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

VISUAL EXTEND 8.0

English User Manual

dFPUG c/o ISYS GmbH

Copyright

1. Introduction

By Rainer Becker

1.1. Why Visual FoxPro 8.0 ? Why Visual Extend 8.0 ?

Where do we come from, where are we standing now, and, after all, where do we want to go? These questions may appear a little bit unusual in the introduction to the manual of a framework. But nevertheless it is worth wile reading, informative and perhaps entertaining. And if we can reach at least some agreement of opinions in the following chapters, we should get along well in the future because then you know more exactly where we want to go and how far your aimed direction matches our ideas, and how it helps you with achieving your objectives.

1.2. The advantages of FoxPro

Perhaps you know the saying, "moving house thrice is just like a home burnt down once". Those who had fought their way from FoxPro/DOS to FoxPro/Windows were very busy with this first move. But as the underlying concept was already quite a modern one, this change could be handled well in most of the cases. And what did a FoxPro developer receive with his development environment – about the following offer still sounding very attractive today:

- No runtime license fees for the applications developed and deployed
- No license cost for the integrated database engine
  (which can be deployed together with the applications to end users)
- No license cost for the integrated report designer
  (which can be deployed together with the applications to end users)
- Integrated tools for generating forms, menus, and reports
- Integrated tools for help file integration, debugging, and installation
- All kinds of tools for automation of various development tasks (Builders and Wizards)
- A programming language of by far greater capability than the script languages which are so fashionable again these days
- Powerful relational database with alternative client/server back-end database (mostly MSDE / SQL Server but MySQL and others are welcome to)
- Hybrid data access by both record-oriented and SQL-based means
- Calculable development workload in the production of data-oriented applications for all fields of usage
- Performant applications on the fastest PC database available

Looking back, the move from FoxPro/DOS to FoxPro/Windows was relatively easy when compared to the step from FoxPro/ Windows to Visual FoxPro. This second step was a substantially bigger challenge due to completely new concepts like object orientation and inheritance, methods, events, and properties. Regrettably and unexpectedly, some developers have – and alas, with success – avoided this change until today. As a result of this, the know-how and the decision-making ability for a possible change are missing. The question is, how long shall this go on?

The advantages of FoxPro enumerated in this first section have been kept up in Visual FoxPro. And the following substantial extensions have been added:

- Completely object-oriented development environment with a lot of capable classes, inheritance, the container concept and the matching new or extended tools
- Hybrid programming approach both procedural and object-oriented – resembling the alternative data access facility on a record-oriented or set-oriented (SQL) basis

- Support of all modern technologies like ActiveX, OLE automation, and meanwhile of course WebServices, XML, and COM, too
- Fast website setup through database integration and string functions, e.g. with Active FoxPro Pages or WebConnect

It was a stroke of genius not only to keep up the hybrid approach of data access (record-oriented and set-oriented), but to establish a model of programming which is now hybrid, also (procedural and OOP). Changing freely between SKIP and SQL now became possible as well as calling a procedure from an objects method. With this comes the tremendously practical feature of directly editing the class definitions and form definitions (so-called metadata). This has always permitted bending the inheritance hierarchy into shape in order to insert or correct intermediate levels later. The concept of hierarchical containers replaced the SimpleFrame interface and turned out to be very practical as well. The SimpleFrame user interface was fostered by Microsoft but did not allow two objects with identical names within one form, thus making the basically interesting reuse of aggregated objects a painful job.

Some programmers had used to copy code and adapt it instead of building an extended standard function, and thus it was their specialty to exchange a small saving of development time for a large lot of maintenance work. In the new Visual FoxPro world, they could of course continue to do this, ruining any thinkable advantage from the very outset. This happened more often than not. On the other hand, the desperate effort to model really everything and anything in classes and to introduce a separate layer of empty intermediate classes for any level thinkable lead to a much too complicated concept of development, a concept nobody can ever communicate to new staff on a project. As always, the truth is to be found somewhere in the middle.

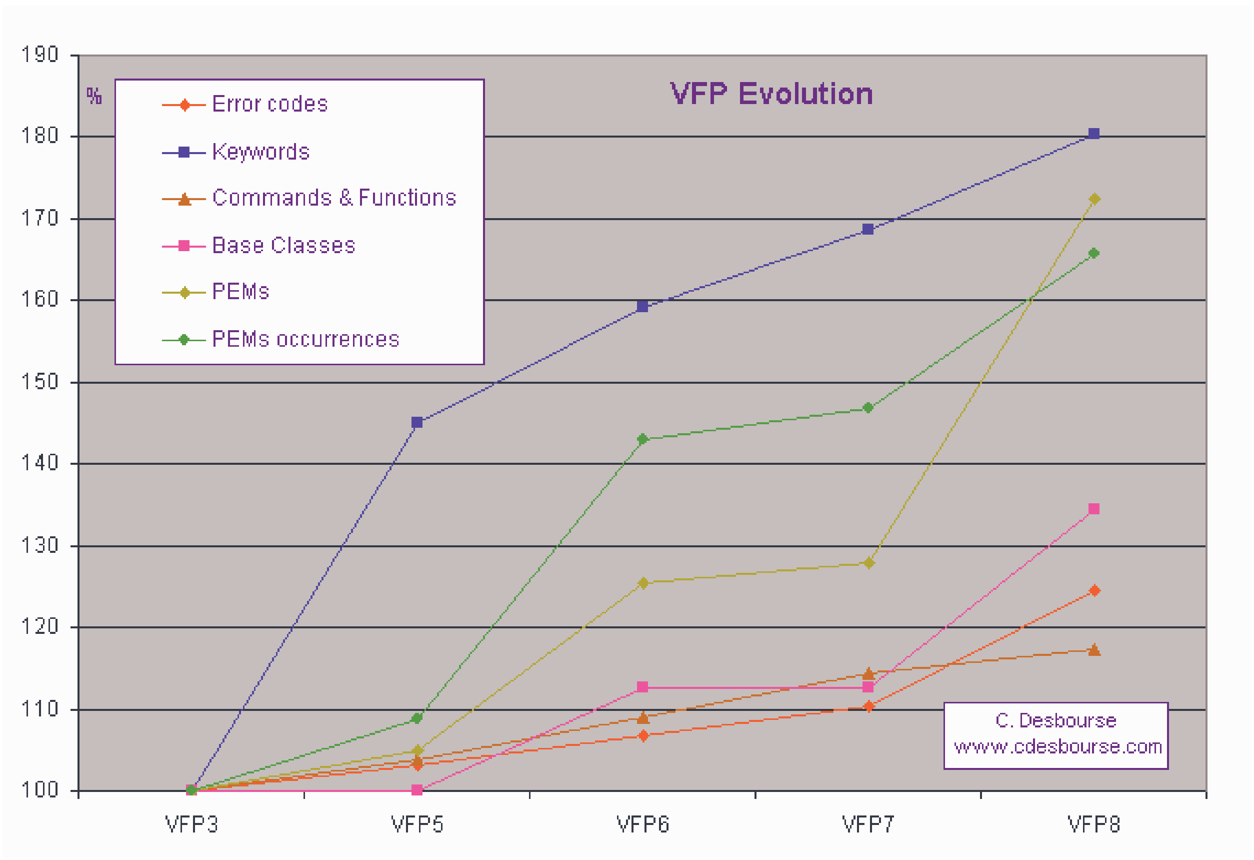
Development really ripened with the large leap in features of the stable and capable version Visual FoxPro 6.0 which is considered the most frequently used version of Visual FoxPro. Owing to its downward compatibility each further update, beginning with Visual FoxPro 3.0, was rather smooth. Therefore people could change to the following version any time, although the German edition of version 7.0 seemed to have several annoying storage leaks.

1.3. Yet more advantages with Visual FoxPro 8.0

The current version Visual FoxPro 8.0 is a substantial step forward from its predecessor, featuring a leap in developing like with the popular version 6.0. You see this especially clearly in the respective treatment by Christian Desbourse which contains a quantitative comparison of all Visual FoxPro versions to date. The translated article has been published in the 12th instalment of the German loose-leaf periodical FoxX Professional, but it is also available online in the dFPUG (German-speaking FoxPro User Group) document portal and in the original English version on Mr. Desbourse’s website

http://www.cdesbourse.com/Technology/t_vfp_evolution_en.htm

Unfortunately, this large step forward in developing is not yet known well enough, and many FoxPro developers do not see it, let alone see the resulting benefit for the practical work.



The most important features of Visual FoxPro, version 8.0 are:

- New tools like Taskpane, Toolbox, and Code Reference search
- Improvements in tools like the report generator, form designer, view designer and others
- New Builders, e.g. the data environment Builder and the XML Web Service Builder
- Extension of the database by auto-increment fields, expressions and SQL functionality
- Totally new and flexible error handling with the TRY-CATCH construct
- Ample event binding within and outside the own application
- At last, custom visual subclasses for page, column, header, option button, command button
- New classes like Collection, XML Adapter, Cursor Adapter or just Empty
- Support of hyperlinks, delayed data binding etc. in objects
- Further improvements in the areas COM Server, IntelliSense – almost everywhere.

It is called the biggest update since Visual FoxPro 6.0, and rightly so. There is plenty to learn for the users, especially for those of us who have left out one or more updates, i.e. for the majority of those who have now changed from Visual FoxPro 6.0 or even 5.0 directly to VFP 8.0 and may learn the innovations of two or three versions in one go. But to pity them is inappropriate, as any soldier of fortune does update his equipment and sharpens his knife – only FoxPro developers are fond of skipping one or two updates, thus missing out 3 to 5 years of evolution in a market where 5 years mean one generation. Of course it is true: Updating takes not only the update fee, but also a lot of time of getting used to the new thing. And why should one update an existing application when the direct benefit to the paying end user is not already apparent?. On the other hand, it is also true that learning in continuous small steps is better than not to update for years and then be forced to take a giant step which in many cases consumes more time than available in everyday work, ending up in a failure.

In this context Microsoft’s announcements are nearly as important as the multitude of practical technical extensions. For example

- Support promise by Microsoft for Visual FoxPro 8.0 until 2010, that is for a longer time than any other product, and an assumed prolongation in the next year for the following version of Visual FoxPro called Europa
- Availability of a Service Pack 1 for Visual FoxPro – the standard argument in Germany, “I will keep waiting for the first Service Pack because a .0-version by Microsoft won’t work anyhow” is refuted
- Availability of an updated OLE DB provider for Visual FoxPro 8.0

A sad thing is of course the general stop of the localization of the user interface and help file for German and Spanish (all other language supports have come to an end with version 6.0 or at the latest with 7.0, but nobody noticed that around here anyway). Therefore the following bits of information are helpful:

- Availability of a German user interface in the document portal of the dFPUG for free
- Availability of a German update book for VFP 8.0 via the dFPUG mail order service
- Announcement of a German documentation/ help file by the dFPUG

Putting together the German user interface, the help file of Visual FoxPro 7.0 (if you have got that one), and the update book for VFP 8.0, almost everything except for the Wizards exists in German. Therefore Microsoft’s cost-cutting renunciation of a localization should have little effect on developers’ everyday work. As soon as the German documentation becomes available, this issue comes to an end anyway.

By the way: There might be an additional argument for you to update to Visual FoxPro 8.0 right now from Visual FoxPro 5.0 and 6.0 or Visual Studio 6.0 / 97. All these old and outdated versions are still allowed to upgrade for low cost and it might be for the last time you will be able to update for such low cost. You should prefer to upgrade old versions right away so you earn an additional year of learning experience (even if you do not use the new version in your production environment) while preserving your update rights for the upcoming version. Alternatively you can spend the money of two updates for one full version in about a year when you want to have the upcoming version Europa with its excellent report writer extensions that you need to have anyhow.

1.4. The advantages of Visual Extend

The framework Visual Extend has never left its FoxPro foundation, and we consider this fact an important reason for the success of the product. The summarizing marketing buzzwords have stayed unchanged for many versions, still hitting the nail on the head:

- Develop your own Office compatible applications
- Contains powerful builders for forms, grids, picklists, and many sophisticated one-to-many forms
- Set up new projects in the language of your choice with the Visual Extend Application Wizard
- Integrates perfectly well into your existing Visual FoxPro environment
- Let the Visual Extend Builders do the hard programming work for you
- With Visual Extend, you too can become a Visual FoxPro professional

The existing versions of Visual Extend support the Visual FoxPro developer by means of the following tools:

- Automated project setup
- Office-compatible standard menus with favorites and toolbar
- User administration and access rights
- Data base maintenance and error log

- Adaptable options dialog and info dialog
- Form Builder with tab definitions for the processing and list display of data
- Grid Builder with incremental search and storage of user settings
- The same Builder combination for 1:n forms
- Builders for various kinds of picklists
- Tools for language administration and message texts
- Integration of reporting functions and filter functions
- Standard classes for Active Desktop, mover boxes, thermometer display, pick date etc.
- Standard functions for primary key generation, logging/audit trail
- Multi-customer usage on one installation and client/server support
- OLE support and help generator

These features successfully cover all standard areas of application development either purely in Visual FoxPro or as a 2-layer application with a back-end database (such as e.g. SQL Server or MSDE) according to the client/server principle. The efforts to implement a complete 3-layer architecture with a separated business logic have not been continued. The demands of this architecture and the level of necessary theoretical background knowledge for developing attractive applications in the area of small to medium businesses are much too high in most of the cases.

Apart from technical reasons there are some further heavyweight arguments in favor of Visual Extend:

- Very long period of practical tests and ripening from Visual FoxPro/Visual Extend 3.0 through the versions 5.0, 6.0, 7.0 and 7.1 to the current version Visual FoxPro / Visual Extend 8.0
- Large basis of users with many hundred users each internationally and in each of the countries of German language which brings about a better independence of local issues
- Solid manufacturer with many years of experience in Visual FoxPro and faithful customers (formerly Devigus Engineering, today the dFPUG itself)
- Copious online support with forum, newsgroups, and a lot of news, documents, lecture manuscripts, articles and slideshows around the product, covering almost every question

Just for the reasons named above, in the German-speaking area more licenses of Visual Extend are sold than of all other suppliers of frameworks in total, and this has a very positive influence on the reliability of the further development and maintenance of the product, even though of course Visual Extend just like Visual FoxPro itself suffers a bit from the often weak willingness of the users to buy an update.

1.5. Even more advantages with Visual Extend 8.0

The current version of Visual Extend keeps moving in the same strategic direction that has made the earlier versions of FoxPro so popular, focusing on a stepwise evolution while keeping up backward compatibility as far as possible. The new version Visual Extend 8.0 brings about the largest update ever. The free intermediate release 7.1 did already guarantee the pure ability to run under the new version of Visual FoxPro, 8.0, thus relieving us of time pressure to ship as soon as possible due to the new FoxPro version. Therefore emphasis could be put on a general overhaul and essential extension in many areas. A brief overview of the most important ones among the new features:

- Integrated extended menu designer
- Report redirection into PDF files
- Report output as mail attachment
- Open-dialog in Windows XP style

- Support for treeview controls
- Builder for cTreeViewForm and cTreeViewOneToManyForm
- Product activation key for 32 modules, configurable by the developer only
- Configurable Internet download scripting for e.g. application updates
- SQL Server database update for clients
- Update page for cPickTextBox Builder and for cPickFieldBuilder
- New table-driven cPickAlternate Builder
- Application Manager as a VFP8 TaskPane
- Use of DataEnvironment classes
- Additional sample applications, sourcecode included
- Enhancements of the Grid and Form Builder
- Extended cSearchDialog with 5+ criteria
- Integration of a backup function
- New XP layout of the login dialog
- Vastly extended OLE automation for WinWord
- cPickDate with additional hotkeys
- New class cDatetime for the input of date/time values

The report designer always came in as the top issue when the dFPUG conducted big opinion polls about the demands on the new Visual FoxPro version 8.0. The new version of Visual Extend expands the report designer by PDF output and eMail output, and hence increases the utility of existing applications without additional labor. The menu designer came in second. It is also being addressed thoroughly and in an elegant manner in the new version.

