is set to .T. all nodes are loaded in the TreeView control when form loads. When the property value is .F., in the TreeView control are loaded only nodes at first level and nodes that are expanded when status of the control is restored. Child nodes that are not visible when

form is loaded, will not load at that time. These nodes are loaded in the Treeview control when their parent node is expanded.

*IRestoreTreeviewStatus* – When this property is set to .T. a list of expanded nodes is saved for currently logged user when the form is closed. Next time when the form is loaded same nodes of the Treeview control are expanded to restore treeview status.

### 8.9.2. Layout settings of the TreeView Control

These settings are correspondent to the TreeView ActiveX control

- Style* :
- 0 - tvwStyleText
  - 1 - tvwStylePictureText
  - 2 - tvwStylePlusMinusText
  - 3 - tvwStylePlusMinusPictureText
  - 4 - tvwStyleLinesText
  - 5 - tvwStyleLinesPictureText
  - 6 - tvwStyleLinesPlusMinusText
  - 7 - tvwStyleLinesPlusMinusPictureText

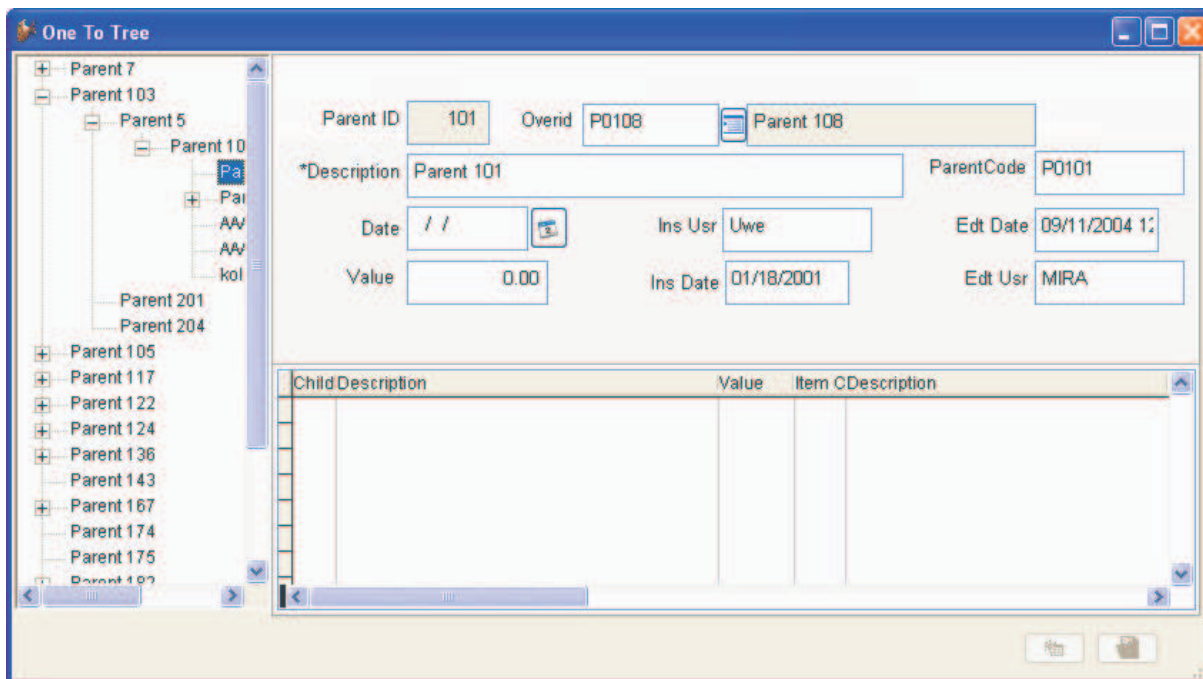
- Appearance* :
- 0 - ccFlat
  - 1 - cc3D

- Border Style*:
- 0 - ccNone
  - 1 - ccFixedSingle

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

### 8.10. VFX – CTreeViewOneToMany Builder

Main purpose of this class is to represent the data contained in a particular table (parent table) in a tree structure along with the powerful functionality that *cOneToMany* class gives the developers. The TreeView structure gives the end-user complete overview of the hierarchical data relations contained in the particular table. Here is an example for a form, based on *cTreeViewOneToMany* class.

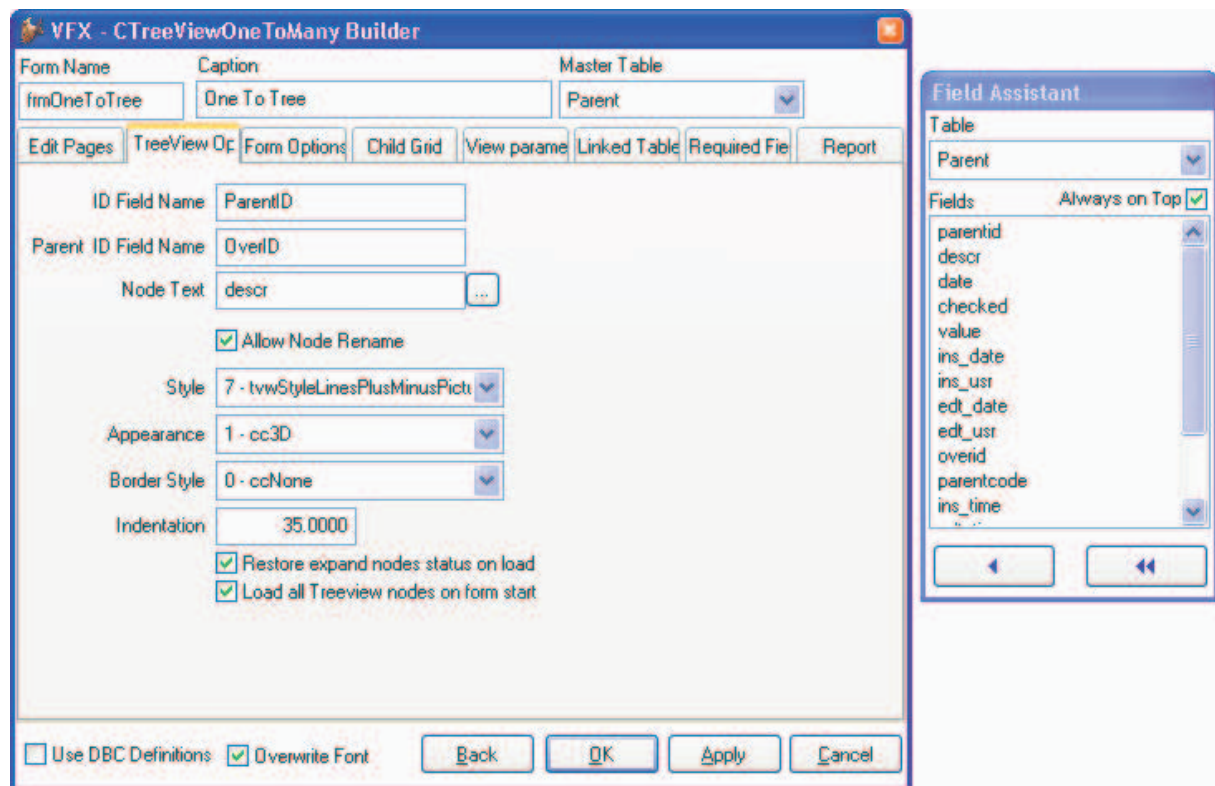


This class is based on class *cOneToMany* (*vfxform.vcx*) and contains a Treeview control based on *cTreeView* class (*vfxappl.vcx*). The class combines the functionality of *cOneToMany* and the advanced data presentation in a hierarchical tree structure. When a particular node in the TreeView control is chosen, 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. Additionally in the lower part of the form can be edited related child tables.

With the VFX – CTreeViewOneToMany Builder you can quick create a form based on the class *cTreeViewOneToMany* and set all necessary properties.



The builder works similar to the VFX - COneToMany Builder. You can make your setting in the EditPage and FormOptions und Child Grid pages in the same way, as for forms based on the class *cOneToMany*. In addition to this you must make settings for the Treeview control in the page TreeViewOptions.

The settings that you need to make are same as in the VFX –CTreeViewForm Builder

### 8.10.1. Databinding of the TreeView control

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

*ParentIDFieldName* - This property holds the name of the field, where will be stored the Primary key value of the parent data 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 value will be used as NodeText in the tree structure. When a field name is used, the developer can allow the end-user to edit the description text directly into the TreeView control. This also depends of the *AllowNodeRename* property. If *AllowNodeRename* is set to .T., the user can edit labels in the TreeView control and this will automatically update the corresponding data field in the main table.

*AllowNodeRename* - This property defines if the user is allowed to edit description text in the TreeView control. Editing the description in the TreeView control is allowed only when the Node Text is based on single table field. The content of the corresponding data field, will be automatically be updated according new description entered in the TreeView control.

### 8.10.2. Layout-Settings of the TreeView Control

These settings are correspondent 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

### **8.11. Enhancements in OneToMany-Forms**

There are several improvements in *cOneToMany* and *cTreeviewOneToMany* form classes, comparing with former VFX versions.

- Buttons for adding and deleting child records are now enabled only if the form is in Edit or in Insert mode.
- The child part now can contain also other controls than *cChildGrid* which can be placed similar to parent edit page.
- The Edit pages in child part of one-to-many VFX forms can be created using builder in same way as you create edit page in parent part of form.

*CChildGrid* class, which is implemented in all one-to-many VFX forms, is now expanded with several new functions.

- When the child alias is based on a view or *CursorAdapter* cursor, the *cChildGrid* class allows using incremental search for columns.
- Clicking on empty area of the *cChildGrid* class adds a new child record.

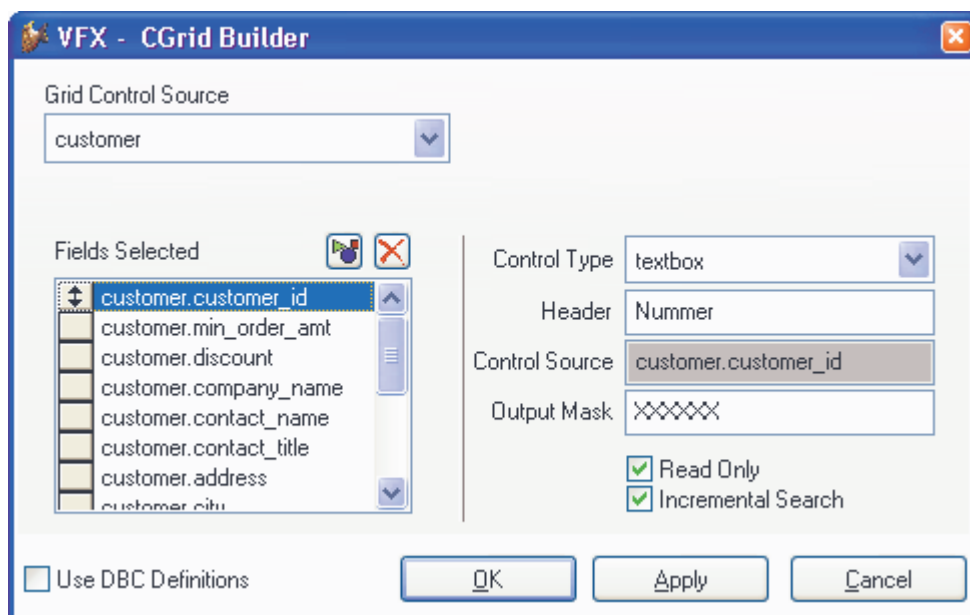
### **8.12. VFX – CGrid Builder**

Although the VFX – Form Builder already generates a Grid Page, you may have the need to make modifications only on a grid. The VFX Grid Builder automates the creation of full featured grids. The resulting VFX Power Grids are powerful yet simple to use and do not have any performance penalties. You will find the features of the power grids extremely useful. The incremental search, the user-specific save and restore of column layout changes, column size and sort order will be appreciated by the users of your applications.

To call the VFX Grid Builder, select the last page in your form, and select the Grid Control. To call the Builder right click with your mouse and select builder. Of course you could also select the grid object in the property sheet and select builder from within the property sheet.



The VFX Grid Builder loads and presents this Dialog:



Since the user interface is the same as the one used on the grid page of the form builder, for a detailed description of all the options, please refer to the description under the topic called *The VFX - CDataFormPage builder*.

### 8.13. VFX – CChildGrid Builder

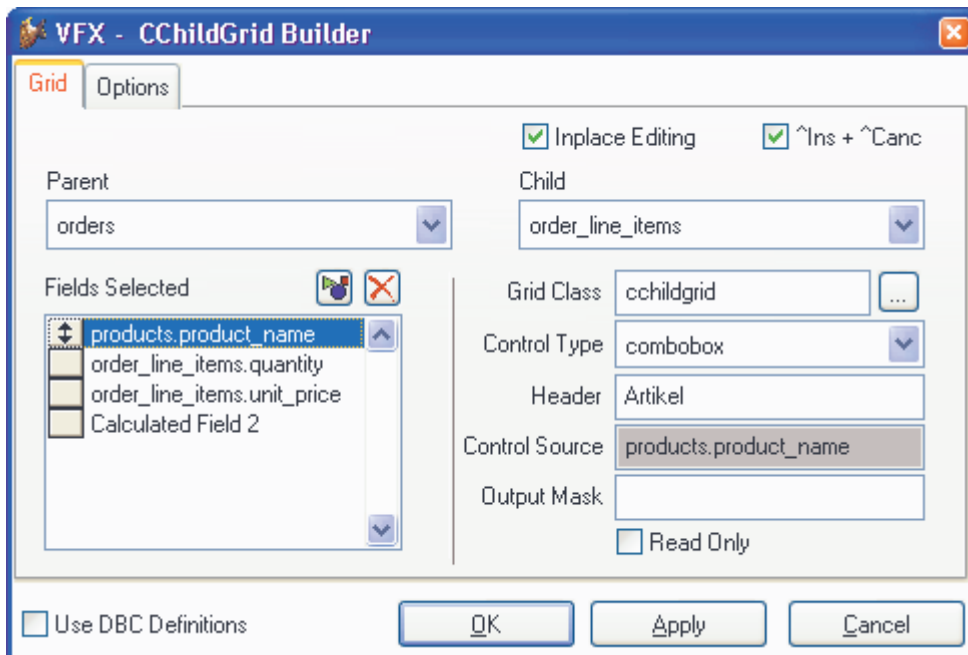
The Child Grid Builder, allows you to enhance the functionality of your child grids. Use this Builder to either customize some fields or to edit the method code *OnPostInsert()*, which will fire whenever a new child record has been inserted. It's similar to the standard VFX data manipulation form, where you have the same granularity of events:

- *OnPreInsert()*
- *OnInsert()*
- *OnPostInsert()*

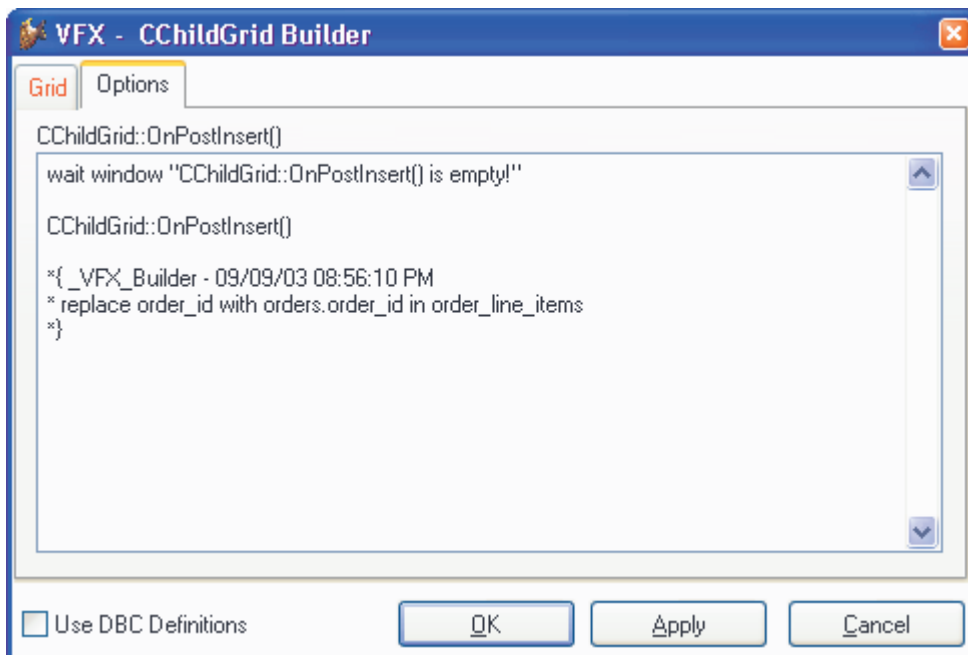
In the *OnPostInsert()* method of the child grid, you have to replace the child table field which makes up the link to the master table typically with a code like this:

```
REPLACE <ChildLinkField> WITH <Parent.ParentField> IN <ChildTable>
```

The Child Grid Builder has the user interface shown below. On the first tab called *Grid*, you can customize the child grid as described earlier:



On the second tab called *Options*, you can edit the code for *OnPostInsert()* method, so that it replaces the foreign key field in the child table with the value of the parent field.



The reason why the VFX Builder can not automatically generate this *OnPostInsert()* code is that you might have the situation, where you have a combined key or other situations, where you have to replace more than one field in the child table. When a single field key is used, the generated code in the example is correct and you need only to remove comment characters in the beginning of the lines.

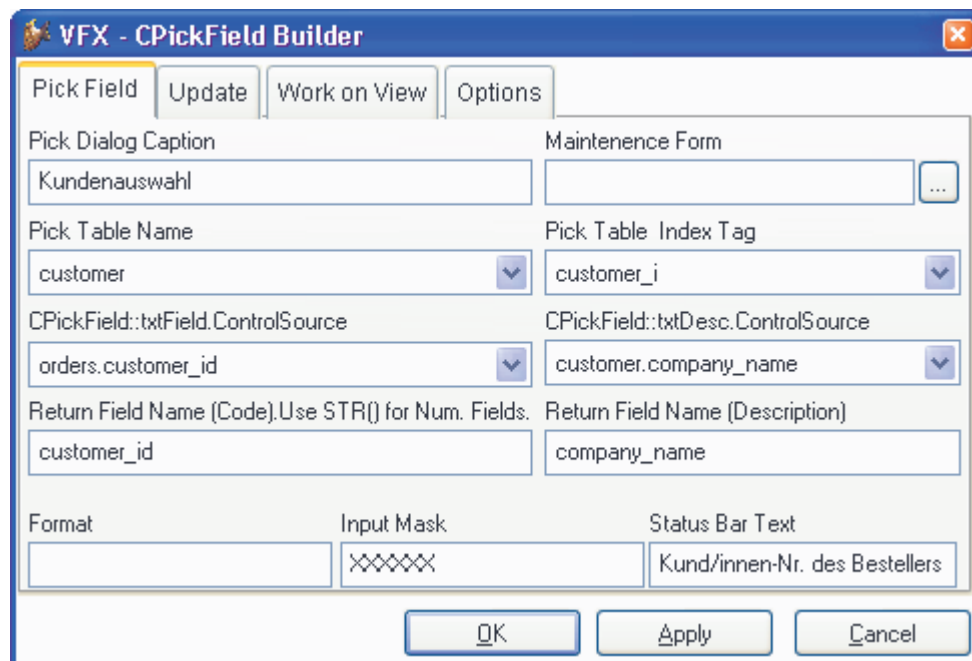
## 8.14. VFX - CPickField Builder

VFX contains several classes for Picklists. A Picklist consists of a text field, a button and a read-only text field. In the text field can be entered a value. When leaving the field it is checked whether the entered value is contained in the table with the picklist values. If no, a Picklist form is started for selection. In the Picklist form the user can select the desired data record. In a read-only text field can be displayed further information from the picklist table. When needed, the user can be allowed to insert new data records in the picklist table. All properties of the Picklist control can be set with the VFX – CPickField Builder. All this without entering a single line of code or text in the property sheet of the picklist container control manually!

To call the VFX Picklist Builder, be sure to select the picklist container control on the form, right click and select Builder from the context menu.

**NOTE:** To select a control which is on a Page, within a Pageframe, within a Form, you have to get used to the Visual FoxPro way of accessing controls in a container hierarchy (Click, RightClick, Edit or to use Ctrl+Shif+Click). A good way to check whether you are on the right object is to have a look at the property sheet's current object.

The VFX Picklist Builder loads and shows the following dialog:



The following options are available on the page *Pick Field*:

**Pick Dialog Caption.** Enter the caption for the picklist form in which the user can select the value he wants to pick.

**Maintenance Form.** If the user doesn't find the desired record in the picklist form, you might want to offer the user the ability to directly call the data maintenance form (view mode or directly insert mode) for the table currently being picked from. Enter the name or the data manipulation form which will be called if the user clicks on the *Edit* command button on the picklist form.

**Pick Table Name.** Select the name of the table or the view you want to validate and /or pick the value from. Here you can select from all tables or views you entered in the data environment.

**Pick Table Index Tag.** Select the name of the Index tag will be used for validation of the user input.

**CPickField::txtField.ControlSource.** Select the control source for the input text field.

**CPickField::txtDesc.ControlSource.** Select the control source of the description field of the picklist field. Make sure that you set a correct relation to the table you are selecting this control source from, otherwise this control will not refresh correctly when you move the record in your main form.

**Return Field Name (Code).** Enter the name of the field from the picklist table or view you want to get the value from. Do not enter any alias, since the pick table will be opened with a random alias.

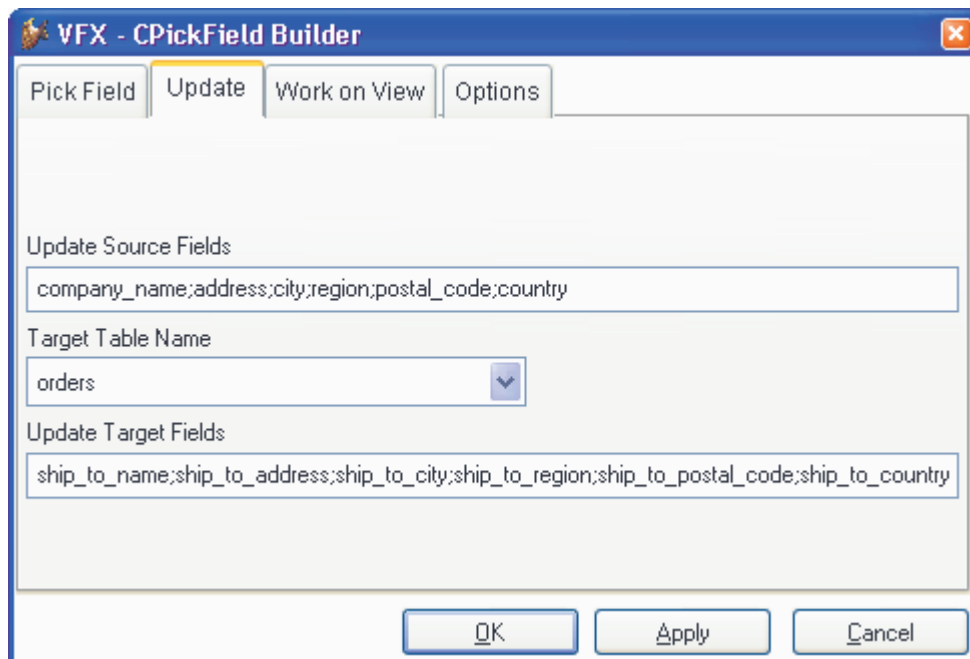
**Return Field Name (Description).** Defines which field from the picklist will be used to get the description of the selected record. Do not enter any alias, since the pick table will be opened with a random alias.

**Format.** The VFX Pickfield Builder takes this property from the database container.

**Input Mask.** The VFX Pickfield Builder takes this property from the database container.

**Status Bar Text.** The VFX Pickfield Builder takes this property from the database container.

The following options are available on the page *Update*:



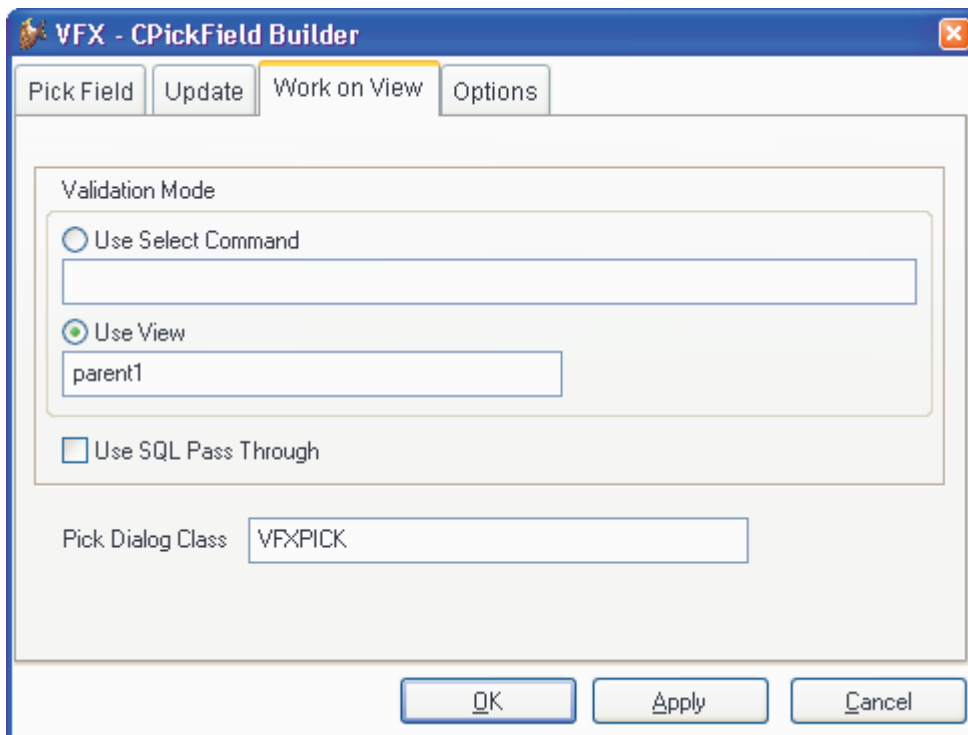
The screenshot shows a dialog box titled "VFX - CPickField Builder" with four tabs: "Pick Field", "Update", "Work on View", and "Options". The "Update" tab is selected. The dialog contains three text input fields and a dropdown menu. The first field, labeled "Update Source Fields", contains the text "company\_name;address;city;region;postal\_code;country". The second field, labeled "Target Table Name", is a dropdown menu with "orders" selected. The third field, labeled "Update Target Fields", contains the text "ship\_to\_name;ship\_to\_address;ship\_to\_city;ship\_to\_region;ship\_to\_postal\_code;ship\_to\_country". At the bottom of the dialog are three buttons: "OK", "Apply", and "Cancel".

**Update Source Fields.** Here you can enter fields from the Picklist table, which values will be stored back into the edited table. When you enter more than one value, you must separate them with semicolon.

**Target Table Name:** Choose the target table. Usually this is the table that is processed in the form.

**Update Target Fields:** Choose the fields for updating. When you enter more than one value, you must separate them with semicolon.

On the page *Work on View*, are available the following options:



**Work on View.** If the datasource that you are picking from is a view, mark this option.

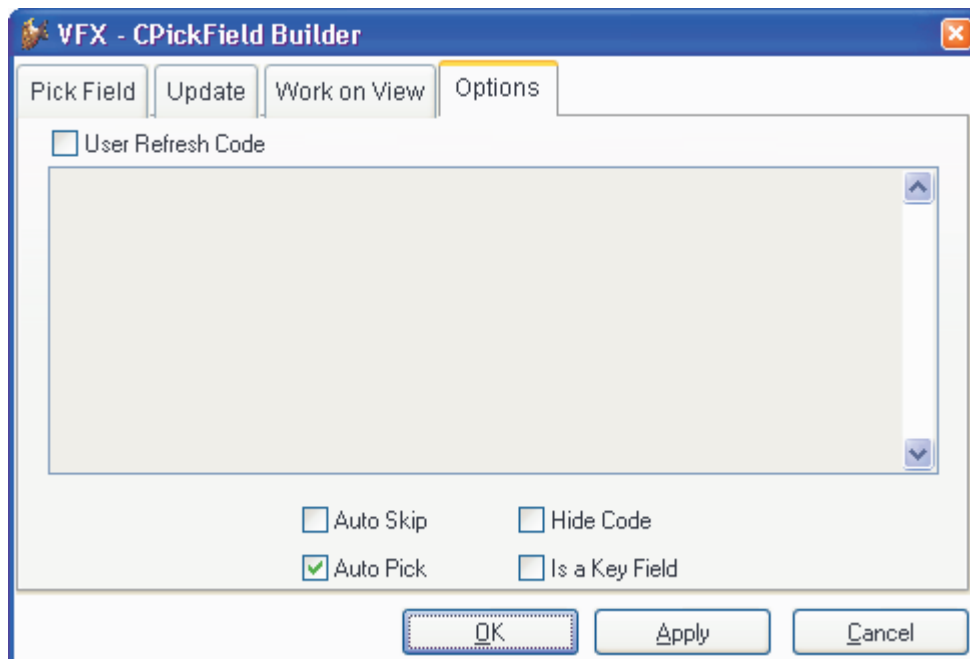
**Use Select Command:** Alternatively you can use a select command or a view for validation of the user input. When you use a select-command, you must ensure through a where clause that at most one value will be selected. Example: „*select customer\_id from lv\_customer where customer\_id = trim(this.txtField.Value)*”

**Use View:** Alternatively you can use a select-command or a view for validation of the user input. When you use a view, enter the name of the view here. The where clause of the view must ensure that at most one value will be selected.

**Use SQL Pass Through:** When you check this checkbox, the select-command contained in the view will be used from the VFX and trough SQL Pass Through will be sent to the remote datasource.

**Pick Dialog Class:** Here can be used your user-defined class for the selection list control. Please note that this class must be inherited from the *CPickField* class.

On the page *Options*, are available the following options:



**User Refresh Code.** Sometimes you need to have custom code in the *Refresh()* method of the picklist container.

**Auto Skip.** Check this option if you want to automatically tab to the next field after selecting a value from the picklist. This sets the *cPickField* property *lUseTab* to *.T*.

**Auto Pick.** Check this option if you want to automatically call the picklist, when the user enters a wrong value. This sets the *cPickField* property *lAutoPick* to *.T*.

**Hide Code.** Check this option if you want to hide the code field in the picklist. This sets the *cPickField* property *lHideCode* to *.T*. The user cannot enter a value, besides he can only choose from the picklist.

**Is Key Field.** Check this option if you want to define this pickfield as a key field which is only accessible when entering a new record and then no more (like the textbox class *ckeyfield*). This sets the *cPickField* property *lKeyField* to *.T*.

**OK.** The selected options will be used and put in the selected picklist object.

**Apply.** Does the same as *OK* but does not quit the VFX – CPickField Builder dialog.

**Cancel.** Cancels the VFX – CPickField Builder process. Any selections and entries will be lost.

This builder is fully reentrant, too. This means that during the development cycle you can call this builder as many times as needed without losing any of the settings which have already been defined.

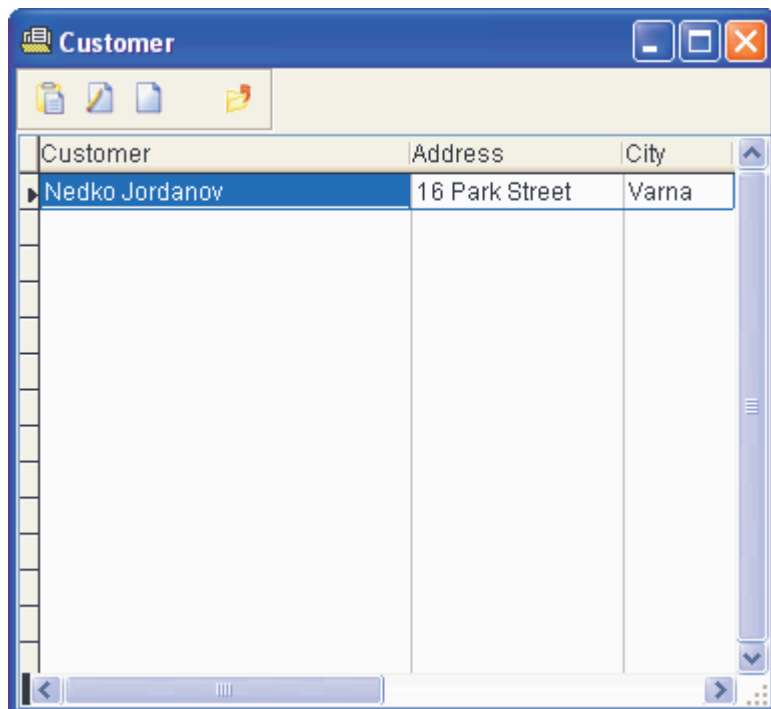
If you use a Picklist Control on a form, it might look similar to this:



The user can call the Picklist by:

- clicking on the command button next the Picklist Input Field (normally a three dot caption Icon),
- doubleclicking in the Picklist Input Field or the Description, or

- by pressing the Function Key F9



From the Picklist Dialog there are (like in any VFX Power Grid) features like:

- incremental search with autosort,
- sorting by doubleclicking on the column header,
- change column width
- autosave grid position and layout

The user can select the desired record by:

- doubleclicking,
- pressing *ENTER*
- selecting the *OK* command button

If the user wants to edit the table which is behind this picklist, he can click the command button *Edit...* and the data manipulation form for this table will be called. When the user need to add a new record, he clicks the *New* button.

### 8.15. VFX – *CPickAlternate Builder*

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 value. When using the class *cPickAlternate*, the primary key of the chosen PickList record will be stores into the correspondent field in the main table, while the user will see the value of another field of the PickList table.

Use the *cPickAlternate* control rather than a Combobox Control, whenever the choice will be made from a table with lots of records. It is also suitable, when if you want to give the user the ability to enter a value, that does not correspondent to the primary key of the Picklist table. The purpose of the class is to provide the end-user with an easy-to-use interface that allows well known values to be entered, instead of program generated primary 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 primary key value will be passed back to the *cPickAlternate* control and will be used to update the related data in the main table.

This class is based on *cPickField* class 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 where will be stored the foreign key value. This foreign key correspondent to the primary key in the PickList table.

With the VFX – CPickAlternate Builder you can easy set all necessary properties of the class.

*Pick Table Name* – Here can be chosen the name of one of the tables from the Dataenvironment.

*Pick Table Index Tag* – This is the name of the Index tag that will be used for search in the Picklist table. This index key corresponds to the value entered in the text box control.

*CPickAlternate.txtField.ControlSource* – The Controlsouce of the text box control. This field will come from the Picklist table.

*CPickAlternate.txtDesc.ControlSource* – The name of the description field. After successful validation its value will be displayed in the description field. This field is also from the Picklist table.

*Return Field Name (Code)* – The name of the field which value from the selection table, will be shown to the user. Usually this field name corresponds to the name, which is filled into txtField.ControlSource. Here should be entered only the field name without the table name. The value of this field must be of character type. If it is necessary, convert the value with TRANSFORM() into a character type.

*Return Field Name (Description)* – The name of the description text field, which will be returned back from the Picklist table. An expression can also be returned. The value is displayed in the description field. The value must be of character type. If it is necessary, convert the value with TRANSFORM() into a character type.

*Return Field Name (Internal Key)* – The name of the field from the Picklist table, which contains the primary key. The relationship from the main table to the Picklist table in the data environment is maintained trough this field.

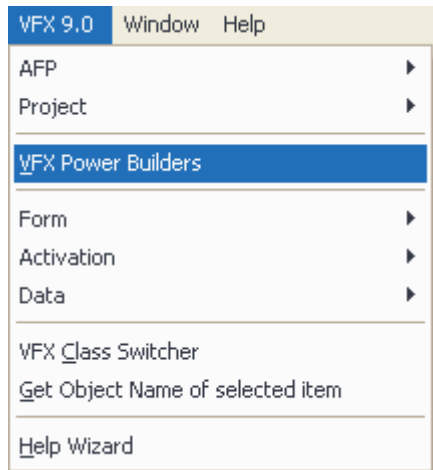
*Control Source Internal Key* – The name of the field from the main table, where the cPickAlternate class will store the key value. This field contains the foreign key to the Picklist table.