Something had to be done about the look-and-feel also. Treeviews are state of the craft, and Windows XP has beautiful looks, but extended picklists and many other extensions of the user interface are naturally indispensable for the production of easily usable applications. In any case, here is also a large variety of possibilities not only to develop new applications faster, but also extend existing applications easily in order to offer a reworked release to customers.

As almost half of the professional developers work in the field of client/server computing as well, the database update feature has been extended for this area. For the others a function for data backup was added which is simple to integrate into one’s own application. It generates ZIP files without further license fees or installation hassle.

And for those who do not only develop in-house, but also offer their software on the free market, we now finally have a sensible solution for updating the application via the Internet, and also a product activation with individual rights for up to 32 modules by means of an activation key freely definable. This method is working well in the sales of Visual Extend itself. We therefore offer an extended variant of it to all developers.

The multilingual ability of Visual Extend has proved its worth, too. The international spread of its users show this. For that reason we offer this feature in an enlarged variant to all developers: The language support of the generated applications has been extended from hitherto German, English, French, Italian and Spanish by Greek, Bulgarian and Czech to 8 languages in total supported by the applications. Additional languages are soon to follow.

As with the Service Pack 1 for Visual FoxPro 8.0 we can announce a new build for Visual Extend 8.0 to. Yes, there were some bugs fixed but additionally we added the following capabilities:

- Extension of menu designer to add code to events and define menu positions
- New version of VFX.FLL (containing ZIP and activation functions) for Windows 98
- Enhanced form builder with better positioning and an rearrange-all option
- Enhanced search dialog for better usability
- Storage of temporary files in the sys(2023)-directory instead of EXE directory

The new build is available at the website http://www.visualextend.com for free download.

1.6. The next versions

For the next version of Visual FoxPro named Europa, a first list of new features has been disclosed for the second half of 2004; among others, you will find

- Extensible report designer with events and rights
- Extensibility of reports at run-time
- Cease of limitations in all areas, including SQL
- Substantial extension of the SQL syntax
- New data types and data base functions
- Docking and resizing for custom forms
- Various graphical functions and optical enhancements
- Extended properties window with Builder integration
- Background compiling with syntax coloration of errors

The main emphasis of Visual FoxPro’s next version, apart from the important issue of the report generator, will be the cease of limitations in any respect and the increase of developers’ productivity. This improves the value of the product for the developer considerably without interfering with frameworks in any way whatsoever. And of course we can already draw up a list of the first ideas about Visual Extend, version 9.0, though not a definitive one:

- Integration of Cursor Adaptor and perhaps XML Adaptor
- Further Builders for the automation of common tasks
- Integration with the Toolbox
- Support for further languages
- Support of the new features in the report designer and in the report generator
- Management of docking positions of user windows
- Interface classes for standard tasks like address management
- New features for the adaptation of the user interface through end users

If you have additional ideas and wishes please contact us in the newsgroup and send your ideas to mailto:vfxwish@visualextend.de as soon as possible.

As it will take some time until Visual FoxPro 9.0 is completed, we have looked for other areas of improvement meanwhile. That is, making further sample applications and ready-to-run solutions to be taken over into the developer's own range of software, e.g. in the subject areas as follows:

- Address management
- Billing
- Bookkeeping/ financial accounting
- Maintenance of online shops
- Offline reader
- Server service

However Visual FoxPro 9.0 and the corresponding version Visual Extend 9.0 lay some time ahead of us. So for the time being you should not worry about that, just be sure that the respective products are obviously under continuous development, and this fact can only be of benefit to you. That is the only thing such product previews shall demonstrate. And they shall prevent you from investing work in the development of features which you are going to receive in the following version anyway. So announcing the next version is useful, but under no circumstances shall it be an obstacle to working your way into the current version of Visual FoxPro and Visual Extend – and using them.

2. VFX 8.0 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 or a data model 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.

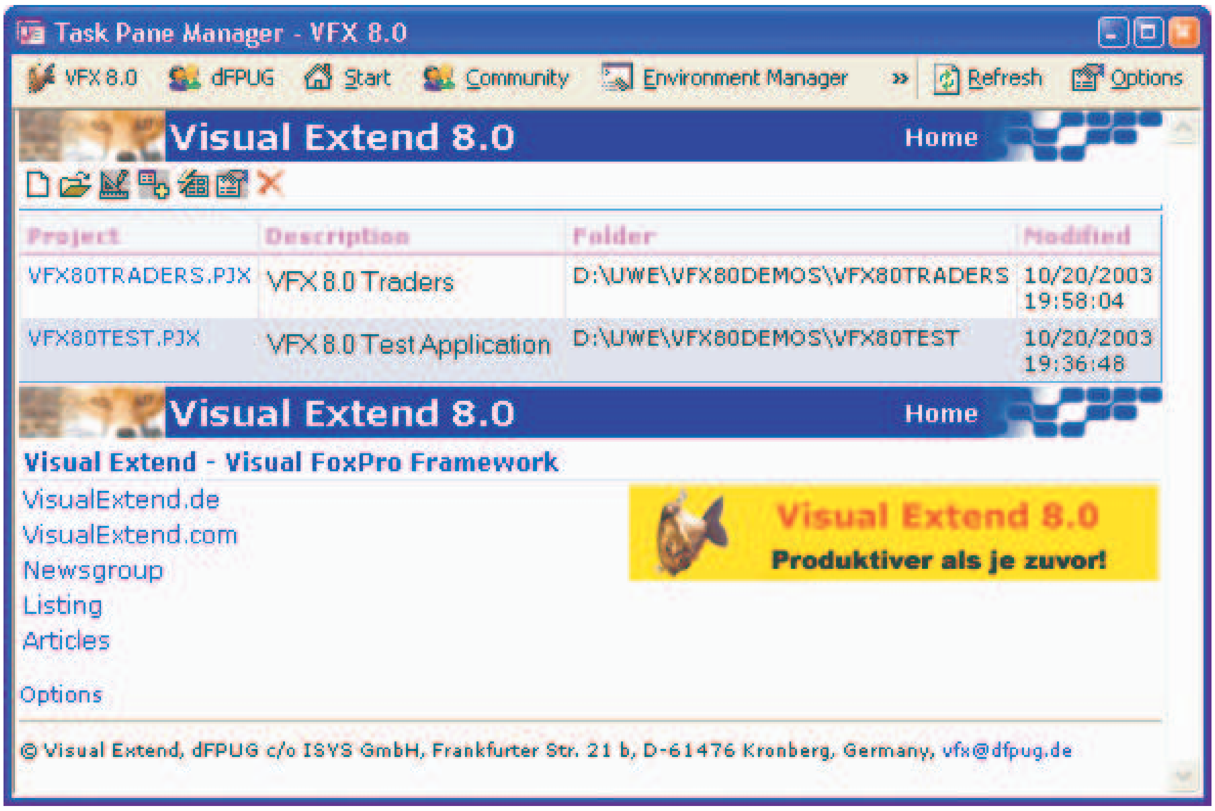
2.1.1. Installation

After the installation of VFX, it is helpful to integrate the VFX Menu into the standard Visual FoxPro Menu. For this purpose, insert the following line into your Config.fpw file:

```
Command = DO <VFX installation path>\builder\vfxmnu.app
```

2.1.2. VFX Task Pane

The VFX 8.0 Task Pane is integrated automatically with the first start of VFP after the installation of VFX 8.0. The VFX 8.0 Task Pane contains among with other, all functions of the earlier VFX versions known VFX Application Manager.



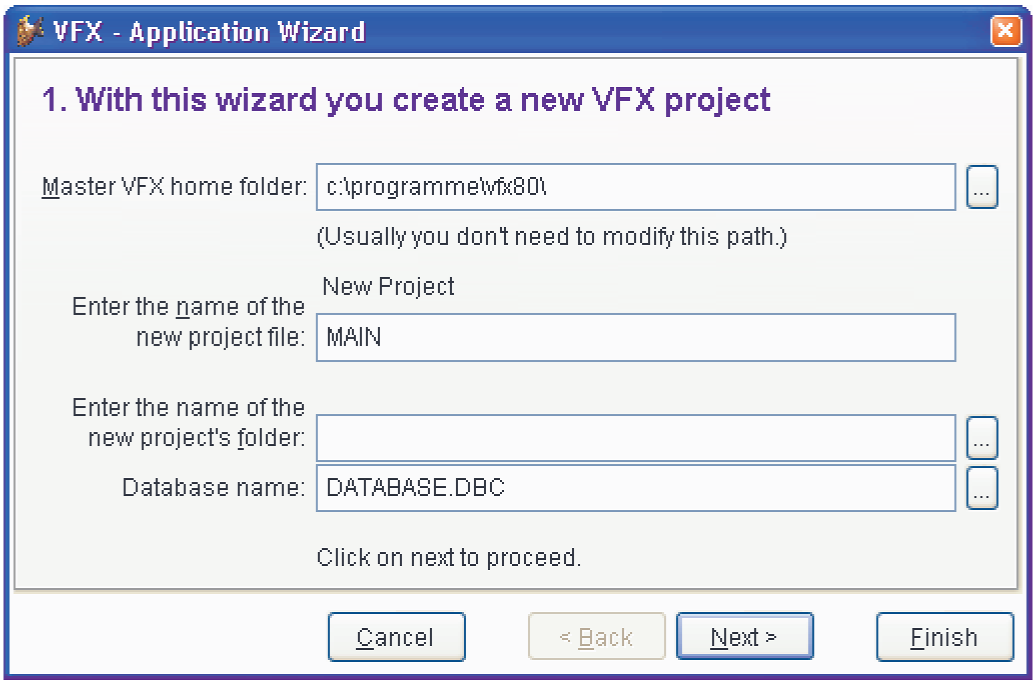
Project	Description	Folder	Modified
VFX80TRADERS.PJX	VFX 8.0 Traders	D:\UWE\VFX80DEMOS\VFX80TRADERS	10/20/2003 19:58:04
VFX80TEST.PJX	VFX 8.0 Test Application	D:\UWE\VFX80DEMOS\VFX80TEST	10/20/2003 19:36:48

© Visual Extend, dFPUG c/o ISYS GmbH, Frankfurter Str. 21 b, D-61476 Kronberg, Germany, vfx@dfpug.de

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 Application Wizard.



1. With this wizard you create a new VFX project

Master VFX home folder: c:\programme\vfx80\

(Usually you don't need to modify this path.)

New Project

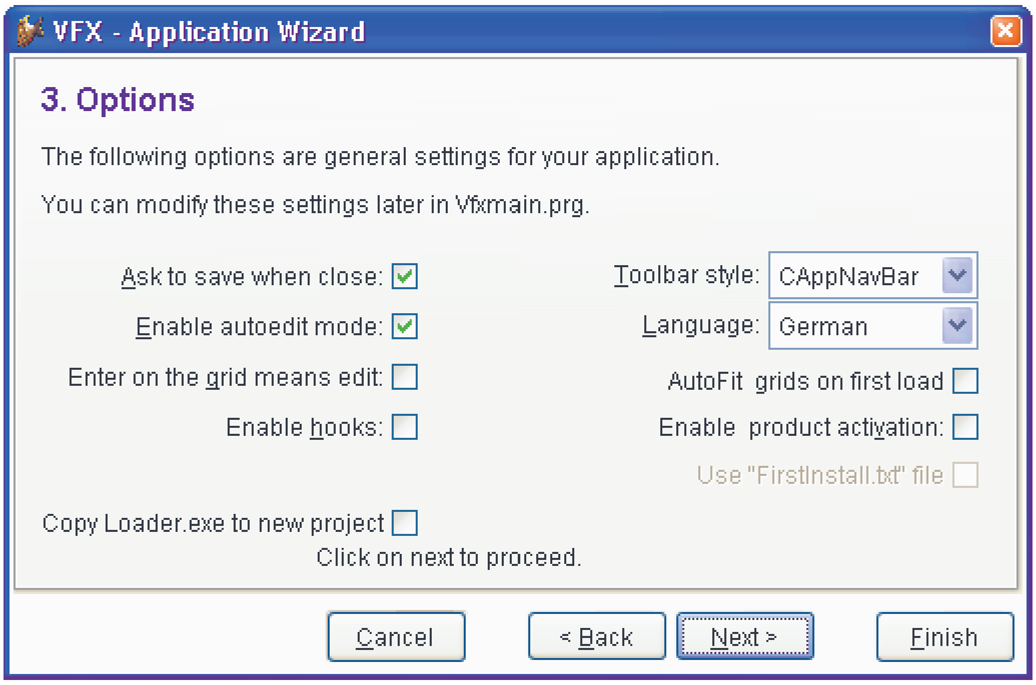
Enter the name of the new project file: MAIN

Enter the name of the new project's folder:

Database name: DATABASE.DBC

Click on next to proceed.

As language for the created application by default will be suggested the language of the used Visual FoxPro version. 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.



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: ✓

Enable autoedit mode: ✓

Enter on the grid means edit: ☐

Enable hooks: ☐

Toolbar style: CAppNavBar

Language: German

AutoFit grids on first load ☐

Enable product activation: ☐

Use "FirstInstall.txt" file ☐

Copy Loader.exe to new project ☐

Click on next to proceed.

2.2. *Functionality of the new Application*

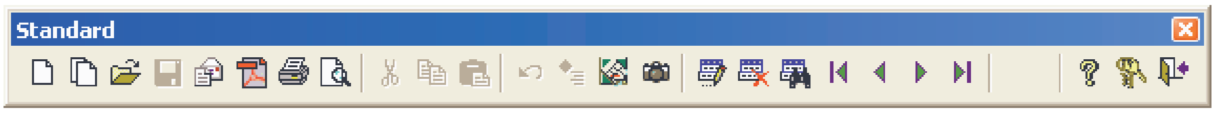
The application created with the Application Wizard can be tested immediately. For this purpose, the main program Vfxmain.prg can be started directly from the project manager. Alternatively, also an App or an Exe file can be created and tested. However, usually this is not necessary during the development.

The application starts with a Splash screen. As picture for the Splash screen is used a png-file, which can be easily edited or exchanged by the developer. It is possible to pass around the login form. After displaying the Splash screen, the main screen is created and the login form is activated. By default, each user of a VFX application must be logged in with a user name and a password. It is possible to skip the login form and to be log the user automatically with the Windows login name. Alternatively, the user administration can be completely switched off.