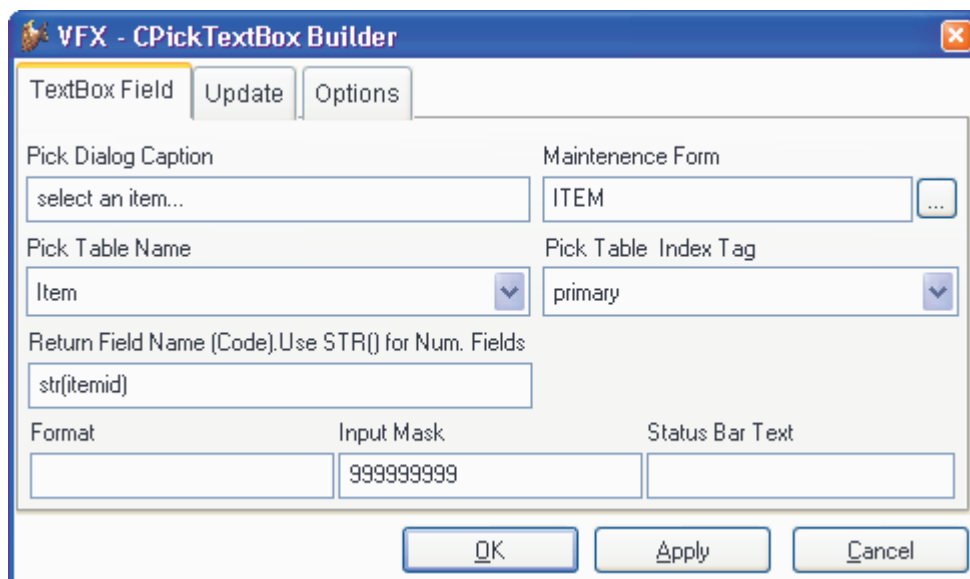
## 8.16. VFX – CPickTextBox Builder

Visual Extend offers a Builder which helps you to create full featured Picklist Controls.

To call the VFX - CPickTextBox Builder, select the column in the child grid, which should become a picklist control and select the option VFX Power Builders from the VFX Menu:



The VFX - CPickTextBox Builder offers a similar user interface like the normal VFX - CPickField Builder and is also fully reentrant:



The screenshot shows the 'Update' tab of the 'VFX - CPickTextBox Builder' dialog. It features three tabs: 'TextBox Field', 'Update', and 'Options'. The 'Update' tab is active. The dialog contains the following fields:

- 'Update Source Fields': An empty text input field.
- 'Target Table Name': A dropdown menu with 'Parent' selected.
- 'Update Target Fields': An empty text input field.

At the bottom, there are three buttons: 'OK', 'Apply', and 'Cancel'.

The screenshot shows the 'Options' tab of the 'VFX - CPickTextBox Builder' dialog. It features three tabs: 'TextBox Field', 'Update', and 'Options'. The 'Options' tab is active. The dialog contains the following fields:

- 'Work on View': A checkbox that is unchecked.
- 'Is a Key Field': A checkbox that is unchecked.
- 'Pick Dialog Class': A text input field containing the value 'VFXPICK'.

At the bottom, there are three buttons: 'OK', 'Apply', and 'Cancel'.

### 8.17. VFX – Combo Pick List Builder

The class is designed for easy creation and maintenance of pick lists. Using this class it is possible to define many pick lists without need to create pick table for every particular list.

The class *cComboPicklist* uses two VFX system tables: *Vfxpdef.dbf* and *Vfxplist.dbf*.

The table *Vfxpdef.dbf* holds definition for pick lists. It contains one row for every defined pick list. For picklist definition can be written code that will be executed when user makes a selection. This is a common code and executes regardless of specific list item that is selected. In *Vfxplist.dbf* table you can write more specific code that will be executed only if that item is selected.

Pick items are defined in *Vfxplist.dbf* table. Field *Picklist* is the identifier of the picklist definition and is foreign key to *Vfxpdef.dbf* table. The fields *Code* and *Describe* are values used for creating the list. Depending on picklist definition, only *Code* column or both *Code* and *Describe* columns are displayed in the list. Fields *Value* and *Proccode* are designated for developer's purposes. In *Proccode* field you can write a specific code that will be executed when that particular item is selected.

For every instance of *cComboPicklist* class you can define if adding new records in the pick table is allowed and the user level required for user to be allowed to add new values.

Properties:

*nParentID* – ID key value from *Vfxpdef.dbf* table

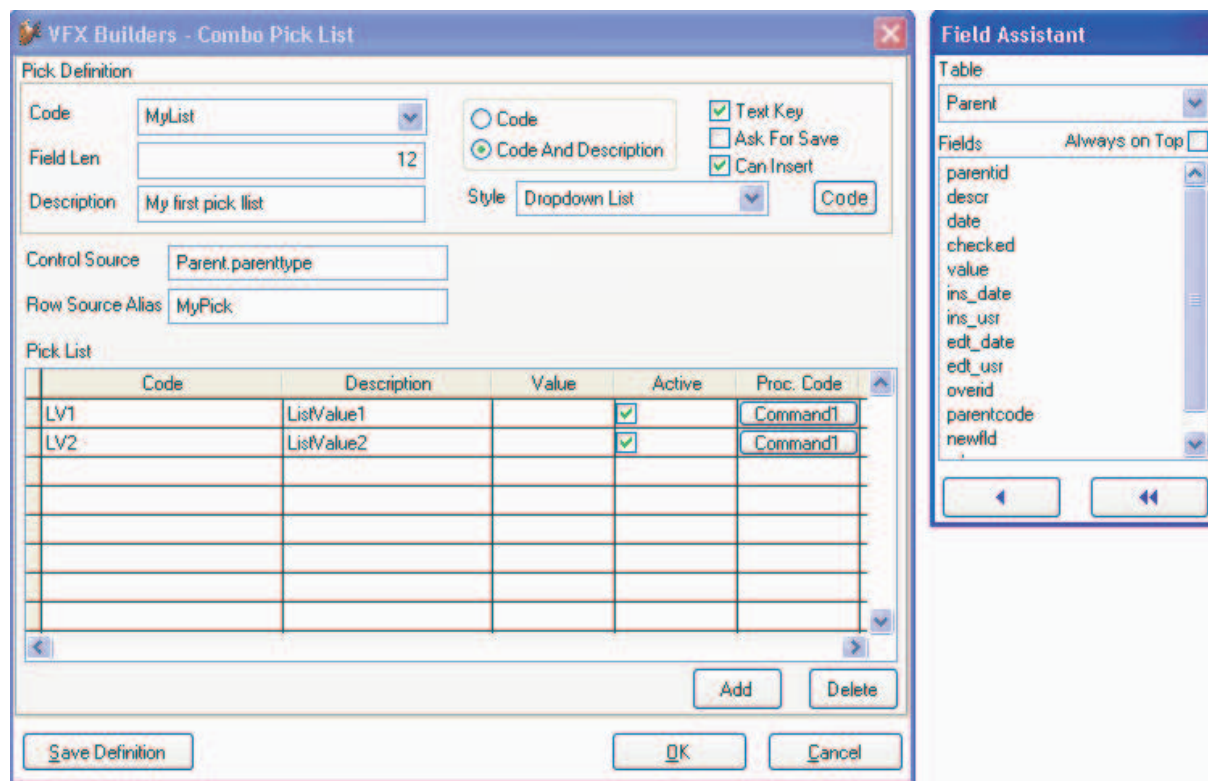
Methods:

*Addnewcode* – this method is executed when the end-user adds a new value in the pick list table. If it is needed to perform additional processing, the code should be placed in this method.

For *cComboPicklist* class it is possible to define two different code snippets. In *Vfxpdef* table in *ProcCode* memo field and in *Vfxplist* table in *ProcCode* memo field.

The code in *Vfxpdef.ProcCode* table is common for the *cComboPicklist* and is executed every time when the selected item of the Combobox is changed. The code in *Vfxplist.ProcCode* table is specific for the particular list item and is executed only when this item is selected from the list.

For every entry in *Vfxplist.dbf* table you can define if this is an active entry or not. Using this approach you do not need to delete entries which will not be used in the future and thus to cause orphan records in your database. When the field will not be used anymore, just set the values in its *Active* field to *.F.*



The settings of the class *cComboPicklist*, as well as content of tables *Vfxpdef.dbf* and *Vfxplist.dbf* can be edited with VFX – Combo Pick List Builder.

For the *cComboPicklist* you have to choose the *ControlSource* and the *Rowsource* alias. If in the form's data environment there is already same alias as the one chosen for *Rowsource*, you are asked if that alias should be used or a new one has to be created. If the *Rowsource* alias is not in the data environment of the form, the builder creates a new cursor object for it and sets its properties accordingly.

### 8.17.1. Pick list maintenance form

Code	Descript
LV1	ListValue1
LV2	ListValue2
SLV1	Second pick list value 1
SLV2	Second pick list value 2

Code: [MyList] (dropdown menu showing MyList and MySecondList)

Picklist: [ ]

Descript: [ListValue1]

Active

Value: [ ]

This form is designated to give the end users of the application ability to maintain used pick list values. For this purpose in every VFX 9.5 project is included the form *VFXPlist.scx*.

User can navigate through the full list or to filter visible data records by list code. It is possible to delete a record from the table, but the proposed approach it to mark unused rows as non-active, in order to avoid orphan records in the database.

### 8.17.2. The class CComboPicklist

This class is designated for easy creation and maintenance of pick lists. Using this class it is possible to define many pick lists without need to create pick table for every particular list.

The class *cComboPicklist* uses two VFX system tables: *Vfxpdef.dbf* and *Vfxplist.dbf*.

The table *Vfxpdef.dbf* holds definition for pick lists. It contains one row for every defined pick list. For picklist definition can be written code that will be executed when user makes a selection. This is a common code and executes regardless of specific list item that is selected. In *Vfxplist.dbf* table you can write more specific code that will be executed only if that item is selected.

Pick items are defined in *Vfxplist.dbf* table. Field *Picklist* is the identifier of the picklist definition and is foreign key to *Vfxpdef.dbf* table. The fields *Code* and *Descript* are values used for creating the list. Depending on picklist definition, only *Code* column or both *Code* and *Descript* columns are displayed in the list. Fields *Value* and *Proccode* are designated for developer's purposes. In *Proccode* field you can write a specific code that will be executed when that particular item is selected.

For every instance of *cComboPicklist* class you can define if adding new records in the pick table is allowed and the user level required for user to be allowed to add new values.

#### Properties:

*nParentID* – ID key value from *Vfxpdef.dbf* table

## Methods:

*Addnewcode* – this method is executed when the end-user adds a new value in the pick list table. If it is needed to perform additional processing, the code should be placed in this method.

For *cComboPicklist* class it is possible to define two different code snippets. In *Vfxpdef* table in *ProcCode* memo field and in *Vfxplist* table in *ProcCode* memo field.

The code in *Vfxpdef.ProcCode* table is common for the *cComboPicklist* and is executed every time when the selected item of the Combobox is changed. The code in *Vfxplist.ProcCode* table is specific for the particular list item and is executed only when this item is selected from the list.

For every entry in *Vfxplist.dbf* table you can define if this is an active entry or not. Using this approach you do not need to delete entries which will not be used in the future and thus to cause orphan records in your database. When the field will not be used anymore, just set the values in its *Active* field to *.F.*

## 8.18. VFX - Parent/Child Builder

Although there is a special VFX – Builder for the creating 1:n-forms, sometimes it is better to manipulate Child-data in its own form. That is in particular the case, when you want to use the child form not only through the main form, but also for direct manipulation. In addition, if you, have many fields on the Child-form, it can be difficult to edit them in a 1:n-form.

A special power of VFX is the use of the Linked Child technology. Thereby two forms are logically connected one to other. One form serves as a Parent form. Any VFX Form class can serve as Parent form. Also the Child form can be based on any VFX Form class.

When moving the record pointer in the Parent form, the content in the Child form will be automatically updated and the data records that correspond to the current Parent are displayed.

If the Child form is based on a table, a filter is used, in order to limit the visible data scope. When the Child form is based on a view, a *REQUERY()* is performed if needed, in order to display the desired data set. Thereby the underlying view must have exactly one parameter, which must correspond to the parent key.

A Parent form can call several different Child forms. A Child form can serve again as Parent for other Child forms.

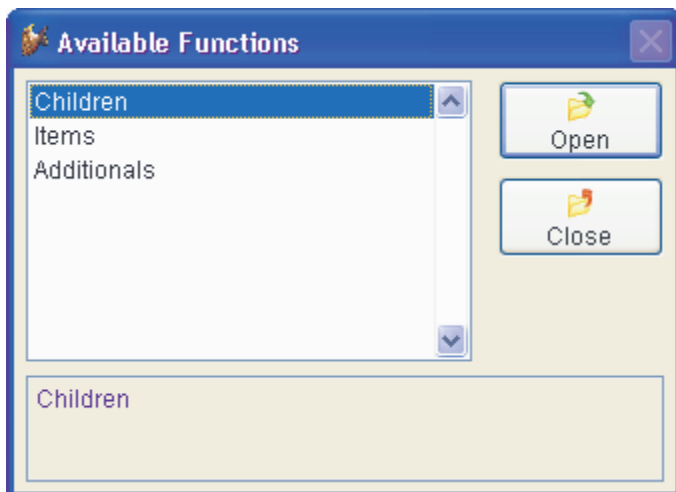
### 8.18.1. Preparing the Parent-Form

In the Parent form, with the help of the form builder, must be selected the following options *Has More Options* (sets the property *lMore* on *.T.*), *Has Child Form* and *Auto Sync Child Form* (sets the property *lAutoSynChildForm* on *.T.*). The form builder generates automatically template code into *OnMore()* and *OnSetChildData()* methods. The code of these methods must be manually adapted afterwards. The Child form is invoked in the method *OnMore()*.

If the user wants to see the available options to the current data record, there are different possibilities:

- User can press the function key *F6*.
- User can select option *Other...* in the *Edit* menu.
- User can click on the *More...* button in the standard toolbar.

Depending on the code in the method *OnMore()* the user will see a dialog, which looks similar to the following:



Calling the *OnMore()* method with the parameter *tnPassThrough* is very useful, if you want to start a form directly on the passed number. You can use this technique, in order to start a form from the *OnMore()* method on a button in a toolbar.

If there is only one option in the *OnMore()* method, the assigned form is opened, without this dialog to appear.

### 8.18.2. Preparing the Child-Form

Additionally, in the Child form with the form builder on the page Options, the VFX developer must select the option *Is Child Form* or to set manually to *.T.* the form property *lChildForm*.

When you call a form, pass the necessary parameters to the *Init()* method of this form. Since the passed parameters are not automatically visible for other methods of the same form, the VFX – form stores the necessary parameters in special properties.

Here is the code of the *Init()* method, which the VFX – Form Builder produces as example for your needs:

```
lparameters tcArg
local lInitOk
if !empty(tcArg)
    if getArgCount(tcArg) <> 0
        this.cCalledBy      = upper( getArg(tcArg,1) )
        this.cFixFieldValue = strstran(getArg(tcArg,2), "@", ";")
        this.Caption       = getArg(tcArg,3)
        this.cFixFieldName = strstran(getArg(tcArg,4), "@", ";")
        this.cFilterExpr   = upper( getArg(tcArg,5) )

        this.lPutInLastFile = .f.

        *****
        ** Set who has called you

        if this.cCalledBy = "<CalledBy>"

            *****
            ** Disable CPickField that are Fix Fields for this form

            *{PickFieldList}*
        endif
    endif
endif
this.SetQueryArg()
lInitOk =eval(this.class+":::init(tcArg)")
```

```
*****
** Insert your extra initialization code here
return lInitOk
```

The template code could look in such a way, if you adapted it to your needs:

```
lparameters tcArg
local lInitOk
if !empty(tcArg)
    if getArgCount(tcArg) <> 0
        this.cCalledBy = upper( getArg(tcArg,1) )
        this.cFixFieldValue = strtran(getArg(tcArg,2), "@", ";")
        this.Caption = getArg(tcArg,3)
        this.cFixFieldName = strtran(getArg(tcArg,4), "@", ";")
        this.cFilterExpr = upper( getArg(tcArg,5) )

        this.lPutInLastFile = .f.
        *****
        ** Set who has called you

        if this.cCalledBy = "PARENT"
            *****
            ** Disable CPickField that are Fix Fields for this form

            ThisForm.pgfPageFrame.Pagel.cntParentid.lFixField = .t.

        endif
    endif
endif
this.SetQueryArg()
lInitOk =eval(this.class+":init(tcArg)")
*****
** Insert your extra initialization code here
return lInitOk
```

The VFX function *getArg()* checks the parameter character string and divides it into its parts. The parts are separated by semicolon.

Look at the example. When we call the Contact form for a certain company, the passed parameter can have the following parts:

```
"COMP;1234567890;Contacts for company ISYS;CONT_COMP_ID;UPPER(CONT_COMP_ID)= '1234567890'"
```

The separated parts of this character string are stored in the already existing form properties, before they can be used within the entire form. Let's see the form properties, which hold the information from the passed parameter character string *tcArg*:

VFX – Form property	Description	Example
<b>cCalledBy</b>	This character string indicates, from which form this form was called.	COMP
<b>cFixFieldValue</b>	The value of the field from the main table ( <b>primary key in the Parent table</b> ).	1234568890
<b>Caption</b>	Title of the child form. Here is a meaningful tip for the parent to which data row corresponds.	Contacts for company ISYS
<b>cFixFieldName</b>	The name of the field in the Child table, which defines the 1:n-relation. This field obtains the value indicated above, if a new data record is added ( <b>foreign keys in the Child table</b> ).	CONT_COMP_ID
<b>cFilterExpr</b>	The (at best) Rushmore optimized Filter expression, in order to show the data records that correspond the criteria of the parent table.	UPPER(CONT_COMP_ID) = '1234568890'

For very large data sets, it is better to work with views. The VFX – processes work exactly in same way. If the Child data originate from a view, it is not necessary to pass the filter expression.



### 8.18.3. Settings in VFX - Parent/Child Builder

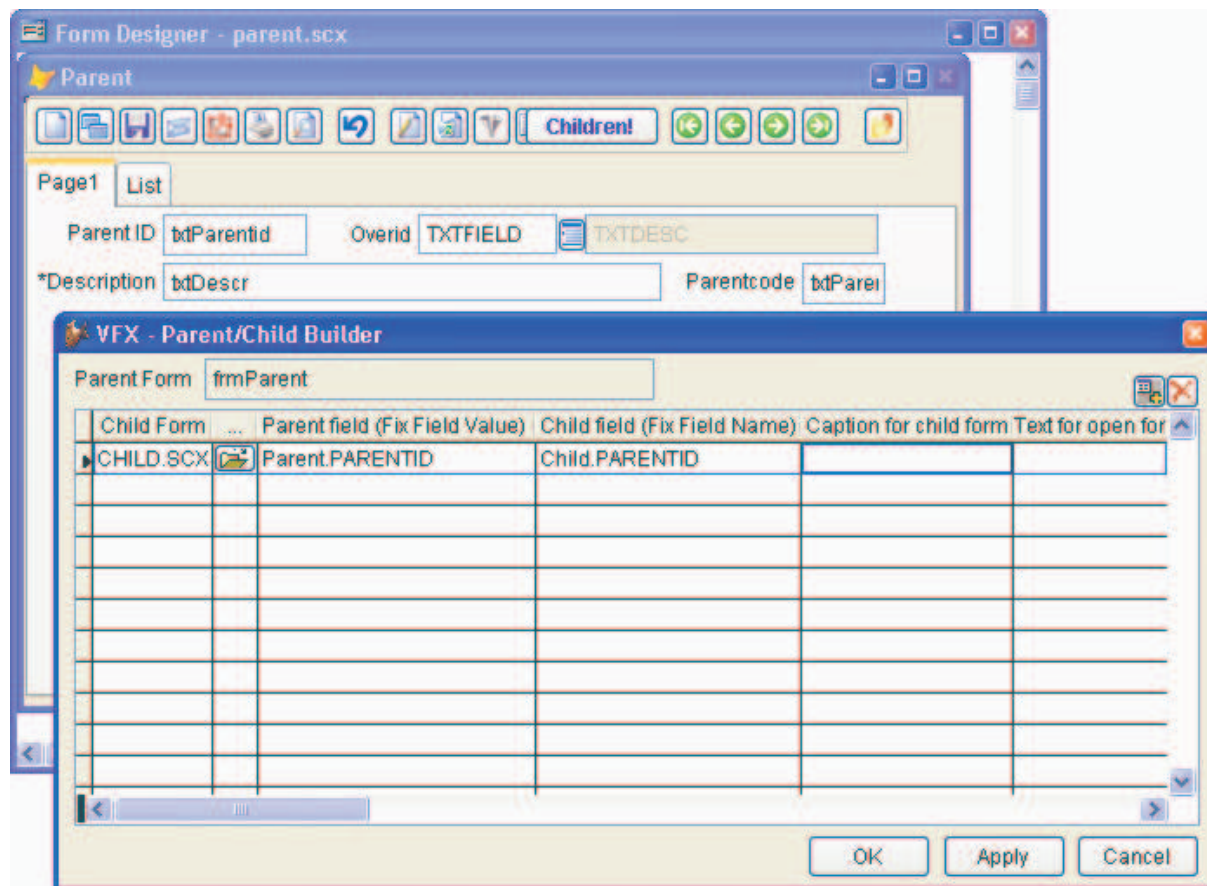
A Child form can be started by setting few properties in the *OnMore()* method of a Parent form. The key of the Parent form will be passed to the Child form. In the Child form are visible only the data, which correspond to the key of the Parent data record. The visible scope in the Child form can be limited alternatively with a filter or a view.

By adjusting some properties in the *OnSetChildData()* method in the Parent form, the simple Child form becomes a Linked Child form. That means that, if in the Parent form the record pointers one moves, the content in the Child form is automatically updated according to the current Parent key.

It is possible several Linked Child forms to be controlled at the same time by a particular Parent form. Both the Parent form and the Child form can be of any VFX Form types. It is possible an 1:n:m- relationship is to be realized, by using a OneToMany form as Linked Child form.

There is a powerful new builder in VFX 9.5 that helps creating code for managing parent-child relation between forms. For easy handling the relationship code, is used the new class *cChildManager*. To start the VFX - Parent/Child Builder, should be opened the parent form. While the parent form is active the VFX - Parent/Child Builder can be started from the VFX menu.

In the builder dialog can be added as many forms as needed.



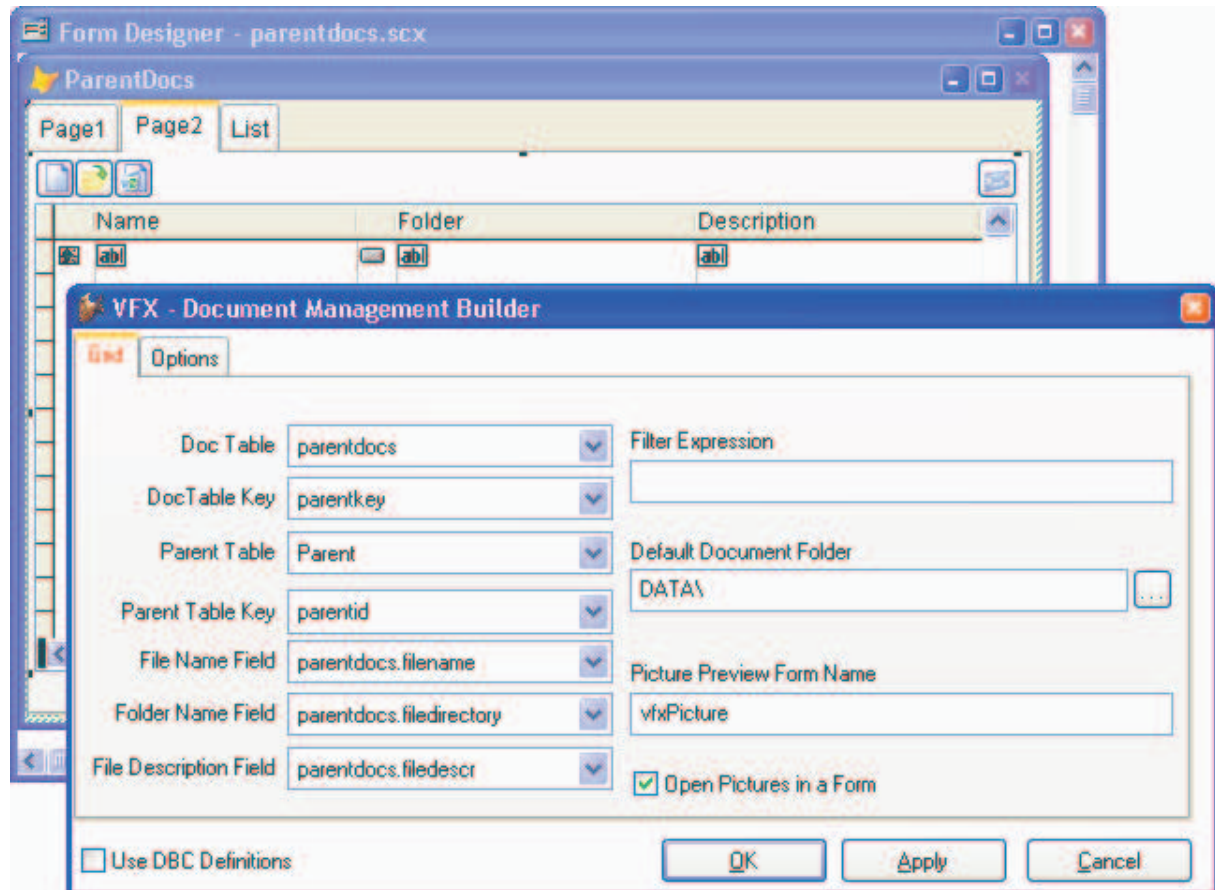
In the column *Child Form* is entered the name of the child form or the child form can be chosen using the Open button. In the column *Parent Field (Fix Field Value)* is entered the name of Id field in the parent alias. The value of this field will be passed to child form when the form is called and on every the record pointer move in the Parent form.

In the column *Child field (Fix Field Name)* is entered the name of the field in the child alias which is foreign key to the parent alias.



## 8.19. VFX – Document Management Builder

The new class *cDocumentManagement* is designated to maintain all kind of documents (i.e. Word, Excel, Powerpoint) related to application data. The *cDocumentManagement* class is a container that manages child records related to current record of main table. It allows the end-user to open related documents and also to send them as attachment in an e-mail.



The class can be added to the existing forms.

*cDefaultDocumentFolder* – Default folder for documents.

*cFilterExpression* – Filter expression to be applied.

*lOpenPicturesInForm* – If this property is set to *.T.*, picture documents are opened in a VFX form. The name of this form is specified in *cPicturePreviewFormname* property. If the property value is set to *.F.*, picture documents are opened with the application, which is associated with their file extension in Windows Explorer. The default value is *.F.*

*cPicturePreviewFormname* – Name of the form to be used to preview picture documents. The default values is *VFXPicture*.

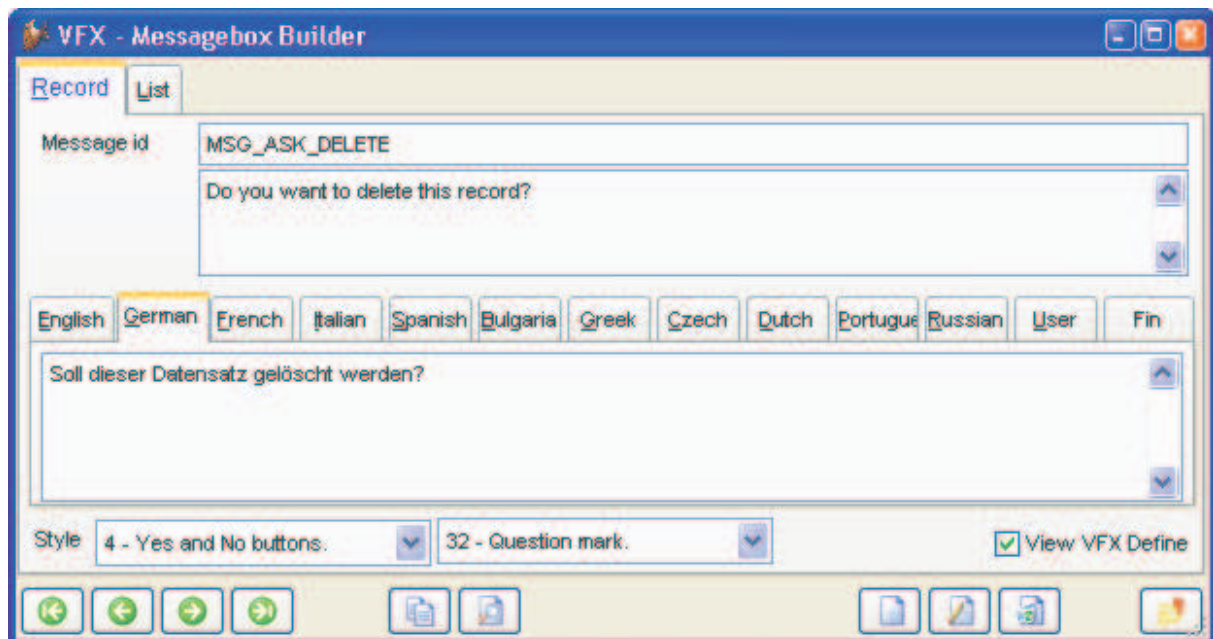
*cPicturePreviewCaption* – String which will be passed to picture preview form and will be used by that form as caption.

## 8.20. VFX – Messagebox Builder

A useful tool for the creation of Messageboxes in different languages is the VFX Messagebox Builder. The texts of the Messagebox are stored in the table *Vfxmsg.dbf*. The command for the invoking the Messagebox is copied into the clipboard and can be pasted from there to the own program source code. As parameter is passed not the

text, but a constant. The Include files with the values of the constants in the desired language are created with the VFX Message editor.

To call the VFX MessageBox Builder, select the menu option *Form, MessageBox Builder* in the VFX Menu:



Click on the button *New* to create a new MessageBox. Then enter in the field *Message id* a meaningful name for the MessageBox. In the Pageframe you can enter the text for each necessary language.

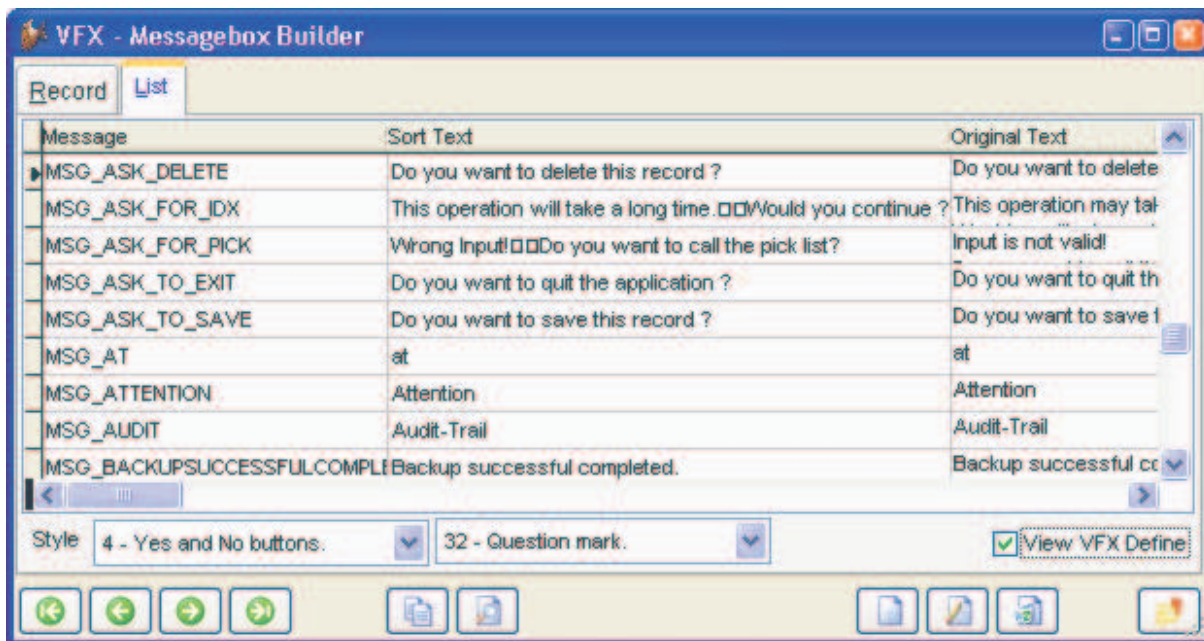
In the row *Style* select desired type of the MessageBox. You can chose among different icons and buttons.

By clicking on the button *Test it!* The MessageBox will be displayed in preview.

Copy in the clipboard the code, created by VFX – MessageBox Builder with the button *Copy code to clipboard*. Then you can paste this code from the clipboard in every part of your code.

For every entry the VFX – MessageBox Builder adds a new data record in the table *Vfxmsg.dbf*.

On the page *List* you have an overview of all existing data records.



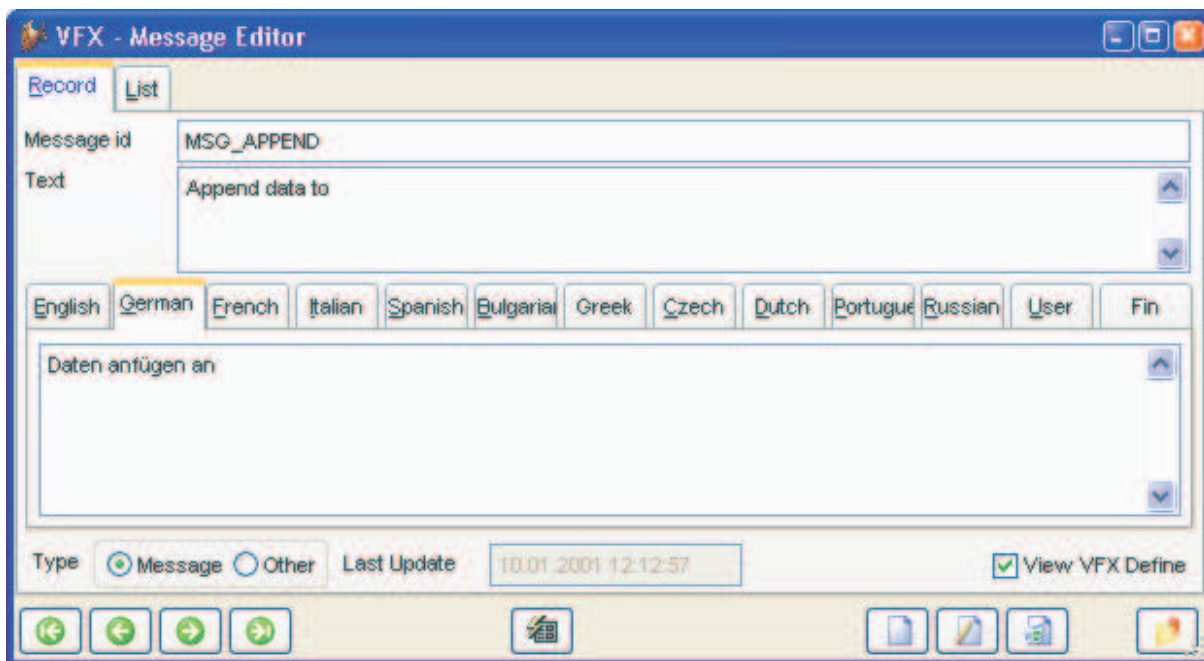
**Tip:** Also if you do not create multilingual applications, you can use the VFX – MessageBox Builder.

### 8.21. VFX – Message Editor

The values of all VFX used constants are located in the free table *Vfxmsg.dbf*. For each language exists a memo field containing the text. With the VFX Message editor these texts can be edited.

The VFX Message Editor is the central place to manage and translate all messages and other language specific text elements such as captions, tooltip texts and status bar messages. From within the VFX - Message Editor you can create all the needed include files (*Usertxt.h* and *Usermsg.h*).

To call the VFX - Message Editor, select menu option *Form, Message Editor* in the VFX Menu:



Click on the button *Make Include File* to create an include file in the language selected in the Pageframe. The Include files are saved in a folder with the name of the respective language under the Include folder of your project.

After the creation of your Include files, you must only copy these into the `\INCLUDE` file of your project, as described in the chapter concerning the multilingual applications.

---

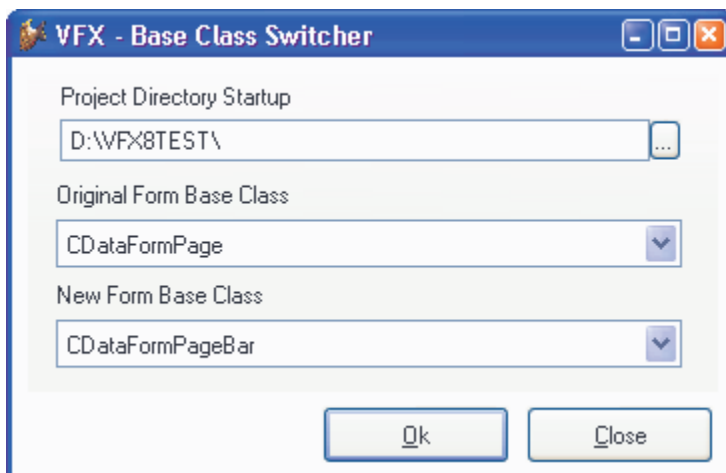
**Tip:** You can mix your own constants with the predefined constants in the table *Vfxmsg.dbf*. Write your constant before or after the VFX - Header and/or Footer.