2.2.1. Usage

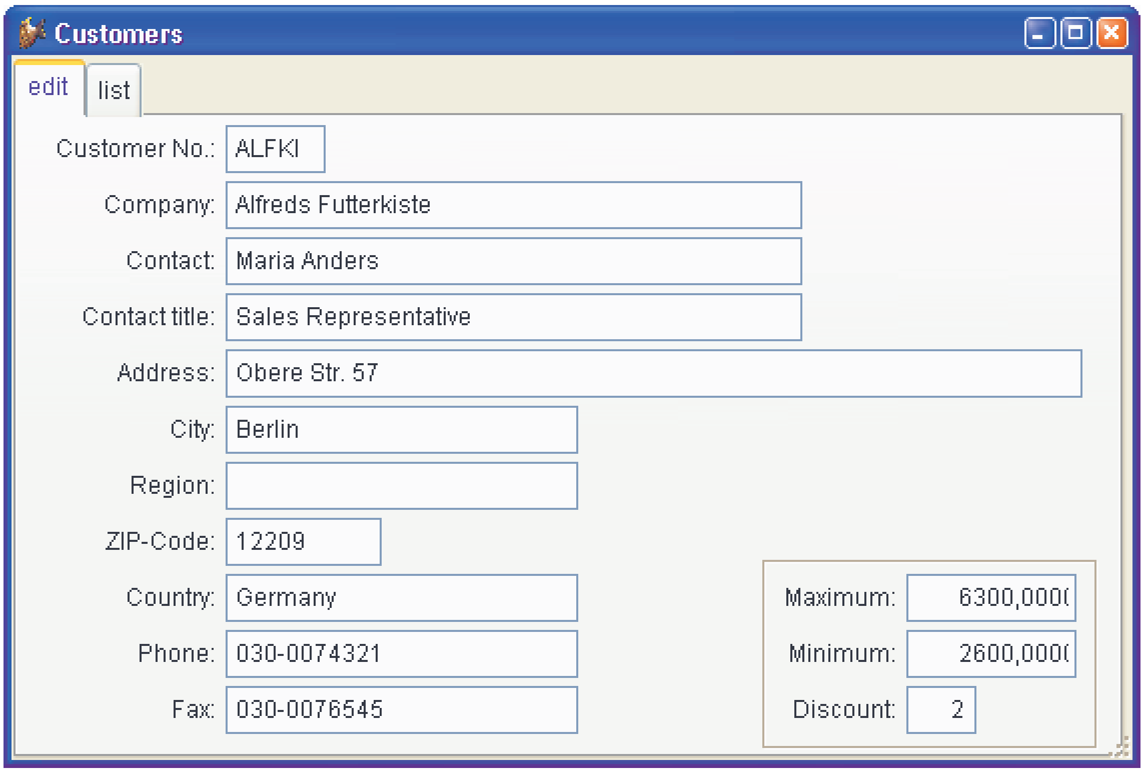
After the login, the VFX application is used similarly like Office applications. Users, who are familiar with the usage of Word or Excel, can practically immediately work productively with a VFX application.

2.2.2. Standard toolbar



All buttons in the toolbar on the screenshot, which are not specially marked, are identical in their function to those from Office products.

2.2.3. Form



Customer No.:	ALFKI
Company:	Alfreds Futterkiste
Contact:	Maria Anders
Contact title:	Sales Representative
Address:	Obere Str. 57
City:	Berlin
Region:	
ZIP-Code:	12209
Country:	Germany
Phone:	030-0074321
Fax:	030-0076545
Maximum:	6300,000(
Minimum:	2600,000(
Discount:	2

If for a form the property lAutoedit is set to true (which 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 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 changed proportionally. Within grids, the size of the control is not changed by default. If 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 with no problems. Through a property of the form (*lMultiinstance*) can be prevented multiple instances.

2.2.4. User list

In VFX is included a user administration. Here are placed a form for editing the user data, a form for defining the user rights and a login screen. A user security level can be set through a numeric field.

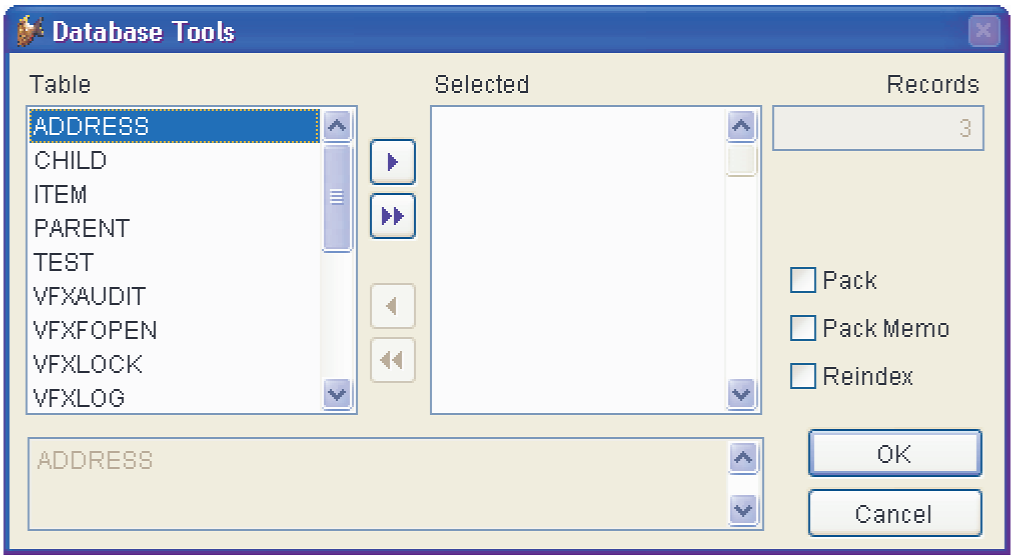
For all fields of the current user data record (from the table *Vfxusr.dbf*), corresponding to the currently logged user, global variables with the prefix gu_ are created. It is possible in each place in the program to check the value of these global variables, in order to decide whether a user may perform a certain action. So can, for example, be restricted the selection of a menu option, which opens a form or working on disable a field on a data manipulation form.

2.2.5. 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, 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.

2.2.6. Database Tools

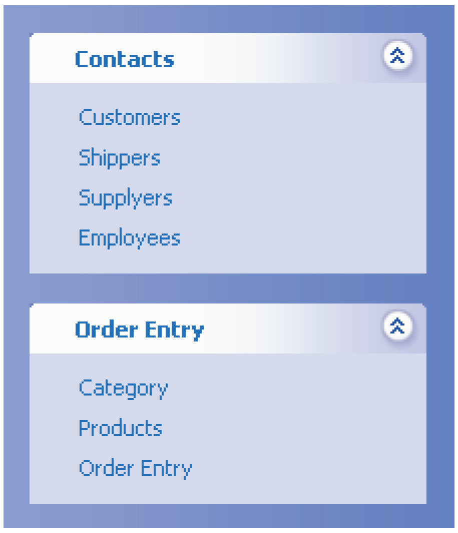
Under the menu option Tools – Database… is invoked a form with a Mover Dialog.



Here tables can be packed or reindexed.

2.2.7. Open-Dialog

By default, forms are started using the Open Dialog. The Open Dialog appears in Windows XP layout. The data for the forms are located in the table *Vfxfopen.dbf*.

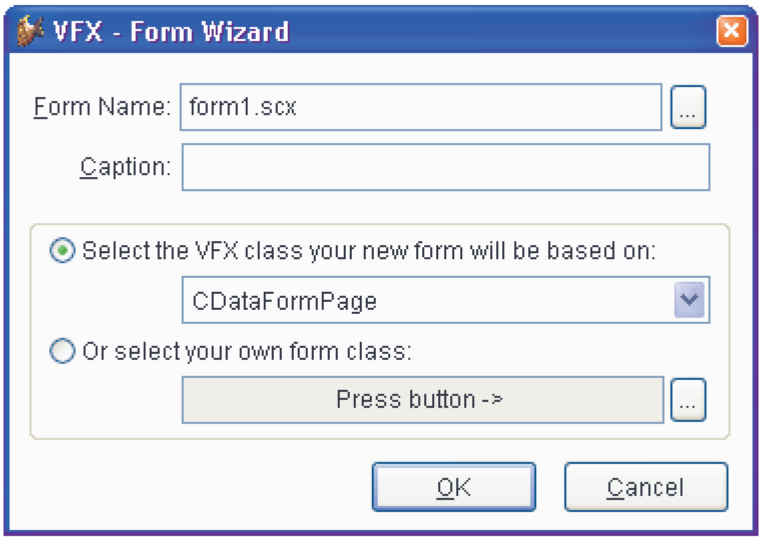


2.2.8. About Dialog

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, included into the project and opened for edition.


VFX - Form Wizard

Form Name: form1.scx

Caption:

[✓ Select the VFX class your new form will be based on:] CDataFormPage

Or select your own form class: Press button ->

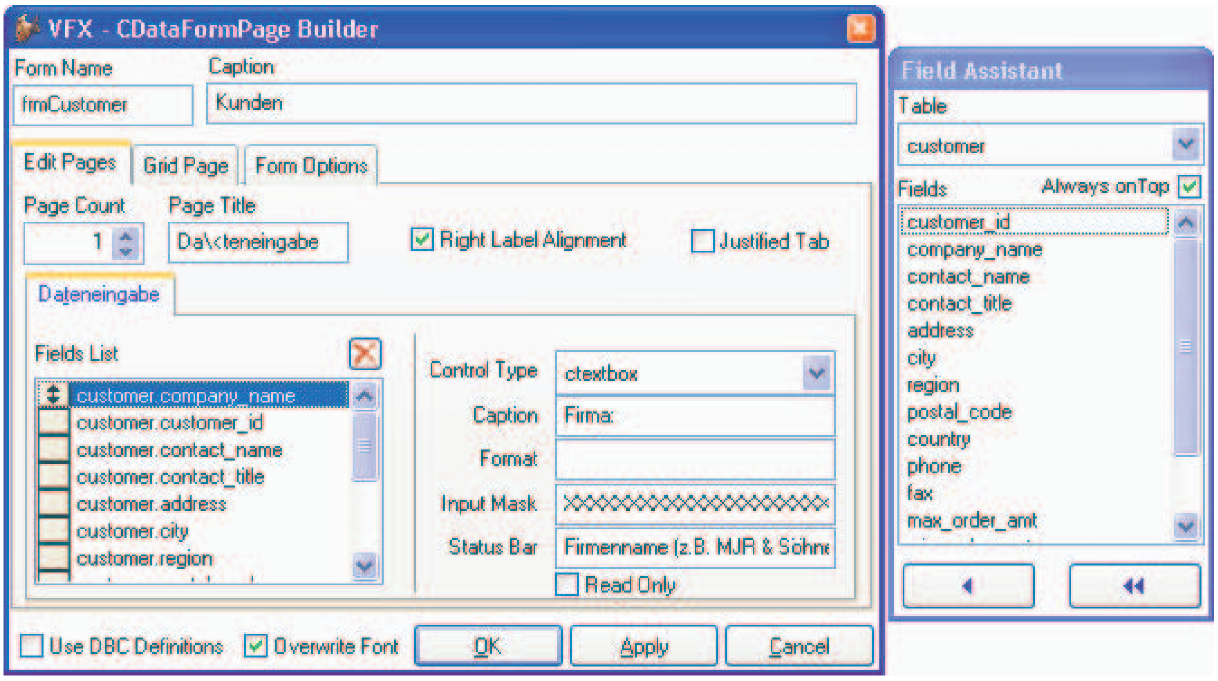
2.4. Data environment

The tables or views, used by the form are set in the data environment. The VFX-Form Builder reads the data environment and allows controls creation, based on the fields of the tables. At run-time, the 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 created 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.



VFX - CDataFormPage Builder

Form Name: frmCustomer

Caption: Kunden

Edit Pages | Grid Page | Form Options

Page Count: 1

Page Title: Da\<teneingabe

[✓] Right Label Alignment

[ ] Justified Tab

Dateneingabe

Fields List:

customer.company_name
customer.customer_id
customer.contact_name
customer.contact_title
customer.address
customer.city
customer.region

Control Type: ctextbox

Caption: Firma:

Format:

Input Mask: XXXXXXXXXXXXXXXXXXXX

Status Bar: Firmenname (z.B. MJR & Söhne

[ ] Read Only

[ ] Use DBC Definitions

[✓] Overwrite Font

Field Assistant

Table: customer

Fields

Always onTop [✓]

customer_id
company_name
contact_name
contact_title
address
city
region
postal_code
country
phone
fax
max_order_amt

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 overwriting the properties.

2.6. The VFX-CGrid Builder

When it is needed to be made changes for the Grid, 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 also reentrant.

2.7. Testing

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. In addition, 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.

2.8. Creating an One-To-Many Forms (1:n)

The top part of the One-To-Many form looks exactly like the standard forms. In the lower part is placed a Child Grid, in which data from a Child table can be edited. It is possible several Child Grids to be placed on a Pageframe. Thereby, the grids can display different columns of one and the same table or data from different Child tables.

2.8.1. The VFX-COneToMany Builder

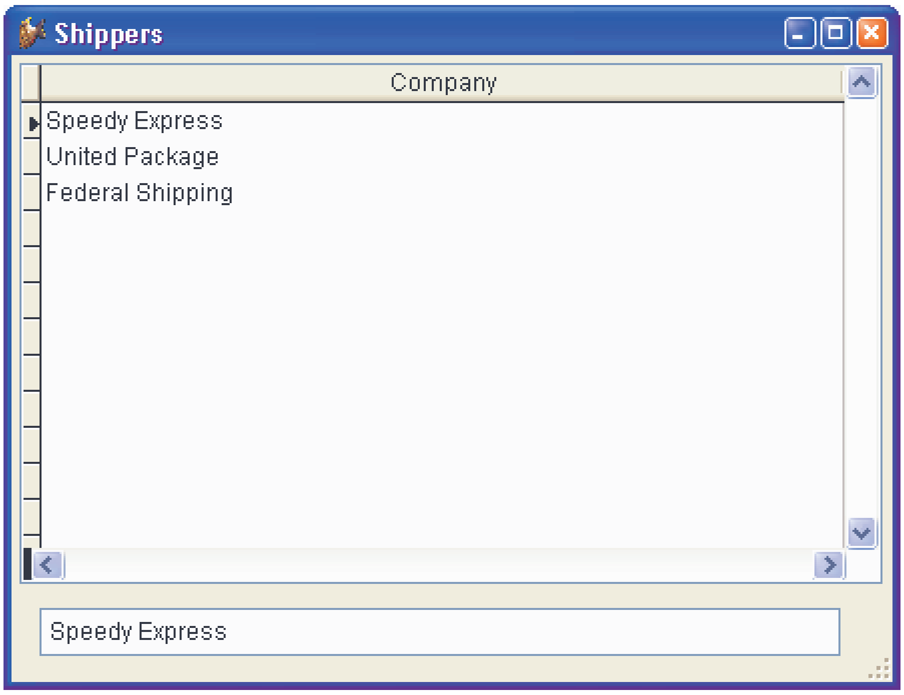
Additionally to the standard Form Builder, the VFX-COneToMany Builder contains a Page for creation of Child Grids. The Child Grid control source table is selected and the columns are arranged.