---

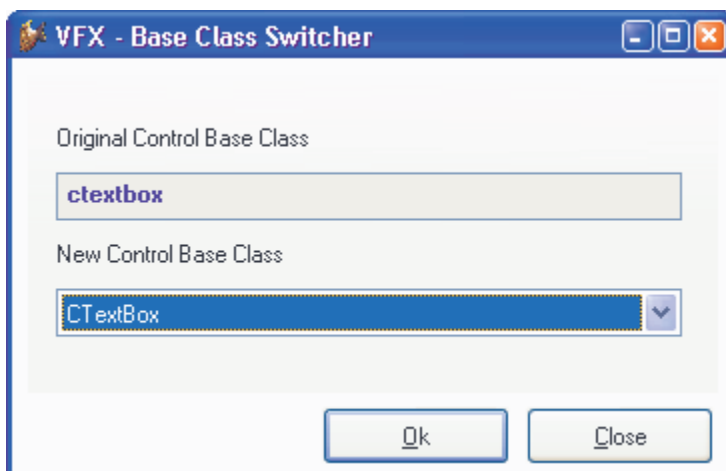
## 8.22. VFX – Class Switcher

The Class Switcher has two functions.

If no form is open, when it is called, the Class Switcher changes the classes of forms in a whole project. For example the form class *cDataFormPageBar* will be replaced by *cDataFormPage*. This Allows easy to provide all forms with navigation buttons and/or remove them again. As particularly helpful, this tools handles the actualization of existing VFX 3-Projekte. In VFX 3 each form had a frame a with button at the lower edge. In VFX 9.5 you can use instead a real toolbar.



When while opening the VFX - Class Switcher, a form is opened for edition, you can change the base classes of the particular objects. For example it is possible a Textbox to be changed to a Spinner.

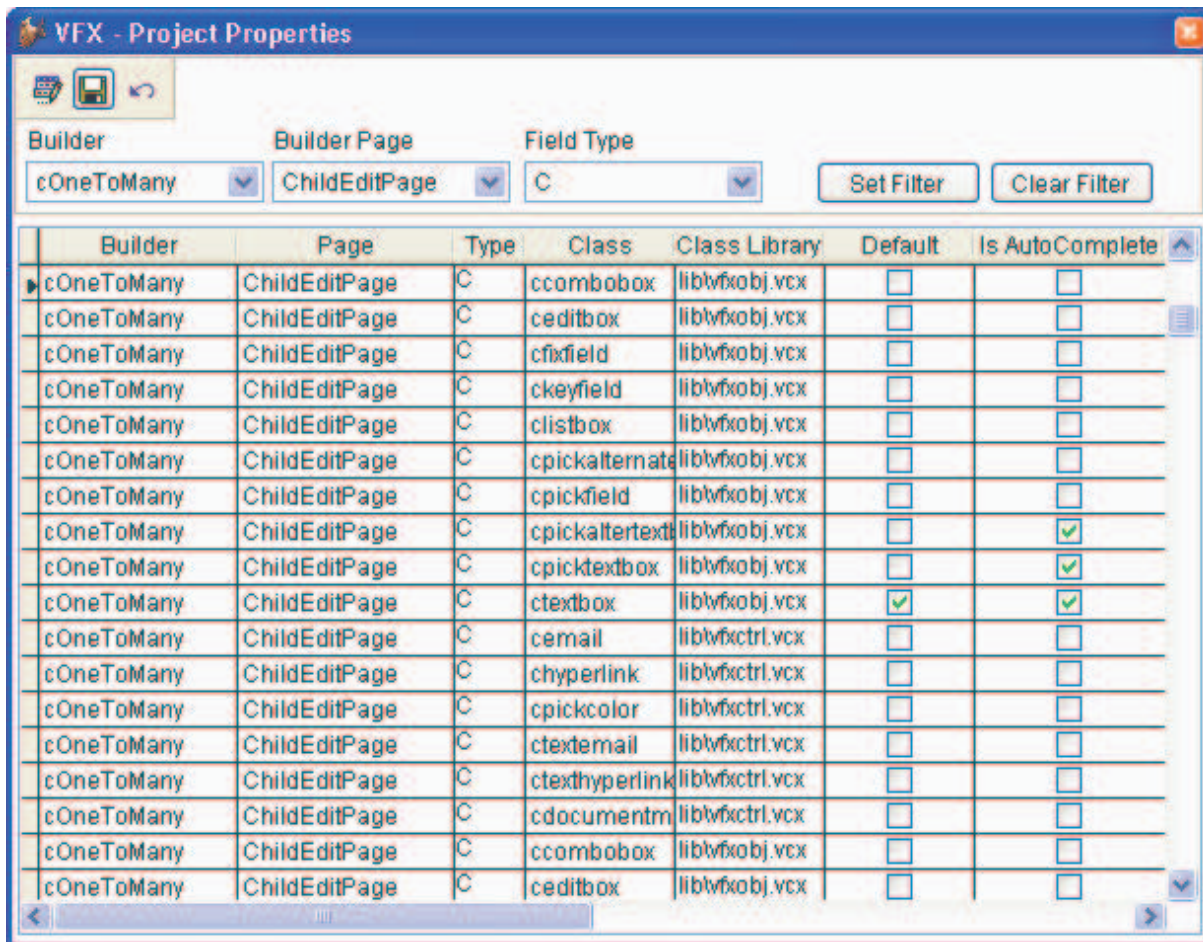


## 8.23. VFX – Project Properties

In VFX can be used own instances of the VFX classes. In the VFX – Project Properties dialog can be registered classes for the particular control types, which can be used. As default here stand the known classes from the



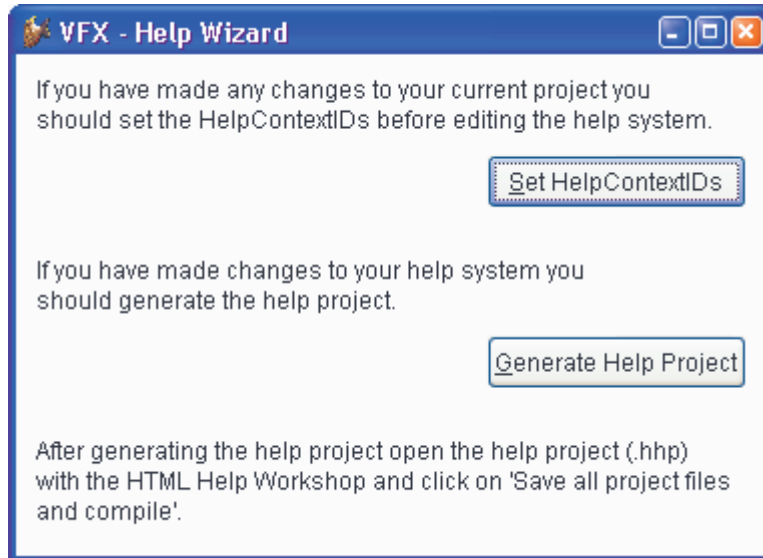
class library *Vfxobj.vcx*. The VFX developer can change these defaults and register own classes, which are stored preferably in the class library *Appl.vcx*. These classes can be used from the VFX – Builders when creating new forms.



## 8.24. VFX – Help Wizard

In VFX has been integrated a new System for creation of CHM Help files.

The VFX – Help Wizard automatically fills out unique *HelpContextIDs* in all controls of a project.



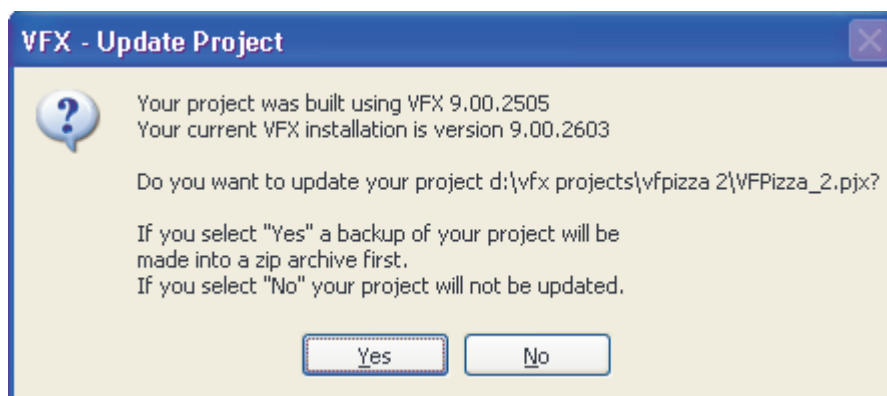
When at application's run-time, the table *Vfxhelp.dbf* is available, help texts can be entered in this table. For this purpose, the form *Vfxhelp.scx* will be opened. The actual help text can be entered into an editbox and will be saved into table *Vfxhelp.dbf*.

With the VFX – Help wizard from the data in the table *Vfxhelp.dbf*, completely automatically can be generated HTM files as well as a help project. It is just necessary to compile this project with the Help-Workshop and the CHM help file with context sensitive help for the entire application is ready.

When at run-time the table *Vfxhelp.dbf* is not available, the usual context sensitive help system will be activated. The CHM help file will be opened and the *HelpContextID* of the active control will be passed as parameter

## 8.25. VFX - Project Update Wizard

Projects, created with previous VFX versions or with previous versions of VFX 9.5, can automatically be actualized to the newest version.



The VFX - Project Update Wizard can be run from VFX 9.5 Menu using *Project, Update Project*. The VFX - Project Update Wizard compares version of the opened project with the currently installed VFX version. When

the project is created with an older VFX version, the developer is asked whether the project should be actualized.

After clicking Yes, the wizard starts. Upon this action a backup ZIP file of your project will be created. The ZIP file is created in the project folder and is named as the project file. If an archive file with this name already exists or cannot be created the wizard cancels its execution.

VFX Project Update Wizard renews VFX class libraries, VFX system report files and the procedure file *Vfxfunc.prg*. The table *Vfxmsg.dbf* in the project is merged with latest *Vfxmsg.dbf* table. All include files are newly generated.

The structure of VFX tables is actualized. Newly added fields and content are automatically added.

Missing files are added to the project automatically. For example these are new bitmaps files or free VFX system tables.

With this the VFX Update Project Wizard completes its work and saves a lot of time the developers. Generally, the updated projects will be capable of run immediately with the new version 9.5 VFX. However the developer should check the project carefully and when necessary make manual adjustments.

The most applications have for instance a specific menu *Vfxmenu*. The Update Project Wizard does not know which menu elements the developer removed. Because of this the wizard cannot add new menu elements. However the menu can be fast extended with new menu element, by comparing with the menu from the VFX 9.5 installation.

Check the new *Vfxmain.prg* and make by hand necessary changes according your application.

In former VFX versions were used public variables for all fields from tables *Vfxsys.dbf* and *Vfxuser.dbf*. In VFX 9.5 properties of an object are used instead.

Example:

```
Old: gu_myFeld New: goUser.myFeld  
Old: gs_myFeld New: goSystem.myFeld
```

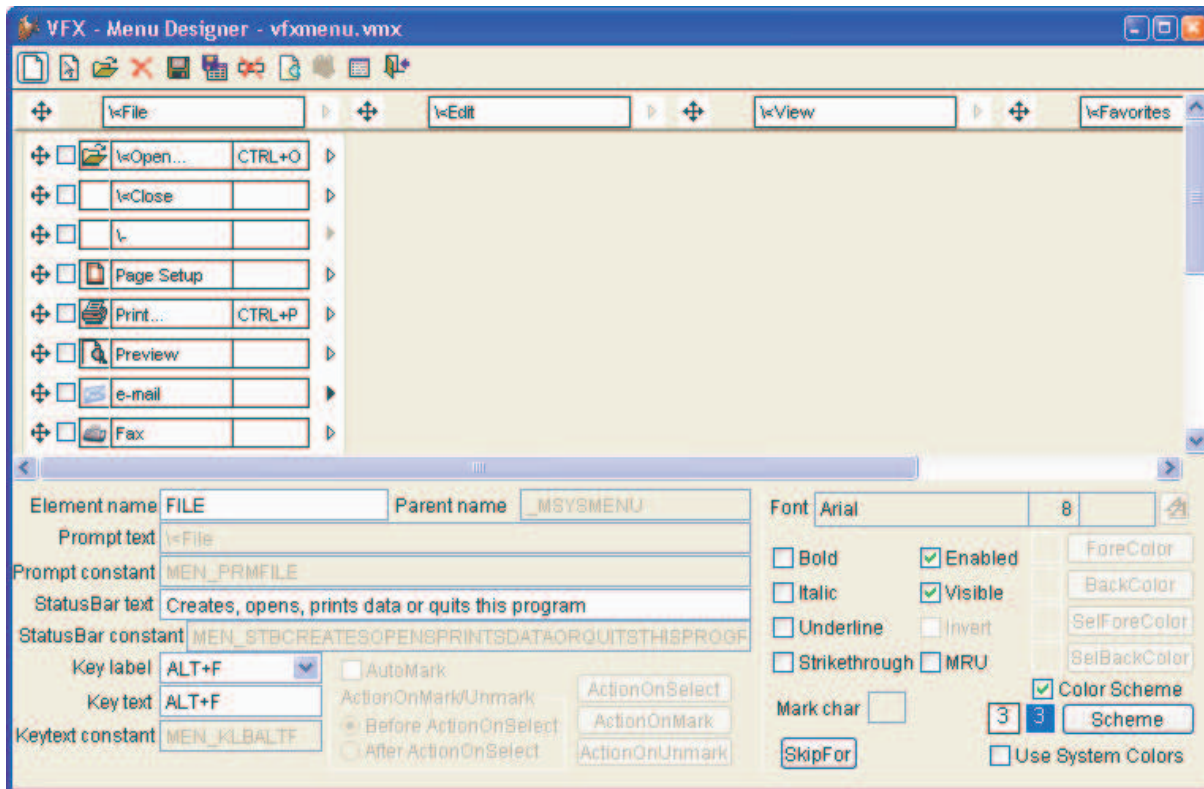
You can find all usages of *gu\_* and *gs\_* with the VFP 9 Code Reference Tool and easy and fast change all occurrences.

## **8.26. PDM – Project Documenting**

With VFX is delivered a special for VFX developers Project and Database Documenting Tool PDM. The PDM can be started from VFX 9.5 menu option *Project, Project Documenting*. It creates fully automatically a complete technical documentation. The documentation is created in HTML format and contains many cross references.

## 9. The VFX – Menu Designer

VFX Menu-Designer (VMD) is a tool for quick development of menus. The VMD is a visual designer where while development, the menu is shown in same way, as it will look like later at run-time. The VMD makes menu development simpler and allows you to set all menu properties, contrariwise of the VFP Menu-Designer, that does not support all menu properties. You can create multilingual menus by marking the check box in the Create New Menu dialog or later - by clicking correspondent button in the toolbar or choosing the option in the menu.



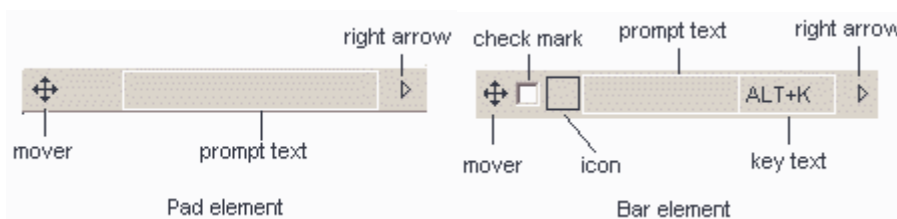
A menu contained in a VFX project can be opened with the VMD directly from the VFP-Project Manager. Alternatively menus can be opened also from the VMD using the Open button in the toolbar or using the appropriate menu option. In the Opening Dialog you can chose between the menu types *mnx* and *vmx*. If a menu, which was not edited with the VMD, is opened, it is automatically converted into the *.vmx* format.

The open menu can be edited visually. Man can add and delete elements and edit properties for the particular elements.

New menu pads can be added by clicking on the right arrow of the last pad in menu bar. Thus can be added a popup. You can add new bar in the popup By clicking on the down arrow at the bottom of the popup.

A menu bar or menu pad can be deleted when element has the focus. Use Menu option *Delete* or press *Ctrl+Del* to delete current element.

Some properties of the elements can be set visually:





### - Prompt text

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

### - Key text

The key text is used to show the user the Hot key or key combination for the element. It should correspond to the Hot key or key combination, chosen in the lower part of the VMD options panel, and is displayed in the menu at run-time to help users choosing relevant keys.

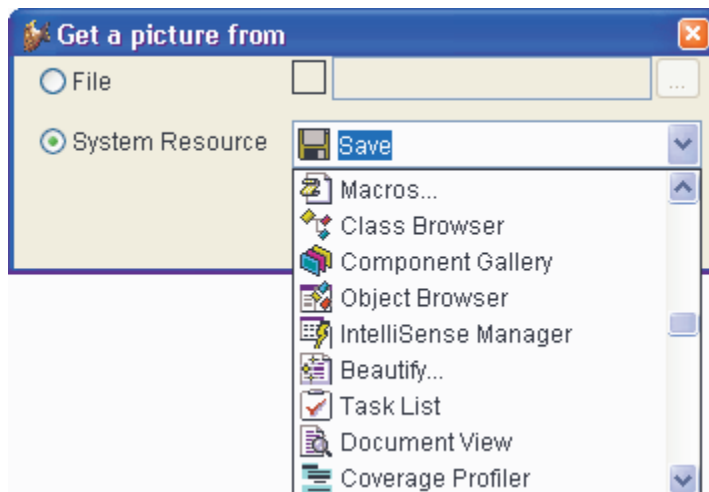
### - Check mark

In order the menu element to behave as a Checkbox, it is necessary to set the property *AutoMark*. When, in addition, you mark the element by *CheckMark* property, the element will appear initially marked when the menu is loaded. For Menu element, which behaves like a Checkbox, you can write specific code that will be executed, when the respective menu bar automatically changes its state to marked (*ActionOnMark*) and when it gets unmarked (*ActionOnUnmark*). The code is edited in Standard Visual FoxPro editing window.


The code that is entered into *ActionOnMark* or *ActionOnUnmark*, can be executed by your choice before or after the code that is executed in *ActionOnSelect*. To set this behavior, choose the desired option *Before ActionOnSelect* or *After ActionOnSelect* from the correspondent option group.

### - Icon

Each bar in a menu can be assigned an icon. This icon could be selected from a Visual FoxPro integrated system resources or another graphical file can be used. For selecting picture, is used the dialog *Get picture from*. It is invoked by clicking on the black border square for the particular menu bar element. In this dialog you can choose between a graphical file and an icon from the VFX system resources.



If for a menu bar element an icon is assigned, and at the same time the element is set to behave as a checkbox, the icon itself serves as a check mark. When the menu bar is checked the icon is displayed sunken, and when menu bar is unchecked – the icon appears normal.

Position of the particular menu element in the menu structure can be changed using drag and drop operation. For this purpose is used “4way” button  at the left part of all pads and bars. In some cases moving the elements is restricted: a pad element can not become a bar element and a bar element cannot become a pad element. In addition, it is not possible to move a bar in its child popup.

Other properties of the menu elements can be set in the Properties panel. These are the font, the foreground and background colors, a message that will be displayed in the VFP status bar, as well as constants that will be used when a multi-lingual menu is developed. All changes will be displayed in the designer in the active element..

With the button *ActionOnSelect* the code that will be executed when the menu element is selected, can be entered in an editor window. Using the button *SkipFor* can be entered a logical expression. When this expression evaluates to *.T.*, this menu element cannot be selected.

The properties that are set are always valid for the active menu element. By default, newly created child elements obtain properties values of their parent

Font properties can be changed by clicking *Font* button. The Standard Windows Font Selection Dialog appears. In this dialog, you can select font name and font size for the element, as well as the font style.

At any time can be displayed a preview for the menu, using *Preview* button in the toolbar or by choosing the option *Preview* from VMD-menu.

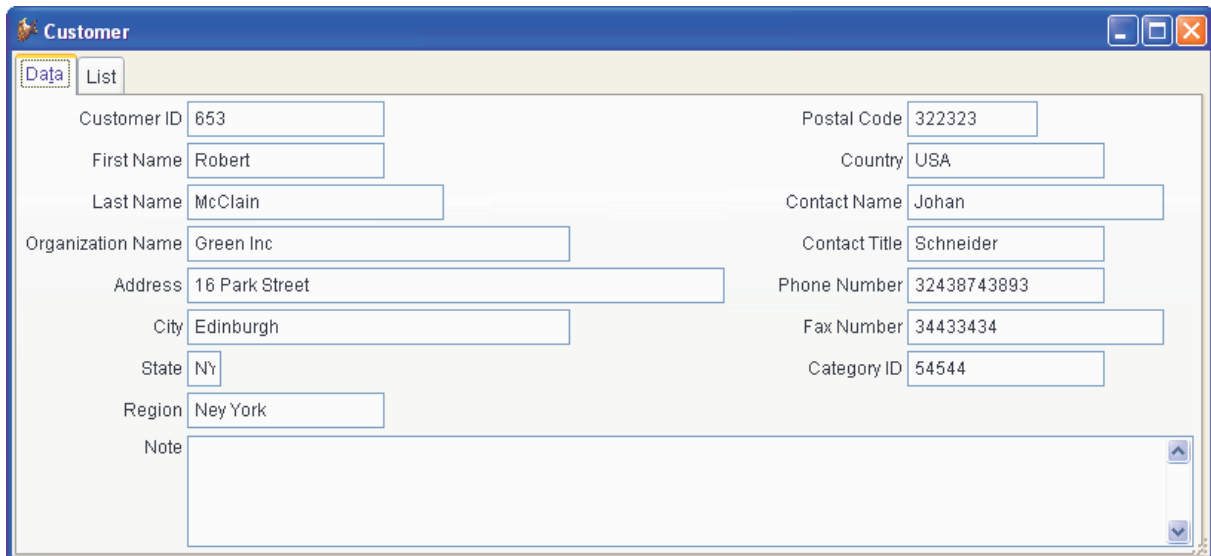
When saving menu, VMD creates automatically the needed include files for language-dependant menus. The subsequent steps after editing menus are not required.

## 10. Features of the created Form

The form that is created with the VFX – Form Builder has lots of useful features. The position of the form in on the workspace, the size of the form (using a resizer the size of the form can be changed by the user at run-time), the latest active page of the Pageframe as well as Grids, Sort order, Column width will be saved personally for every particular user. When an user closes a form and then opens it again, the form is displayed exactly same as when it was closed.

### 10.1. CDataFormPage Form User Interface

The default user interface for a standard data manipulation form is the following when it is not in edit or insert mode:



Customer ID	653	Postal Code	322323
First Name	Robert	Country	USA
Last Name	McClain	Contact Name	Johan
Organization Name	Green Inc	Contact Title	Schneider
Address	16 Park Street	Phone Number	32438743893
City	Edinburgh	Fax Number	34433434
State	NY	Category ID	54544
Region	Ney York		
Note			

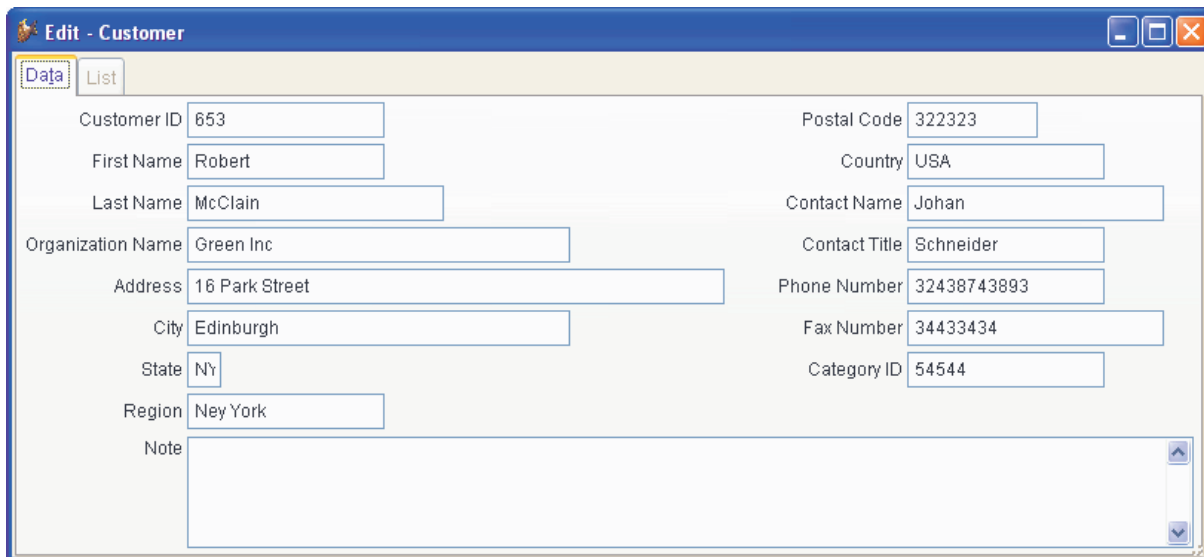
If you are in edit or insert mode, the caption of the form changes and the toolbar controls are automatically synchronized.

---

**NOTE:** For input of large data, you can directly call *Ctrl+N* while already in insert mode. This allows incomparable fast data input for multiple records. For the same handling optimization reason, the table navigation keys are available also while in edit or insert mode.

---

Depending on the setting of the application class property *nAutoEdit*, resp. the form property *lAutoEdit* the user can start typing and the form switches to the edit mode like shown here:

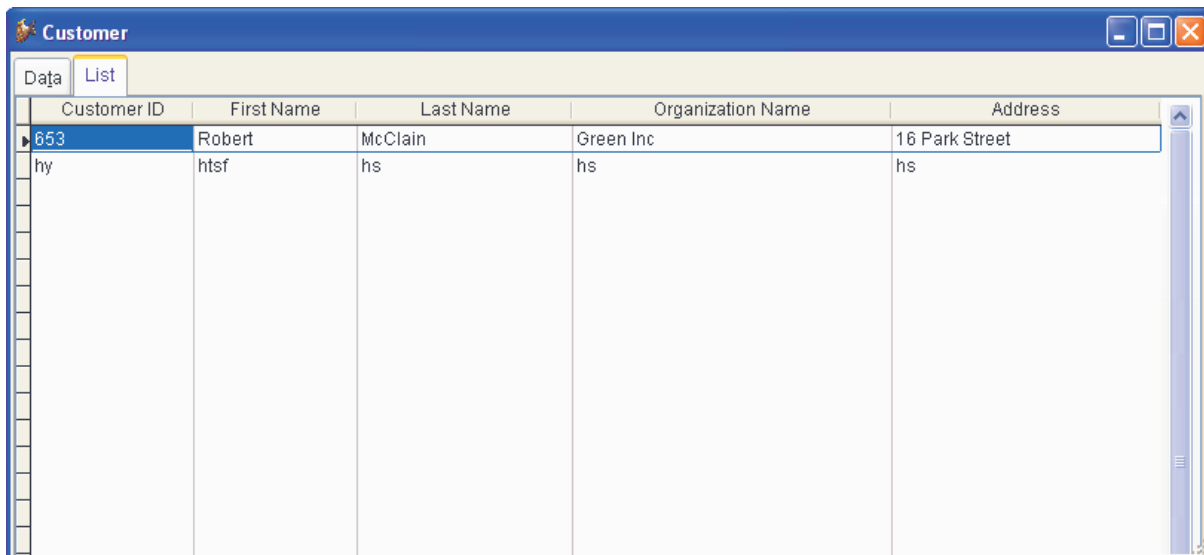


The buttons in the toolbar as well as the menu entries will be activated corresponding to the form's status.

### 10.2. The VFX Power Grid

In all columns in a grid an incremental searching is enabled by default. Also doubleclicking on any column header is a grid automatically sorts it by the correspondent column. If there is no existing index for the column, a temporary index will be created by VFX automatically.

When it is necessary to expand search for additional column, press *Ctrl* key and meanwhile click on other column header. The consecutive sorting order is displayed in the headers trough numbers in parentheses.



A double click on the column header sorts the column. The next double click toggles the sort order.

After clicking in a column can be started entering of a search string. The sort order will be set on this column and entered string will be searched incremental. The entered string will be displayed in the status bar.

Search : Martin

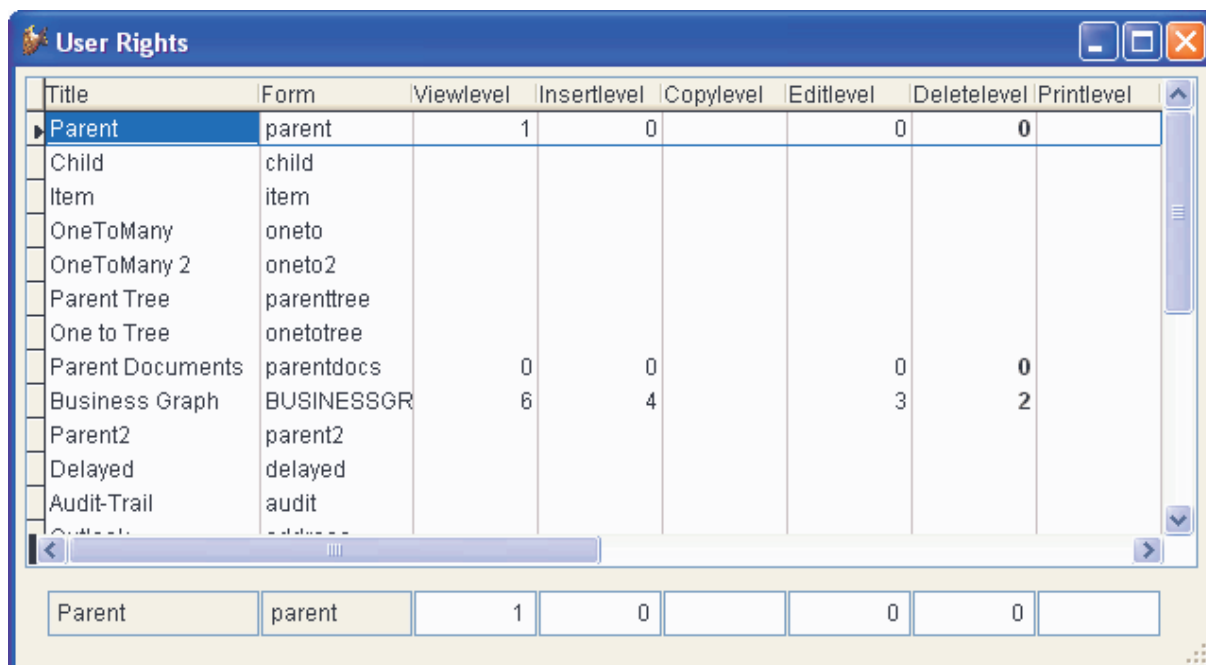
Using the VFX CGrid Builder, you can define, for which columns you want to have the Incremental Search capability enabled. This allows your user simply to type characters, numbers or even date while in any column and VFX automatically sorts data by this column and positions the record pointer on the first match. During the Incremental Search, the searched item is displayed in the status bar and corrections can easily be made using the Backspace key.

VFX shows the current sort order in the grid column header. The developer can choose among the following types of visualization:

- No visualization
- Underlining the caption
- Visualization using different colors
- Visualization through p and down arrow, similar to the Windows-Explorer

### 10.3. Form based on the class *cTableForm*

In the forms based on the class *cTableForm*, the search grid and other controls are placed next to each other or one under another in a container. A typical *cTableForm* based form is the Administration of the user rights.



## 10.4. Form based on the class *cOneToManyForm*

The screenshot shows a window titled "OneToMany" with a blue title bar. Inside, there is a "Page1" tab and a "List" button. The form contains several fields: "Parent ID" (value: 1), "Ins Date" (value: 10/15/199), "\*Description" (value: Parent 1 Or, maybe, there should be an), "Ins Usr" (value: ADd), "Date" (value: 06/11/2003), "Edt Date" (value: 12/06/200), a checked "Checked" checkbox, and "Edt Usr" (value: rosen). Below the form is a grid with columns: Child ID, Description, Value, Item ID, and Description. The grid contains five rows of data.

Child ID	Description	Value	Item ID	Description
54	Child 54		3	Item 3
56	Child 56		3	Item 3
66	Child 66	454	999	Item 999
67	Child 67	33	999	Item 999
78	Child 78	44	8	Item 8

The operations on the master table are identical to the standard data manipulation form. The toolbar and the Menu *Edit* apply to the master table.

The Child data rows are edited directly in the Grid. Only if you are in Edit- or Insert mode of the master table, you can write into the child grid, insert new child records or delete the currently selected child record. All operations on the child records are handled using the optimistic table buffering. If you select undo changes, the changes made to all child records of the current master record are also reverted. If you select save changes, all changes applied to the master record and to all child records of the current master will be saved.

A click on the empty area of the child grid adds a new child record.

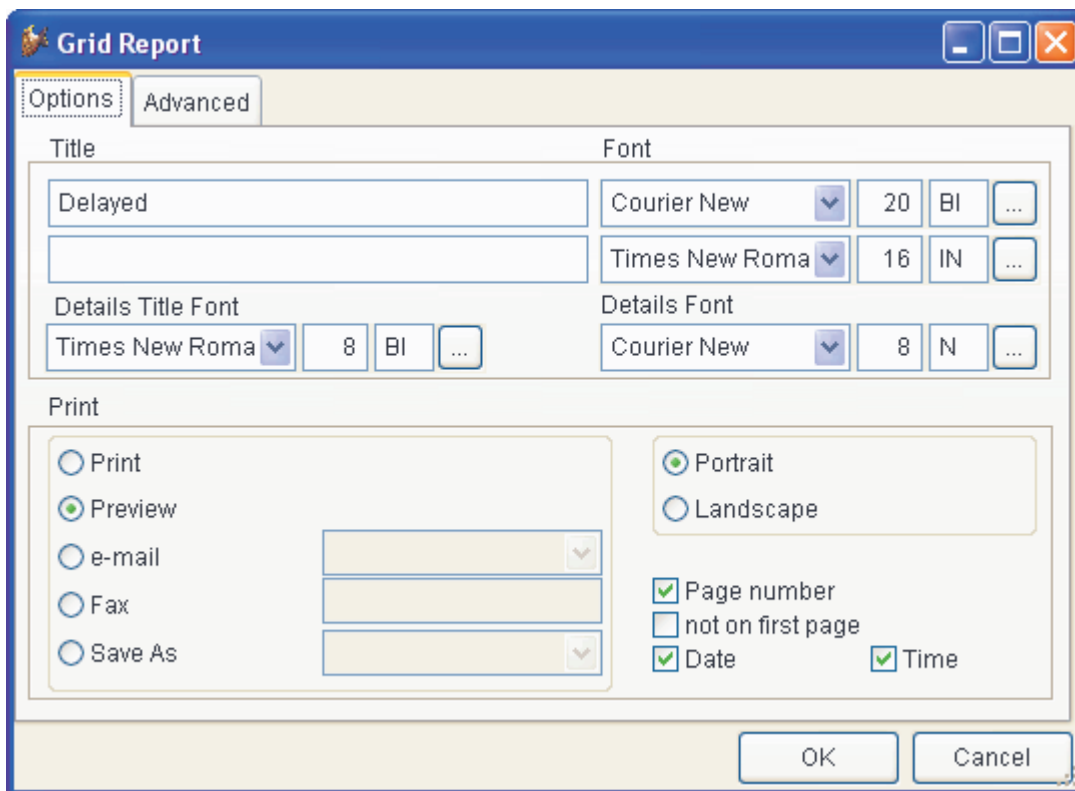
When the child data is based on a view or on a CursorAdapter, incremental search can be performed for the child data.

One of the most interesting features of VFX is the special Picklist control, which you can easily add to any of your Child Grids using the VFX - CPickTextBoxt Builder. The Picklists can be accessed in edit or insert mode.

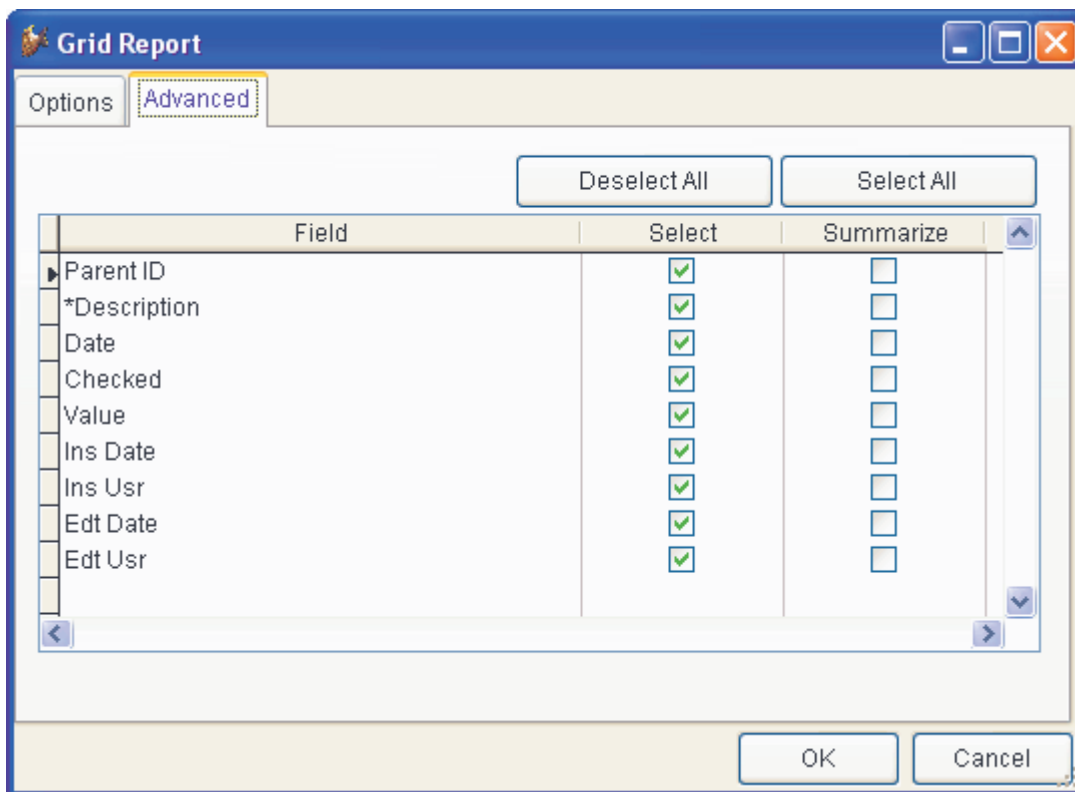
By a double-click in the *CpickTextBox* or by pressing the function key *F9* the Picklist will be shown.