2.8.2. The VFX-CChildgrid Builder

In the builder for Child Grids, the data of a Child Grids can be edited. The difference to the normal Grid builder consists of the fact that with the builder for Child Grids can be edited the code of the OnPostInsert method. In this way, a new Child data record can be linked to a Parent data record, when the key of the Parent data record is stored in the Child data record. This happens in the OnPostInsert method. VFX completely generates the code into the OnPostInsert method. However, the code is commented. After check by the developer, the comment characters can be removed. A rewrite of the code is usually necessary, only if compound keys are used.

2.9. CTableForm

Another type of form is the *CTableForm*. With this form, the list Grid and the control members are represented each next to other or among themselves. Therefore, it is suitable in particular for forms with only few input fields.



Company
Speedy Express
United Package
Federal Shipping

Speedy Express

2.10. More Functions

Through a form property (*lMore*) 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 will be created an array containing parameters passed to the called VFXMore form in which can be selected between the available functions. For example can be started Child forms.

2.11. Creating Parent/Child-relations between Forms

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.

2.12. Picklists

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.

2.13. Picklists in Child-Grids in One-To-Many Forms

Also within Child Grids in One-To-Many Forms can be used Picklist controls. The settings for it are also made through a special builder.

2.14. Adding a toolbar to a Form

The possibility to add a toolbar for a form is very user-friendly. The toolbar can be normally created with VFP. In the Click method of the toolbar buttons a method of the active form is naturally called. For example:

```
_screen.activeform.mymethod()
```

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 in a per user basis.

2.15. Client/Server-Applications

2.15.1. Using views

A table or a view can be used alternatively as data source for a form. If a view is used, the property *lWorkOnView* of the form must be set to .T.

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. In addition, the use of views is possible with Picklists. Thus, a VFX application can be used as front-end for example for a SQL server or other remote data sources.

2.15.2. Entering the View parameters

There is a special VFX Form class, designated for the input of the view parameters. A form is based on the class *cAskViewArg*.

The controls, which contain view parameters as ControlSource, can be copied through the clipboard from the Data Manipulation Form into the form for input of view parameters. In a property (*cviewparameter*) is written the name of the respective View parameter. The form for entering view parameters can be called in the init method of the data manipulation form or for example through a button.

2.16. Changing properties of the Application object

In the main program, Vfxmain.prg programmatically is created an instance of the class of the application object. Here, in the code, it is possible to be changed VFX methods and properties without having to make changes at the class libraries.

2.17. Mover-Dialog

The Mover Dialog is a practical tool for selection of relative small data. The Mover class will be programmatically instantiated. Parameters are an array for the selection list and an array for the selected elements, which also contains the resultant chosen elements after closing the dialog.

2.18. 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.

2.19. 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. In addition, the data environment window is opened.

2.20. 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, global variables with the prefix gs_ are created for later use and are initialized with the values of corresponding fields.

2.21. Multi-lingual Application, VFX-LangSetup Builder

At the time when a new VFX project is created, can be selected between the languages German, English, French, Italian, Spanish, Greek, Bulgarian and Czech. Accordingly to the selected language the Include files are copied into the include folder of the new project.

When someone wants later to translate its application into another language, he will have to start the VFX LangSetup builder for each form. This builder creates an assignment for each Caption of a form. At run-time to the Caption is assigned the value of a constant.

The constants can be created and edited with the VFX Message editor. Before the compilation of the application just copy the Include files of the desired language into the include folder under the project folder and compile the application.

2.22. Updating the user’s database

VFX contains procedures to automatically accomplish an actualization of the database at the customer side. For this purpose, under the data folder a folder named update is stored. Into this folder should be copied all tables of the database, just without data. At the program start, the database in the data folder is updated. In this way can be added new tables in the database, new fields in tables, new index keys and new views. In the same way no more needed tables, fields etc. are deleted. Subsequently, all files in the update folder are deleted. Free tables can also be updated with this method.

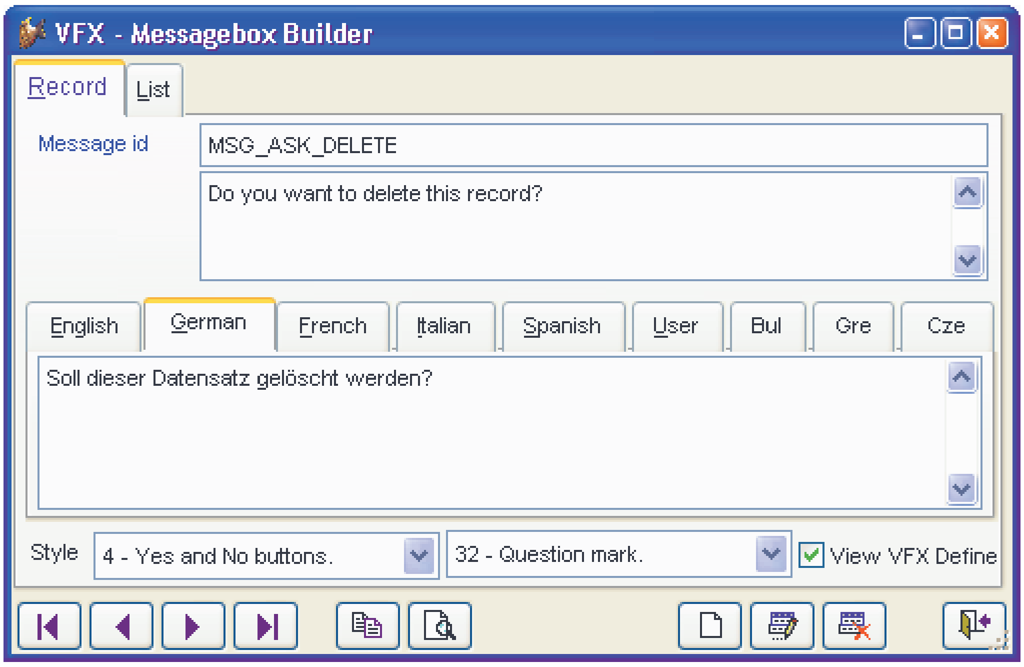
In VFX 8.0 is also supported the creation and actualization of SQL server databases.

2.23. VFX-Class Switcher

With the VFX Class Switcher, it is possible later to be changed the base class for a control. Therefore, for example, it is possible to make a spinner or a Picklist from a text box control.

2.24. 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.



Message id: MSG_ASK_DELETE

Do you want to delete this record?

Soll dieser Datensatz gelöscht werden?

Style: 4 - Yes and No buttons. | 32 - Question mark. | ✓ View VFX Define

2.25. 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.

2.26. Hooks

VFX offers direct intervene within all important methods through Hooks. As example, let us 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 Replace command. If the OnPostInsert method returns .F., a Tablerevert() is invoked and thereby the new data record will be immediately deleted again.

Additionally to these possibilities, an Eventhook is inserted in most 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.

3. Introduction

3.1. Overview

Visual Extend is an Application Development Framework for Software Developers working with Microsoft Visual FoxPro 8.0. Visual Extend includes Builders, which assist the Software Developer in its daily work and dramatically speed up the software development process without scarifying 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 over Visual FoxPro from scratch is a major undertaking. Not only because it is 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 modular design of Visual Extend, every Software Developer can decide whether to use all or only some parts of the Visual Extend Application Development Framework. The Object Orientation of Visual Extend allows the Developers to subclass existing Visual Extend Classes to further customize or enhance their preferred Development Environment.

Visual Extend is not just a Foundation Class Library. It is 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
- Other Visual Extend Developer Productivity Tools like a 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, including optional 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 File List in File Menu, actually open Window in Window Menu
- Incremental Search including Autosort in all VFX Grids
- Alternate Sort by Doubleclicking on any Column Header in any VFX Grid
- Show 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

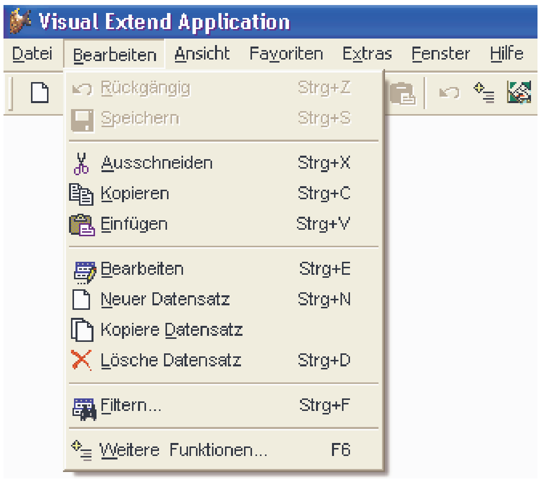
- Auto-save and Restore of Size and Position of Picklist Forms including any Picklist Grid Layout Changes.
- Powerful Picklist Object within Childgrid
- User Access Management including Password Encryption
- Auto use of Network Logon Names including Autologon Feature
- User Security including Form Level Security View, Edit, Insert and Delete restrictions
- Database Tools for Packing, Reindexing and Repairing of local Tables
- Complete run-time Error Tracking System
- System Lock-Table for optional semaphore locking schemes
- 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 online 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

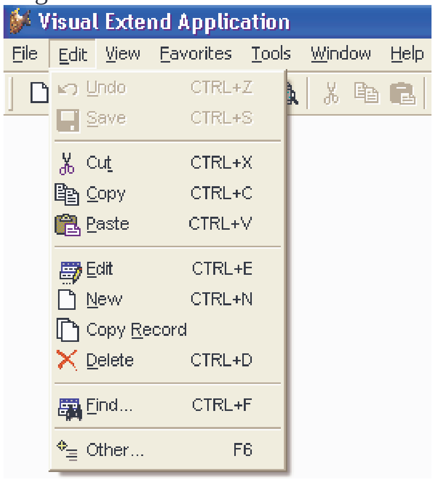
Software Developers using Visual Extend will appreciate the following features:

- 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, 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 loosing the reentrance feature of the VFX Builders as long as you add/remove all controls using the VFX builders!
- Builders for Standard Forms including Master and Child Form technique (Calling and Called By)
- Builder for Power Grids
- Builder for all your Picklist needs
- Builders for classical OneToMany Forms as well as advanced OneToMany Forms including Tab Dialog for the Master and another Tab Dialog for multiple Childs all on one Form
- All Builders get the Field Descriptions and other properties automatically from the Data description
- Form Builders will automatically size any Textbox Controls according current Field Length
- Use the VFX form builders on own VFX based form classes 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 Application Manager.
- Easy subclassing of the application class and setup of the environment class
- Easy setup of application specific main toolbars
- Linked child form techniques
- The complete Application Development Framework covers already all user interface elements in English, French, German, Italian, Spanish, Bulgarian, Czech and Greek. Start a new Application in the language of your choice without need of translating a single word of the Visual Extend Application Development Framework

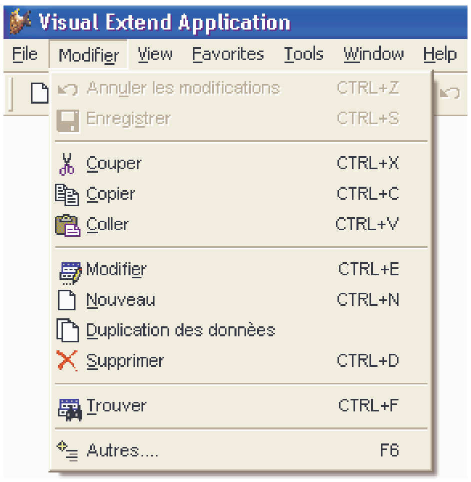
German



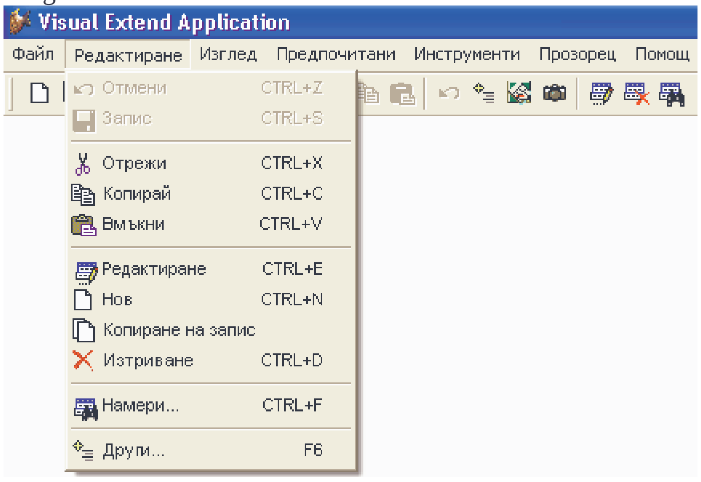
English



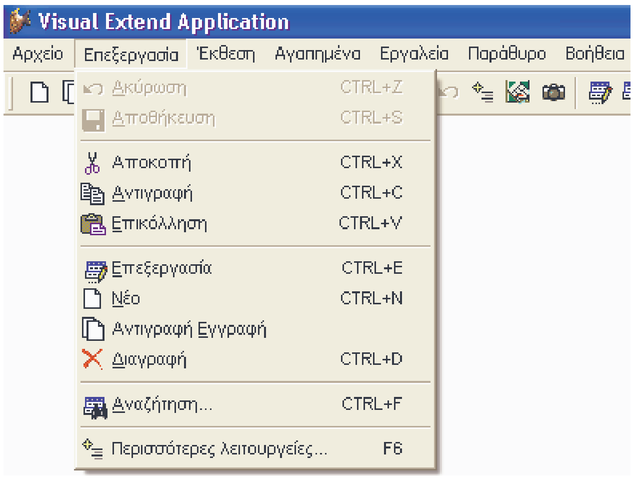
French



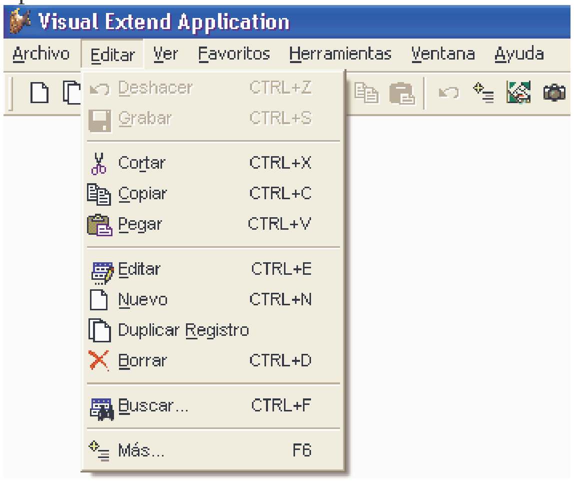
Bulgarian



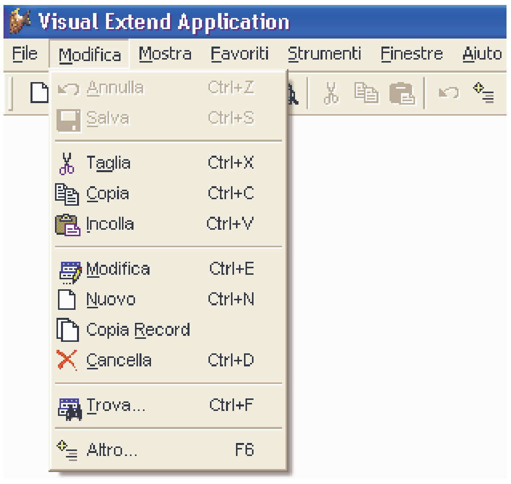
Greek



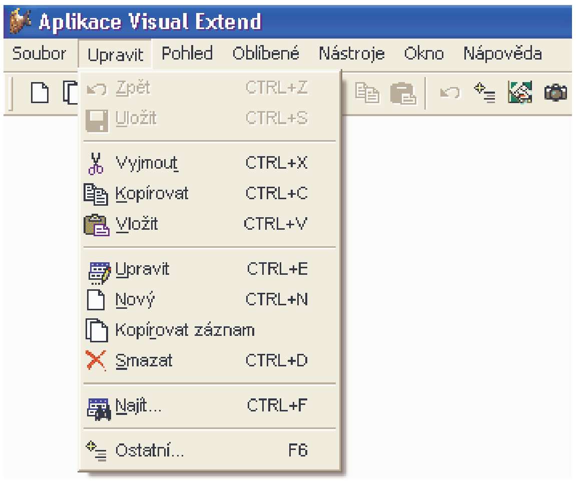
Spanish



Italian



Czech



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

4. Included Tools

4.1. VFX-Class Library

You will find the class library files in the subfolder *\VFX80\LIB*. For a detailed description of all class library files including its classes, properties and methods, please refer to the VFX technical reference online help file.