## 10.5. Printing

By default in all forms can be printed a list without need to have report prepared for this purpose. At run-time VFX application creates a temporary report, based on the appearance on the search page.

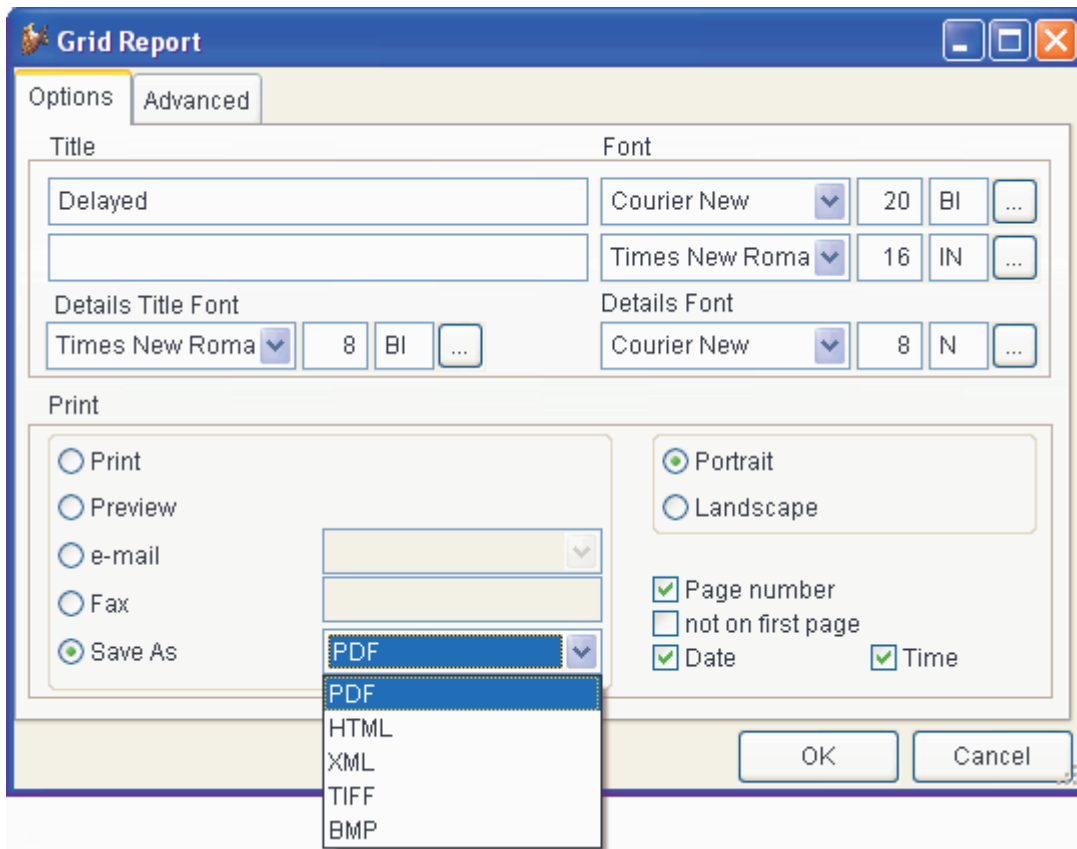


Before printing the user can eliminate undesired columns from the columns list. The width of the columns corresponds to the width of the columns in the Grid.



VFX 9.5 supports several possibilities to save the VFP 9.0 report output in files. Generated reports can now be saved in several new formats. The supported formats are PDF, HTML, XML, TIFF and BMP. These file types can also be used to create an attachment and e-mail the report.

In Grid report dialog, when *E-mail* or *Save As* option is chosen, the type of generated file is selected from a Combobox among all available export formats.



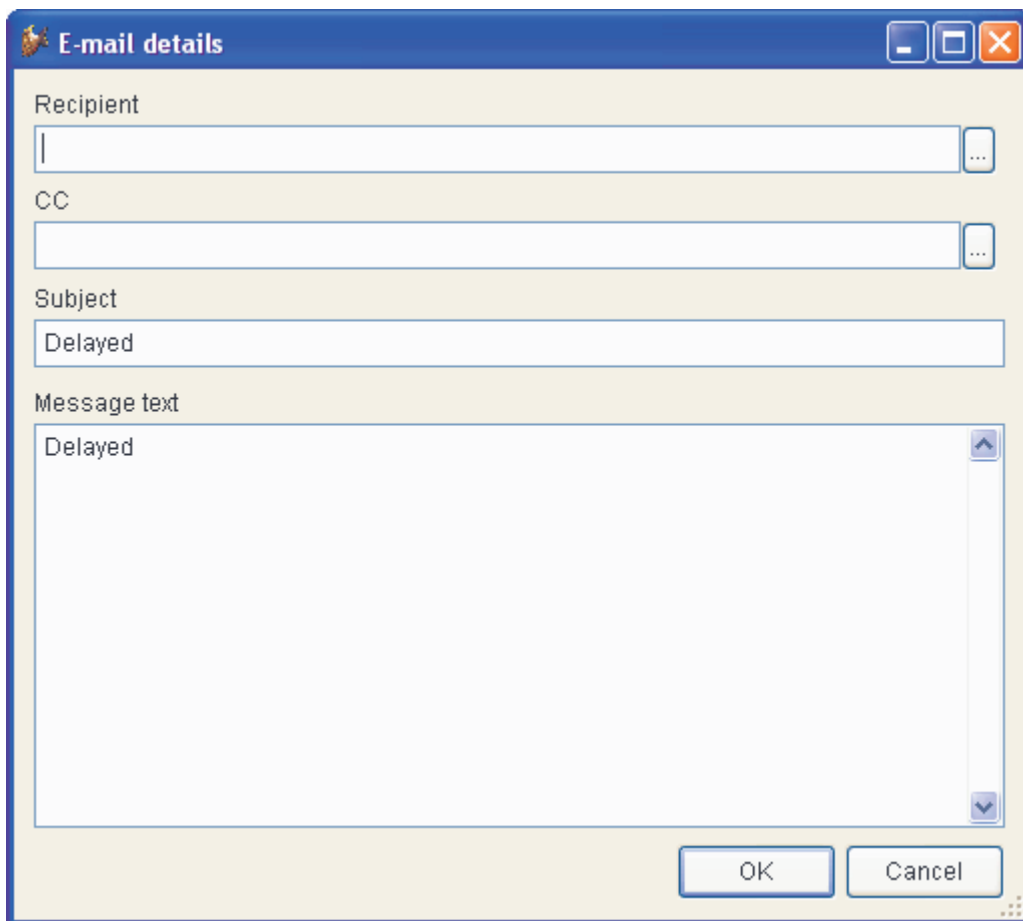
If the desired format is TIFF or BMP, a group of subsequently numbered files is generated. For every page in the report is created a separate file. File names are generated using the file name chosen by the user and a numeric value representing page number.

## 10.6. Sending E-Mail

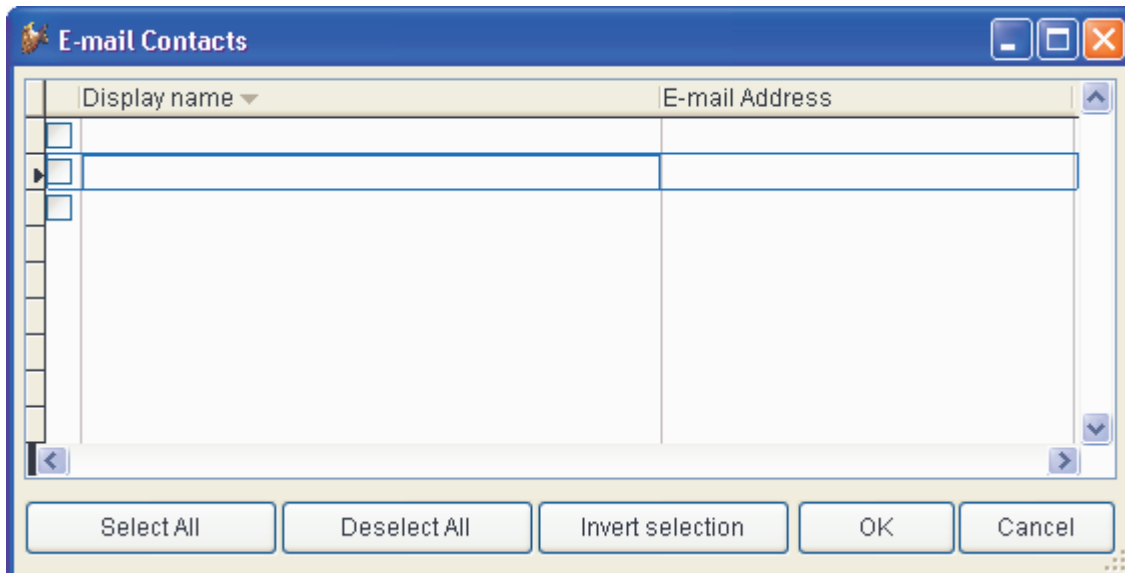
All file formats that can be exported as a report output can also be sent as an e-mail attachment.

In *E-Mail details* dialog can be entered recipients; CC recipients, BCC recipients list and a text for e-mail body. The visibility of BCC Textbox is controlled by *goProgram.UseBCCRecipients*. When this property is set to *.T.*, it is possible to enter BCC recipients list





For every recipients list user can invoke the *E-mail contacts* dialog to select a list of e-mail addresses saved in the Outlook Address book. The dialog is called from the ellipsis button next to every recipients list field.



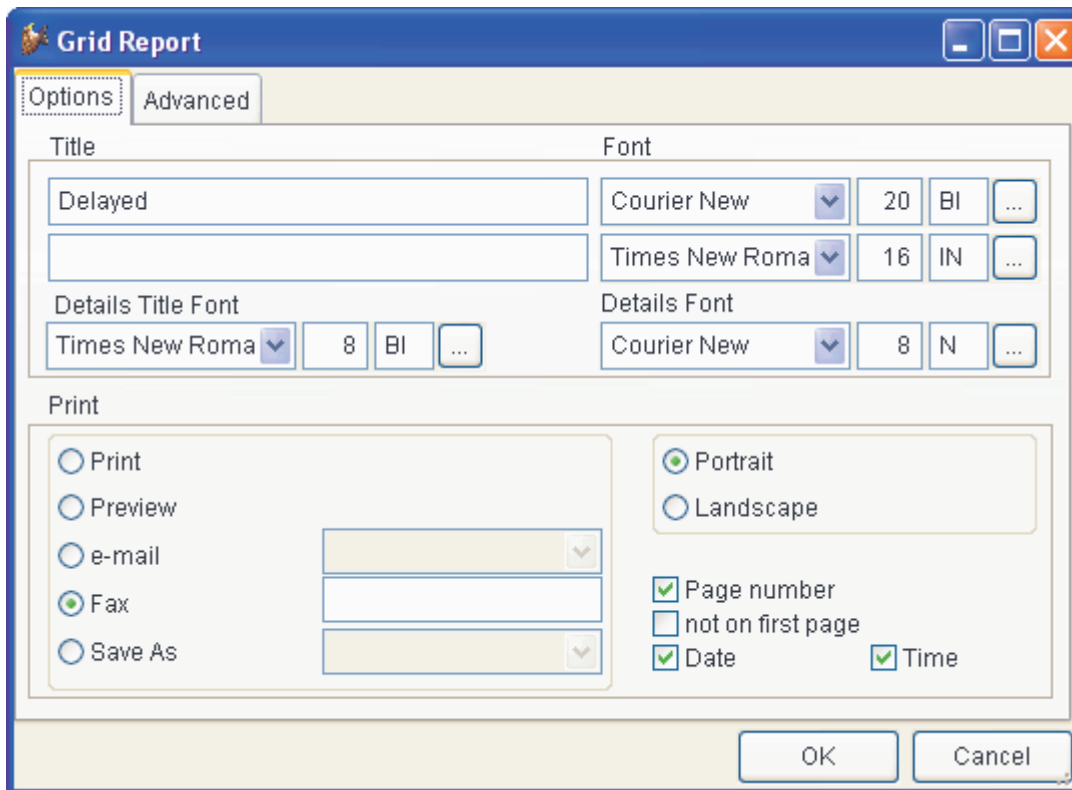
After clicking OK, all addresses, selected in *E-mail contacts* form are passed back into correspondent recipients list.

### 10.7. Sending Fax

Another feature for report output is the ability to send generated report as a fax. When the user chooses Fax option for report output the fax number should be entered in the textbox.

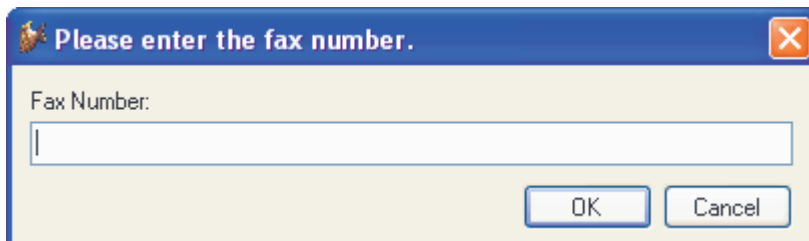
The report is sent to a fax printer and the specified number is automatically used for that, without need of any additional actions by the end-user.

VFX 9.5 uses the fax application FRITZ!fax by AVM and Winfax by Symantec. VFX 9.5 recognizes when one of these fax applications is installed. When a fax application is found, it is used for report output to the corresponding fax-printer driver.



The fax number is passed to the fax application directly by the VFX application. The end user will not meet the fax application dialogs.

When a particular form uses an individual report file, the end-user can enter the fax number in an additional dialog:



## 10.8. Filtering

The visible data scope in a form can be narrowed by setting a filter expression. For this purpose VFX provides a ready-to-use dialog. You can include as many fields as you need, linked with „and“ or „or“ logical clause.

It is possible to combine many different search criteria. The search criteria will be saved per user and form. And are available to be used on the following application executions.

In Search dialog the user can enter only valid values. For every chosen data type, in the Operator column, are available only operators which can be applied to this data type. It is possible to enter values from same data type.

Field	Operator	Value	A/a
Parent ID	More than	65763.00	
Ins Date	More than	01/11/2003	

In the column Value are placed many controls. The property *CurrentControl* of this column is changed, depending of the data type of the field, chosen in the column *Field*.

When a fields of *Character* data type is chosen, in the column *Value* is shown a textbox control, where can be entered any characters. Additionally is available an operator *Contains*. In this case the filter expression is created with \$ operator. The rightmost column in the grid is used in this case to specify if string comparison will be case sensitive.

When a *Numeric* fields is selected in column *Field*, in the column *Value* is shown a textbox that will allow the user to enter digits only.

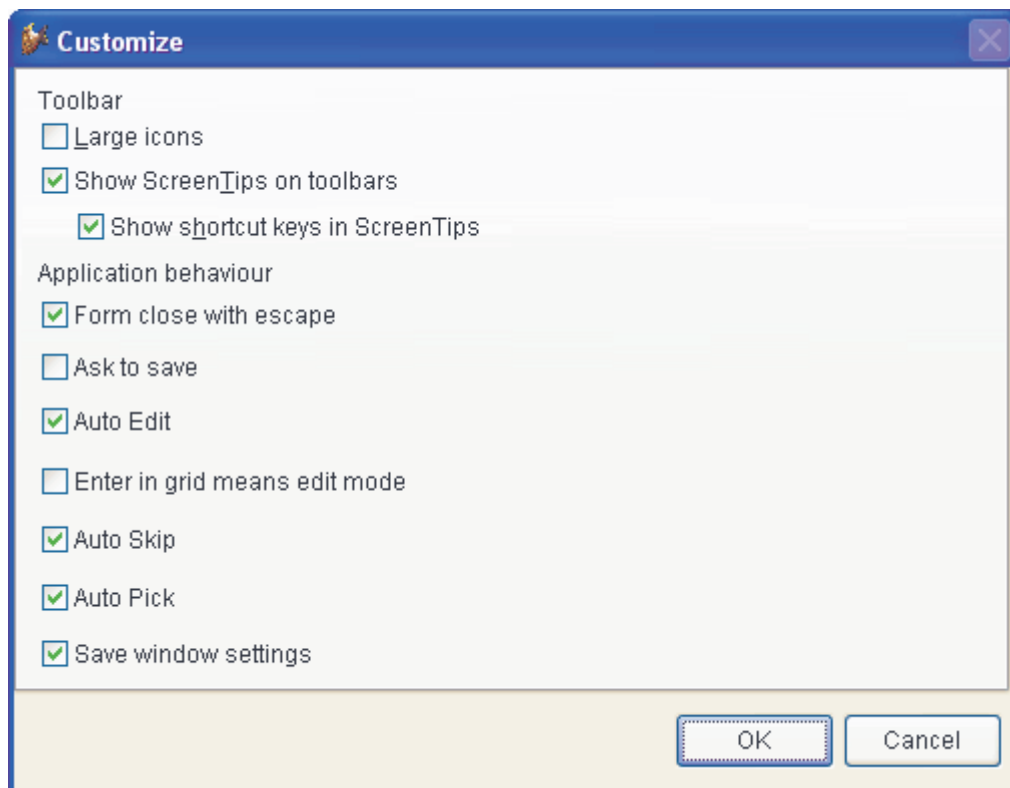
When a field of *Date* or *Datetime* data type is selected, the *Inputmask* of the column *Value* will be set accordingly.

When a field of *Logical* data type is chosen, in the Combobox in the column *Value* can be chosen between True and False. In this case use does not enter a value manually.

In such case it is not possible for end-user to enter non-valid value.

## 10.9. Layout

The layout of the application created with VFX 9.5 is improved with new XP style icons. New icons are developed for toolbar and menu entries.



Users are able to change their personalized layout setting under the menu option *Tools, Customize*. It is possible to switch between large or small toolbar icon size. They can set whether tool tip texts for toolbar buttons will be displayed. When Show screen tips option is marked, additionally can be specified if hot keys will be displayed in tool tip texts. For example the hotkey combination for the button *New* is *CTRL+N*.

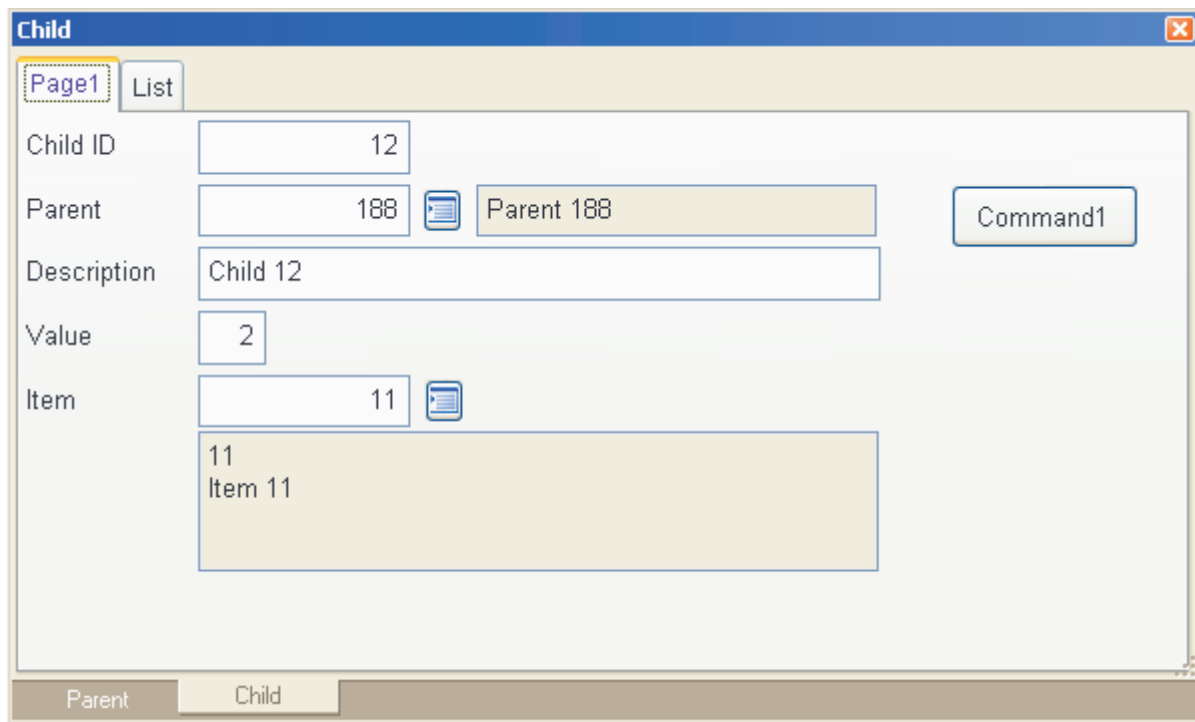


New for layout of VFX forms is the ability to select a bitmap as background for a page in the pageframe in all data forms. The picture can be set in VFX – Form Builders.

In VFX – Form Builders, it is also possible instead of setting background picture for a page in the pageframe, to define background color.

## 10.10. Dockable Forms

VFX 9.5 supports dockable forms.



The docking behavior of forms is controlled by *goProgram.nDockable* property of the application object. If this property is set to -1, for every particular form are used its own settings. When *goProgram.nDockable* contains a value greater than -1, this value is assigned to Dockable property of all forms.

---

**NOTE:** When the *WindowType* of the form is set to *Modal*, the value of the property *goProgram.nDockable* is not applied to the form. Generally, modal forms cannot be docked.

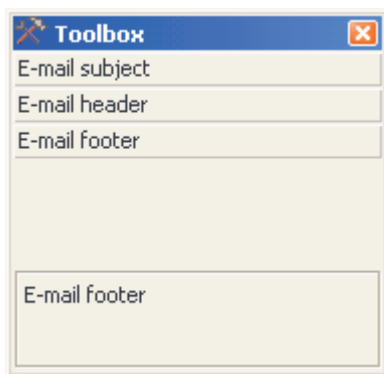
---

The dock-status and the dock-position of a particular form is saved for every user in the resource table *Vfxres.dbf*.

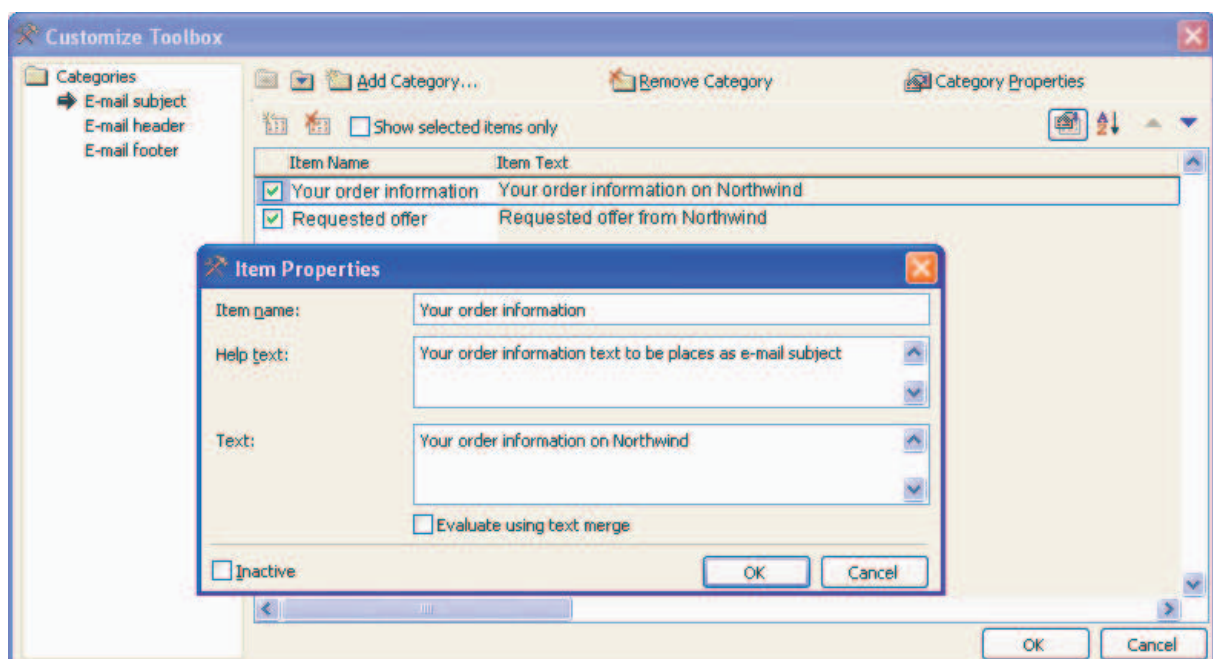
## 10.11. VFP Toolbox for end-users

The useful toolbox functionality of VFP 9 is extended to end-users, too. The toolbox is fully integrated into VFX 9.5 and adjusted to VFX specifics. Users can define specific text that can be easily dropped-down into text boxes and edit boxes in forms. Dragging an item from the toolbox dropping it into a Textbox or Editbox controls in a form will place the item text at the cursor position.

Similar to VFP toolbox, text items are grouped into categories.







On right mouse click on Toolbox windows and choosing Customize Toolbox, categories and items can be added, edited and deleted.



For every category can be specified Category name and Help text. For Items can be entered Item name, Help text and Item text.

Category names and Item names are shown. The correspondent Help text is displayed in an Editbox in the lower part of the Toolbox window as description for the active item. The Item text is placed in the destination control at the cursor position.

Using the buttons  and  the user can change the display sequence of the categories in the Toolbox window. Items can be move within category using  and  button.

## 10.12. Treeview

The class *cTreeView* class is now enhanced for better performance while loading. For every user, when the form is closed, in the resource table *Vfxres.dbf* is saved current state of all TreeView nodes: expanded or collapsed. On next opening of the form all nodes of the TreeView control are shown in the state in which they were when the form was closed.

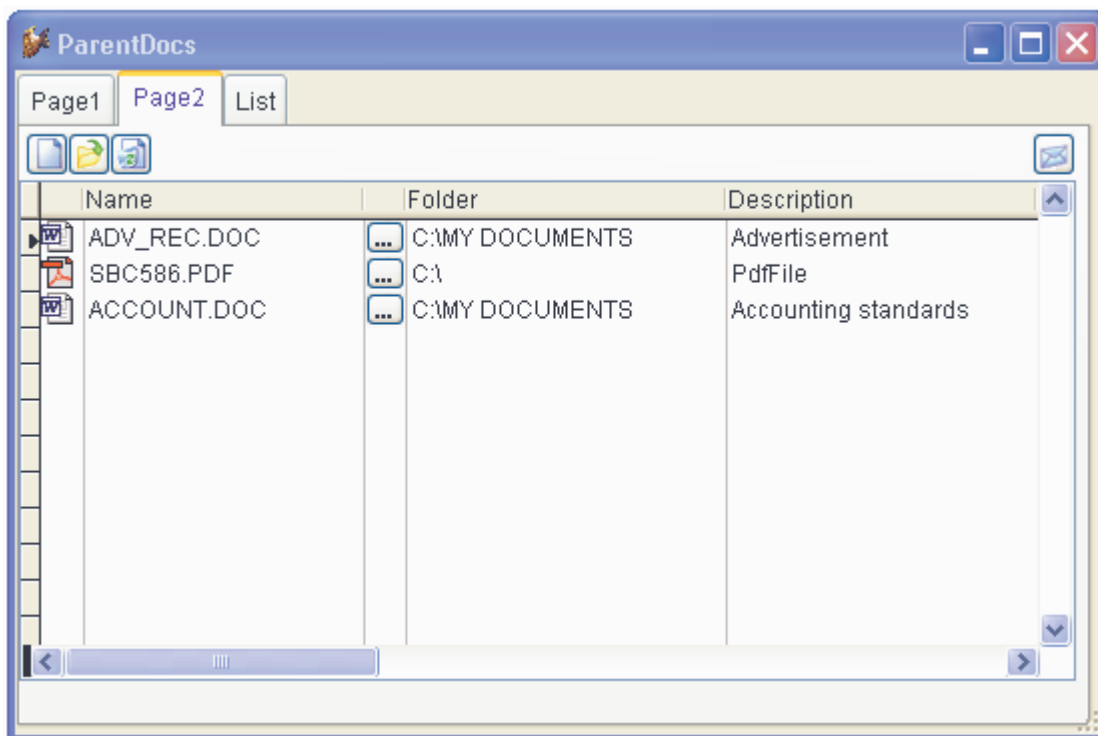
It is possible now, the grid report in forms based on *cTreeviewForm* and *cTreeviewOneToMany* form classes to be printed as a treeview structure.

A context menu with New, Rename and Delete options is added to the Treeview control

### 10.13. Document management with the class *CDocumentManagement*

The new class *CDocumentManagement* is designated to maintain all kind of documents (i.e. Word, Excel, PowerPoint) related to application data. The *CDocumentManagement* class is a container that manages child records related to current record of main table. It allows the end-user to open related documents and also to send them as attachment in an e-mail.

The class can easy be added to the existing forms.



### 10.14. About Dialog

A link label is placed in the About dialog, for displaying the End-user license agreement (EULA). This link invokes a dialog form, where the EULA can be read and printed from.

The End-user license agreement text is kept in the table in a data row with *Type = "EULA"* and *Filename = "EULA\_" + lcLangID* (where *lcLangID* is the abbreviation string for the particular language). In this way it is possible to provide an End-user license agreement text for every language.

### **10.15. Further end-user features in VFX 9.5**

- Support of the incremental search even when the field contains .NULL.
- Localized Hotkeys for the class cpickdate and a multiline tool tip text
- New classes: e-mail with e-mail client call, hyperlink with Internet Explorer call, numerical text box with calculator call, TAPI, file selector with open file dialog
- Conform to visible =.F. for Grid-Columns for the search dialog and the print dialog
- Progress bar by the actualization of the customer database
- Script for download and install Adobe Acrobat reader (for PDF documents)
- Keyboard support in the XP opening dialog
- Drag and drop support in Mover dialog
- Position on last viewed record when form is reopened
- Support of the *HighLightStyle* property in Grids
- Improved memo fields visualization in Grids
- When all entries in favorites menu are deleted, the corresponding menu bar is also removed



## 11. Data Handling

### 11.1. Data Handling conception

One of the greatest innovations in VFX 9.5 is the completely new conception of the data access. Don't worry, existing applications are fully compatible with the new data access conception. As till now, it is possible to work further directly with tables or views on local or Remote-data sources.

The new data access is rather an additional possibility to access various data sources. VFX 9.5 supports the VFP class CursorAdapter for data access. The VFP class CursorAdapter can be considered as a small revolution in the data access from VFP applications.

So far, the data access in VFP and VFX always worked with the help of a DBC. Experienced programmers could also access data by SQL Pass-Through, but here we will not discuss this detailed. The data access through a DBC is well known for us and it is trustworthy and reliable. However, the data access through a DBC has also a few disadvantages. A DBC is nothing else than a table. The file extension is changed by DBF in DBC because it concerns a special table. In the DBC is kept information about the database structure and the integrity, as well as information about connections if working with Remote datasource.

Users could manipulate the DBC data. Connection information about the Remote data source, including user name and password, is kept as a clear text and is fully readable if the DBC is opened, for example, with Excel. The idea to work without DBC has two general reasons. The connection information must be protected against unauthorized access and modification. The application can very easy be ported from a DBC data source to a remote data source.

Exactly these goals can be achieved by using the CursorAdapter class. CursorAdapter objects can be added to the Dataenvironment in same way as tables or views. CursorAdapter class can be inherited and VFX provides the class *cBaseDataAccess* from the class library *Vfxctrl.vcx*. This class is a underlying class for all created CursorAdapter objects used in a VFX application.

All handy features of VFX forms, like incremental search in grids, filter and print functionality are also available for forms that use CursorAdapter objects as datasource. This functionality is implemented in same way as for forms using tables or views.

CursorAdapter object, based on the class *cBaseDataAccess*, uses a connection manager class, which developers know from former VFX versions. This ensures that all CursorAdapter objects will use one common data handler. This approach not only optimizes usage of data access resources, but also considers connections license policy, so that no redundant connections to the datasource are created.

The connection specific information that will be used by the Connection Manager object is read from the file *Config.vfx*. Similar to the DBC, where can be stored connection information for more than one connection, in the file *Config.vfx* can be stored information about several connections. The connection can be created either to a DBC or to any remote data source using either a DSN entry or a connection string. In order to protect the file *Config.vfx* against unauthorized access, it is password encrypted. The password that is used for encryption is keyed up in the property *goProgram.configpassword* and in this way is included into the compiled EXE file.

Through additional entries in *Config.vfx* file the application can connect to different data sources by switching between defined data sources. The file *Config.vfx* can contain multiple connections definitions. When multiple connection definitions are saved, on application start the end-user sees a selection dialog. This feature is equal with the ability to specify many databases in the system table *Vfxpath.dbf* well know from the former VFX versions.

### 11.2. Concept of new applications

When with VFX 9.5 is started a new application development, the new data access concept should be seriously taken into consideration. If the data access of the newly developed VFX 9.5 application is exclusively based on

the class *cBaseDataAccess*, thanks to its portability it is possible later to switch between different data sources without any problems.

In this way the application development can start with using a DBC as datasource. With VFX - CursorAdapter Wizard, are be created CursorAdapter classes, correspondent to every table in the database. These CursorAdapter classes will be used then as a datasources in all forms.

### 11.3. VFX - CursorAdapter Wizard

The VFX - CursorAdapter Wizard creates one CursorAdapter class for every table that is included in the database. Using these generated CursorAdapter objects can be accessed data from a form for example. The CursorAdapter Wizard can access any type of data source, used in VFX and use it as a foundation for generation of CursorAdapter classes

Later, the generated CursorAdapter classes, can be edited with VFP CursorAdapter Builder. In particular, it is necessary to take into consideration which parameters is it meaningful to be used for the CursorAdapter classes.

By default these CursorAdapter classes are based on *cAppDataAccess* class and are saved into *Appl.vcx* class library. However, in the Wizard it is possible to be changed underlying class and target class library.

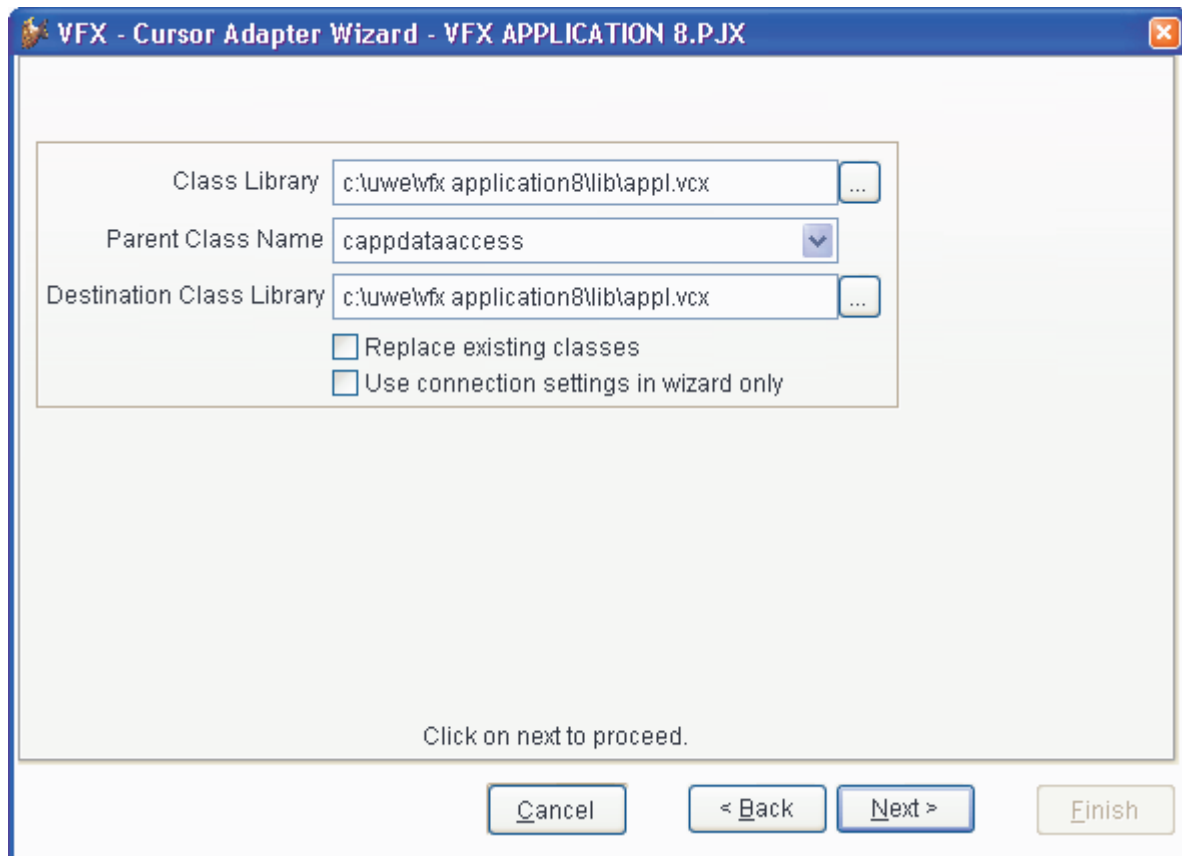
Wizard leads the developer through three steps.