4.2. VFX-Wizards and Builders

All VFX Wizards and Builders are located in the \VFX80\BUILDER\ folder:

Builder	File	Description
VFX Wizards and Builders	VFXBLDR.APP	The following VFX Wizards and Builders will help you to create professional Visual FoxPro Applications in record time: √ Application Wizard for the generation of a new Application √ Form Wizard for the generation of a new Form √ Form Builder (including multi page, **reentrant**) √ Grid Builder (**reentrant**) √ Picklist Builder (**reentrant**) √ OneToMany Builder (including multi page for master and multi page for multiple children, **reentrant**) √ ChildGrid Builder (**reentrant**) √ Picklist Builder for Picklists within child grids (**reentrant**) ***How to start:*** *If you follow the installation instructions you can call the VFX builders using the right mouse anytime when the corresponding object has been selected.*
VFX LangSetup Builder	LANGBLDR.APP	Automizes the generation of the code for the VFX LangSetup method. Great help for creation of a multi lingual application. ***How to start:*** *Either from the VFX Menu (DO VFXMNU) by selecting LangSetup Builder or by starting the left mentioned APP.*
VFX Messagebox Builder	MSGBLDR.APP	Automizes the generation of messagebox dialogs and the associated constants for the include files. ***How to start:*** *Either from the VFX Menu (DO VFXMNU) by selecting Messagebox Builder or by starting the left mentioned APP.*
VFX Message Editor	MSGEDIT.APP	Automizes the localization of messages and other captions as well as the generation of the associated include files. ***How to start:*** *Either from the VFX Menu (DO VFXMNU) by selecting Message Editor or by starting the left mentioned APP.*

All VFX 8.0 Form, Grid and Picklist Builders are now 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 the 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 VFX80\LIB\BUILDER\VFXCODE.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
VFX Task Pane	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 and is located in the BUILDER folder.
VFX Menu	VFXMNU.APP	Call this program if you want to have a special VFX menu from where you can call the VFX Application Wizard and other useful VFX Builders. **Tip:** If you follow the installation instructions, this menu will automatically be loaded whenever you start VFP.
VFX Class Switcher	<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.
VFX Object Name Picker	<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. New Developer Tools

In addition to the well-known Builders, existing in former VFX versions, in VFX 8.0 are available new Power Builders for following Classes:

- cTreeViewForm
- cTreeViewOneToMany
- cPickAlternate

To assist the Product activation are used two wizards: (see also Application Protection using Activation Key)

- 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

Wizard for creating SQL Metadata:

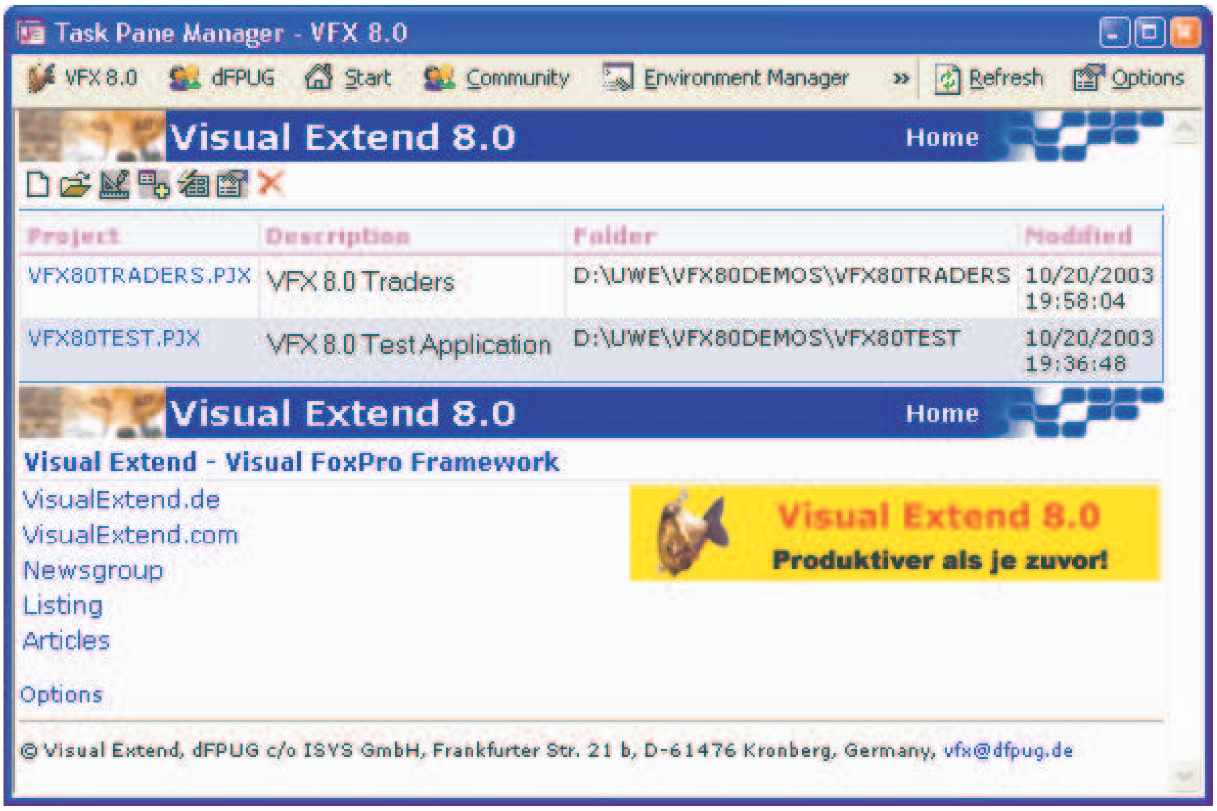
- Metadata Wizard

The new VFX-Menu-Designer:

- VMD (Visual Extend Menu Designer)

4.5. The VFX 8.0 Task Pane

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



Project	Description	Folder	Modified
VFX80TRADERS.PJX	VFX 8.0 Traders	D:\UWE\VFX80DEMOS\VFX80TRADERS	10/20/2003 19:58:04
VFX80TEST.PJX	VFX 8.0 Test Application	D:\UWE\VFX80DEMOS\VFX80TEST	10/20/2003 19:36:48

© Visual Extend, dFPUG c/o ISYS GmbH, Frankfurter Str. 21 b, D-61476 Kronberg, Germany, vfx@dfpug.de

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 8.0 Task Pane and sets current path to the project folder.
Add Project	Adds an existing VFP-Project to the VFX 8.0 Task Pane.
Rebuild	Rebuilds all files in the project, selected in the VFX 8.0 Task Pane. After rebuild, the Project will be opened for edition.
Properties	Opens the VFX – Project Properties for the project, selected in the VFX 8.0 Task Pane.
Delete	Deletes the project, which is selected in the VFX 8.0 Task Pane.

5. Installation

5.1. Hardware- and Software- Requirements

Since VFX is an extension to the Microsoft Visual FoxPro 8.0 Development System, you need a Hard- and Software Environment under which you can run Visual FoxPro 8.0. Please refer to the Visual FoxPro Hardware and Software Requirements for further information.

5.2. The VFX 8.0 Installation

Run the setup program called *VFX80Setup.msi* and follow the instructions on the screen.

Make sure to install VFX 8.0 with the installation program we provide into a new folder, do not install VFX 8.0 in the same folder as earlier VFX versions!

VFX 8.0 has software copy protection. You can install VFX 8.0 and when you start the VFX builders, a dialog tells you your personal Registration Key. All you have to do is to register your copy online on our VFX Registration Web site and we will email your Activation Key, which you can enter on your system. We deliver two different types of Activation Keys: One, which is limited to 30 days and another, which is unlimited.

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. The only way to get an Activation Key is by registering on our registration site on the web.

http://www.visualextend.com

We are convinced that in today's world where things change dramatically fast, the shortened payback time of a mayor investment like VFX has been for us, the investment of both, our customers and our own has to be protected the best possible.

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

5.3. Setup the Visual FoxPro Environment for VFX

You must have Microsoft Visual FoxPro 8.0 properly installed, before you can start working with VFX 8.0.

Next, you should ensure that the VFX 8.0-Menu appears automatic each time, when you start your Visual FoxPro 8.0. We suggest the following way:

Let us assume that VFX is installed in folder C:\Program Files\VFX80.
Add this line in the file CONFIG.FPW in the VFP 8.0-folder:

```
command=do "C:\Program Files\VFX80\Builder\vfxmnu.app"
```

Adapt the path if necessary

NOTE: If you don't have a file named **CONFIG.FPW** just create an empty one with notepad.

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 *\VFX80\BUILDER*.

- **Searchpath:** *\VFX80\BUILDER* will be added to the Searchpath

At first start of VFP after VFX 8.0 installation, the VFX 8.0 Task Pane will be incorporated into VFP Task

Pane automatically.

Very Important: Make sure that you are 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 8.0 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.

5.4. Important about the Template Classes setting

On the Forms page of the Options dialog, you can define template classes for Formsets (not supported by VFX!) and Forms. If you set the *CDataFormPage* as your template class for new forms, all newly created forms will automatically be based on this class. However, be very careful: **This means that physically all references are mapped to this class library file, and not to your project specific class library files! We suggest not working with Template Classes settings on Form level at all, use following form wizards instead.**

5.5. Important about the Creation of new Forms using the VFX Form Wizard

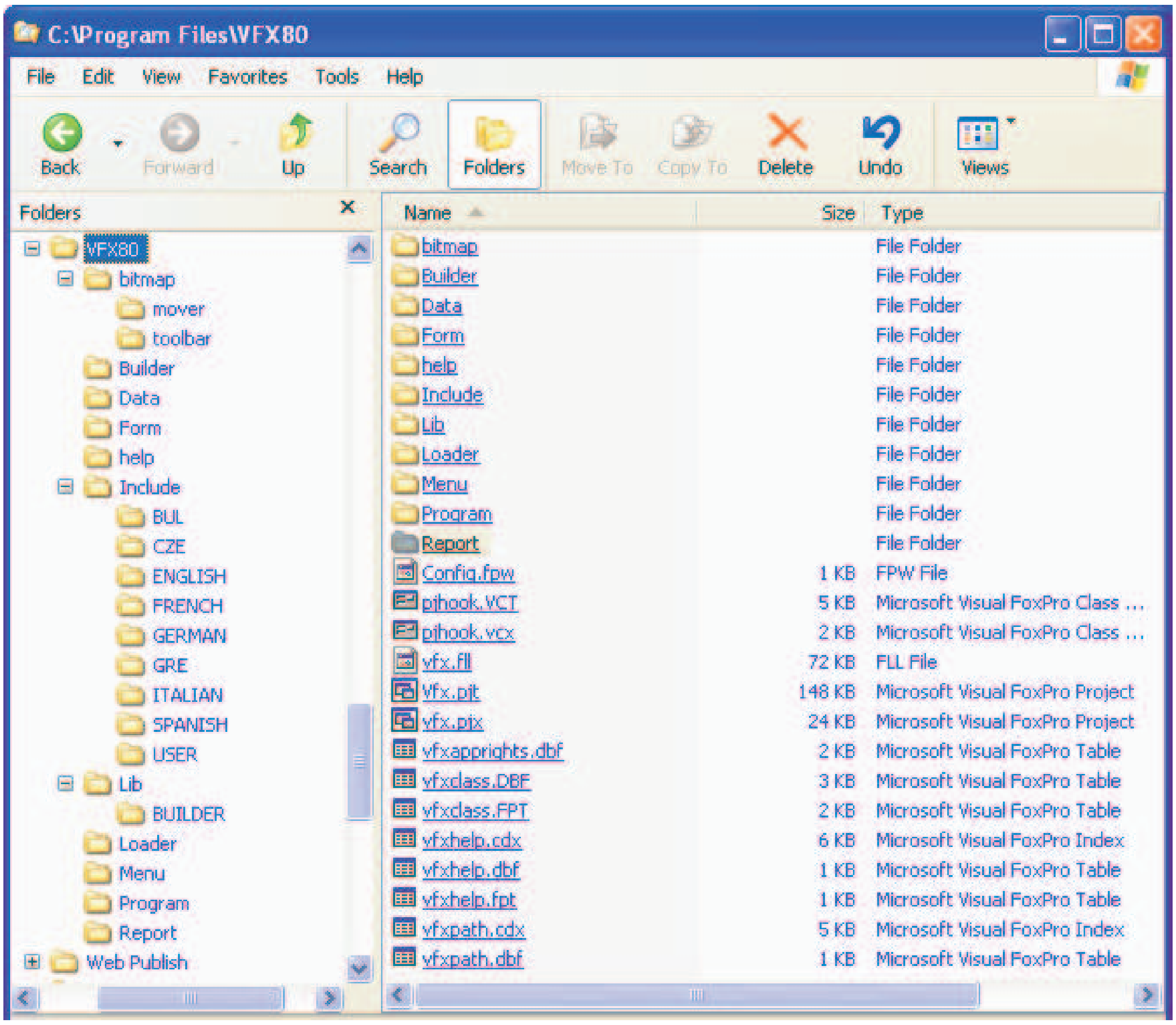
To create a new form, drag i.e. the class *CDataFormPage* and drop it on the new form. Since on a new form Visual FoxPro automatically creates by default a *FormSet1* and a *Form1*, you will have to delete them both in the *Form* menu! **If you are tired to create new forms this way, you should use the VFX Form Wizard, located in the VFX menu! This tool is probably the best reason not to use template form setting, as described above.**

5.6. Important for developers of multilingual applications

To allow multilingual application development, you must have the files for the *desired* language in the folders *\include* and *\menu*. All VFX forms already contain code in the LangSetup() method so that they show the labels, captions and tooltips in the chosen language. There are no language specific components in the VFX class libraries so they can be used without any changes in all languages.