#### 11.3.1. Choosing the Datasource

The screenshot shows the 'VFX - Cursor Adapter Wizard' dialog box. The 'ODBC' radio button is selected. Under the 'ODBC' section, the 'Use DSN' radio button is selected. The 'DSN' dropdown menu is set to 'Northwind'. The 'User Name:' and 'Password:' text boxes are empty. The 'Generate SQL Connection String' section is unselected. The 'Use connection string' radio button is also unselected. The 'Server Name:', 'User Name:', and 'Password:' text boxes are empty. The 'Use Trusted Connection' checkbox is unchecked. At the bottom, there is a message 'Click on next to proceed.' and four buttons: 'Cancel', '< Back', 'Next >', and 'Finish'.

This data source will not become data source for the compiled application. It is used for building CursorAdapter classes only. When application runs, the actual DataSource will be read from *Config.vfx* file. In this way you can set different data source for the particular end-users.

#### 11.3.2. Choosing the underlying class and class library



If the option *Generate SQL Connection String* is selected, on second step it is also required to choose database from the list of databases which reside on the selected SQL Server.

On this step are to be chosen underlying CursorAdapter class and destination library.

The default values are:

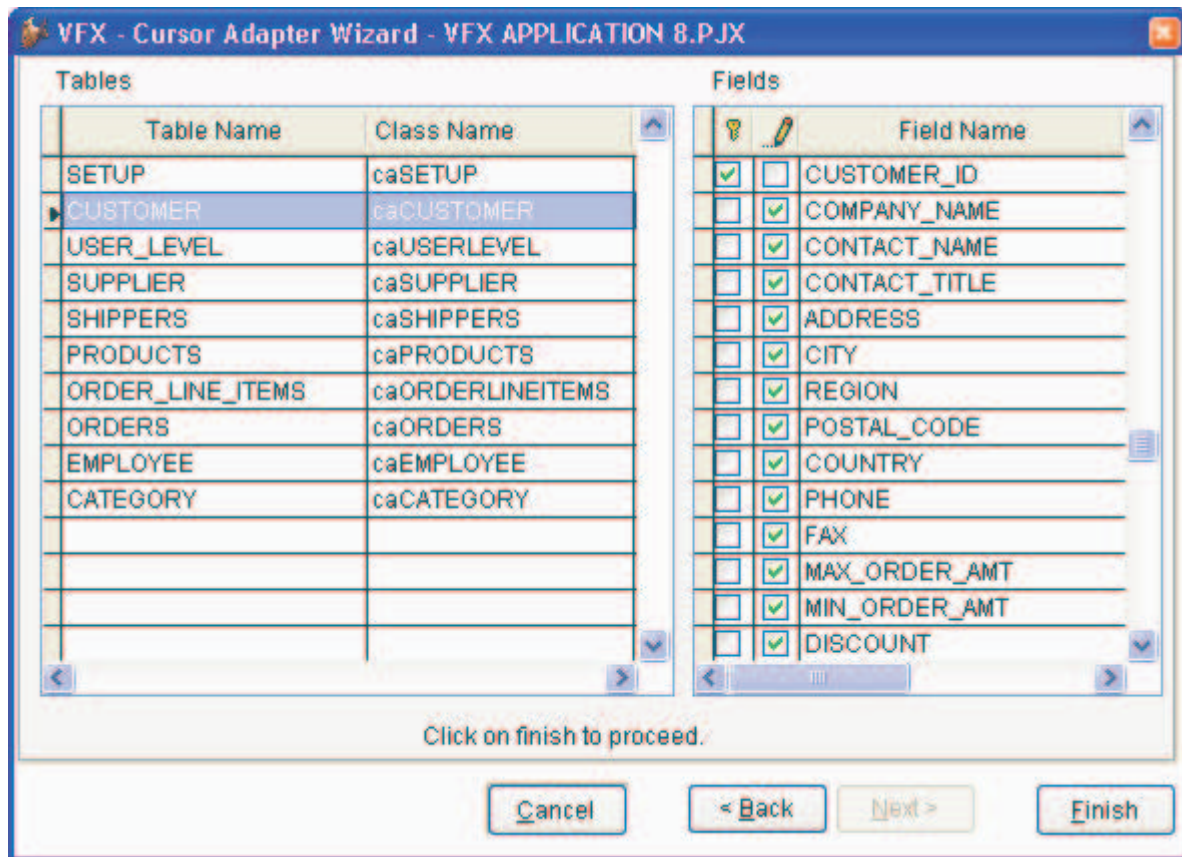
**Class Library:** *Appl.vcx*

**Parent Class Name:** *CAppDataAccess*

**Destination Class Library:** *Appl.vcx*

It is also possible to set whether classes that already exist in destination class library should be overwritten by checking *Replace existing classes* option.

### 11.3.3. Choosing Tables



Last step for building CursorAdapter classes shows a list of all tables and fields for every table. When you navigate through the table list in the left grid in the dialog, in the right grid is displayed the list of fields for correspondent table.

By default fields which are primary key for the table are marked as key fields for the created CursorAdapter class. All other fields by default are marked as updateable fields. You can change these preset values accordingly.

As a result the Wizard creates one CursorAdapter class, corresponding to every table in the chosen database. For generated classes are filled following properties: *CursorSchema*, *Tables*, *SelectCmd*, *KeyFieldList*, *UpdateableFieldList*, and *UpdateNameList*.

## 11.4. Data access with CursorAdapter

Builders of VFX 9.5 use CursorAdapter objects in the Dataenvironment. CursorAdapter objects can be used in Dataenvironment in same way as local or remote views are used.

In all VFX Builders and Wizards, CursorAdapter objects can be used as datasource. CursorAdapter objects can be used as datasource also in Picklists.

VFX 9.5 includes a CursorAdapter class, which ensures the base functionality, needed to access application's data. This is the class *cBaseDataAccess*, placed in the class library *Vfxctrl.vcx*. This class should be used as an underlying class for all CursorAdapter objects, used in the application. It ensures that the whole application uses one common data connection and does not create unnecessary connections.

### 11.4.1. The class CBaseDataAccess

The new *cBaseDataAccess* class is based on VFP CursorAdapter class, providing VFX developers with extended data access functionality. The class *cBaseDataAccess* uses common application settings and allows you to develop applications which can be easily ported to different data sources. The data access settings for the class *cBaseDataAccess* are saved in *Config.vfx* file

When an instance of the class *cBaseDataAccess* is loaded its properties are set accordingly the correspondent *goProgram* properties. The property *goProgram.cDataSourceType* is used as *DataSourceType* for the newly created object. If *NATIVE DataSourceType* is used, the class uses the database path and database name, referred by *goProgram.cDataDir* and *goProgram.cMainDataBase* properties. For all other values of *DataSourceType*, to obtain *DataSource* value is called *GetConnection* method of the connection manager object. The object is specified in *cConnMgrName* property.

In the class library *Appl.vcx* is placed the class *cAppDataAccess*, which is an instance of the class *cBaseDataAccess*. The developer should place and additional specific settings for application data access in this class *cAppDataAccess*.

#### Properties

*cConnMgrName* – The name of the object which should be used as a connection manager object. This connection manager object is used as a *DataSource* for the class *cBaseDataAccess*. By default this is *goProgram.oConnMgr*.

*cExecuteAfterCursorFill* – This property is aimed to contain code which needs to be executed after *CursorFill* method is executed. Here can be placed code for any necessary data processing as well as code for creating indexes. Using a property it is possible specify code to be executed when the object is created at run time.

*Filter* – Filter expression to be applied on the retrieved data for the created cursor

*Order* – The index tag specified here will be used for the created cursor. Index tag can be defined in the VFX – Data Environment Builder.

#### Methods

*CreateIndexes* – The code in this method is generated by the VFX - Data Environment Builder. Here are placed command to build needed indexes for the created cursor. This method is called immediately after *CursorFill* method

### 11.5. Managing data access using the file *Config.vfx*

When developing the application, the choose *DataSource* used in the CursorAdapter Builder refers resources, local for the developer's computer.

It is not always possible the users' computers to use resources named in same way. For instance, the name of SQL Server instance on customer's network is always different than name of your local SQL Server running on your computer and used in development.

Even if on the development computer and on the customer computer a SQL server database should be used, the name of the SQL server on both computers will be different. Hence, as a rule, on the development computer and on the customer computer is necessary to use different connectivity data.

This is the reason at runtime not to use *DataSource* which were set at design time. At run time *cBaseDataAccess* class uses the object *goProgram.oConnMgr* – an instance of class *cConnectionMgr*. Settings for *goProgram.oConnMgr* object in turn, are read at run time from the file *Config.vfx*

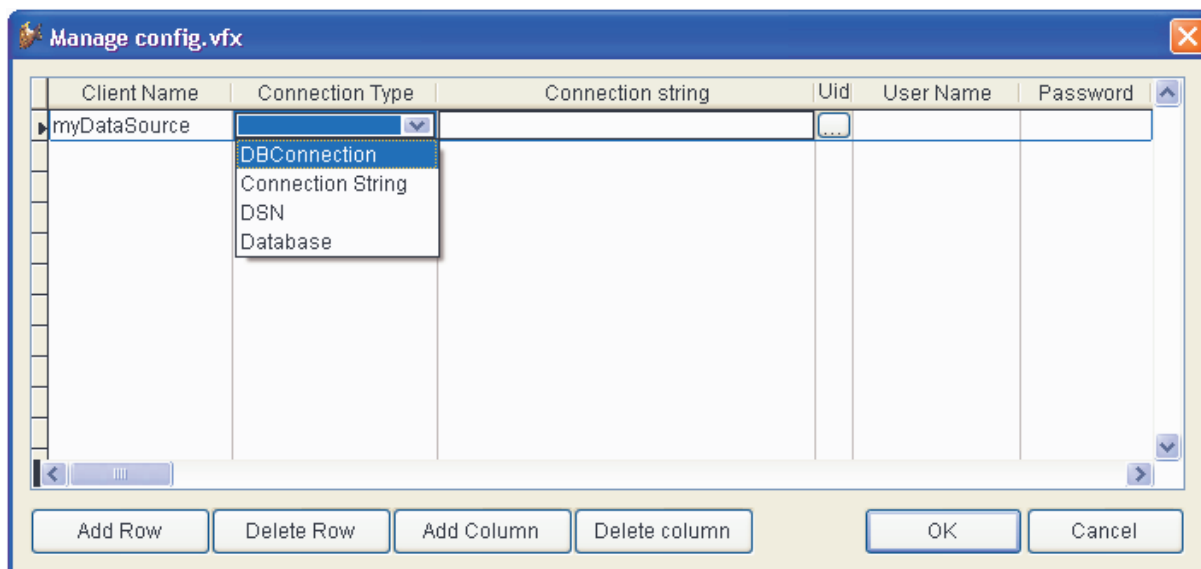
VFX provides an own connection-manager class to establish a connection to the datasource. This connection-manager is instantiated as a Child object of the application object and is available through the reference *goProgram.oConnMgr* at run-time.

CursorAdapter objects based on the class *cBaseDataAccess* use the object *goProgram.oConnMgr*, an instance of the class *cConnectionMgr*, to establish a connection to the datasource. The settings for the object *goProgram.oConnMgr* are read from the file *Config.vfx*. In this file is kept the information about the datasource used by the application.

The file *Config.vfx* contains data, encrypted for safety reasons, that is used to connect to the customer database, for example, type of the datasource, connection string and other. The password for encoding the file is stored in the property *goProgram.cConfigPassword*. VFX developers should define this password themselves.

The file *Config.vfx* can be created by the developer and deployed along with the application. If by the start of the application the file *Config.vfx* does not contain any entries (or is corrupted), the dialog *Manage Config.vfx* is invoked and the user should define the datasource entries.

Later, users with administrator's rights can edit the file *Config.vfx* using the menu *Tools/Manage Config.vfx*



For every particular client you can define different DataSource type and specify corresponding VFP native database or ODBC connection to the database. The file *Config.vfx* can contain multiple records. When there are more than one record in it, a *Client selection* dialog is invoked.

It is possible to use a connection, defined in a VFP database. For this, choose *DBConnection* as connection type. At run time the connection name is assigned to *cDBCConn* property of *goProgram* object. In the file *Config.vfx* is defined the name of database to be used for different clients. At run-time the connection information is retrieved from the database, specified for the currently selected client and this connection is further used to connect to remote data.

To use ODBC connection, can be used a connection string or an existing DSN. If a *Connection String* option is chosen for DataSource type the ellipsis button can be used to invoke a dialog that helps the user to build a valid ODBC connection string.

When *DSN* will be used, additionally can also be specified username and password to be used while establishing connection. If you do not provide username and password for the DSN and the connection requires authentication, a login screen will pop up asking user to enter username and password.

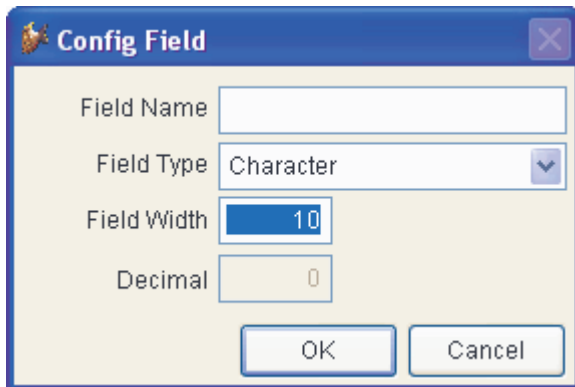
The file *Config.vfx* corresponds to the table *Vfxpath.dbf*, known from previously VFX versions. All fields, included in *Vfxpath.dbf* table are also included in *Config.vfx* file



In the file can be defined several entries, which access different types of datasource. Thus a user of the application can decide by the application start whether he wants to work with a VFP database or with different server databases.

Through the encryption of *Config.vfx* file, for the VFX application is attained a security that was not achieved before.

By analogy with *Vfxpath.dbf*, developer can add own defined fields to *Config.vfx* file and use these fields later at run time. The button *Add Column* invokes a dialog window, where you can specify name and type of the field to be added



### 11.6. Switching between DBC and SQL Server

When a VFX 9.5 application is so constructed that the data access is accomplished exclusively through CursorAdapter classes, it is possible afterwards to switch between a DBC and a SQL Server database very easily.

Let's assume, we have developed an application that uses a DBC as a datasource. By the development we have ensured data access everywhere in the application occurs only through CursorAdapter classes. Now a customer would like to run this application with a SQL Server database.

For this purpose, first it is needed to port the VFP database onto SQL Server. We can do this using the VFP Upsizing wizard, but also other tools, such as xCase for example, are suitable for this task.

For the access to the SQL server database can be provided a DSN. However, this also brings a security risk because the DSN can be manipulated. It is more secure in the file *Config.vfx* to be chosen a connecting string for the data access. Thereby connection information is independent of other settings at operating system level and keeps all information about the data access saved within the application.

The SQL Server database is installed on the server of the customers. The finished application is delivered with an empty *Config.vfx* file. In this way, the *Manage Config.vfx* dialog for keying up the datasources is automatically invoked by the application start at the customer's side. The connection to the SQL Server installed by the customer can be defined with user's name and password and it can be edited within the application.

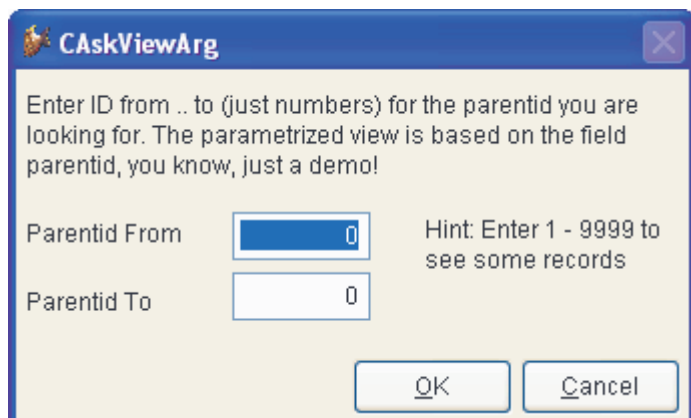
### 11.7. Forms based on views

While the development of VFX great attention was paid to the fact that it must work both directly with VFP tables, and also with local views and with remote views. In particular, Views cannot have any index keys. VFX must also provide a temporary index file in each case, in which an ordering is needed.

Views can be used as data source for every VFX Form type. It also possible to base OneToMany forms or Parent/Child constructions on views. The usage of views is possible with Picklists, too. Thus, an VFX application can be used as front-end for example for a SQL server or other remote data sources.

In most cases Views are parameterized. The parameters must be defined before querying the data of the view. For entering the view parameter, VFX provides the form class *CAskViewArg*. The Data Manipulation Form is created, as usual, using the VFX – Form Builder. The property *IWorkOnView* will be set to *.T*. For the view in the data environment the property *noDataOnLoad* will be set to *.T*. This means, that by the loading of the form the view will be opened without querying data.

Now a form, based on the class *CAskViewArg* will be created. The controls, which contain fields as *ControlSource* that will be used also as view parameters, can be copied through the clipboard from the Data Manipulation Form, on the form based on the class *CAskViewArg*. In the property *cviewparameter* is specified the name of the View parameter. For the controls could be added appropriate labels. With this the form is ready and can be saved.



Now the form based on the class *CAskViewArg* must be called from the Data Manipulation Form. This happens on the end of the *Init()*-Events:

```
do form <Form for input of the View parameter> with this
```

It is also possible the form for input of the view parameters to be called again at run-time. If the call is made from a control, for example from the *Click()* Event of a button, the call must look in this way:

```
do form < Form for input of the View parameter> with thisform
```

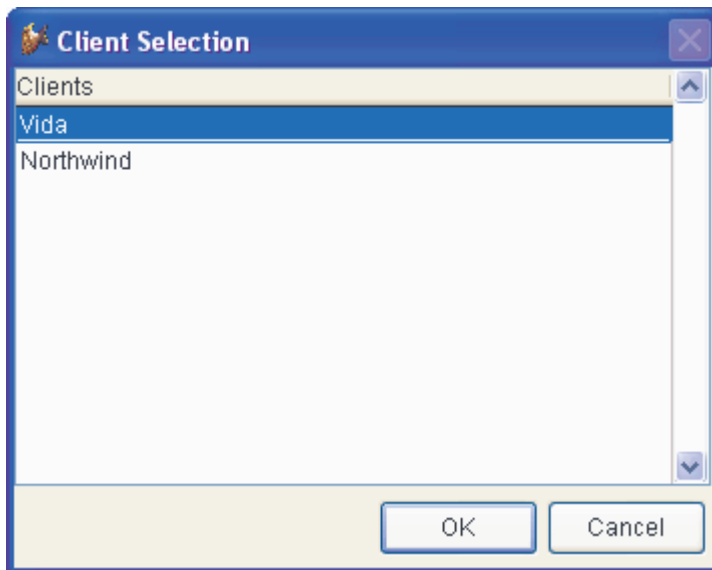
This is all what you need to do to work with views. All further is handled by VFX.

## 11.8. Multi-Client-Support

Usually VFX – Application works with just one database, as it was set in the VFX – Application Wizard. When needed, it is possible to build client management feature. Therefore the property *cDataDir* of the application object *cFoxAppl* in *Appl.vcx* should be set to an empty string.

When the datapath is empty, at run time the VFX Application searches for a file *Config.vfx*. The usage of the file *Config.vfx* is discussed above in the topic *Managing data access using the file Config.vfx*. When the file *Config.vfx* does not exist, VFX – Application checks for the table *Vfxpath.dbf*. This table must be located in same folder where the executable program file is. When this table has exactly one data row, will be used data path, stored there. If the table contains more than one row, when the application starts, a Client Selection dialog is invoked, for choosing desired database.





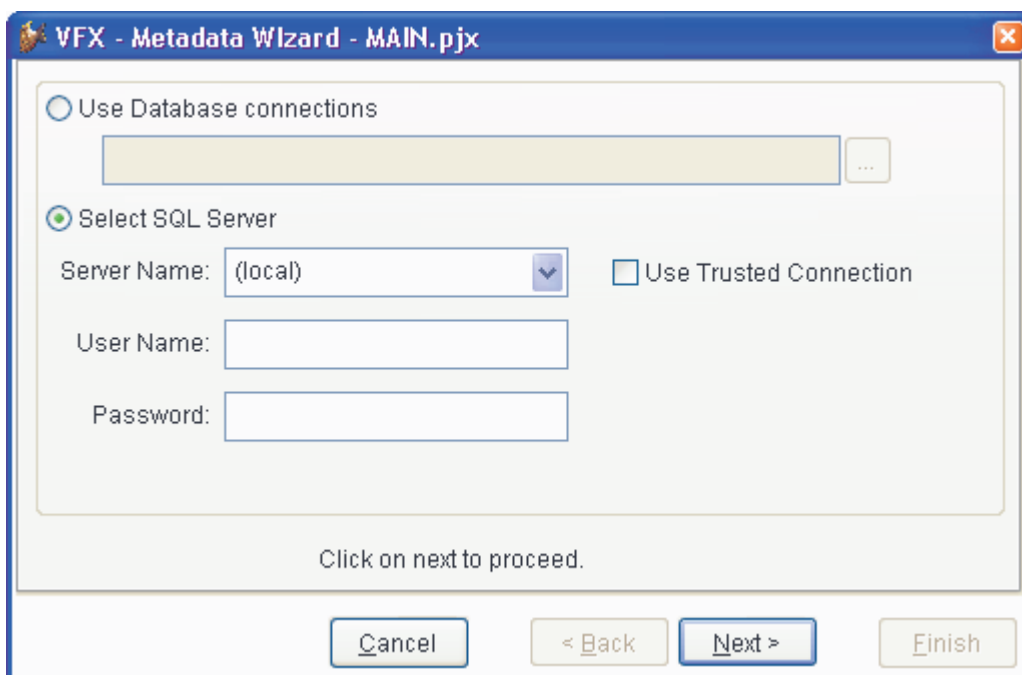
## 11.9. Updating the user's database

### 11.9.1. VFP-Databases usage

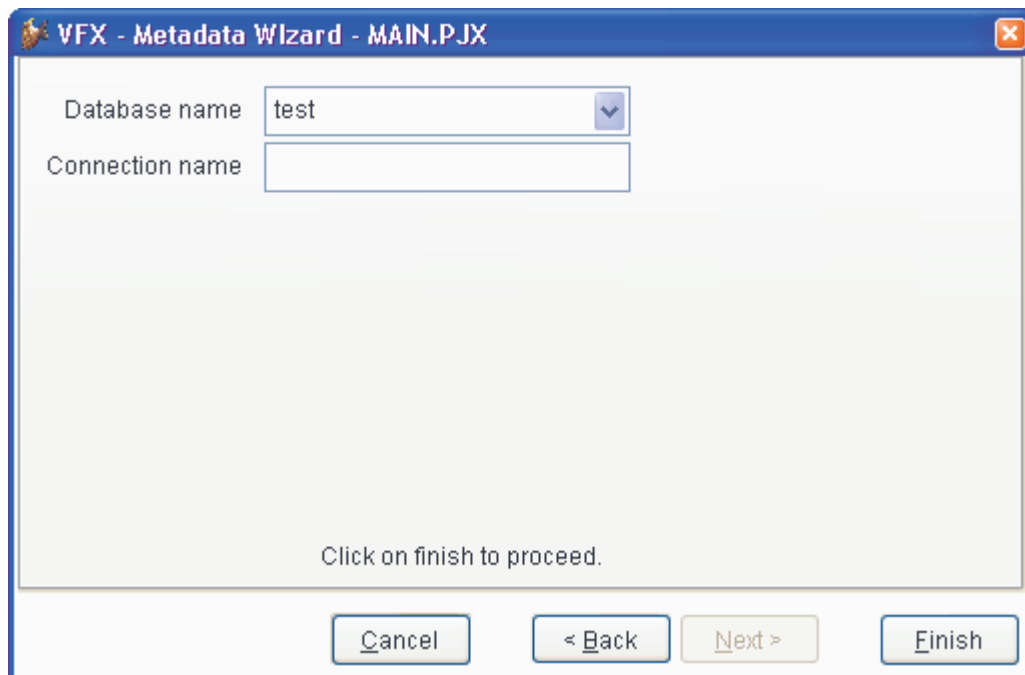
VFX includes procedures to automatically update the database at customer's side. The tables which should be updated, will be copied, without data, in the *Update* folder, under the Data folder. On next start VFX – Application finds out presence of the tables in the Update folder and updates the database. Free tables can also be updated in this way, too. On application start the database in the data folder is updated. Using this approach to the database can be added new tables, new fields in tables, new Indexes and new views. Also fields and tables that are no more used are deleted in this way. Finally all files from the *Update* folder.

### 11.9.2. SQL Server-Database usage

The Metadata Wizard helps you to prepare metadata definition of your currently used database structure. The Metadata can be used for database update at customer's side.



Alternatively you can pick out the connection among the connections to a SQL server created in a VFP database or the SQL server can be manually selected.



The Metadata Wizard creates a table *Datadict.dbf*. This is a free table, where the structure of the SQL Server 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 *Datadict.dbf* is deployed to end-users the data structure update will use it and again scanning existing connections will update user's database structures.

### 11.10. Index files

VFX ensures optimum use of existing index keys. For the incremental search in VFX power Grids VFX automatically considers all existing index keys of the opened table. An index key with *UPPER()* clause is expected for character fields. For date fields is expected an index key with *DTOS()* clause.

If VFX does not find a suitable index key, a temporary index file is created. This index file is deleted, as soon as the form is closed. Furthermore the index file is deleted, if the form changes into edit mode or into insert mode as well as by deleting data records. That is meaningful because if temporary index files are opened running transactions, for example as used in the RI code, would lead to VFP run-time error. VFP does not permit temporary index files, if you work with transactions.

When transactions are used in a form, after the data manipulation the former valid index keys can be set again, if it is necessary. For the user it seems that the selected sorting sequence remains the same. You can set this in VFX – Application Builder by marking *Recreate temporary index files after editing* option.

If in a form and in any code called from it, no transactions in the used tables are invoked and also no RI code is placed, you can set in VFX – Application Builder not to delete temporary index files during the data manipulation. For this, mark the following options: *Disable clearing indexes when editing data*, *Disable clearing indexes when inserting records* resp. *Disable clearing indexes when deleting records*.

Temporary index files are deleted every time when a form. is closed

## 12. Application Protection using Activation Key

The main purpose of using product activation is to prevent unauthorized use of the application on unregistered computers just by copying it.

For a new created project, the application protection through product activation can be switched on in VFX – Application Wizard on Page 3 Options by marking the Checkbox *Enable product activation*

Later this setting can be changed in VFX – Application Builder. The property *goProgramm.UseActivation* must be set to *.T.* in order to enable product activation. When the property *goProgramm.UseActivation* is set to *.F.* the application will not be protected using product activation..

For every application can be defined up to 32 different end-user rights. Every right can be activated independently of the other rights.

### 12.1. List of used terms

*System specific value* – A System specific value, for example the serial number of a hardware component or the creation date of a particular file or a value of an entry in the Windows Registry. Both the used file name and the used Windows Registry key name should be predefined by the developer.

*Activation rule* – For every application can be defined an unique activation rule. This rule consist of consists of a number of system-specific values, which combination unique identifies a particular PC. Also text proceeding functions can be used, when defining Activation rule.

*Installation key* - This is the character string that contains information about the PC gathered using the activation rule. The Installation key is by the developer to prepare an Activation key for the concrete user.

*Activation key*- This is a character string that holds concrete assigned rights for a particular PC. The activation key will be created by the developer based on the Installation key. The Activation key is useless for any other PCs

*Installation date* - This is the date, when the application was started first time on a PC.

### 12.2. How it works

When the application protection using Activation key is activated, on start of the application the object *goProgram.SecurityRights* is created with child properties named according user rights that the developer has defined. Every of these property could have three different values:

*-1* – The application is not activated. In this case the developer can decide what action should be executed. For example, for the user could be denied access to several functions, as long as application is not activated.

*0* – The application is activated, but the user does not have right to perform this action.

*1* – The application is activated and the user has right to perform that action

When application protection using Activation key is enabled, the activation key along with data of the first start of the application is stored in an INI file. The developer can choose the name of this INI file, so that every application uses its own INI file. The default name is *VFX.ini*. The INI file will be stored in the Windows folder.

The Activation key is encrypted using the activation rule. The protection can be improved further by including character constants, keys from the Windows Registry and by the creation date of an arbitrary file. This combination can be different for every particular application, so that every application can have its own unique activation rules

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

By default the protection creates the INI file at when the application is started for first time. The current system date at this moment will be stored in the INI file. This date is then available for later use as the value of the property *goProgram.InstallationDate*. The weakness of this type of protection is that the user can delete the created INI file and the INI file will be created again on the next start of the application with a new date.

To avoid such user's action, the developer can choose to use an additional protection level. It is base on usage of a special file, deployed along with the application installation files. By default this file is named *FirstInstall.txt*. Its name can also be set using *cFirstInstal* property of the class *cActivation (Appl.vcx)*. The file *FirstInstall.txt* will be stored into Windows folder.

When the developer chooses protection with the file *FirstInstall.txt*, the application behavior will be following: When the application is started, first it will check for the INI file. If the file exists, the date of first start of the application will be used to set *goProgram.InstallationDate* and all the rest of the user rights will be created according to the activation key.

If the INI file does not exist, then it is assumed that this is the first run of a newly installed application. To verify if this is really first application start, additional will be checked if the *FirstInstall.txt* exists. If the file exists, it is considered that the application is really started for first time. Then the installation date will be stored into the INI file and the *FirstInstall.txt* will be deleted. If an unfair user tries to re-activate the application by deleting the INI file, application will be terminated, if *FirstInstall.txt* does not exists. This enhanced protection ensures a higher security level. However the developer must not forget to include the file *FirstInstall.txt* in the application installation files and to set it to be placed into the Windows folder.

When the user wants to activate the installed application, he must send his installation key to the developer. The installation key can be sent to the developer using three different ways. The desired method can be set using the property *nRegWay*:

- 0 – Installation key is displayed in a dialog window. The user can copy the key and paste it in another application (for example in an e-mail message).
- 1 – Installation key is stored into a file. Later this file can be send to the developer. The name of the file must be defined in the property *cParamFile*.
- 2 – The Installation key is stored into a file and at same time is sent as an e-mail attachment to the developer. The name of the file must be defined in the property *cParamFile*. The e-mail address of the developer must be defined in the property *cRegEMail*.
- 11 – After invoking the registration dialog, the installation key is stored in a file. This file can be sent later to the developer. The file name is defined in the property *cParamFile*.
- 12 – After invoking the registration dialog, the installation key is stored in a file and is sent immediately as an e-mail attachment to the developer. The name of the file must be defined in the property *cParamFile*. The e-mail address of the developer must be defined in the property *cRegEMail*.

The installation key is a numeric value, 10 digits long. The end-user can send the installation key to the developer by e-mail or could enter it on a registration web page. Form the VFX menu *Activation, Customer List* is started the VFX – Customers List dialog. The developer uses the installation key in the VFX - Create Activation Key wizard to create an activation key for the user. Then the generated activation key it is sent to the user and the user can activate the application by entering the activation key in the registration form. It is also possible just to store the activation key file in the same folder where the EXE file is placed. On next start the application will read the activation key from the file.

The activation information will be stored on the user's PC into an INI file. The name of the INI file is defined by the property *cINIFileName* of the class *cVFXActivation(Appl.vcx)*. The default value is *VFX.ini*.

The developer can choose whether the simple application protection will be used or additionally the file *FirstInstall.txt* will be used, that will be used at first start of the application. The name of this file is defined in the property *cFirstInstall* of the class *cVFXActivation* (*Appl.vcx*). The default value is *FirstInstall.txt*

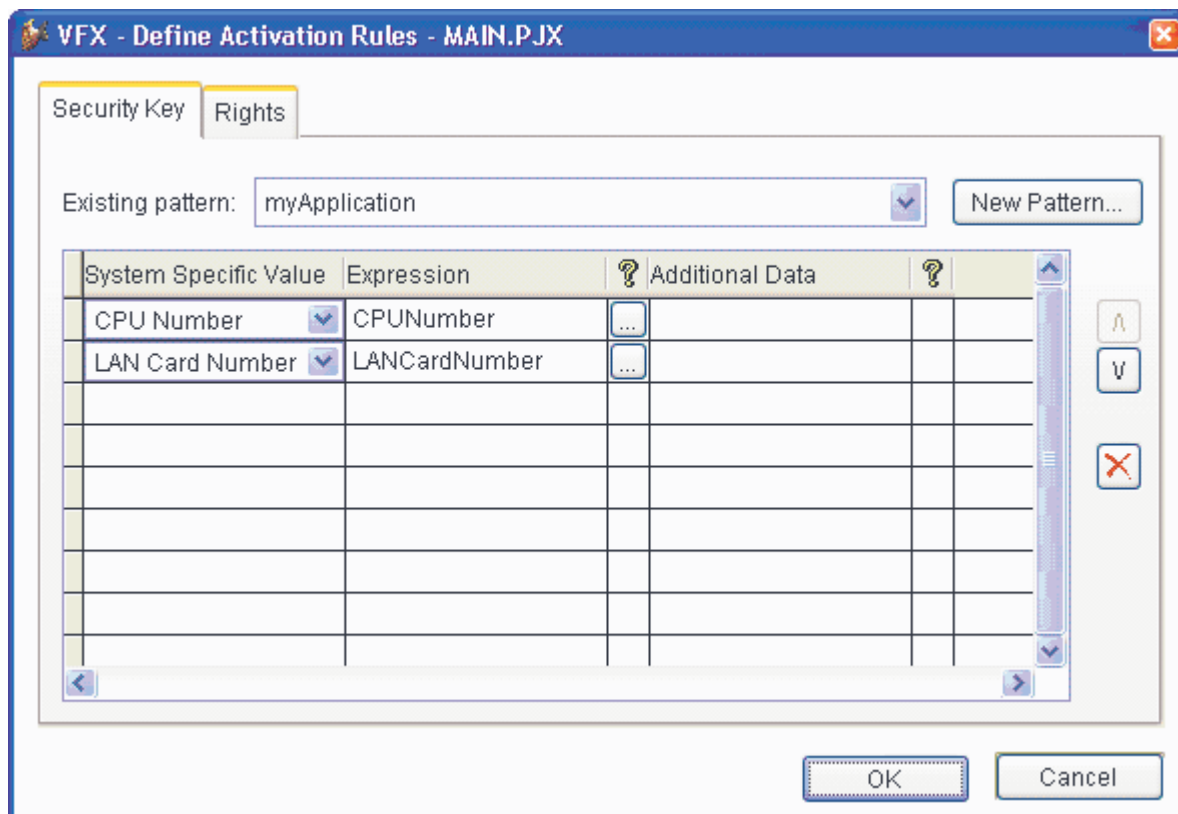
If the file *FirstInstall.txt* will be used, it must be deployed along with the applications' installation files. The installation application must store this file in the Windows folder and the activation object will erase this file when the application is started for first time. At this time the installation date will be stored into the INI file. Later on every application start the INI file will be checked if the stored installation date is available. When the date is missing, and if the file *FirstInstall.txt* is also missing, the object assumes that the installation is not valid and the application will be terminated.

When the file *FirstInstall.txt* is not used, the INI file will be created, in case it is missing.

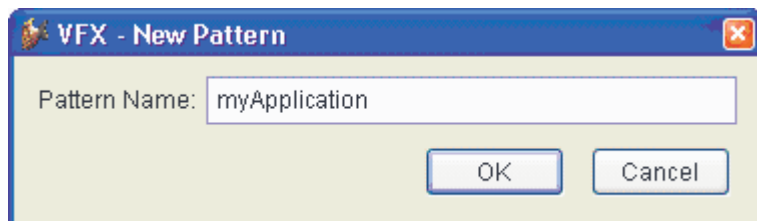
The installation date can be specified in two ways: Either the System date-time will be used or creation date-time of a particular file will be used. If the creation date-time of a file will be used, the name of this file should be stored in the property *cRegFileName* of the class *cVFXActivation*.

### 12.3. Defining Activation Rules

The VFX – Define Activation Rules dialog is started from the VFX menu *Activation, Define Activation Rules*.



When you start the *Define Activation Rules* Wizard for the first time for a particular project, you are asked whether you want to create a new rule for this project.



On the Page Security key of the wizard is placed a Dropdown combo that you can use to change the rule for the current active project. In the grid under it, can be added as many rows as needed. Using all rows in the Grid a key will be generated and stored into the property *cActPattern* of the class *cVfxActivation*. Based on this key, at the user-side, application determines which system specific values must be used to generate the Installation key. The Installation key ensures that the application will run only on the computer, for which the Activation key is created.

In the first column of the grid a System specific value can chosen. In the Drop-Down combo are listed all possible hardware parameters that can be used for creation the Installation key. In addition to this additional string operations can be used for further processing on this value.

For example, instead of the whole hard disk drive Serial number, only the last four digits could be used for creating the Installation key. In the Combobox in the first column *HDD Factory Serial Number* row should be chosen. The correspondent VFX system variable is named *HDDFactoryNumber* and it is automatically filled in the second column of the Grid. To use only the last 4 digits, you need to write the following expression in this column: `RIGHT(ALLTRIM(HDDFactoryNumber), 4)`.

When one of the system specific values: *File Creation Date* or *Registry Key Value* will be used, must be specified additional parameters. When the File Creation Date of a file will be used, the name of the file must be entered. When Windows Registry Key Value will be used, must be specified the name the registry key. These additional parameters are entered in the column *Additional Data*

On the user's PC will be created an Installation key from the activation rule. All parameters, included in the Activation rule will be used. If just one of these parameters of the PC changes, the application registration becomes invalid and the user need to obtain a new activation key, corresponding to the new hardware details.

In the grid can be added as many rows, as needed. Rows in the Grid can also be reordered using the arrow buttons in the right part of the wizard. Reordering the rows in the grid changes the Activation rule.

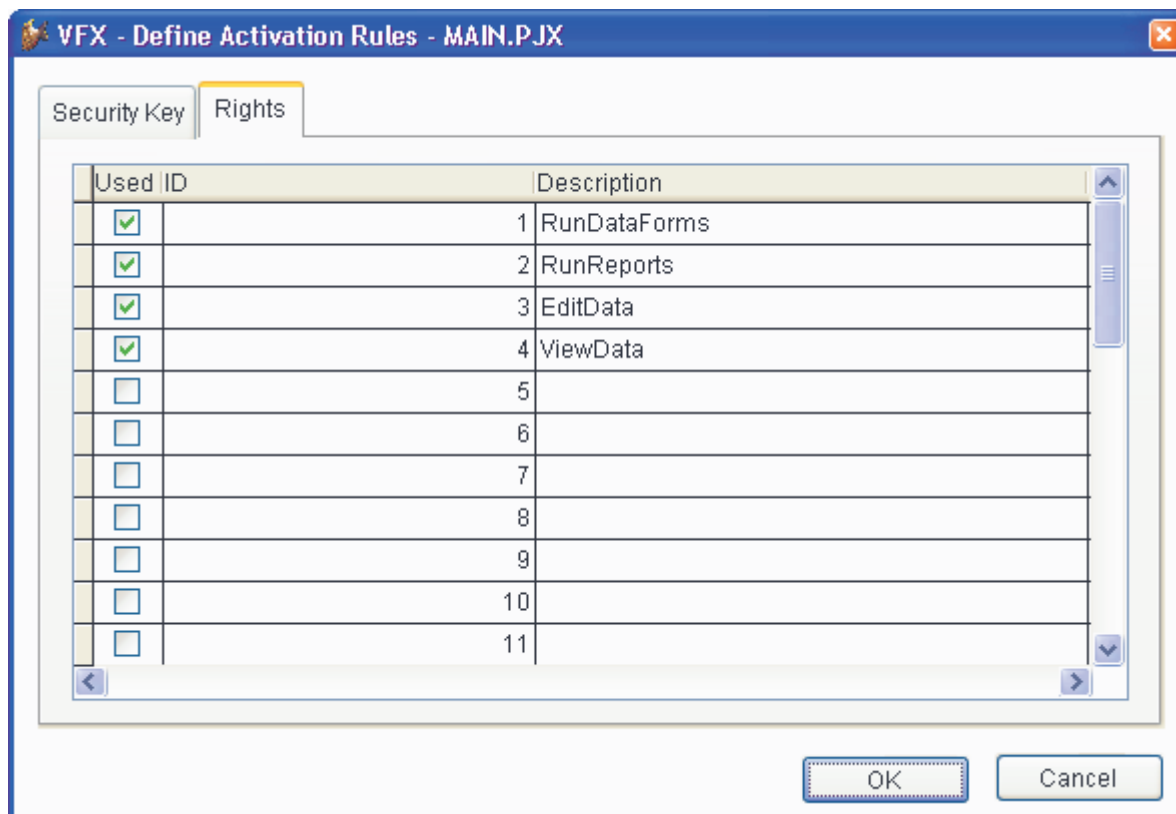
When you save defined activation rules this pattern is saved for later use in the property *cActPattern* of the class *cVFXActivation* (*Appl.vcx*).

---

**IMPORTANT:** The values of the property *cActPattern* must NOT be deleted! Without this value it is not possible to create Activation key!

---

On the page *Rights* can be defined up to 32 different user rights. In this way can be controlled the access to 32 modules in the application For example, can be defined rights that allow the user to: start a form (*RunDataForms*), print reports (*RunReports*), Modify data (*EditData*), Visualize data (*ViewData*) etc. At run-time of the application the respective rights can be checked and correspondent actions can be performed.



All user rights are available for use at run-time as properties of the global object *goProgram.SecurityRights*, so they can be accessed from any part of the application.

While the application is still not activated, all user rights have the value -1. When the application is activated, a user right has value 1, when the action is allowed and 0, when the action is not allowed.

To define a rights in the wizard, first must be marked the checkbox in the first column. Then a name for the right should be entered. At run-time the application will create a property of the *SecurityRights* object with this name.

Because of this, the names should conform to VFP rules for variables naming

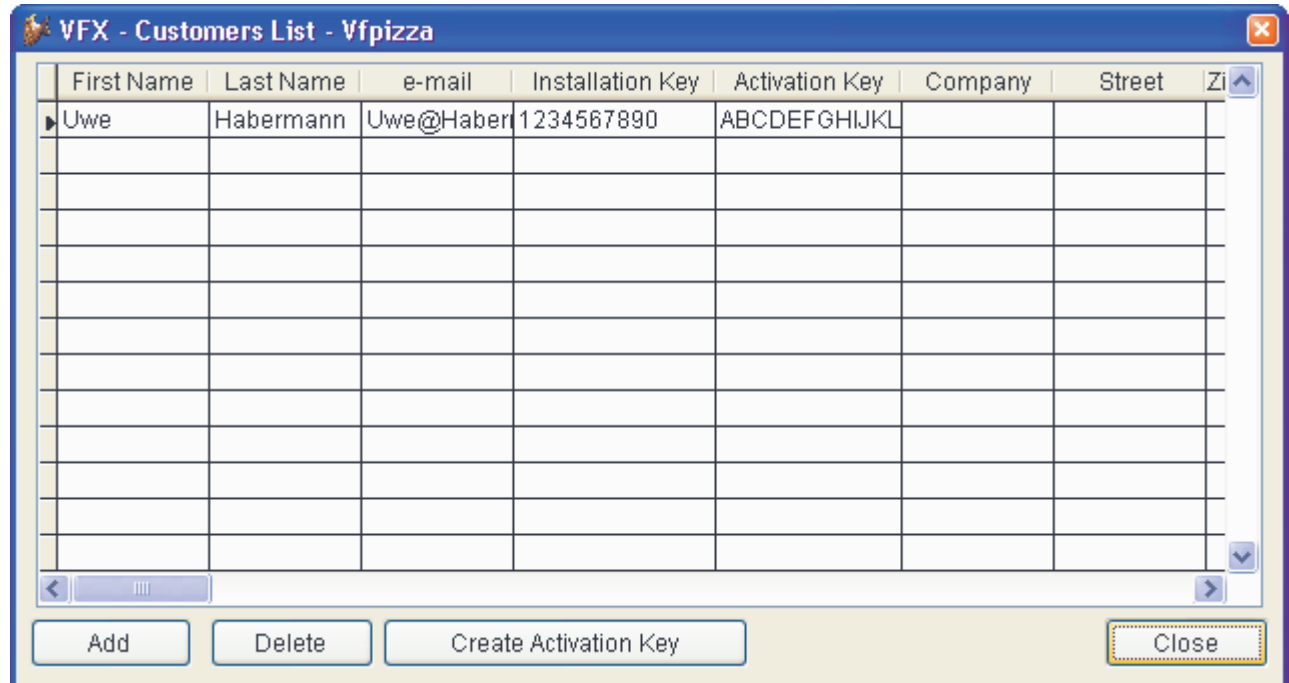
---

**NOTE:** Application rights are specific for every different application. The rights that are already defined cannot be used for another application. When similar rights will be used, they have to be defined again. The application rights are saved in the table *Vfxapprights.dbf*, in the project folder.

---

## 12.4. Generating an Activation key

VFX 9.5 applications can be protected against unauthorized usage with an Activation key. This Activation key holds information for the application, whether the user is allowed to perform a particular action. For every action must be chosen correspondent user right.



First Name	Last Name	e-mail	Installation Key	Activation Key	Company	Street	Zi
Uwe	Habermann	Uwe@Haber	1234567890	ABCDEFGHIJKL			

In the form *Registered Customers* can be added and deleted customers' data. For every user is kept the installation key and granted user rights, so it is possible easy and fast to create and resend activation key when necessary.

*Create activation key* button invokes the well known *VFX Create activation key* form and after setting user's rights and clicking *OK* button in that form, *Installation key*, *User rights* value and *Activation key* are saved back in Customers list.

When the user sends his Installation key, an Activation key should be generated. This Activation key informs the application whether the user may perform a certain action. For every action should be defined corresponding *right*.

When the button *Create Activation Key* is clicked, is invoked a Dialog with the user rights, defined for the active project.



ID	Description	User has this right
1	RunDataForms	<input checked="" type="checkbox"/>
2	RunReports	<input type="checkbox"/>
3	EditData	<input checked="" type="checkbox"/>
4	ViewData	<input checked="" type="checkbox"/>
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		

The Installation key of the user is entered in the lower part of the dialog. The Installation key can be inserted using the clipboard or can be read from a file.

When user rights that will be active are marked, the Activation key will be generated by clicking on the button OK. Generated Activation key will be stored in the file *<Projectname>.xak* in the project folder. The Activation key or the file must be sent to the user, to be used for application activation.

When to the user are given rights, correspondent to the example above, to perform all data processing actions, but not to run reports, the rights loaded at run-time will look like this:

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

When the end-user starts an application that requires activation (and when the application is still not activated), the installation key is automatically generated. Depending of the value of the property *nRegWay*, the generated Installation key is either shown in a form or a file that can be sent via e-mail is created. After the user has received the activation key, he can enter it in the registration window or store the file containing the Activation key into the application folder. This will activate the application.

Later, when the user chooses menu option *Help/Register...* In this case the generated installation key is only shown in the form regardless of the value of the property *nRegWay*.

## 12.5. *cVFXActivation class properties*

*cFirstInstall* – This property contains the name of a file. Depending of the existence of this file, the class decides whether the application is started for the first time. If this property contains an empty string, it is not possible to be checked whether the application is started for first time. The date of the start of the application will be saved in the INI file without further checks.

*cINIFileName* – The name of the INI file, where the date of first start of the application and activation information will be saved. Default value is *VFX.ini*.

*cParamFile* – The name of a file, where will be saved the Installation key. Depending of the value of the property *nRegWay*, this file can be sent by e-mail or processed in other ways.

*cRegMail* – In this property should be stored the E-mail address of the developer, where the file containing the installation key, will be sent, if the value of the property *nRegWay* is 2.

*cRegFileName* - Here can be entered the name of a file, that the application installation will create. The creation date of this file will be used to determine installation datetime. If this property is left blank, system datetime at the first start of the application will be used.

*nRegWay* – In this property can be specifies how the developer will receive the Installation key.

0 – The Installation key will be displayed in Dialog and the user can copy the Installation key and paste it in another application.

1 – The Installation key will be stored into a file. The user can send this file to the developer later. The name of the file should be entered in the property *cParamFile*.

2 – The Installation key will be stored into a file and sent to the developer as an e-mail attachment. The name of the file should be entered in the property *cParamFile*. The e-mail address of the developer should be entered in the property *cRegEMail*.

## 13. Creating multilingual Applications using VFX

VFX is well prepared for creation of multilingual applications. You can choose between runtime localization and design time localization.

### 13.1. Localizing at design time

When creating a new VFX project can be chosen between different application languages. In the new project are generated include files for the selected language accordingly.

If it is needed later to translate the application into another language, for every form the VFX - LangSetup Builder. This Builder constructs an entry for every Caption of the form. At run-time the value of a constant is assigned to the Caption.

The constants can be edited with the VFX - Message editor. Then, before compiling the application, include files for the desired language are simply copied in the project folder and lets the application is built.

The interface elements occur in the following areas:

- User interface of the existing functionality in the Visual-Extend class library and all dialogs.
- User interface of your own applications.

You don't have to worry about first point.

The user interface of the existing functionality in the Visual Extend class library and all dialogs exists in eight languages and you don't have to translate a single word, to create an application in one of the supported languages. If you need the Visual Extend class library in another language, you can easily extend *Vfxmsg.dbf* by your own.

**We would appreciate it, if you would send us your translations of the VFX messages (*Vfxmsg.dbf/cdx/fpt*) in another language that is still not supported by VFX, so we could make them available to other developers as well. Thank you!**

Checklist for the creation of multilingual applications using VFX:

- √ Use the *USERTXT.H*, resp. *USERMSG.H* include files which are generated from the *VFX Message Editor* to manage and store all language specific user interface elements from your application. *The repository for messages as well as captions, tooltip texts and status bar messages is the table VFXMSG.DBF. In this table you can find also all VFX messages and captions which have been translated already.*
- √ In your application define constants rather than the text strings directly. i.e. use **WAIT WINDOW Loc\_Text1** rather than **WAIT WINDOW "MyText..."**.
- √ Use the *USERDEF.TXT* include file for application specific constants which are the same for all languages for better organization of your localization work.
- √ Use the *VFX LangSetup Builder* to create the VFX form method named *LangSetup()* to place the localization code by adding your captions, tooltip texts and so on into this method using define constants. (The VFX LangSetup Builder automatically generates the code for the *LangSetup()* method and updates the *VFXMSG.DBF* table with messages and captions).
- √ Translate your text into the different languages using the *VFX Message Editor*. The VFX Message Editor then generates the include files for the different languages in the folder *\INCLUDE\LanguageDir* whereas *LanguageDir* represents the folder name for the language you plan to translate. (As mentioned above, the VFX specific language includes have already been translated into various languages and you don't have to translate a single word from them.)
- √ To build your application for another language, define the *ID\_LANGUAGE* constant in the *VFXDEF.H* include file and copy the include files from the *\INCLUDE\LanguageDir* back into the current *\INCLUDE* folder of your project.
- √ Rebuild all and test your translated application. You will receive a separate EXE File for every Language.

## 13.2. Run-time Localization

With VFX 9.5 it is possible not only to develop localized applications, but also to allow end-user to change application language at run time.

Runtime localization feature is controlled by the property *goProgram.lRuntimeLocalization*. When the value of this property is set to *.T.*, the working language for the application can be chosen in the Login dialog. Additionally, at run-time, the language can be changed using the language selection combobox, in the main application toolbar.

The property *goProgram.lRuntimeLocalization* can be set using the VFX – Application builder.



When runtime localization is used, a global object named *goLocalize* is created at run time, when application starts. This object has child properties, corresponding to every message row in the table *Vfxmsg.dbf*. For every record in *Vfxmsg.dbf* table is created a child property of the object *goLocalize*. The name of the child property is generated from *Message\_ID*, prefixed with *c*. For instance if the *Message\_ID* in the table *Vfxmsg.dbf* is *CAP\_APPLICATION\_TITLE*, the correspondent property of localization object will be named *goLocalize.cCAP\_APPLICATION\_TITLE*. At run-time, the localization object and its properties are always available.

You can use *goLocalize* object properties every time you need.

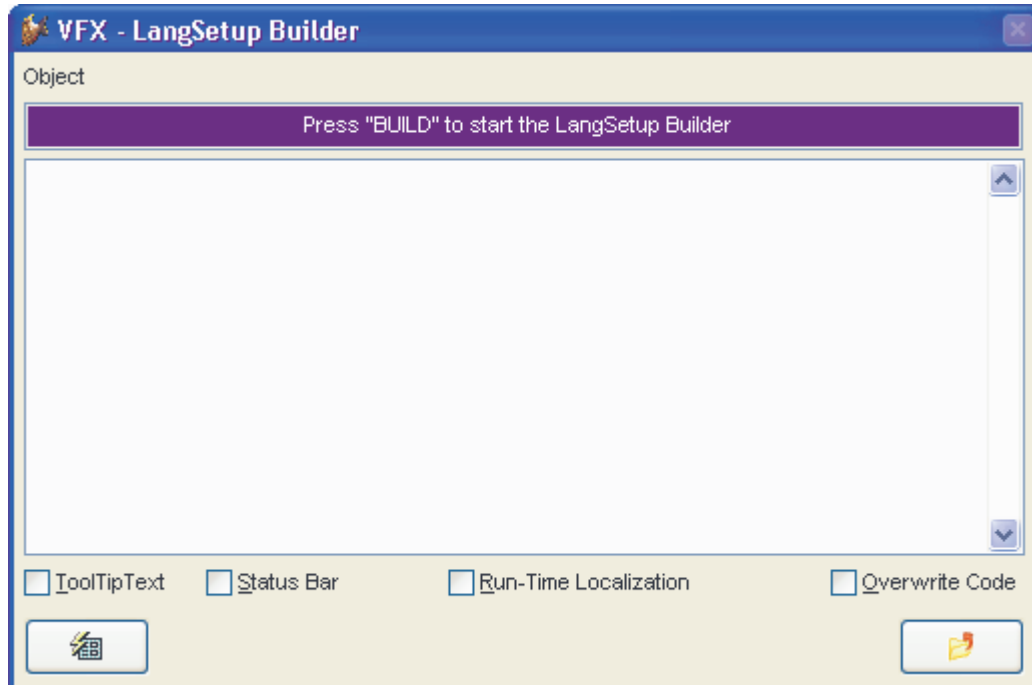
The last used language is kept on a per user basis in the resource table *Vfxres.dbf*. Next time when same user logs in the application, the last chosen language will be automatically selected.

## 13.3. VFX – LangSetup Builder

The VFX LangSetup Builder automates the creation of the needed code within the *Langsetup()* method. You will need this, if you have to supply your application in more than one language. The goal of this builder is to extract all potential captions, tooltip text and Status Bar messages, generating unique constants and putting

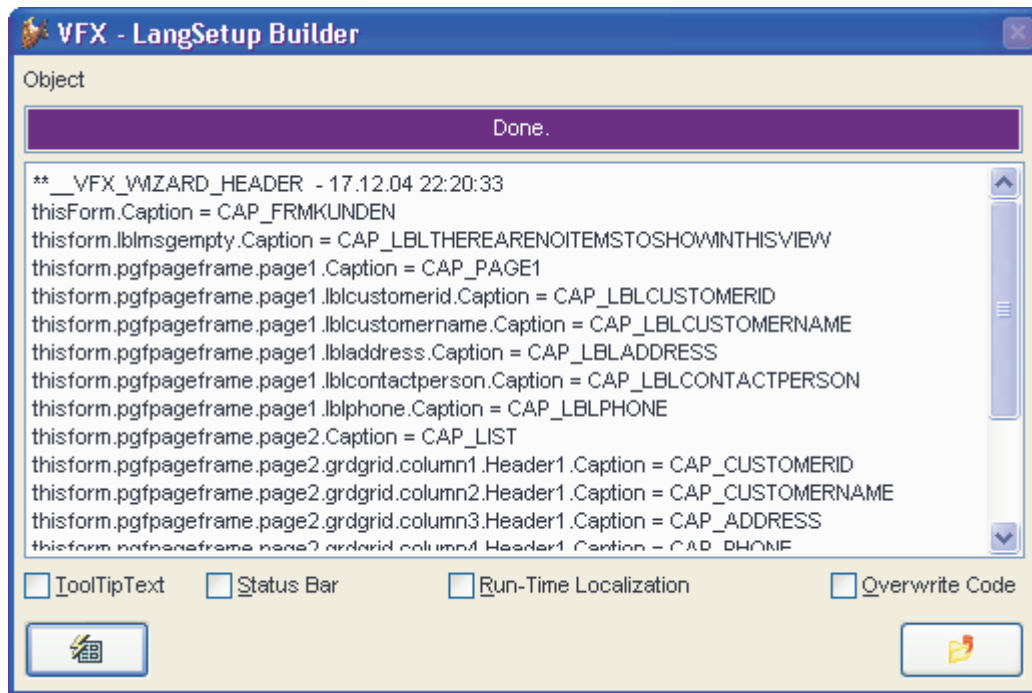
them into the *Vfxmsg.dbf* table, the master table for all include files. After this process, you can use the VFX message editor, described later in this documentation, to translate the text in the different languages.

To call the VFX LangSetup Builder, first open the form for which you want to generate or analyze the captions, tooltip texts and status bar messages. We could say all texts which are applicable for the translation. Then select the following option *Form, LangSetup Builder* in the VFX menu



Mark the Checkboxes correspondent to the desired options. Click on the button *Build* to start the generation of the code for the *LangSetup()*method.

After generation look at the code that were generated for the *LangSetup()*-Method. . If you marked the *Overwrite Code* Checkbox, the generated code will be written in the *LangSetup()*-Method of the currently open form in design mode. The caption codes will be put into the VFX message table *Vfxmsg.dbf* where you can edit and translate them into other languages.



In the Include File *VFX.h*, the constant *\_LANG\_SETUP* defines, whether the *LangSetup()* method will be executed or not. In the *LangSetup()* Method, will be checked whether this constant exists and only if it exists, the code in this method will be executed. This is made for speed optimization of the native forms.

```
#DEFINE _LANG_SETUP          .T.
```

In the Include File *Vfxdef.h* the *ID\_Language* constant defines the current language of your application

```
...
#define ID_LANGUAGE "ENG"
...
```

The first time you create your application using the VFX Application Wizard, the applications will be generated based on your language settings in the VFX Application Wizard screen. If your application has to be translated to another language, other than the one you are currently working on, you will have to switch the *ID\_Language* constant. Please refer to the chapter *Create multilingual Applications using VFX*, for further details.

## 14. VFX.fil

The file *VFX.fil* contains several functions, needed for product activation, data backup as well as for accessing SQL Server and for Internet access. The *VFX.fil* must be deployed along with the application files to the customers. The functions of the *VFX.fil* are described in details.

### 14.1. Product activation

*GetAppRights(lcRightsBin, This.Hex2Bin(This.cActPattern))* – Provides information about user rights from the product activation. An example for the implementation of this function can be found in the class *cVFXActivate* in the method *CheckActState*.

Return Value:

- 0 – Operation completed successfully.
- 1 - Invalid activation key length.
- 2 - Invalid activation key structure.
- 3 - Encryption error.

*GetFileCreationDateTime(cFileName)* – Returns the creation date/time of a file. An example for the implementation of this function can be found in the class *cVFXActivate* in the method *Init()*

*cFileName* – Name of the checked file.

Return Value: A date/time value as string.

*GetSysInfo(This.Hex2bin(This.cActPattern))* – This function creates the installation key. An example for the implementation of this function can be found in the class *cVFXActivate* in the method *CheckActState*.

### 14.2. Data backup and archiving

*CreateZipArchive(tcPath, tcFileMask, tcArchiveFullPathName, tcFeedBackFunction, tnCompressionLevel, tlRecurseSubfolders, tcPassword)* – Creates a Zip archive file. An example for the implementation of this function can be found in the class *cArchive* in the method *CreateArchive*.

*tcPath* – Path to the folder to be archived.

*tcFileMask* - Names of the files to be archived.

*tcArchiveFullPathName* – Full pathname for the created Zip archive.

*tcFeedBackFunction* – Name of a function or a method, which will be called from *CreateZipArchive* for progress feedback.

*tcFeedBackFunction(cCurrentOperatedFile, nState, nAllFilesSize, nZIPedFilesSize, nArchiveCurrentSize)* – This function or method will be called from *CreateZipArchive* every time

- when Zip file that should be created already exist,
- before a file would be added to archive,
- after a file has been added to archive,
- after an archive has been successfully created,
- when an archive cannot be created,
- there are no files to be added to the archive.

*cCurrentOperatedFile* – Name of the file that is processed at the moment.

*nState* - Status

- 1 – The file *cArchiveFullPathName* already exists.
- 2 – Adding *cCurrentOperatedFile* to the archive started.

- 3 – The file *cCurrentOperatedFile* were added to the archive.
- 4 – File *cCurrentOperatedFile* cannot be added to the archive.
- 5 – Archive creation completed successfully.
- 6 – The archive cannot be created.
- 7 – No files or folders to be added to the archive.

*nAllFilesSize* – Total size of all files which will be added to archive.

*nZIPedFilesSize* – Total size of files added to archive so far.

*nArchiveCurrentSize* – Current size of the archive.

*tnCompressionLevel* – The ZIP algorithm uses different compression levels. The acceptable values are -1 through 9:

- 1 – Default compression level
- 0 – No compression
- 1 – Fastest execution
- 6 – Default compression
- 9 – Best compression

The values not listed here allow more precise selection and thus are a compromise between speed and compression level. The standard compression can be reached alternatively with the value-1 or with the value 6.

*tlRecurseSubfolders* – When the value of this parameter is set to *.T.* the subfolders are included in the archive recursively. The files, corresponding to the file mask *tcFileMask* are also searched in subfolder. If the value of this parameter is *.F.*, files, being in subfolders, are not processed.

*tcPassword* – Here must be passes a password if the archive should be protected. If no password protection is required, an empty string must be passed. For the password are allowed all characters, except CHR (0).

Return Value:

- 0 – Cancel archiving operation.
- 1 – The files will be added to archive.
- 2 – Continue archiving operation.

*ExtractZipArchive(tcExtractFilesFolder, tcFileMask, tcArchiveFullPathName, tcFeedBackFunction, tcPassword)* Extracts files from Zip Archive file. An example for the implementation of this function can be found in the class *cArchive* in the method *cExtractForArchive*.

*tcExtractFilesFolder* – Folder, where files extracted from archive will be stored.

*tcFileMask* - Names of the files to be extracted. Multiple file names can be specified in a semicolon separated list. Meta-characters can be used.

*tcArchiveFullPathName* – File pathname for the Zip archive.

*tcFeedBackFunction* – Name of a function or a method, which will be called from *ExtractZipArchive* for progress FeedBack.

*tcFeedBackFunction(cCurrentOperatedFile, nState, nAllFilesSize, nZIPedFilesSize, nArchiveCurrentSize)* – This function or method will be called from *ExtractZipArchive* every time when

- Current file that would be extracted already exist,
- Extraction of a file begins,
- Extraction of a file ended,
- A file cannot be extracted from the archive,
- The extraction of all files completed successfully,



The extraction of all files did not complete successfully.

*cCurrentOperatedFile* – Name of the file extracted at the moment.

*nState Status*

- 1 – File *cCurrentOperatedFile* already exist.
- 2 – Extraction of file *cCurrentOperatedFile* started.
- 3 – Extraction of file *cCurrentOperatedFile* finished.
- 4 – File *cCurrentOperatedFile* cannot extracted.
- 5 – Extraction operation completed successfully .
- 6 – Extraction operation did not complete successfully.

*tcPassword* – Password for extraction from the archive. If no password will be used, an empty string must be passed.

Return Value:

- 0 – Cancel extraction operation.
- 1 – Continue extraction operation.
- 2 – Overate existing file with the file from archive.

### 14.3. SQL Server

*GetSQLServers(@cServersString, @cErrorString)* – Enumerates all available SQL-Servers. Extracts files from Zip Archive file. . An example for the implementation of this function can be found in the function *TryConnecting* in *Vfxfunc.prg*

*cServersString* - String containing names of all available SQL-Servers, as a comma separated list.

*cErrorString* – Contains eventually error message in case an error occurred.

Return Value: Number of available SQL-Servers.

*GetSQLDataBases(cServer, @cDBString, cUser, cPass, @cErrors)* – Retrieves all databases on a SQL Server

*cServer* – Name of the SQL Server, from where databases will be retrieved.

*cDBString* – A string containing all available databases in comma separate format.

*cUser* – Username used to login to the SQL Server.

*cPass* – Password used to login to the SQL Server.

*cErrors* – String containing eventually error message in case an error occurred.

Return Value: 0 – Operation completed successfully.

### 14.4. Internet, E-Mail and Support functions

*URLDownload2File(vcUrl, cFileName, cFeedBackFunction, cCancelDownload)* – Downloads a file from internet. An example for implementation this of this function can be found in the class *cDownload* in the method *download*

*cUrl* – URL address from where the file should be downloaded.

*cFileName* – File name or Full path name. Here will be saved the downloaded file.

*cFeedBackFunction* – Name of a function or a method, which will be called from *URLDownload2File* for progress FeedBack. The function or method must accept two parameters.

*cFeedBackFunction(nCurrentAmount, nFileSize)*

*nCurrentAmount* – Amount of bytes, already downloaded.

*nFileSize* – Size of the downloaded file.

*cCancelDownload* – Name of a variable or property that controls the download process. The variable or property will be automatically checked.

*cCancelDownload* = *.F.* – The download operation will continue.

*cCancelDownload* = *.T.* – The download operation will be canceled.

Return Value: *0* – The download completed successfully.

*Get\_PS\_Printers* (*nPrinterType*, *@cPrinterNames*, *@nPrinterNamesLength*) – Return names of all installed post script printer drivers. An example for implementation of this function can be found in the class *cCreatePDF* in the method *CheckPSPrinter*.

*nLocation* – Printer location to be searched.

1 – Search for local printers.

2 – Search for network printers.

3 – Search for both network and local printers.

*cPrinterNames* – Contains names of all installed postscript printer drivers in a comma separated list.

*nPrinterNamesLength* – Length of the returned string

Return Value: *0* – The operation completed successfully.

*Add\_Printer*(*cPrinterName*, *cPrinterPort*) – Automatically installs a printer driver. An example for implementation of this function can be found in the class *cCreatePDF* in the method *CheckPSPrinter*.

*cPrinterName* – Name of the printer driver that will be installed.

*cPrinterPort* – Port the printer driver that will be installed.

Return Value: *0* – The operation completed successfully.

*Encrypt*(*cStringForEncrypting*, *cPassword*) – Encrypts a string with a password. An example for implementation of this function can be found in the class *cDunConnectiom.cmdOk* in the method *Click()*

*cStringForEncrypting* – String to be encrypted.

*cPassword* – Password, to be used for encryption.

Return Value: Encrypted string

*Decrypt*(*cStringForDecrypting*, *cPassword*) – Decrypting a string with a password. An example for implementation of this function can be found in the class *cDunConnectiom.cmdOk* in the method *Init*

*cStringForDecrypting* – String to be decrypted

*cPassword* – Password, to be used for decryption

Return Value: Decrypted string

*GetAxControlSize*(*nhWnd*, *@nWidth*, *@nHeight*) – Retunes the size of an ActiveX-Control. An example for implementation of this function can be found in the class *cCalendar* in the method *Resize*

*nhWnd* – Handle of the Active-X control.

*nWidth* – Width of Active-X control.

*nHeight* –Height of Active-X control.

Return Value:

.*T.* – The size of the Active-X control was successfully determined.

.*F.* – The size of the Active-X control cannot be determined.

*SetModemConnection(cConnectionName, cPhoneNumber, cUserName, cPassword)* – Creates a Dial-Up connection entry. An example for implementation of this function can be found in the class *cDownload* in the method *EstablishDUNConnection*. For successful execution of this function, a modem driver must be installed.

*cConnectionName* – Name of the connection that will be created.

*cPhoneNumber* – The phone number that will be dialed.

*cUserName* – User name for the connection.

*cPassword* – Password for the connection

Return Value:

.*T.* – The Dial-Up connection entry was created successfully

.*F.* – The Dial-Up connection entry cannot be created.

*CheckInetConn(cCheckURL, cDUNConnName, nHWnd)* – This function checks whether a connection with Internet exists. For this purpose will be made an attempt to access an URL in the Internet. An example for implementation of this function can be found in the class *cDownload* in the method *CheckInterneCconnection*.

*cCheckURL* – This URL will be checked to ascertain that a connection with Internet exist.

*cDUNConnName* – Using this Dial-Up connection entry will be established a connection, if necessary.

*nHWnd* – Handle of the calling window.

Return Value:

*0* – A connection with the Internet exists.

*-1* – The connection establishment operation was canceled by the user.

*-2* – No connection with Internet.

*-3* – An error occurred.

*24* – The Dial-Up connection entry named *cDUNConnName* does not exist.

## 15. VFX – AFP Wizard

This Wizard generates a ready to run active AFP web pages, from an existing VFX 9.5 forms.

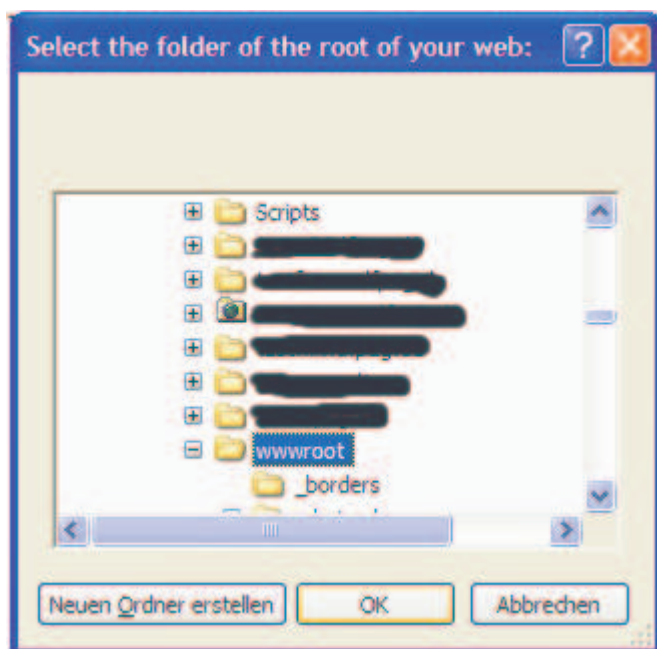
An actual version of AFP (Active FoxPro Pages) can be found on <http://www.afpages.de>.

At the moment the Wizard supports only forms which work with DBF tables. CursorAdapters classes will be supported in a future version.

The Wizard works with forms, based on one of the VFX form classes *cDataFormPage* or *cTableForm*. Further form classes VFX will be supported in a later version.

At the first start of the Wizards the used Metadata table *Vfxafpmeta.dbf* is created in folder `C:\Documents and Settings\All Users\Application Data\DFPUG\Visual Extend\9.5`.

The default suggested output path for the generated AFP pages is obtained from the Registry `HKLM\SOFTWARE\Microsoft\InetStp`.

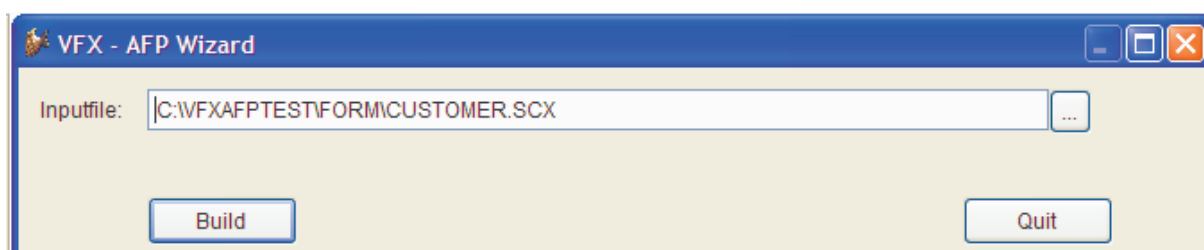


Choose here the local web path in which the files should be saved.

On every run of the Wizards it is automatically checked if additional required files exist. When needed these are automatically created.

The folders are *vfxafpstyle* for stylesheets, *vfxafpimage* for the pictures and *vfxafpjs* for the Javascript which is currently used in the Grids.

Now the Wizard appears.



Pick up the VFX form and click on *Build*.

---

Note: The latest used form will be selected automatically.

---

The login dialog is invoked, exactly like as you had start the application.

The AFP pages are generated and then can be run under

[http://localhost/myFolder/frm\\_FormName.afp](http://localhost/myFolder/frm_FormName.afp)

In case of error:

The error, which appears if a form is (seriously) changed and then immediately the Builder is run, is

*Error loading form*

after leaving the login dialog. This is caused the resource file, which must be deleted first with the user's management in the running program.

Run your application, log in and then go under user's management on the page *Editing*.

There you can click the button *Delete settings*.

### 15.1. Vfxafpmeta.dbf description

The table has following fields:

ckey	Contains the class names
cdesc	A short description
cmemo	The content for instance the HTML code
lparam	.T. means this is a parameter for input control
lCode	.T. means that the content of cmemo will be passed through execscript.
nvers	The actual version number

There are 5 parameters:

Outputpath	The path which must be given by the first start of the Wizards.
Prefix	The Prefix which is placed in the beginning of every form name. Default = <i>frm _</i>
Postfix	The suffix which is appended to the form name.
Extension	The extension of the generated files. Default = <i>.AFP</i>
Postfixexec	the suffix for the EXEC files, which contain the code to process the inputs.