5.7. Installation Files Overview

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



- VFX80
  - bitmap
    - mover
    - toolbar
  - Builder
  - Data
  - Form
  - help
  - Include
    - BUL
    - CZE
    - ENGLISH
    - FRENCH
    - GERMAN
    - GRE
    - ITALIAN
    - SPANISH
    - USER
  - Lib
    - BUILDER
  - Loader
  - Menu
  - Program
  - Report
- Web Publish

Name	Size	Type
bitmap		File Folder
Builder		File Folder
Data		File Folder
Form		File Folder
help		File Folder
Include		File Folder
Lib		File Folder
Loader		File Folder
Menu		File Folder
Program		File Folder
Report		File Folder
Config.fpw	1 KB	FPW File
pjhook.VCT	5 KB	Microsoft Visual FoxPro Class ...
pjhook.vcx	2 KB	Microsoft Visual FoxPro Class ...
vfx.fll	72 KB	FLL File
Vfx.pjt	148 KB	Microsoft Visual FoxPro Project
vfx.pjx	24 KB	Microsoft Visual FoxPro Project
vfxapprights.dbf	2 KB	Microsoft Visual FoxPro Table
vfxclass.DBF	3 KB	Microsoft Visual FoxPro Table
vfxclass.FPT	2 KB	Microsoft Visual FoxPro Table
vfxhelp.cdx	6 KB	Microsoft Visual FoxPro Index
vfxhelp.dbf	1 KB	Microsoft Visual FoxPro Table
vfxhelp.fpt	1 KB	Microsoft Visual FoxPro Table
vfxpath.cdx	5 KB	Microsoft Visual FoxPro Index
vfxpath.dbf	1 KB	Microsoft Visual FoxPro Table

The VFX home folder serves as the central location of all VFX components and is the base for all projects you create with the VFX Application Wizard (described later in this document).

NOTE: Do not work within this project directly, it is NOT intended to be worked with directly. Use the Application Wizard to create a new Project instead as described later in this documentation.

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

6.1. Objective

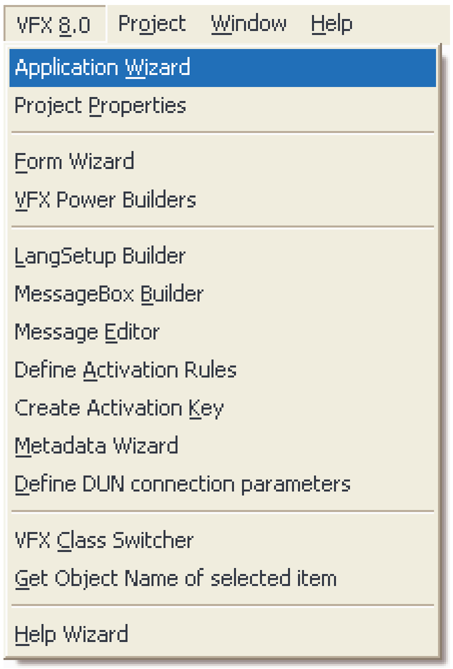
If you want to start a new project, you could set up the whole directory structure manually, copy all the needed support files, like the class library, some standard forms, some configuration files, bitmaps, and so on. That is where the VFX Application Wizard pays off: it does the whole set up of a new 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

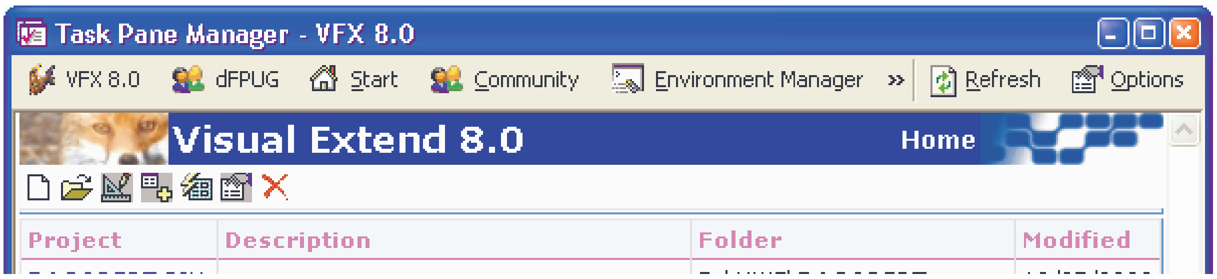
Close all forms and make sure you do not have open any class libraries of the VFX project. The best is to quit Visual FoxPro and restart it before you run the application Wizard.

6.3. The VFX - Application Wizard

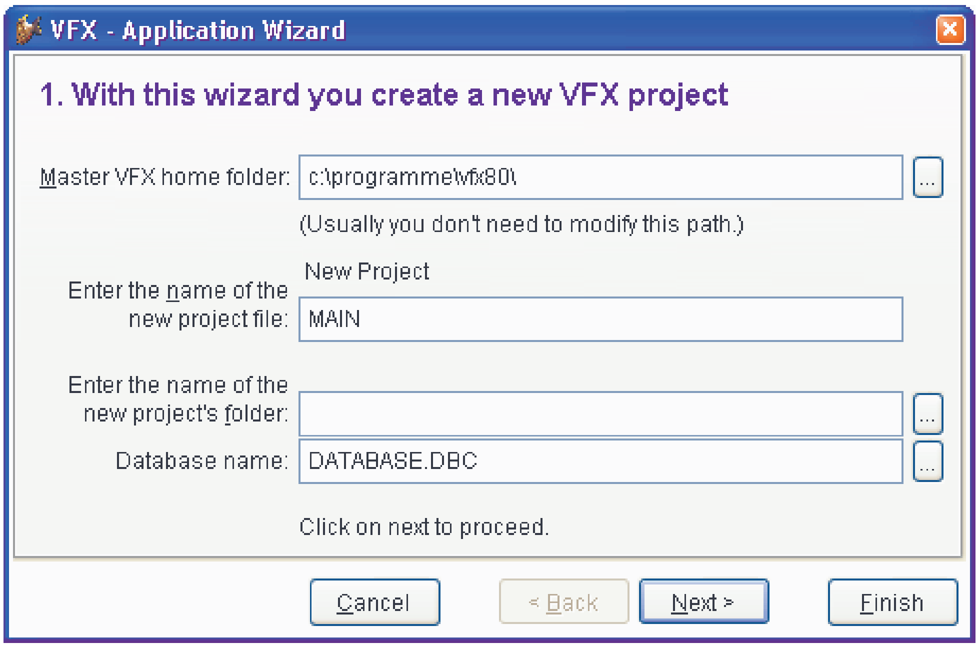
Select the option *Application Wizard* in the VFX 8.0-Menu.



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



The VFX-Application Wizard appears:


VFX - Application Wizard

1. With this wizard you create a new VFX project

Master VFX home folder: c:\programme\vfx80\

(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.

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

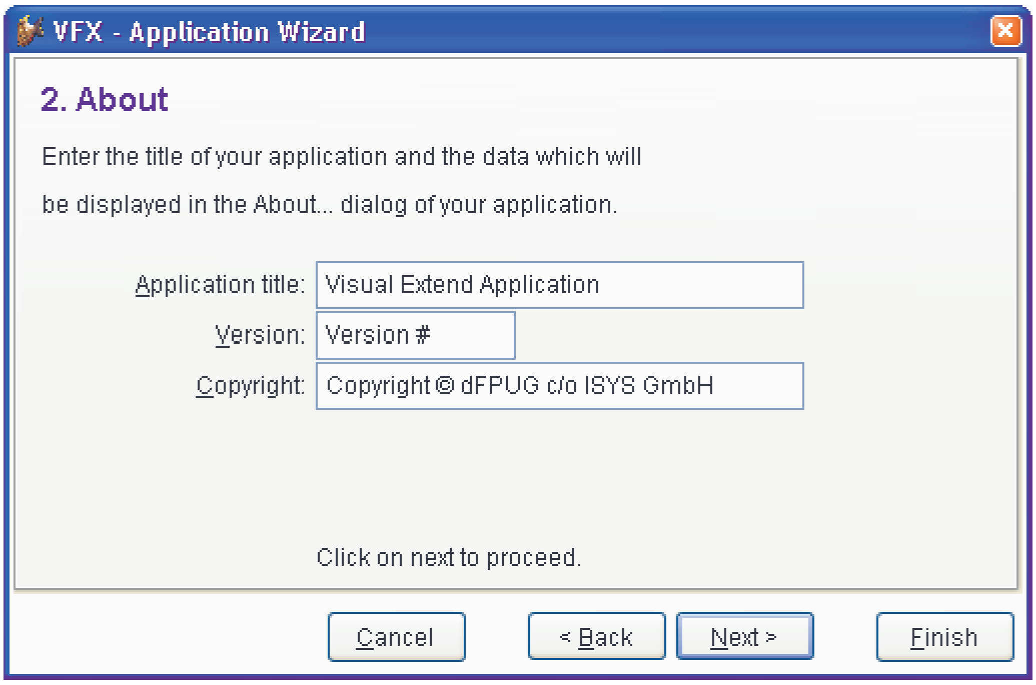
Master VFX home folder: Locate or type the VFX home directory where all VFX support files are located. Usually the default value is correct and you do not need to make any changes.

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.

Database Name. Enter the Name for your new Database Container (DBC). Type the name of your Database Container, without any path and file extensions.

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



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

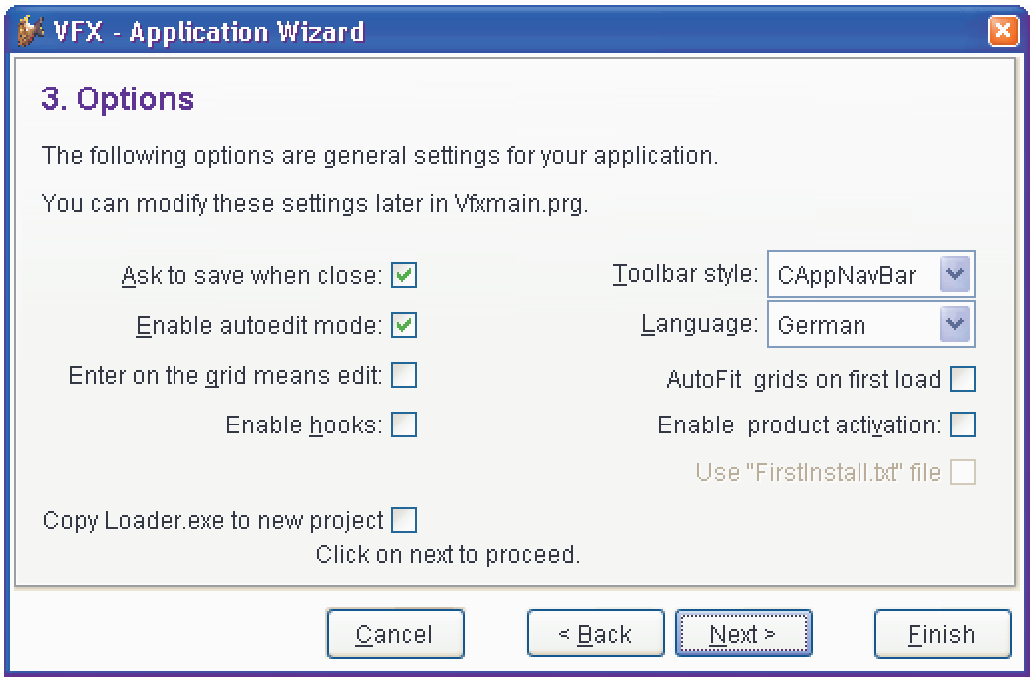
Click on next to proceed.

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:



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: ✓

Enable autoedit mode: ✓

Enter on the grid means edit: ☐

Enable hooks: ☐

Toolbar style: CAppNavBar

Language: German

AutoFit grids on first load ☐

Enable product activation: ☐

Use "FirstInstall.txt" file ☐

Copy Loader.exe to new project ☐

Click on next to proceed.

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.

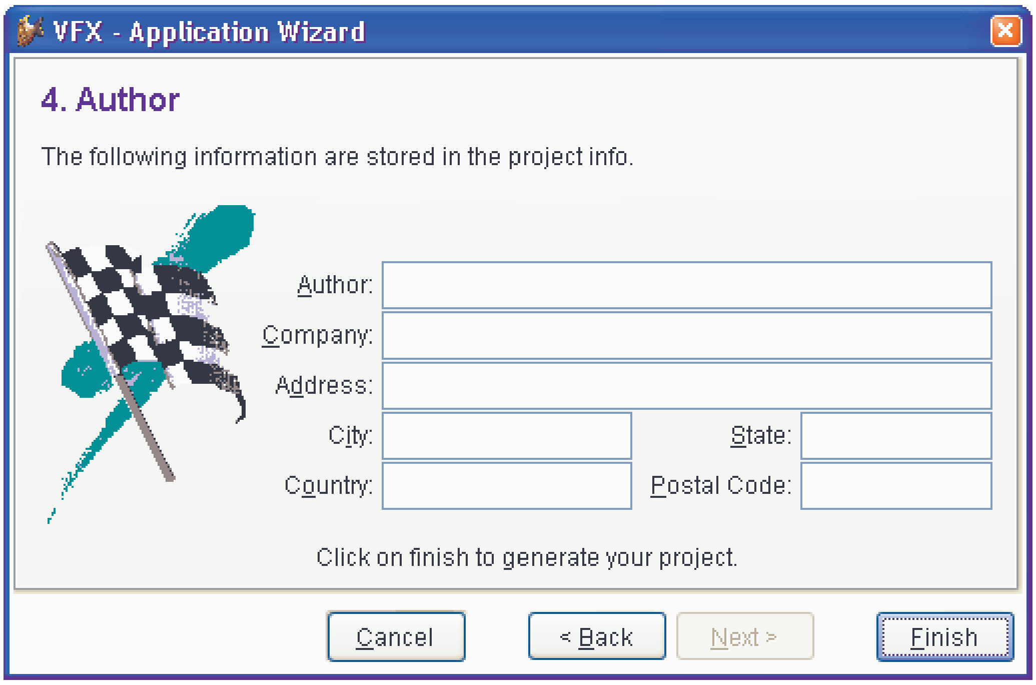
Toolbar style: Select which toolbar Style Class you want to use. *CAppNavBar* includes the record navigation and other editing controls in the main toolbar. *CAppToolbar* includes no record navigation and not all editing controls.

Language: Select the desired language for your new project. Currently you can choose from English, French, German, Italian, Spanish, Bulgarian, Czech and Greek.

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.

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.

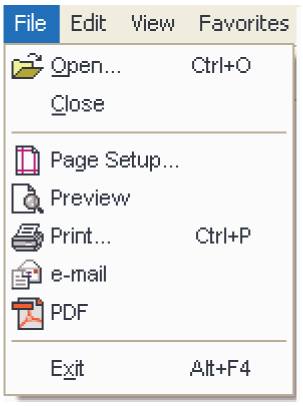
7. Discussion of the Generated VFX - Application

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 Access Control List, 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



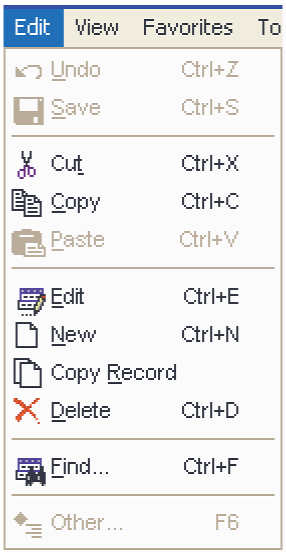
The complexity of the menus is reduced using the standard *File/Open* dialog. The user will open a form always through a common *Open Dialog* for which VFX makes a suggestion for the layout. By default, the Open Dialog is displayed in Windows-XP style docked to the left of the screen.