Every used class in the form is presented with two records.

Most simply this can be explained with the Pageframe which consists of Pageframe and page.

Within a Pageframe can be placed many Pages. Thus the Pageframe must also be closed at the end.

The beginning code is in the record pageframe then comes the record page, all elements contained in it like textboxes or labels and now the code must end with Page\_end and Pageframe\_end which contain closing code.

In case of pageframe this is:

```
<div id="<<cname>>" class="pageframe" style=" position: relative;
width: <<nwidth>>px;height: <<nheight>>px; z-index:<<nlevel>>; left:
<<nleft>>px; top: <<ntop>>px" >
```

And in pageframe\_end is also this

</div>

In this way will be processed with every object, every class.

If a class is not filled, the base class is automatically searched and is pulled up. Thereby is organized a little object-oriented hierarchy.

## 16. Advanced development techniques

### 16.1. Adding a form in the Open-Dialog

VFX gives you a template for an Open-Dialog. Of course you can adapt this dialog to your needs or provide your own dialog.

In addition to the Open-Dialog (*Vfxfopen.scx*), existing in former VFX versions, in VFX 8.0 is available a new Open-Dialog in Windows XP style (*Vfxxpopen.scx*). This new Open-Dialog is activated by default. If you wish, you can switch to the old Open-Dialog, using the property *goProgram.lxpOpenStyle*.



*lxpopenstyle*

.T. – the new Open-Dialog in the Windows-XP-style will be used.

.F. – the old Open-Dialog (*Vfxfopen.scx*) will be used.

The Group Headings in the new Open-Dialog are read from the new table field *Vfxopen.groupcap*. The state of the particular groups (expanded or collapsed) is stored for each user.

The File/Open-Dialog uses the table *Vfxfopen.dbf*. For each form the VFX – Form-Builder adds automatically a data record in the table *Vfxfopen.dbf*. Here is the structure of the table *Vfxfopen.dbf*:

<i>VFXFOpen-Field</i>	<i>Description</i>	<i>Example</i>
<b>ObjectID</b>	This field is used, when the Open-Dialog <i>Vfxfopen.scx</i> is used. For this the property <i>goprogram.lxpopenstyle</i> must be set to .F. Usually the VFX Open-Dialog has two pages. ( <b>Tip:</b> You can set the Pagecount property of the page frame to any value, in order to change the number of the pages in the form <i>Vfxopen.scx</i> .) If you want your form to appear on the page 1 of the page frame, enter in this field PAGE1. For the following pages enter PAGE2, PAGE3 etc.	PAGE1
<b>ObjectNo</b>	Enter a number for the order sequence in the list. 1 will become the first element, then follows 2 etc.. The Ordering will be used on each page.	1
<b>GroupCap</b>	This field is used when the Open-Dialog <i>Vfxpopen.scx</i> is used. For this purpose the property <i>goprogram.lxpopenstyle</i> must be set to .T. This field contains a group caption. The grouping takes place according to the entries in field ObjectID. The GroupCap must be entered only for the first entry of a group.	Contacts
<b>Title</b>	Enter the title, which appears in the list window.	Customers
<b>Descr</b>	Enter a description text, which will be shown, when the user selects this entry.	Address list
<b>Form</b>	Enter the names of the called forms.	ADRE
<b>Parameter</b>	If you want to pass parameters to the form, you can enter them here.	
<b>Viewlevel</b>	The user level, which is required to run the form (for example 1 = Admin, 2 = Head user, 3 = Common user etc.)	1 (only administrators can run this form)
<b>NewLevel</b>	The user level, which is required in order to be able to add new data records in the form.	1 (only administrators can add new data records)
<b>EditLevel</b>	The user level, which is required in order to be able to edit the data records	1 (only administrators can edit the data records)
<b>Eraselevel</b>	The user level, which is required in order to be able to delete data records on this form.	1 (only administrators can delete data records)
<b>Favorites</b>	This form can be added to the Favorites menu.	.T.
<b>PrimaryKey</b>	The Primary key used to manage the Favorites menu.	ID
<b>FavorDescr</b>	Description for entry in the Favorites menu.	
<b>InetLevel</b>	Access right for AFP forms	1 (only Administrators can access AFP-Forms)

## 16.2. System settings in Options Dialog


In the Options Dialog the fields of the table *Vfxsys.dbf* can be edited. The programmer can also add to this table fields with global settings. At run-time values of all fields of *Vfxsys.dbf* table are available as properties of the global object *goSystem*

## 16.3. Active Desktop

The Active Desktop gives a professional look to the applications. On the otherwise empty screen are placed icons and options. By moving the mouse over the pictures is displays the associated menu below the pictures. In the menus there are underlined menu options similar to hyperlinks in the Internet Explorer, that can be simply clicked and an action is started. In most cases as action will be started a form.



The class of the Active Desktop is in the class library *Appl.vcx* and according to the wish of the developer can be extended with any controls.



**Simple**

- [Parent](#)      Parent form which acts as parent form in a linked child scenario plus more...
- [Child](#)      The same child form, just called directly, why not...
- [Item](#)      Item table, shows the cTableForm class, very handy...
- [OneToMany](#)      OneToMany form with parent -> child, almost a classic...
- [OneToMany2](#)      OneToMany form item -> child, you are flexible, aren't you...
- [ParentTree](#)      Parent Tree form shows the cTreeView class
- [OneToTree](#)      Shows the cTreeViewOneToMany class

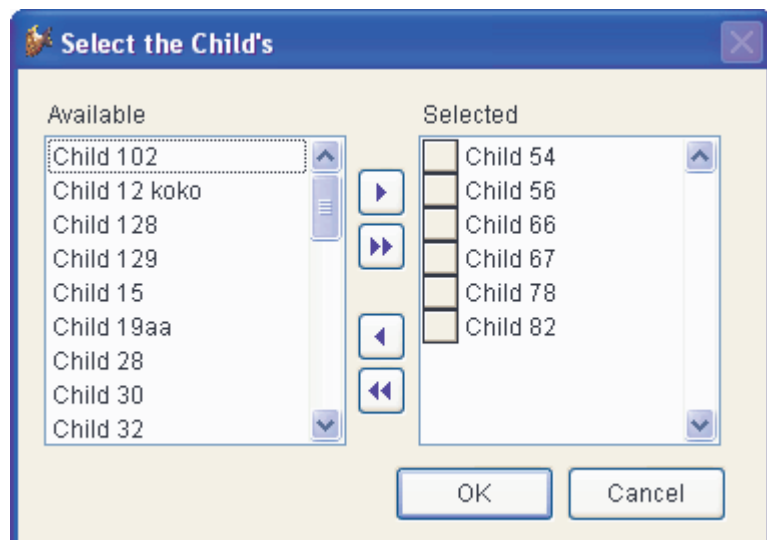
The Active Desktop can be used in addition to or in place of the Open Dialog forms.

### 16.4. More Functions

Through a form property (*IMore*) the button *More functions* in the standard toolbar can be activated. In the Click method of this button is called the *OnMore* method of the active form. This method already contains a template code, which can be easily changed. Here is created an array containing parameters passed to the called *VFXMore* form, where in a dialog can be selected between the available functions. For example can be started Child forms.

### 16.5. Mover-Dialog

The VFX Mover dialog is an efficient control, which you can use in your applications. The VFX Mover dialog gets two arrays, passed as parameters. The first array contains elements available for selection. These elements are displayed in the left list box. The second array contains the selected elements. The second array can be emptied with a call from the mover dialog. The user can select any number of elements.



Here is an example code for the usage of the VFX Moverdialog controls in practice:

```

LOCAL laSource[1,1], loMover

*--prepare the array of all available items
SELECT keygrp_id, keygrp_name FROM keygrp INTO ARRAY laSource

*--create the mover object based on the VFX Class CMoverDialog
loMover = CREATEOBJECT("CMoverDialog")
*--set the caption
loMover.Caption = CAP_KEYFIELDGEN
*--set the property which defines which column from the array get's displayed
loMover.cntMover.nColToView = 2
*--enable multiple selections
loMover.cntMover.lstSource.MultiSelect = .T.
*--pass the array of all available items
* here you can also pass a second parameter if you want to define, which
* elements from the array must appear as already selected
loMover.cntMover.SetData(@laSource)
*--show the mover dialog
loMover.Show()

*--Result: The Public Array _gaMoverList contains the selected items, use it
* and release this Public Array after you have done.

```

After the creation of the object *loMover* you have the complete control over it and you can change all desired properties and methods.

---

**NOTE:** In order to obtain a detailed technical description of the VFX Class Libraries, including all properties and methods, please read in the VFX technical reference.

---

## 16.6. OLE-Classes

It is possible to use OLE automation for Word, Excel, Outlook and PowerPoint from VFX applications. The most important functions are available in classes.

## 16.7. Debug-Mode

Through setting a constant the application can be started in Debug mode. In Debug mode is available an additional menu, with which assistance the debugger can be started at any time. In addition the debuggers can be started by one right-click with the mouse on a form. Also the data environment window is opened.

VFX uses a constant in *VFX.h* include file which defines whether your application runs in debug mode or not. By default the following code is placed in the *Vfxmain.prg* to call the *debugmode* method with a true parameter to activate debug mode depending on *\_DEBUG\_MODE* constant setting:

```

#ifdef _DEBUG_MODE
    goProgram.DebugMode(.t.)
#endif

```

If you don't want Debug Mode code execution, in the Include File *VFX.h*, comment the line, where the *\_DEBUG\_MODE* Constant is defined:

```

...
* #DEFINE _DEBUG_MODE          .T.
...

```

## 16.8. Delayed Instantiation

The load time of a form essentially depends on the number of controls, which must be loaded with the form. But however not all controls of a form are immediately visible when a form is started. If user works with a page frame, first only the controls of a page are visible. The controls of the others, not visible at first pages, do not have to be loaded at all. Only first time when the user activates another page, the controls present at this page must be loaded

The Delayed Instantiation is supported by VFX by the very practical function *AddPageDelay()*.

For this purpose, all controls of a page frame must be saved in a container as a class. To accomplish this, in the VFP form designer, mark all controls of the current page and select *Save As Class* option in the menu File. The class should be stored in the class library *Appl.vcx*. This class library is available to the developer for storing own classes. When saving as class, VFP automatically fills a container with the selected controls. The name of the class should be selected in such a way, that the reference to the form and the page of the pageframe are easily seen. Now the controls stored as a class can be deleted from the page frame.

The function *addpagedelay()* is used to load the container of the form at run-time. The call must be included into the *Activate()* Event of the current page and looks in this way:

```
AddPageDelay(thisform, this, 'x', '<Name of the class>')
```

It is recommended first to develop and test a form without Delayed Instantiation. If the form is nearly finished, it can be reorganized to use Delayed Instantiation. Pay attention that references to particular controls must be changed. Before the conversion to Delayed Instantiation it could refer to a text box for example like this:

```
Thisform.pgfpPageframe.Page1.txtMytextbox
```

After the conversion to Delayed Instantiation the reference looks in this way:

```
Thisform.pgfpPageframe.Page1.x.txtMytextbox
```

The x here is the name of the container, in which are the controls of the page.

## 16.9. Important VFX – Methods

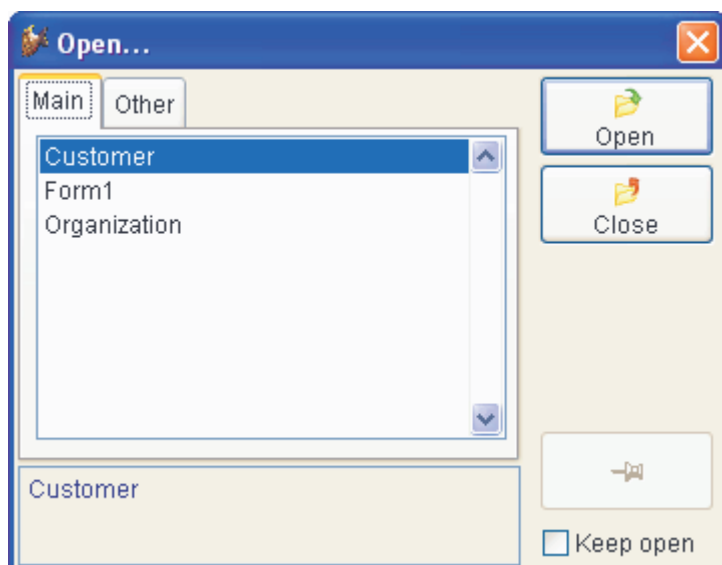
### 16.9.1. Form methods

#### *Valid*

VFX offers a Valid method on Form level. This method is always called, when the data of the form are to be saved. Here should be performed all validations. If this method returns the value *.F.*, the saving procedure will not continue and the form remains in edit mode. The data is saved by returning of *.T.*

#### *OnMore*

With the assistance of this method it is in particular possible to call Child-forms. If you like, a ready Template-code can be generated in the form, from the VFX - Form builder. Depending on the application, only few values of this method need to be adapted from the developer.



In the *OnMore()* method at run-time will be shown a Dialog, where the user can select the Child-form that will be called.

### *Onpostinsert*

This method is called immediately after adding a new data record, before the user to receive the opportunity for data manipulation.

Here can also be filled default values in the fields. This method also gives you possibility to assign primary key values.

### *Onrecordmove*

Each time, when the record pointer is moved, this method is called. Here can be shown or updated, values which do not originate from the database.

## 16.9.2. Methods of the Application object

### *OnPreStart*

In this method can be written code that will be executed when the Start method is called.

### *OnPostStart*

In this method can be written code that will be executed after the call of Start method.

## **16.10. VFX primary key generation**

It is possible you not want to show the primary key of some tables to the users. But for a correct database design you want to use a primary key. For this and similar situations VFX offers a function, which makes possible generation of primary keys and operates in a multi-user environment exactly the same way, as in an client/server-environment.

The modular design of the VFX class hierarchy gives you the possibility to make modifications after inserting a new data record. Apart from many other functions, VFX provides a method with the name *OnPostInsert()*, which is invoked in the moment, when a new data record is added. Normally VFX provides methods for all important events, which are executed automatically before, during and after the event. In this case, when a new data record is added, the following methods are available:

- *OnPreInsert()*
- *OnInsert()*
- *OnPostInsert()*

In addition, there is a property that specifies whether the user can add new data records. This property is named *lCanInsert*.

---

**NOTE:** For further information please read the VFX Technical Reference.

---

In order to generate a primary key, you could insert the following code into the *OnPostInsert()* method of your form. The code calls the function *GetNewId()*. The parameter specifies the table, for which the key is generated.

```
DODEFAULT()  
REPLACE comp_id WITH GetNewId('CUSTOMER') IN customer
```

The last generated key value is stored in the table *Vfxsysid.dbf*.

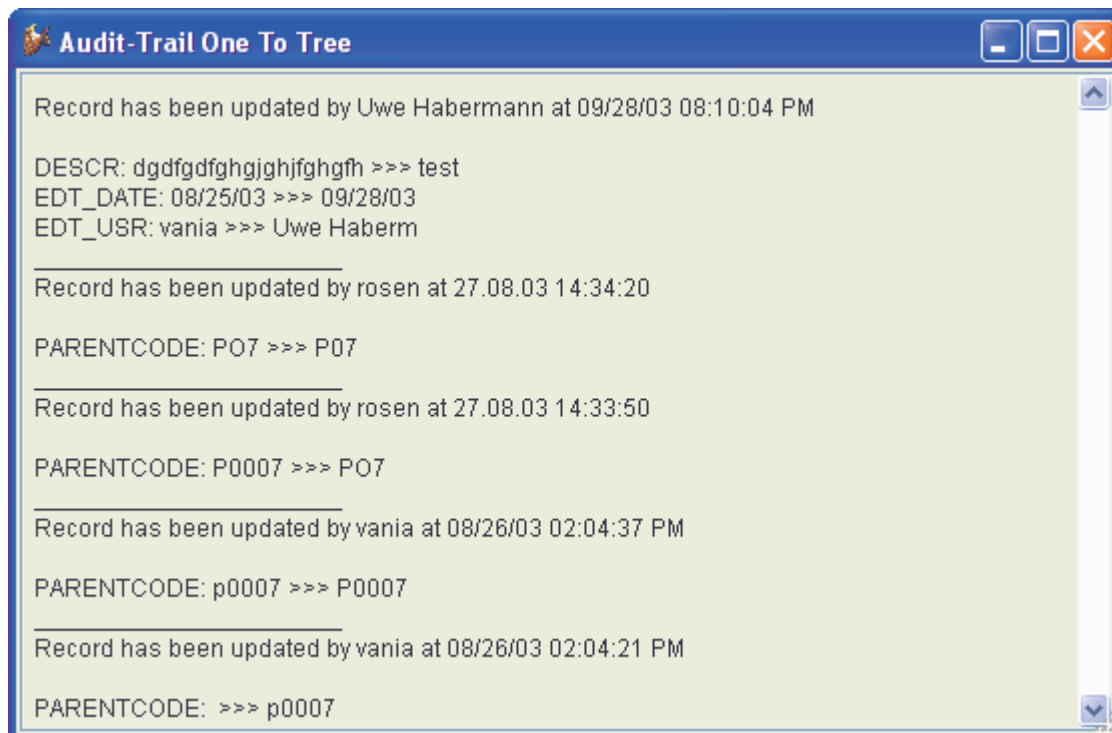
## 16.11. Data manipulation tracking

The Data manipulation tracking (Audit-Trail) logs changes made to the data. VFX uses trigger to determine changes in the data. The trigger functions are placed in all tables.

- `_audit_insert()` tracks the insertion of new data record
- `_audit_update()` tracks all changes
- `_audit_delete()` tracks the deletion of data record

An Audit-Trigger can be linked with a RI-Trigger with a logical *AND*:

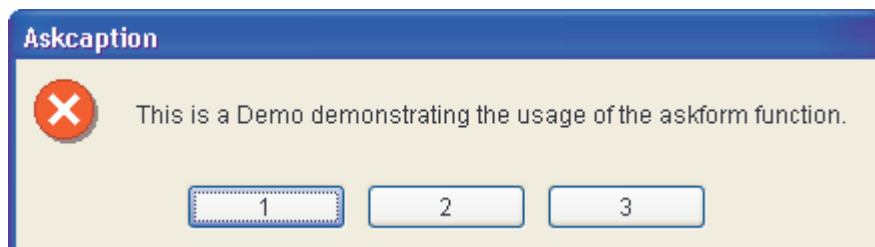
```
__ri_delete_parent() AND _audit_delete()
```



On a button in the standard Toolbar can be viewed the activity log for the current data record.

## 16.12. Askform

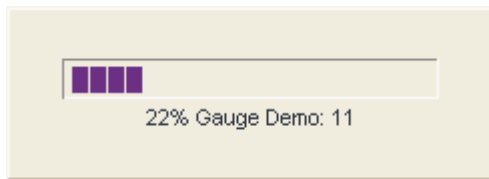
The Ask form corresponds to a Message box, however it has an extended functionality. The Captions of the three (maximum) command buttons can be passed as parameters. In addition it is possible to be specified a timeout for the message box. When the timeout is reached without user action, is returned a value, which corresponds to pressing the default button.



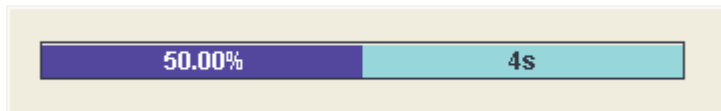
An example for implementation of the function `Askform()` is in the form `Parent.scx` in the demo application VFX90Test.

## 16.13. Progress bar

VFX offers two different ways to indicate the progress of long lasted operations. The simple variant, realized with the form class *cGaugeWin*, displays a bar that indicates the progress.



With the form *Vfxmtr.scx* can be represented a progress bar by indicating the remaining execution time.



Examples for the use of both progress indicators are in the form *Parent.scx* of the demo application VFX90Test.

## 16.14. Date selection

### 16.14.1. cPickDate class

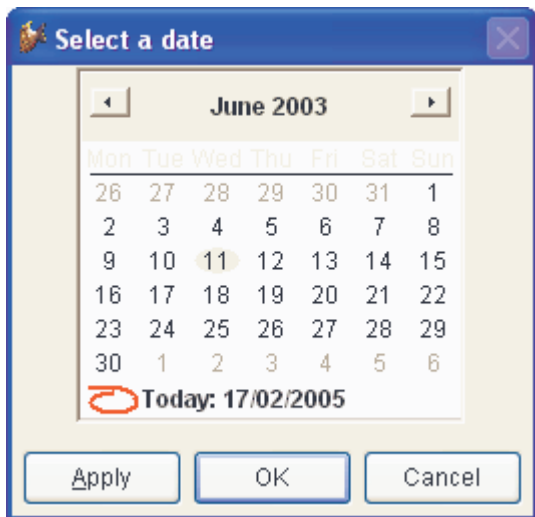
The class *cPickDate* contains a text box for entering a date, as well as a button for calling a calendar control.



For the Textbox for date selection are available the following hotkeys:

+, =	Next day
-, _	Previous day
T, t	Today
F, f	The first day (beginning) of the current month
L, l	The last day of the current month
Y, y	New Year 's day
E, e	End of the year
P, p	Previous month
N, n	Next month

For the calendar is used the Microsoft MonthView ActiveX control. When creating a Setup for deploying the application, this ActiveX control (Mscmct2.ocx) must be included also into the Setup. VFP 9 provides a Merge module for this.



### 16.14.2. cDatetime class

Additionally there is the class *cDatetime* can be used for entering *Datetime* data type values.



This class contains a *cPickDate* control for date input. All functions of the *cPickDate* control, as for example the calendars or the hotkeys, are available.

You can switch to 24-hours format by using SET HOURS TO 24. This setting can be made global for all forms in the function *formsetup()* in *Applfunc.prg*.

The control source of the class *cDatetime* is set in the property *cControlSource*. The control source must be of *Datetime* data type.

### 16.15. Report selection

When for a form must be printed different reports, the class *cRSelection* offers a suitable selection dialog. The available reports are read from tables. It can be made difference between reports, which are visible for all users and reports, which are visible for particular user only.

An example how it can be used is in the form *Reports.scx* in the demo application VFX90Test.

### 16.16. The Microsoft Agents

The agents are nice characters, which loosen up the use of VFX applications.



In VFX80Test the form *Agent.scx* shows a simple example for the possible usage.

## 16.17. The VFX – Resource table

VFX applications use a VFX Resource Table () to store all information about forms the user has opened since the last initialization. This information is used to set the size of the Form, the Grid layout, as well as the current Sort Order.

VFX applications do not use the Visual FoxPro resource table *Foxuser.dbf*, but use the free VFX Resource Table *Vfxres.dbf* instead.

Here are the settings stored on a per user basis in the VFX Resource Table.

Setting	Description	Remark
<i>Position and size of the form.</i>	The user sees the forms always appearing exactly the same way he closed it.	Personal Form Setting.  <b>Note</b> that this also applies to the picklists the user called!
<i>Any layout changes applied to grids.</i>	The user sees the grid of any form exactly the way he left it. This applies to column width, as well as position and does not depend of the type of column. (this works for calculated fields, too).	Personal Grid Settings.  <b>Note</b> that this applies also to the picklists the user called and in the advanced one-to-many form with multiple child grids!
<i>Actual sort order within a Data manipulation form.</i>	The last sort order will always be restored automatically, regardless whether an index tag exists or not. If it does not exist, VFX will create it automatically.	VFX creates unique named Index Files in the temporary Windows folder and deletes them when the user closes the form.  <b>Note</b> that this applies also to the picklists the user called!
<i>Position and state of the form toolbar.</i>	If you have form-specific toolbars, the user sees them always at the same, position and docking state they left.	
<i>Form Toolbar Hide</i>	If the user decides not to work with the form-specific toolbar and close it, the Toolbar will not appear again, until the user reactivates it through the <i>View/ Toolbars</i> Menu Option.	

You can reset this VFX Resource File by clicking the *Clear Resource* command button from the user list form described earlier. This will delete all entries from the VFX Resource File.

## 16.18. The Include Files

Since include files play an important role in VFX Application Development, let's take a look what types of Include Files VFX uses:

Include File	Included by	Language specific	Content/Description
<i>FoxPro.h</i>	VFX.H	No	Standard FoxPro Definitions.
<i>FoxPro_Reporting.h</i>	VFX.H	No	Reporting constants for in VFP.
<i>ReportListeners.h</i>	VFX.H	No	Constants for the VFP ReportListener class.
<i>ReportListeners_Loc.h</i>	VFX.H	Yes	Localized texts for the VFP ReportListener class.
<i>UserDef.h</i>	VFX.H	No	Language independent Constants used in your own Application.
<i>UserMsg.h</i>	VFX.H	Yes	Language Specific Text for messages used in your own Application. Optionally generated by the VFX message editor, when type MESSAGE has been selected.
<i>UserTxt.h</i>	VFX.H	Yes	Language Specific Text for the about dialog, captions and Tooltiptext used in your own Application. Optionally generated by the VFX message editor, when type OTHER has been selected.
<i>VFX.h</i>	VFXMAIN.PRG	No	Sets the <i>_DEBUG_MODE</i> , <i>_LANG_SETUP</i> , <i>_DBCX</i> and other core constant and includes the other Include Files.
<i>Vfxdef.h</i>	VFX.H	Yes	Sets the <i>ID_LANGUAGE</i> constant and defines other not language specific VFX constants.
<i>VfxGlobal.h</i>	VFX.H	Yes	Constants for fields from user list and options. This file is used for backward compatibility with former VFX versions.



<i>Vfxmsg.h</i>	VFX.H	Yes	Language Specific Messages used in VFX User Interface.
<i>Vfxoffice.h</i>	VFX.H	No	Used in the Office Classes Word, Excel and Outlook.
<i>VfxToolbox.h</i>	VFX.H	Yes	Contains constants for the VFP Toolbox.
<i>VfxTxt.h</i>	VFX.H	Yes	Language Specific Captions and Tooltip text used in VFX User Interface.
<i>_FrxCursor.h</i>	VFX.H	Yes	Constants for the VFP ReportListener class.

The application wizard generates most of the constants automatically when you create a new application. You have to make changes in *VFX.h* include file, if you want to change the debugging mode and in *VFXDef.h* include file if you want to change the current language.

In order to prepare Visual FoxPro for a new compiling, you must make a change in the file(s), which include the Include files. The command *clear program* in the command window deletes all compiled programs in main memory. Additionally the files *Program\\*.fxp* and *Menu\\*.fxp* under the application folder, should be deleted. You should include the file *VFX.H* into your forms, if you use constants in your forms

### 16.19. OLE drag & drop

In VFX applications OLE drag & drop is available in three different ways. By default in Grids OLE drag & drop is enabled. The entire content of a grid can be copied, for example to Excel with one mouse-click.

When needed, it is also possible to switch OLE drag & drop for particular controls. By default this feature is switched off and can be set using the VFX – Application Builder into the property *nOLEenableDrag* of the application object.

```
nOLEenableDrag=1  && 0 use form setting (default), 1 enable, 2 disable
```

Further, it is possible to copy the data of all controls of a Page in a Pageframe into another OLE drag & drop capable application. This feature is also switched off by default and can be enabled, if necessary, using the VFX – Application Builder, changing the property *nPageOLEdragdrop* of the Application object.

```
nPageOLEdragdrop=1  && 0 use form setting (default), 1 enable, 2
disable
```

### 16.20. Hooks

VFX offers direct intervene within all important methods through Hooks. As example let's look at the *OnInsert()* method of a form. The *OnInsert()* method is called, if a new data record is to be added. First the method *OnPreInsert()* is called. Only if this method passes back *.T.* as a return value, a data record will be added. After adding the data record the *OnPostInsert()* method is called. Here can be entered data into the new data record, for example with the command *Replace*. If the *OnPostInsert()* method returns *.F.*, a *TableRevert()* is invoked and thereby the new data record will be immediately deleted again.

An elegant way to intervene in the sequence of functions of VFX methods, without having to change the classes, is the use of Hooks.

An Eventhook is inserted in most of VFX methods. If the Eventhooks is activated, in each Eventhook the function Eventhook Handler is called. As parameters to this function will be passed the name of the calling method, a reference on the current object and a reference on the current form. Then, through a Case construction can be implemented individual code. Thereby can be affected the sequence of VFX of functions execution in each practically place.

The concept of the Hooks was extended in VFX 9.5. So far it was possible trough a Hook to implement your own code block within a VFX method. Via the return value of the Hooks you could control whether the still following VFX code in the method should be executed further or not. The return value, which the VFX method supplied thereby, could not be affected and was hard predefined in VFX.

Now with the extended Hooks in VFX 9.5 the return value of the method of the Hook can be additionally controlled.

Hooks are saved in the file *Vfxhook.prg*. Using Hooks can be enabled in VFX – Application Builder by setting the property *nEnableHook = 1*. *NEnableHook* is a property of the application object.

In this example the *FontColor* of all controls, that are disabled, is set to Black.

```
function EventHookHandler(tcEvent, toObject, toForm)
  local lContinue
  lContinue = .T.
  DO CASE
  CASE UPPER(tcEvent)=="INIT"
    IF PEMSTATUS(toObject,"disabledforecolor",5)
      toObject.disabledforecolor=;
      eval(left(rgbscheme(1,2),at(", ",rgbscheme(1,2),3)-1)+")")
    IF PEMSTATUS(toObject,"disabledbackcolor",5)
      toObject.disabledbackcolor=;
      eval("rgb("+substr(rgbscheme(1,2),;
      at(", ",rgbscheme(1,2),3)+1))
    ENDIF
  ENDIF
  ENDCASE
  return lContinue
endfunc
```

## 16.21. Business Graph

Statistical reports are an endless digits list, quite hard to read and analyze. The best way to demonstrate calculated results is to create colorful graphical presentation. The new class *cBusinessGraph* gives to the VFX developer the great chance better to represent and print application data as a graph just in few minutes.

To draw the graph, *MSChart* ActiveX control is used and series data is read from a table or cursor. The data to be shown can be obtained from any cursor. Every column in the cursor is considered to be one series in the graph. It is supposed one of fields in the cursor to hold label texts. If you do not specify field which contains labels, all fields in the cursor are used as series data. Fields, containing series data, should be of numerical data type. Additionally you have to provide series titles, which are displayed as legend text.

### Properties

*cAliasName* – Alias of the cursor, contained data series.

*cGraphTitle* – Title for the graph.

*cLabelField* – Name of the field which will be used to set data labels in the graph.

*cLegendTitles* – A comma-separated list, containing series titles.

*lShowLegend* – When the value of this property is set to *.T.*, a legend will be displayed in the graph.

*nGraphType* –Graph type:

- 0 - 3D BAR
- 1 - 2D BAR
- 2 - 3D LINE
- 3 - 2D LINE
- 4 - 3D AREA
- 5 - 2D AREA
- 6 - 3D STEP
- 7 - 2D STEP
- 8 - 3D COMBINATION
- 14 - 2D PIE
- 16 - 2D XY

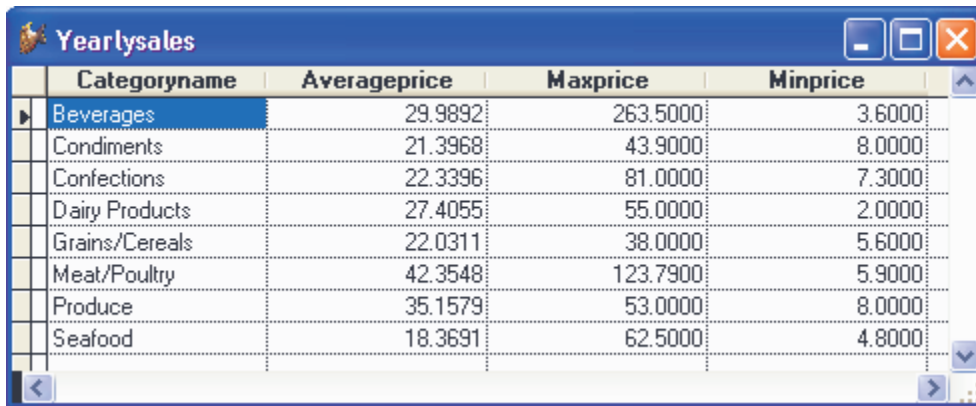
## Methods

*DrawGraph* – Draws the graph using provided data and control's settings. All series data, labels and titles are set accordingly control settings. All data points and legend are actualized.

*OnPrint* – Prints a copy of the graph using the hardcopy report *Hardcopy.frx*.

### 16.21.1. Example

Let's assume a part of the application creates the cursor shown below. It is required to use it for creating a business graph.



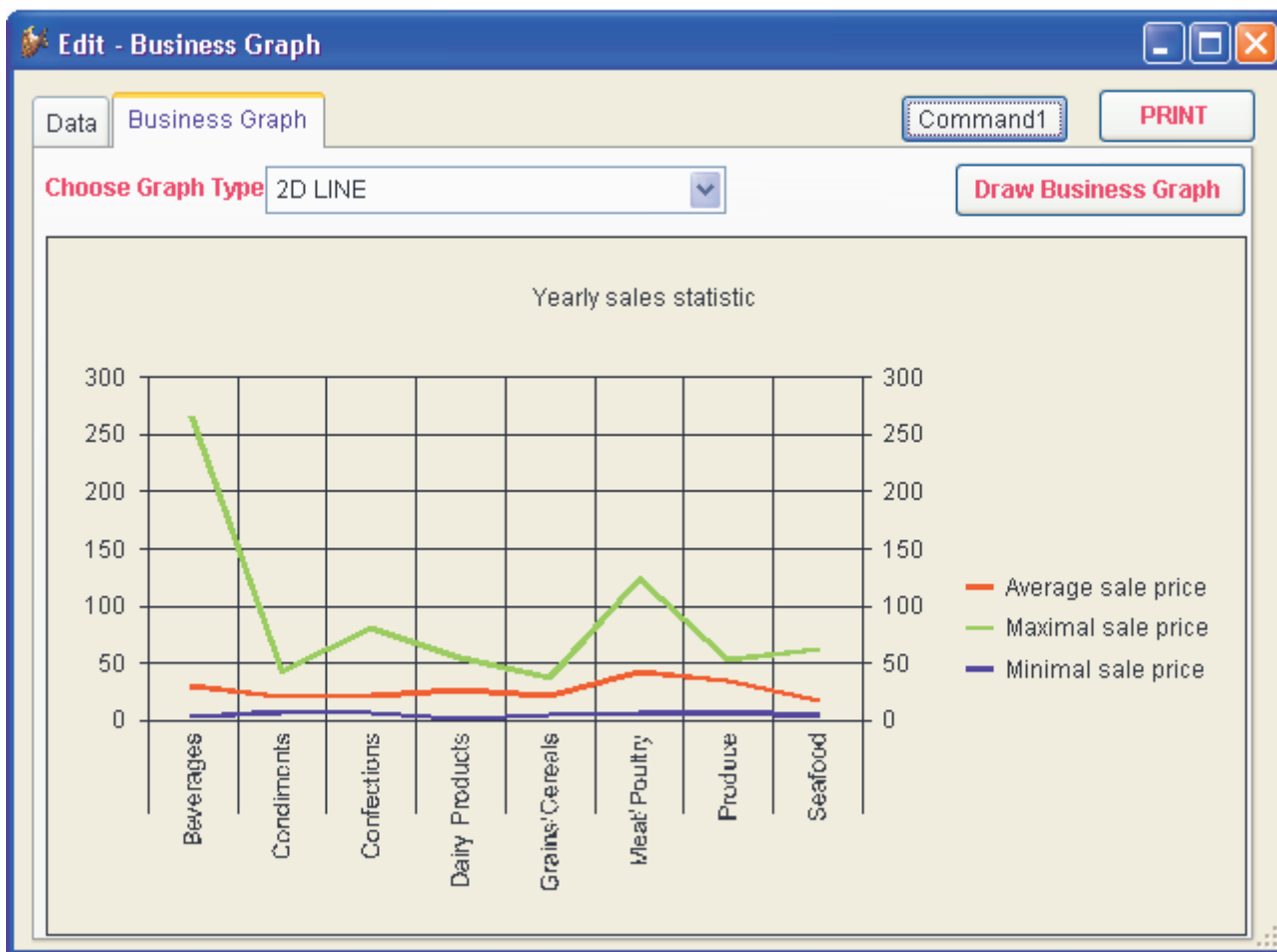
	Categoryname	Averageprice	Maxprice	Minprice
▶	Beverages	29.9892	263.5000	3.6000
	Condiments	21.3968	43.9000	8.0000
	Confections	22.3396	81.0000	7.3000
	Dairy Products	27.4055	55.0000	2.0000
	Grains/Cereals	22.0311	38.0000	5.6000
	Meat/Poultry	42.3548	123.7900	5.9000
	Produce	35.1579	53.0000	8.0000
	Seafood	18.3691	62.5000	4.8000

The class *cBusinessGraph* can be placed in any form. For the example data shown above should be set the following properties of the object:

```
.cAliasName = "YearlySales"
.cGraphTitle = "Yearly sales statistic"
.cLabelField = "CategoryName"
.cLegendTitles = "Average sale price, Maximal sale price, Minimal sale price"
```

In the property *cLabelField* should be entered the name of the fields containing text description. In property *cLegendTitles* are listed texts, which need to be used as a legend for data series. The sequence of texts must be same as fields sequence in the cursor.