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 four selections appear and are only one click away from being selected again.

The *File/Exit* command also conforms the *Office Compatible* principles.

7.1.2. Edit Menu

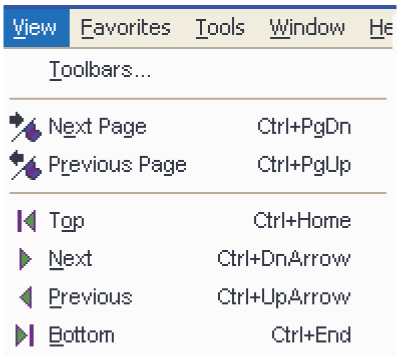


Here are placed all *Data Manipulation Functions*, which apply to the currently selected record as well as the possibility to call the *Find* and *Other Function* forms. Some menu options might be disabled, depending on the mode of the current form which can be either in

Edit/Insert mode (oForm.*nFormStatus* = 1 or 2) or
View Mode (oForm.*nFormStatus* = 0)

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

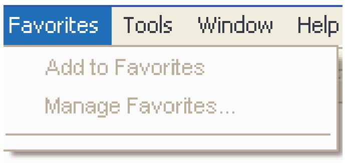
7.1.3. View Menu



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.

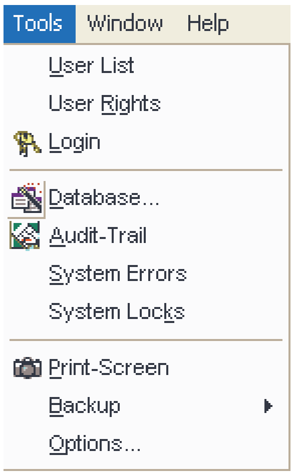
For detailed information, please refer to the *Discussion of the VFX Data Manipulation Form* chapter later in this Document.

7.1.4. Favorites Menu



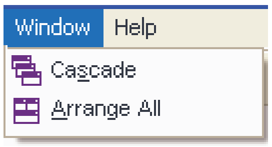
This is the VFX Favorites menu. The first option is to add the currently selected record to the favorite menu. The second is to manage the favorites. At the bottom all currently available favorites grouped by form are displayed as additional menu options at runtime.

7.1.5. Tools Menu



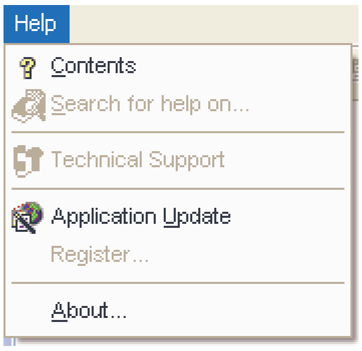
For further detailed information regarding the features described above, please refer to the chapter *User List*, *User Rights*, *Login*, *Database Tools*, *Audit-Trail*, *System Errors*, *System Locks*, *Print-Screen* and *Options Dialog* later in this documentation.

7.1.6. Windows Menu



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

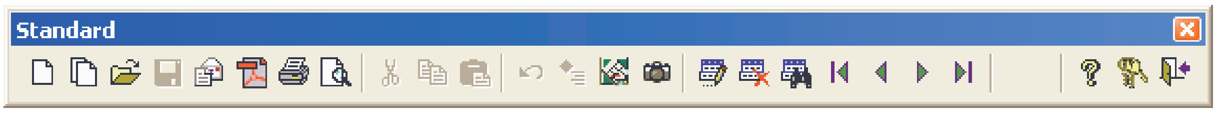
7.1.7. Help Menu



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.



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

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

Besides the standard toolbar, VFX also offers the possibility to define toolbars, which are associated to individual 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.

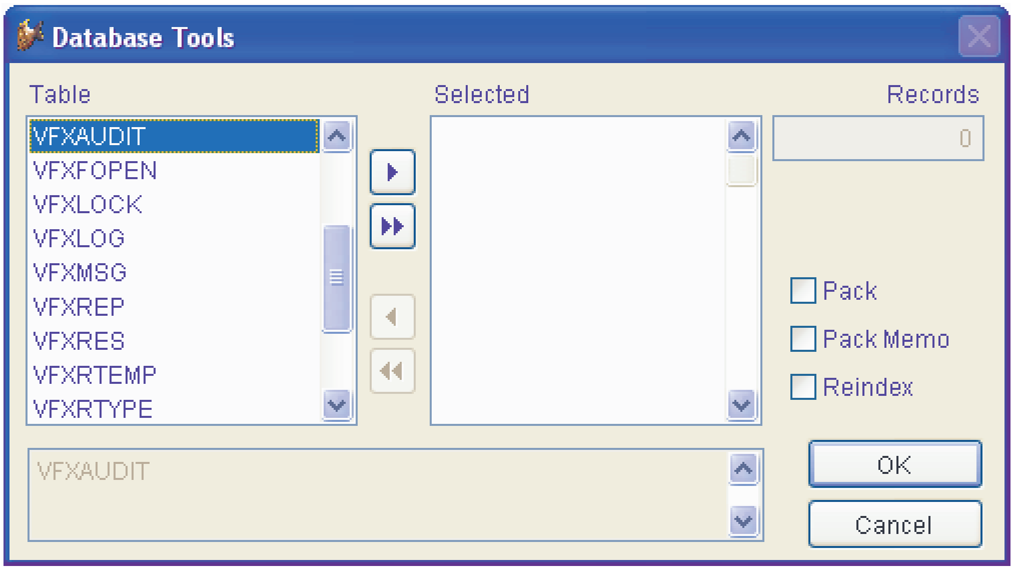
NOTE: For a detailed technical description about the usage of form-specific toolbars, please refer to the separate VFX Technical Documentation.

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 definitively not all, possible applications with their special needs. It is definitively 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. The Database Tools

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



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 of your application.

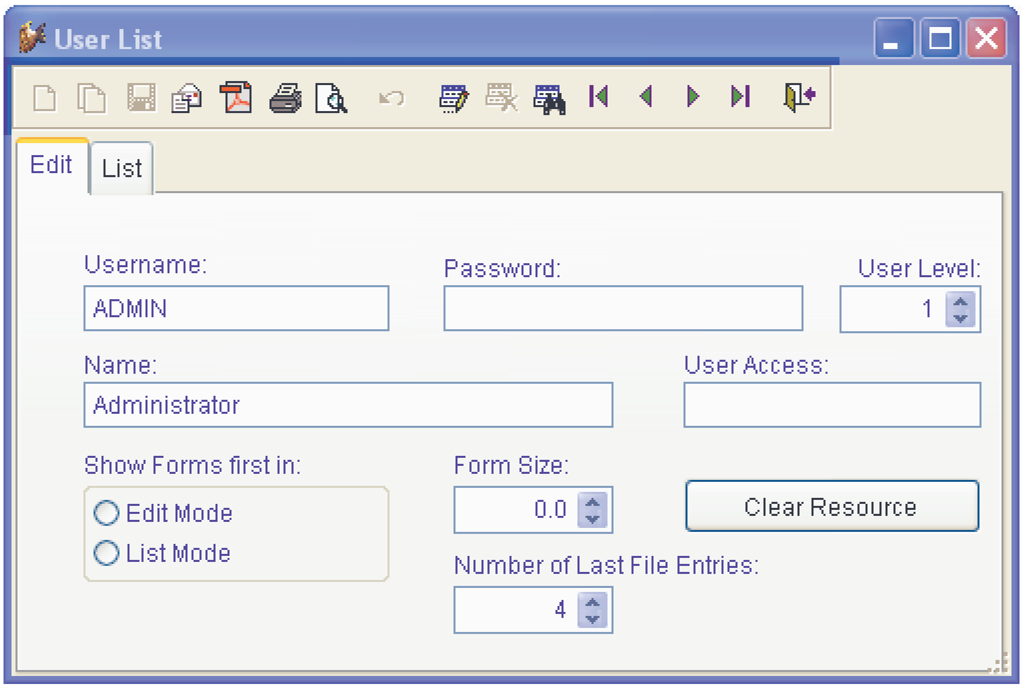
NOTE: If you like the above Mover Dialog, you will be happy to hear there is a VFX Mover Class, which allows you to easily integrate Mover Dialogs into your own applications!

7.3. The User List

In every multi-user application, you must have a user list. First, you need to define who has access to your application, which is the username and password and what the security level per user is. Another very important function of the user list is the possibility to permanently store personal settings on a per user basis.

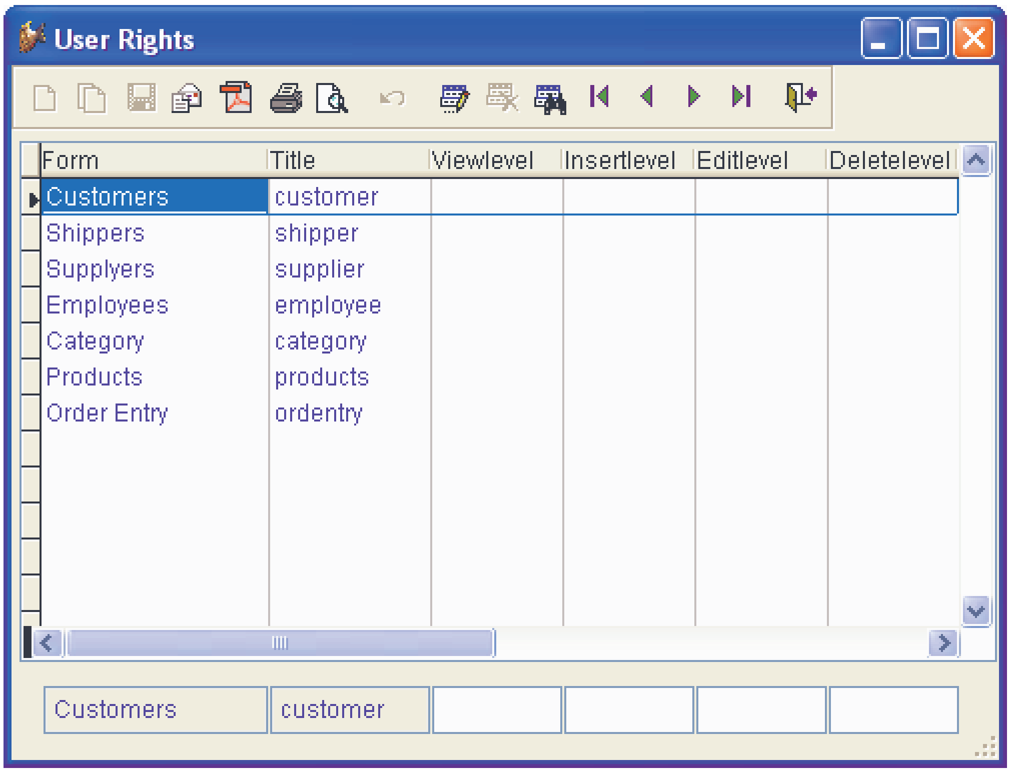
The file in which user-specific information is stored, is the free table *VFXUSR.DBF/CDX*. If you want to take advantage of features like long field names, you have to place this table into the database container.

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



Username: ADMIN
Password:
User Level: 1
Name: Administrator
User Access:
Form Size: 0.0
Number of Last File Entries: 4

Users can clear their VFX resource file, 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 file click on the command button called *Clear Resource*.



Form	Title	Viewlevel	Insertlevel	Editlevel	Deletelevel
Customers	customer				
Shippers	shipper				
Supplyers	supplier				
Employees	employee				
Category	category				
Products	products				
Order Entry	ordentry				

Customers	customer				

The user access rights are defined by the user security level. 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.

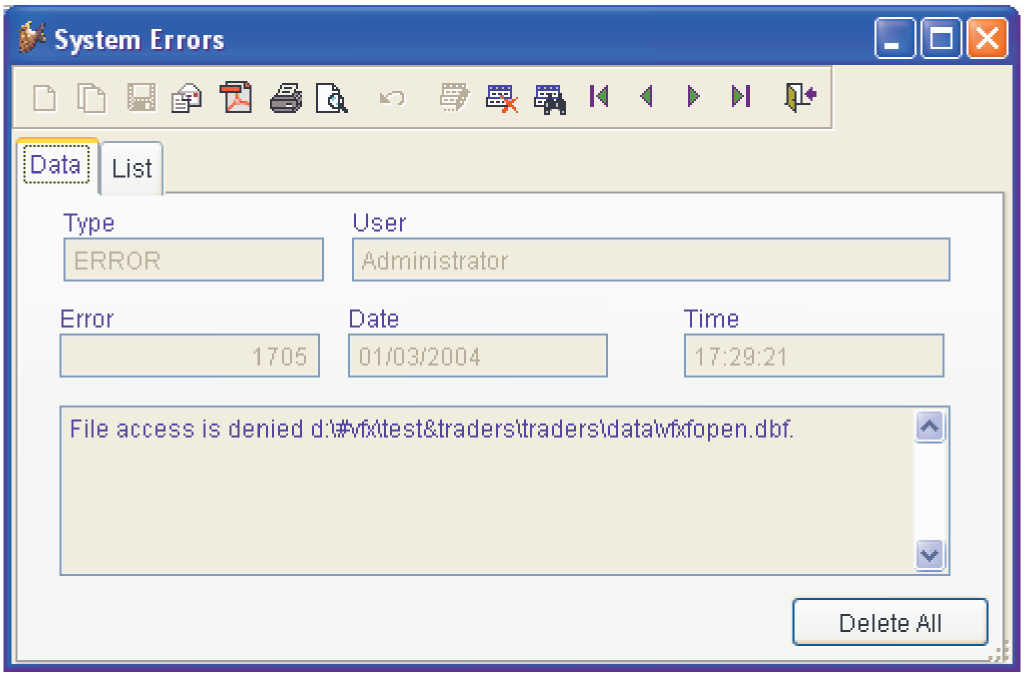
NOTE: Users cannot view or alter other user accounts with a higher security level than their own. Security levels starts with 1 (Administrator) and ends with 99 (lowest security level). Additionally, you can define an access string for further customization of your security needs. For additional security 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.

7.4. 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.



Type: ERROR
User Administrator
Error: 1705
Date: 01/03/2004
Time: 17:29:21

File access is denied d:\#vfx\test&traders\traders\data\vfxfopen.dbf.

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.

7.5. The System Locks

In heavily used multi-user applications, a message as *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.