Now, calling the *DrawGraph* method the following graph is drawn.



## 16.22. Toolbars

### 16.22.1. Use the Main Toolbar you like

It's a good practice to create a new class library file for your application (or company) specific needs. We prepared this for you and called it *Appl.vcx*. To make your live as easy as possible, we already created two classes within this *Appl.vcx*:

*cAppToolBar* and *cAppNavBar*.

The first is the standard toolbar and the second is one which you may use if you do not want to have the navigation and other buttons on the form.

***CAppToolBar*:**



*CAppToolBar* will be used, if you are working with the VFX forms with the navigation and other buttons on the form.

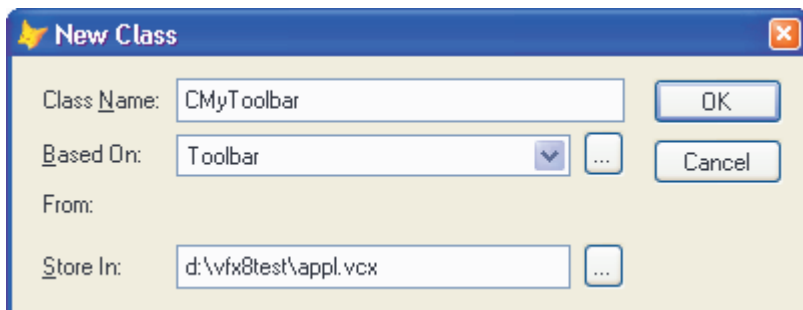
**CAppNavBar:**

*CAppNavBar* will be used, if you are working with the VFX forms without the navigation and other buttons on the form.

To switch from one main toolbar to another is needed to change the values of the property *cMainToolBar* in the application class *CFoxAppl* in *Appl.vcx*:

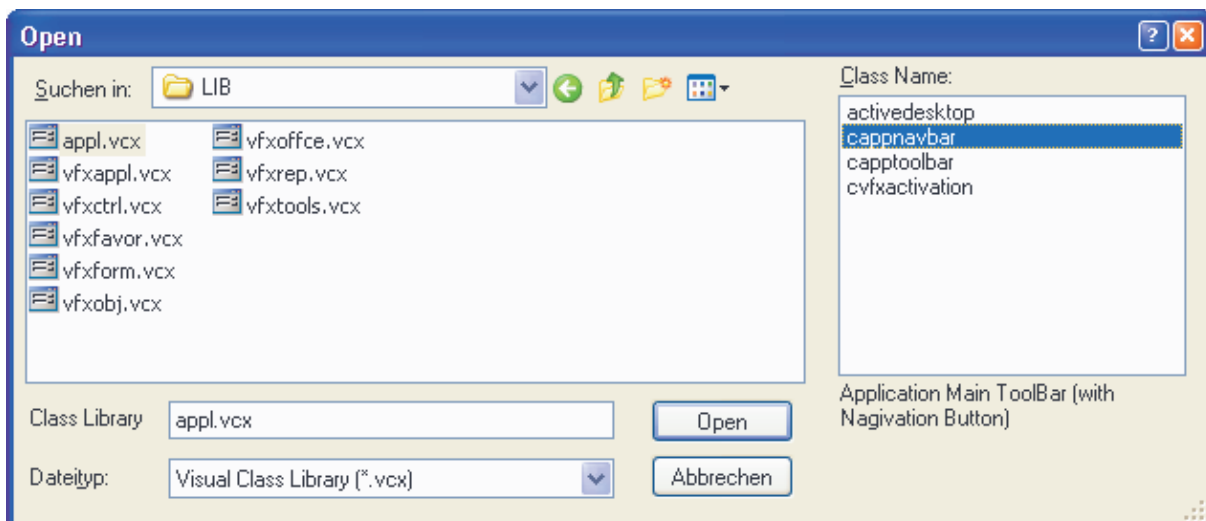
You could use the *CAppToolBar* or *CAppNavBar* toolbar class, to develop most of your office compatible applications. But of course, you can create also other toolbars. All you have to do is create a new class which inherits the *CToolBar* class or even the *CAppToolBar* or *CAppNavBar* class.

Select *New* while in the project manager on the class tab or on any class library object, you will see the following Dialog:



**Class Name:** Enter the name of the new class. Assume we name it *CMyToolBar*.

**Based On:** Click the ellipsis button, and in the following Open Dialog, select the Class *cAppToolBar* (or *cAppNavBar*) from the VFX Class Library *Appl.vcx*.



**From:** The reference to the VFX Library File called *Appl.vcx* will automatically be displayed

**Store In:** If your application-specific library file does not yet exist, just type in the full path and name, otherwise pick it using the ellipsis button (GetFile Dialog).

Now, you have to modify your toolbar class. Do this using the Visual Class Designer.

## Add a Custom button

Visual Extend offers predefined buttons for the easy creation of toolbars. Drag the class *cToolbarbutton* from the VFX class *Vfxctrl.vcx* onto your toolbar and adapt the following properties and methods of the newly added command button:

**Click Event:** Add the command which you want to execute, whenever this command button will be clicked. Assume we want to run a *Customer* form. In this case we place the code

```
goProgram.RunForm( "CUSTOMER" )
```

into the *Click()* event.

**Picture:** Select a *BMP* or *ICO* file to be used on your toolbar command button.

Make sure to place the desired code into the *Refresh()* Event of every toolbar icon (or the toolbar directly) for proper refresh of your icons. If you call a modal form, VFX will automatically disable all toolbar icons, but it's up to you to reactivate the toolbar icons again. This could happen with the following code in the refresh event:

```
this.enabled = this.parent.cmdopen.enabled
```

The above code would automatically synchronize the toolbar icon with the state of the file *Open* toolbar icon, which will always be synchronized with the actual state of the application.

## Add a Separator

Start with a separator, which separates the last standard Icon from the first application-specific on the main toolbar.



Use this icon from the Visual FoxPro Form Control Toolbar and drop it on your toolbar as needed.

### 16.22.2. Adding a toolbar to a Form

The possibility to add a toolbar for a form is very user-friendly. The toolbar must be based on *cToolbar* class and saved into *Appl.vcx* class library.

The name of the toolbar is entered in a property *cToolbarClass* of the form. VFX invokes the toolbar automatically, if the form is active and it hides again when another form becomes active. Of course the state and the position of the toolbar are saved on a per user basis.

Usually, in the *Click()* method of the toolbar buttons is called a method of the active form. For example:

```
_screen.activeform.myMethod()
```

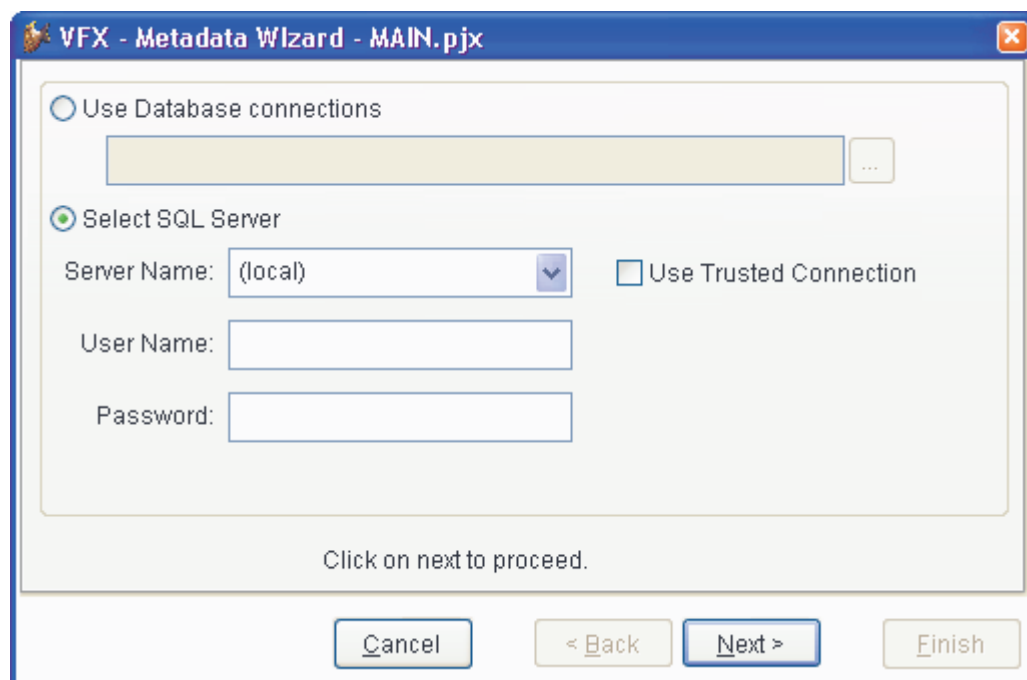
For example, to open a Child form using a button in the toolbar, add into the toolbar a button, based on the class *cToolbarClass*. In the *Click()* event of the button write this code:

```
_screen.activeform.onmore(1)
```

This is all. Since VFX guarantees the fact that the toolbar is visible only if the correspondent form is active, we can be sure that *\_screen.activeform* exists. The *OnMore()* method is called within this form and receives as parameter a passed value *1*. Thus the form is notified, that the first array element of the *OnMore()* method will be used, without activating the *OnMore()* Dialog

### 16.23. CWizard-Class

The class *cWizard* makes it possible to create assistants. The user will be led step by step through the process. A good example for the use of the class *cWizard* is contained in the VFX Wizards itself. The VFX – Metadata Wizard is based on the class *cWizard*.



### 16.24. CDownload class

This class allows you to download files from the Internet. If necessary, the downloaded files can be run and further actions can be performed. In particular in this way can be performed application installation from the Internet;

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

Macros are character strings that contain consecutive commands, defined in the macro language. User-defined macros can be developed. An example can be found in the field *Install\_GS* in the table *Vfxsys.dbf*. With this macro the Ghostscript application is downloaded from the Internet and installed.

The class uses the Internet site URL, stored into the *goProgram.cConnectionCheckURL* property, to check if there is an existing Internet connection. If necessary, a Dial-Up connection will be automatically established. When there are no dial-up network entries, a new entry will be created. The connection information can be predefined by the developer. If necessary, in a dialog, user can change the phone number to be dialed, user name and password to access the network resources.

#### Properties

*LastErrorNo* – This property contains the number of the last error (in case some were encountered). It can be used to check what the reason for the latest raised error is

*LastErrorText* – When an error occurs, the description text for the error is stored in this property

#### Methods

*ExecMacro* (*vcMacro*, *InNoRun*)

*vcMacro* – Macro-language script to be executed

*InNoRun* – When this property is set to *.T.* the downloaded file will not be run

### 16.24.1. Macro language commands

„D:“ *URL*

The downloaded file will be searched at this internet address. This command automatically runs the downloaded file, after successful download, if *InNoRun* parameter is set to *.F.*

„C:“ *nTimeout; lPartial; lTopLevelForm; lResultOnError; SearchedString*

Waits while the window with caption *SearchedString* appears

*nTimeout* – Timeout in seconds – when the expected form does not appear within this timeframe, a timeout error will be generated

*lPartial* – when the value of this parameter is *.T.*, it is enough to find searched string as a part of a window's caption. When the value of the parameter is *.F.*, this specifies that the window's caption must match exact searched string.

*lTopLevelForm* – When the value of this parameter is *.T.*, the string is searched only in the caption of the top-level forms.

*lResultOnError* – With this parameter is controlled the behavior of the script, in case the form is not found within the given timeout period. If the existence of that form 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 script can continue regardless of the fact that window did not appeared, the passed value should be *.T.*

*SearchedString* –String that will be searched within form's captions.

„W:“ *nTimeout; lPartial; lTopLevelForm; lResultByError; SearchedString*

Waits until the window than contains searched string in its title is closed.

*nTimeout*– Timeout in seconds. When expected form does not close within it, a timeout error is generated

*lPartial* – when the value of this parameter is *.T.*, it is enough to find searched string as a part of a window's caption. When the value of the parameter is *.F.*, this sets that the window's caption must match exact searched string.

*lTopLevelForm* – When the value of this parameter is *.T.*, the string is searched only in the caption of the top-level forms.

*lResultOnError* – With this parameter is controlled the behavior of the script, in case the form is not found within the given timeout period. If the existence of that form 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 script can continue regardless of the fact that window did not appeared, the passed value will be *.T.*

*SearchedString* –String that will be searched in form's captions.

„X:“

Closes the top level window. In advance, using “C:” command, must be ensured that the desired window is at top level.

„K:“ *nKeyCode1; nKeyCode2; ...*

Places the listed key codes into the Windows keyboard buffer

„U:“ *URL*



From this internet address will be downloaded a file. The downloaded file will not be run, regardless of the *lnNoRun* parameter's value

### 16.24.2. Example

Explanations for the Ghostscript Installation macro:

*D: ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/AFPL/g811/g811w32.exe*

Downloads the file *gs811w32.exe* from the Internet and runs it afterwards.

*C: 30; .F.; .F.; .F.; WinZip Self-Extractor - gs811w32.exe*

Waits until window with caption "WinZip Self-Extractor - *gs811w32.exe*" appears.

*K: 43*

Sends the key code "Enter" to the active window. This will start the files extraction.

*C: 60; .F.; .F.; .F.; AFPL Ghostscript Setup*

Waits until a window with caption "AFPL Ghostscript Setup" appears.

*K: 43*

Sends the key code "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 opened. This window shows the installation process and the script execution 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 message that the Setup were finished successful.

*X:*

Closes the last window.

With this the Ghostscript installation is completed.

### 16.25. CCreatePDF class

This class is creates report files in PDF format. It receives as parameters the alias name of the used cursor, name of the resultant PDF file that will be created, name of report file to be used for the report creation as well as an optional expression that will be used as FOR clause to filter out report data.

To be able to create a file in PDF format it is necessary GhostScript and a Postscript Printer driver to be installed on the computer. The class checks if GhostScript is already installed. Otherwise, GhostScript will be automatically downloaded from the Internet and installed. The *cDownload* class is used to perform download from the Internet. In the memo field *VFXSys.Install\_GS* is a placed a script, which will be used Ghostscript downloading and installation. Refer to the description of the class *cDownload* for more details.

If there is no Postscript printer driver installed on the computer, the class installs the standard Windows printer driver according to the value stored in the *goProgram.PSPrinterToInstall* property. Usually, this installation does not need user interaction

The report will be passed to the Postscript driver and the result is saved into a file. Then the Ghostscript transforms this postscript file into a PDF-file.

#### Properties

*LastErrorNo* – This property contains the number of the last error (in case some were encountered). It can be used to check what the reason for the latest raised error is

*LastErrorTest* – When an error occurs, the description text for the error is stored in this property

## Methods

*Create\_PDF(tcAlias, tcRezFile, tcFRXName, tcFor)*

*tcAlias* – Alias name of the report data, to be exported in PDF file.

*tcRezFile* – Full path and name of the created PDF file.

*tcFRXName* – Report name, which will be used for creating PDF file

*tcFor* – Conditional expression that will be used in FOR clause to filter out exported data.

Method returns the value *.T.*, if the file is successfully created. When the PDF file cannot be created the value *.F.* will be returned. In this case the number and description text of encountered error are stored in *LastErrorNo* a *LastErrorText* properties of the class.

### 16.26. CEmail class

This class gives the developer the ability to send e-mails just by passing a few parameters to the method *Send\_Email\_Report*. In addition to this to the e-mails can be attached additional report exported files in PDF format.

## Properties

*LastErrorNo* – This property contains the number of the last error (in case some were encountered). It can be used to check what the reason for the latest raised error is

*LastErrorText* – When an error occurs, the description text for the error is stored in this property

*oEmail\_Attachment* – This property will be used internally. It contains a collection of added attachments.

## Methods

*AddAttachment(tsAlias, tcFileName, tcReport, tcFor)*

Fills the E-Mail object information about e-mail attachment files that will be created and sent with the created e-mail. The information for all PDF attachments that have to be prepared is stored in the *oEmail\_Attachment* property. When you pass data alias of an opened table or view and report name the class automatically creates PDF file using this report and alias. An additional expressions can be specified, that will be used to filter out reported data. If no data alias is specified and no table is open in the current work area, the class considers that a file prepared in advance will be used for the attachment. In this case the file must exist at the time when the method *Send\_Email\_Report* is called.

*tcAlias* – Alias name that will be used for report and for PDF export.

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

*tcFRXName* – Name of the report that will be used to generate PDF file.

*tcFor* – Expression, used as FOR clause, to filter exported to PDF records.

*Send\_Email\_Report(tcEmail, tcSubject, tcText)*

Sends an e-mail. When the sent e-mail will have attached files, they must be defined in advance by calling the method *AddAttachment*.

*tcEmail* – E-Mail recipient address

*tcSubject* – Subject of the e-mail

*tcText* – Body text for the e-mail

### *ClearAttachment*

Clears all currently assigned attachments.

The *AddAttachment* method can be called as many times as needed. The alias name of your data table or view, name of the created file, formatting report name and optional expression to be used as FOR clause should be passed. Then the method *Send\_Email\_Reports* method must be called. All PDF files will be created and attached to the created e-mail. Also the files, which you have previously created and passed to the *AddAttachment* method without report name and alias name parameters will be attached.

## **16.27. CArchive class**

This class is designated to back-up and restore user's data. The files are archived in ZIP format. The name of the created archive file is constructed using the name of the data folder and the current date and time. For example the data folder is named "Data" and the archive will be created on 4<sup>th</sup> of November, 2004 the name of the created archive will be Data20041104.ZIP.

### *Properties*

*OverrideFile* – With this property is specified what action will be performed if a file with same name already exists

0 – Cancel operation, if a file with the same name already exists.

1 – When a data backup is made, new files will be added to the archive and existed in the archive files will be updated. When a restore operation is performed, existing files will not be overwritten.

2 – When a data backup is made, the entire archive file will be overwritten. When a restore operation is performed, existing files will be overwritten with the files extracted from archive

*OperationSuccessfully* – Contains the result from the last operation

.T. – If operation have completed successfully,

.F. - If operation have not completed successfully

### *Methoden*

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

*lcFileLocation* – Full path to the folder which contents will be backed-up.

*lcMask* – File mask for archived files. Example "\*.DBF;\*.FPT;\*.CDX"

*lcArchFilePathName* – Archive name (including full path) that will be created

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

*ZipProgress* (*tcCurrentOperatedFile*, *nState*, *nAllFilesSize*, *nZIPedFilesSize*, *nArchiveCurrentSize*)

Callback-function for the *CreateZipArchive* function (in *VFX.fl*)

*tcCurrentOperatedFile* – The name of file being currently added to the archive

*nState* – Current operation:

- 1 – File exist
- 2 – Adding the file to archive started
- 3 – The file was successfully added file to archive
- 4 – Cannot add the file to archive
- 5 – Archiving operation completed successfully

- 6 – Archiving operation did not completely successfully
- 7 – No files to be archived

*nAllFilesSize* – Total size of all files which will be archived

*nZIPedFilesSize* – Total size of files added to archive so far

*nArchiveCurrentSize* – The current size of archive

Return value: 0 – Cancels archiving operation. 1 – Continue adding files to existing archive and overwrite existing files in archive. 2 – Overwrite existing archive file

*ExtractFromArchive(lcArchFileForExtract, lcPathForExtract)*

*lcArchFileForExtract* – Full Zip file path name that will be extracted

*lcPathForExtract* – Destination folder where the files from the archive will be extracted

*UnZipProgress(tcCurrentOperatedFile, nState, nArchiveFilesSize, nUnZIPedFilesSize)*

Callback-Function for *ExtractZipArchive* function (in *VFX.fl*).

*tcCurrentOperatedFile* – Name of currently extracted file from archive.

*nState* – Current operation

- 1 – File already exist
- 2 – File extraction started
- 3 – File extracting finished
- 4 – File cannot be extracted;
- 5 – Extracting form archive completed successfully;
- 6 – Extracting form archive did not completed successfully.

*nArchiveFilesSize* –Size of archive

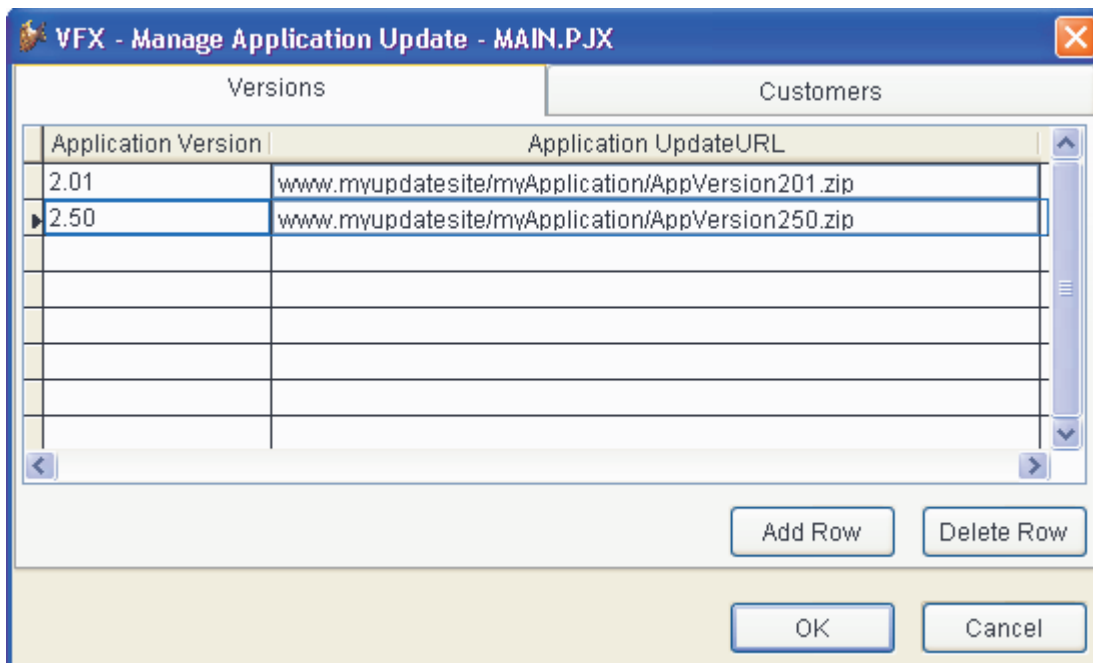
*nUnZIPedFilesSize* – Total size of part of archive already extracting

Return value: 0 – Cancel entire operation. 1 – Do not extract current file. 2 – Overwrite existing file.

## **16.28. Application update**

The possibilities for the application update at the customer side over the Internet were extended. The developer can define a list of the customers who are entitled to download updated application versions and to install it. This customer list is stored in a coded file on the web server and is downloaded before the real actualization into the customer's PC and is checked. Together with the customer list a version list is downloaded. The purpose of this version list is to handle application update dependent on the program version currently installed at the customer side.

Both lists can be edited from VFX 9.5 menu, under *Activation, Manage Application Updates* option.



In the column *Application Version* is entered the application version number. In the column *Application Update URL* is the link to the file that need to be downloaded, in order to update that particular application version.

At customer side, the actualization process executes in two steps. In the first step is executed a download script that downloads the customer list and the version list. The name of the file containing the customer list is *UpdateCustomer.vfx*. The version list file is named *UpdateVersion.vfx*. The download script for these both files is in the field *UpdateApp* in the table *Vfxsys.dbf*.

In the second step, after both files are downloaded and decrypted, it is checked whether the user is entitled to update application. The download link for the updated application version according his current application version is obtained from the file *UpdateVersion.vfx*.

### 16.29. VFP Toolbox for developers

VFX supports the VFP Toolbox for developers. When a project is open, VFX libraries, used in this project, are added to the toolbox.

### 16.30. Further development with VFP

The entire VFX 8.0 project is based on normal VFP source code. At any time the created application can be developed further with VFP, even when on the developer's computer VFX is not installed.

### 16.31. Error handling

In VFX 9.5 is implemented an enhanced handling of run-time errors. Now the customer can send run-time log to the developer by e-mail. The customer is informed about the contents of the error report. Sending the error report as an e-mail to the developer is the fastest way to localize and remove problems in the application. The E-mail address of the developer is assigned to the property *goProgram.csupportemail*. The value of this property can be changed with the VFX - Application Builder.

### 16.32. Troubleshooting Guide

**Error "cap\_application\_title not found":** The include files could not be found. Make sure that the current directory is the directory of the project you are working with! **Tip:** Try this in the command window: CD ?.

Shut down, restart, set directory to your application directory, open project, select rebuild all and run. If you included your own include files, make sure to recompile your source program before you rebuild your project!

---

**Hint:** Select the option "Properties" (last option in the right click menu when editing a PRG) and then in the dialog "Compile before saving". This guarantees that you have always recompiled PRG's as described later on in this guide.

---

**Changes in include files don't go through:** Make a change in the file which includes the Include file, quit Visual FoxPro, delete the compiled FXP's, restart, select the correct project directory, rebuild your project. Tip: Try also the CLEAR PROGRAM command which clears all compiled programs from memory. If you make changes in an include file which affect a form, make sure to open the form and save it, otherwise the changes in the include file may not affect the form. If your include file changes do still not go through make sure to delete all FXP files and rebuild all again.

**Important! Working folder:** Make sure that the current Directory is the Directory of the Project you are working with! Try this: CD ?

---

**NOTE:** You should better use the VFX Application Manager to open and switch projects.

---

**Created forms are based on the library in another directory rather than on the (expected) library of my application:** This is only a problem if you work simultaneously on different projects or on different versions of the same project. To solve incorrect links, temporarily rename the directory of your project and open all of your forms. Open all Forms and Classes and locate the correct library from your application when necessary and select save.

**Incremental Search and other VFX Power Grid Features do not work:** Make sure you called the Grid Builder as described in the User Manual.

**Incremental Search says Feature not available:** You must set the buffering mode to 3, otherwise no IDX files can be generated. You probably have it set to 5!

**OneToMany Form does not refresh the childtables when I move the record in the parent:** Make sure that you set the OneToMany Relation correctly in the form's Data Environment! All you have to do is to drag the relation from the parent primary key to the child's foreign key. Don't set any other properties. **Tip:** Make sure that you did **NOT set the OneToMany property** of your OneToMany relation in the form's Data Environment to true. Setting this property to true mimics the SET SKIP TO command and therefore is NOT at all what we want...

**My picklist does not work with numeric fields:** Make sure to set the *CPickfield* class property *cReturnExpr* to *TRANSFORM(Field)* rather than *Field*. The rest will work as with character fields.

**Changes in PRG's do not go through:** Issue CLEAR PROGRAM and retry. Or, better yet, set the edit option to "Compile before saving"!

**Project Rebuild failed:** If you have some library elements in another language than the desired or have general concerns that the project did not recompile correctly, start the rebuild all option from the VFX application manager described above.

---

**NOTE:** The include files and the menu files must be checked manually!!! Don't expect a German version if the include files are in English.

---

### 16.33. Further enhancements for developers

- The VFX Builder is called when a Pageframe is selected.
- Support of views and Cursoradapter classes by displaying the Audit Trails
- Support of all VFX classes in Builders
- As a separator in all VFX properties, comma or semicolon can be used alternatively

- Additional new fields *cins\_time* and *cedt\_time* to keep the last processing time
- If *readonly* property is set to *.T.*, *tabstop* is set to *.F.* automatically
- *cpickfield* builder: the properties *cfieldlist* and *cfieldtitle* can be keyed up directly in a textbox in the Builder
- *Vfx* system tables can be optionally included in a SQL Sever database.
- New builder for Audit-Trail triggers generation.

## 17. Remote support

In VFX 9.5 is integrated the viewer part of the remote control program Radmin. End users can start the remote control by choosing Help, Remote control from the main menu. The remote control is carried out through the Internet.

### 17.1. How does the Remote control work?

#### How does the Remote control work?

A connection using the IP protocol, using port 4899 by default, is established between the customer's PC and the supporter's PC. VFX uses exclusive IP connections, established over the Internet. For IP connections within a LANs the remote control program Radmin can be easily configured by hand.

In order to use remote control, the customer's computer must be connected to Internet. The IP address must be accessible through Internet. The port 4899, which is used by Radmin should not be blocked by a firewall.

The advantage of Radmin is that no installation on the customer's PC is required. Only two files are used on the customer computer for running Radmin: *R\_Server.exe* and *Adm.dll*. This file *R\_Server.exe* can be placed in any folder.

To initiate the remote control, the customer's computer has to establish an internet connection. Usually, when dial-up connection is used, a dynamic IP address is assigned to the customer's computer. Hence, this IP address is not known to the supporter. For this, the VFX application registers the actual IP address of the customer's PC as a subdomain of DynDNS. Thus the Supporter can connect to the customer's PC via the Internet using the subdomain name.

### 17.2. Prerequisites

The developer should prepare the VFX application for remote control. Therefore, for the support of the developed application, must be registered a subdomain in DynDNS. The registration is free of charge.

The encrypted registration information is stored in the VFX application in the table *Vfxsys.dbf* in the memo field *dyndns*, so that the registration information on the customer's PC is not readable. The encryption is made with the password *cConfigPassword*. This password should be assigned to the property *cConfigPassword* of the application object – *cFoxAppl* class in *Appl.vcx*.

The edition of the DynDNS registration information is available through the menu *Data, Manage Vfxsys.dbf* in the VFX menu.

1. The content of the field *dyndns* is composed by four lines:
2. DynDNS User name
3. DynDNS Password
4. Subdomain-Name
5. Password for accessing Radmin on customer's PC

### 17.3. Subdomain registration

Thanks to the organization Dynamic DNS Network Services it is possible to be registered sub domains free of charge. Every developer should register a dynamic DNA at <http://www.dyndns.org/services/dyndns>.

A user name, a password and an E-mail address are necessary for creating an account with DynDNS. The Subdomain name can be chosen on your wish. It can be selected from among large number of Domain names.

Example: *myCompany.dnsalias.com*

In this example *myCompany* is the selected Subdomain-Name. *Dnsalias.com* is the existing DynDNS Domain Name.



At the Subdomain registration, an account with user's name and password must be created. The account can be configured with the login information. The login data can also be set at the URL above.

The IP address associated to the Domain name can be changed arbitrarily often and with different methods. Detailed descriptions to all methods can be found on the website [www.dyndns.org](http://www.dyndns.org). The VFX application calls an URL to register the actual IP address of the customer's PC. The URL has the following format:

```
http://username:password@members.dyndns.org/nic/update?hostname=myCompany.dnsalias.com
```

When this URL is entered in the Internet Explorer, an HTML page with the word *Good* is loaded.

Because the Internet browser sends the own IP address to the server, it is not required to pass the IP address separately. The Internet server must know to which address the answer must send back. Dyndns uses automatically this IP address for the registration of the Subdomain.

#### **17.4. The remote support application Radmin**

The remote control application Radmin can be downloaded by the website [www.radmin.com](http://www.radmin.com). The documentation is available on this website also.

Radmin is a shareware application and can be registered at a reasonable price. At the moment the full version which is necessary for the Supporter workplace costs 35 US\$. A license for a customer costs 15 US\$. Customer licenses can be acquired only in bundles from 50 licenses.

Like VFX, Radmin is also protected using an activation key.

If the customer wants to use the remote support, Radmin can be used immediately. If after the 30-day trial period, an attempt to establish a connection is made, the Supporter is called on to transfer a registration key to the customer computer.

The registration key can be transferred by the Supporter to the customer's computer through the Radmin connection.

Beside the remote control, Radmin offers the chance for file transfer.

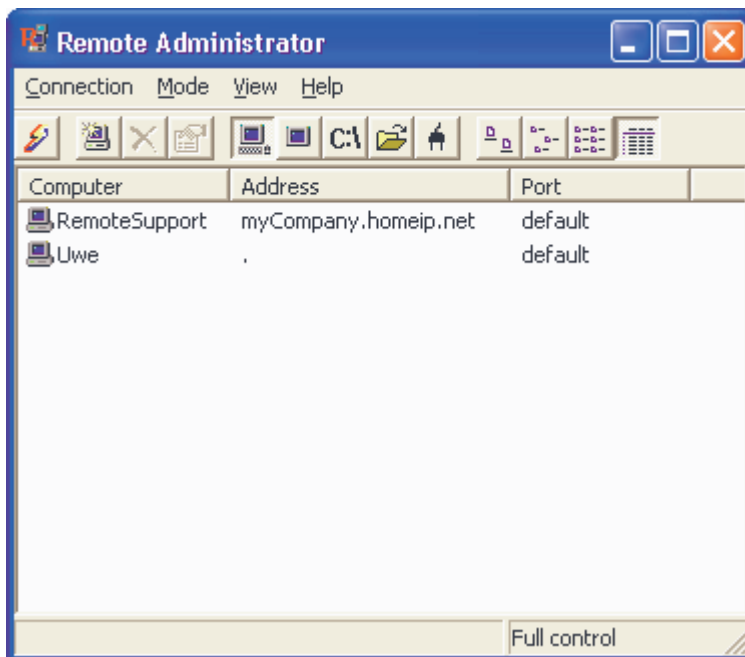
#### **17.5. The remote control from the point Supporters**

The customer should begin the remote control connection only after consultation with the Supporter. The remote support application allows unlimited access to the customer's PC and thus is a considerable security risk for the customers!

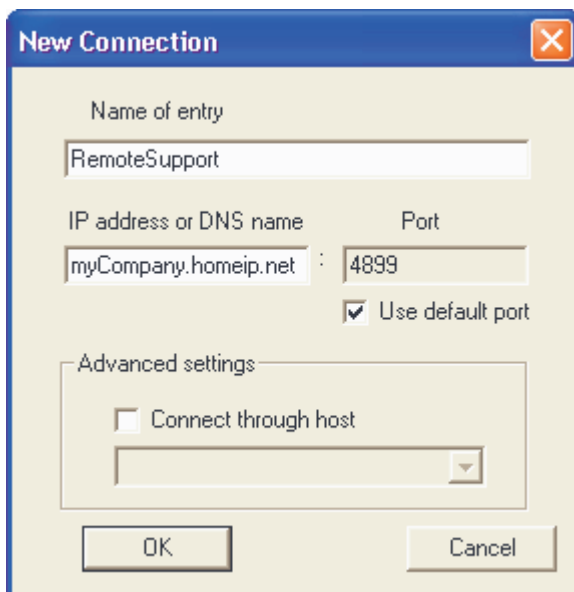
Hence, the access to the customer's PC should be protected by a password.

It is not very likely that a waiting Radmin server is found quickly in a dynamically assigned IP address in the Internet by hackers. In addition, the access to the customer's PC is protected by a password for accessing the customer's PC which must be given while establishing the connecting by the Supporter.

In the Remote administrator Viewer is created an entry for the application Support.



In the properties of the Remote entry the Subdomain name is entered in the field IP address.



Now the customer's PC can be found in the Internet through the Subdomain name. The Supporter needs to create only one entry for the remote support of all customer computers.

After establishing successful connection, the customer's PC can be operated in the Radmin-Viewer's window exactly like the own PC.

## 18. Documentation

Beside the user manual there is lots of on-line documentation for VFX. To this belongs in particular the technical reference, which is available as Windows help file. In it for each class library, for each class are described each method and each property. In a Tutorial are described the solutions with VFX based on typical developers' questions. Directly from the technical reference can be started videos (Avi files). There are 10 videos with approx. 45 minutes duration altogether. In the videos is described and shown the development of forms for Fileserver- and Client-/Server-databases. A great assistance in the training for the VFX beginner.

### 18.1. Support

Support for VFX can be found in the dFPUG forum (<http://forum.dfpug.de>). There are both a German and an English section for VFX. Alternatively, these sections can also be read and edited as newsgroup (<news://news.dfpug.de>).

Further information to the product can be found in the Internet on the Visual Extend website (<http://www.visualextend.de>). From here it is also possible to download the demo application, the entire documentation and the current full VFX version. An extensive collection of further VFX documents is available in the document portal dFPUG (<http://portal.dfpug.de>). Current information can be received from the free dFPUG eNewsletter, in the VFX section (<http://newsletter.dfpug.de>)

## 19. Summary

As we saw VFX provides a complete development environment, which does not leave any wishes open. All important settings of VFX classes, in particular of the form classes, can be completed with reentrant builders. Practically, all properties and functions described in this article can be set without programming, only by using the builder.

Even so, it is practically to be able through Hooks to intervene in each place in the program workflow.

As far as VFX source code is supplied and it is developed with VFP itself, the developer has unlimited freedom to make own extensions or adaptations that are needed.

The performance of VFX applications is so good, as it can be for pure VFP applications. The nesting level is not deep. Most classes have only 1 to 2, maximally however 4 nesting levels behind them. In order to accelerate further loading of extensive forms, can be used Delayed Instantiation. This is also supported by VFX with easy handled functions.

The applications created with VFX provides to the developer with a very professional look and an Office-compatible user interface.

With all this, VFX offers an unrivalled cost/performance ratio. It offers to every developer a rich source of ideas and numerous complete problem solutions.

### **19.1. Your feedback is important for us!**

Your feedback is highly appreciated! Mail us your feedback to [visualextend@dfpug.de](mailto:visualextend@dfpug.de) or visit our VFX Newsgroup at <news://news.dfpug.de>.

Thanks to all VFX customers for the great feedback provided so far!

VFX 9.5 - More productive than ever before!