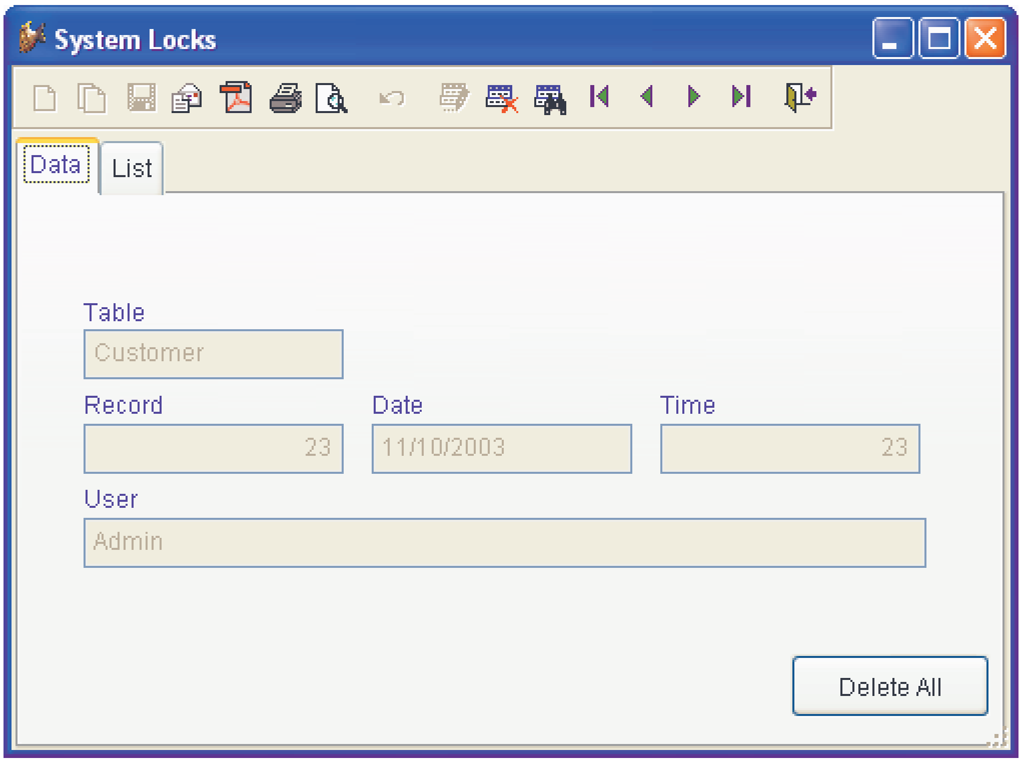


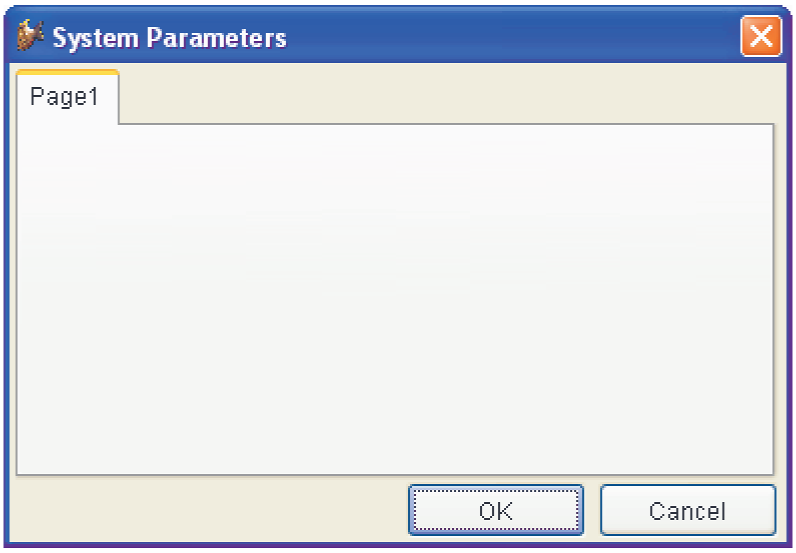
Table		
Customer		
Record	Date	Time
23	11/10/2003	23
User		
Admin		

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.

7.6. Options

VFX offers a table called *VFXSYS*, which stores application-specific settings.



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

VFX Application Wizard creates a *VFXSYS* Form, which is reedy to use. This form inherits from the class *CSystemDialog*. All you have to do is to create your fields in the *VFXSYS* table, put the controls on the above dialog with the Control Source Property pointing to the variable with the prefix *gs_*.

VFX creates for every field in the *VFXSYS* table, a public variable with the prefix *gs_* and handles automatically the save & restore of these values.

In contrast to the application-specific settings, you may also have settings, which are user-specific. Like with the application specific settings, VFX creates for every field in the *VFXUSR* table a public variable with the prefix *gu_* and handles automatically the save & restore for these values.

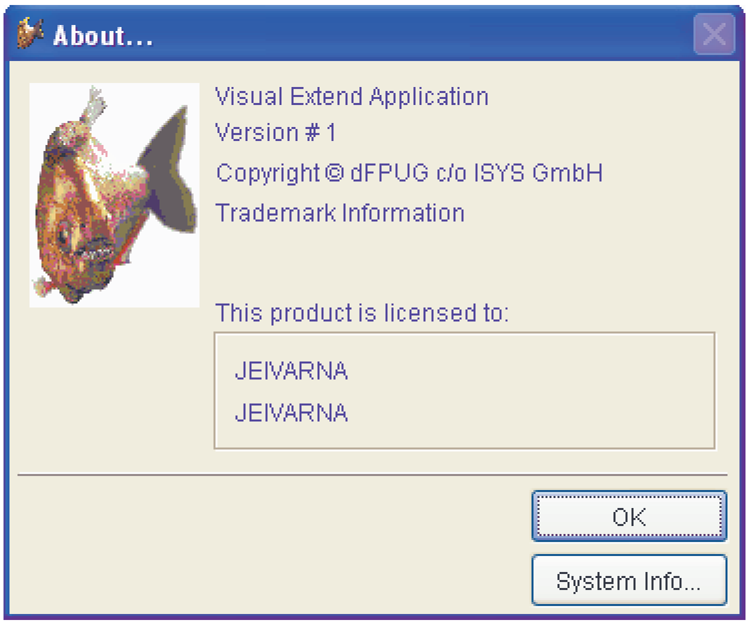
Assume you have a field called *TEST* in the *VFXUSR* table. In that case, you will see a public variable called *gu_test*, which will pick up the value from the field *TEST*, whenever the user logs on VFX writes the content of the public variable *gu_test* back to the table *VFXUSR* whenever the user logs off.

In this way, it is easy to store and retrieve user-specific settings. The only thing you have to do to support user-specific settings, is add a field in the table *VFXUSR*. It is so easy. Try it out!

7.7. The About Dialog

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

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


About...

Visual Extend Application
Version # 1
Copyright © dFPUG c/o ISYS GmbH
Trademark Information

This product is licensed to:

JEIVARNA
JEIVARNA

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 8.00 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 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

8.1. Objective

Creating 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 big 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. In addition, adding pages to the pageframe or changing the grid columns is very easy with the reentrance feature of the VFX form builders.

8.2. Result

Please refer to the Chapter *Discussion of the VFX Standard Data Manipulation Form* later in this Document to get an idea about the User Interface of the standard data manipulation forms created using VFX.

8.3. Preparation

8.3.1. Setup the Database container

First, you need to setup the database container of your application. Define your tables, fields and indexes.

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.

8.3.2. Creating a new Form

Start the VFX-form Wizard from the VFX-Menu. Give the forma name and choose the class on which this form will be based. It is possible to choose among VFX-Forms classes or user-defined forms classes. User-defined forms classes must be inherited from VFX-Forms classes. The generated form will be saved in Form folder under the current project’s folder and the Form designer will be opened.

8.3.3. Setup the Form Data Environment

Set the data environment for the created form. VFX form builder automatically retrieves the information from the data environment and uses it for the generation process.

8.4. The VFX Form Builder

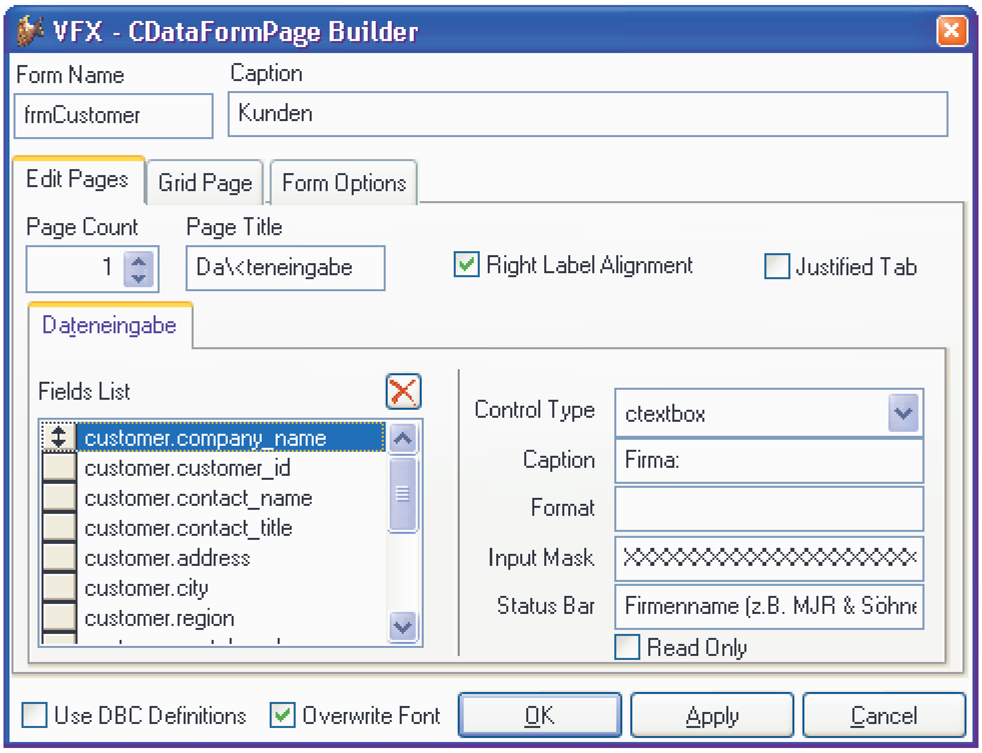
8.4.1. Call the VFX Form Builder

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

NOTE: If you receive a message indicating that there is no builder available for the currently selected object, or the standard Visual FoxPro Builder appears, make sure that you followed the installation instructions in this document and that you selected the form object and not another object on the Form. **One common mistake is that you select the PageFrame Container instead the form itself. Check the object in the Property Sheet if you are not sure whether you selected the form object.**

The VFX Form Builder loads and shows a user friendly Dialog:

8.4.2. The VFX Form Builder User Interface


VFX - CDataFormPage Builder

Form Name: frmCustomer
Caption: Kunden

Page Count: 1
Page Title: Da\<teneingabe
[✓] Right Label Alignment
[ ] Justified Tab

Fields List:
- customer.company_name
- customer.customer_id
- customer.contact_name
- customer.contact_title
- customer.address
- customer.city
- customer.region

Control Type: ctextbox
Caption: Firma:
Format:
Input Mask: XXXXXXXXXXXXXXXXXXXXXXXXX
Status Bar: Firmenname (z.B. MJR & Söhne
[ ] Read Only

[ ] Use DBC Definitions
[✓] Overwrite Font

The VFX Form Builder has an intuitive User Interface.

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 will see it displayed in the form builder’s caption. If your form has a variable caption, depending how the Form will be called, do not worry too much about this caption, just use a more or less descriptive caption in that case.

The VFX - CDataFormPage builder has a pageframe with three pages named *Edit Pages*, *Grid Page* and *Form Options*. On the *Edit Pages*, you define the pageframe, which will be used to display and edit the fields you selected. On *the Grid Page*, you define the *Data Grid* and on the *Form Options* Page, you set different form options.

The following options are available on the Edit Pages:

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, which offers all fields currently available in the data environment.

Control Type. Define for all selected fields, which control type you want to use. Following control types are available:

Control Type	*Description*	*VFX Class Library*
<Default>	The class that you have chosen as default display class, in the database container.	
CTextBox	Normal Text Box	VFXOBJ.VCX
CKeyField	Textbox for editing of Identification Key	VFXDBOBJ.VCX
CFixField	Textbox for editing field, which is linked to a master Table. Used for forms where the form has been called from a master form, receiving a fix value from the master. I.e. Orders for a Customer, in that case, the Customer Field would be a FixField, because it will not be accessible, in the case the form has been called from the Customer Form.	VFXDBOBJ.VCX
CPickField	Field to enter a value, which will be validated against a table or view including description and other automatic information fetching.	VFXDBOBJ.VCX
CEditBox	Edit Box for Memo Fields or other large character fields.	VFXOBJ.VCX
CComboBox	Combobox.	VFXOBJ.VCX
CListBox	List box.	VFXOBJ.VCX
CCheckBox	Checkbox for logical fields.	VFXOBJ.VCX
COptionGroup	Option group.	VFXOBJ.VCX
CSpinner	Spinner Control for numerical fields.	VFXOBJ.VCX

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.

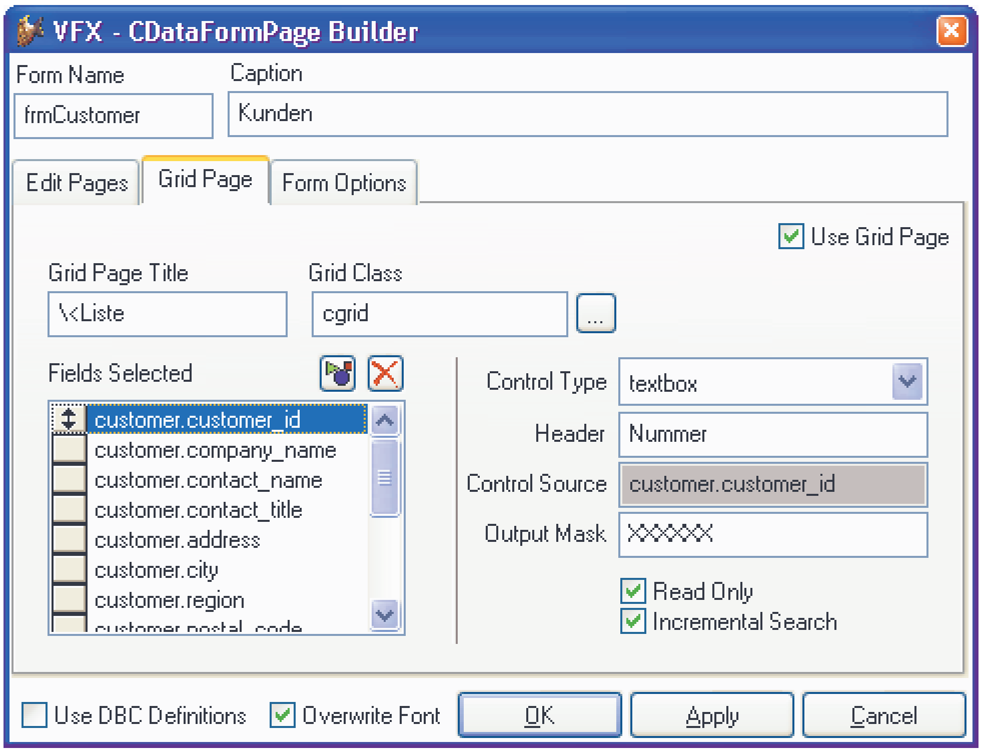
Format. Format property for the selected field. The default will be read from the database container.

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.

The following options are available on the Grid Page:


VFX - CDataFormPage Builder

Form Name: frmCustomer
Caption: Kunden

[✓] Use Grid Page

Grid Page Title: \<Liste
Grid Class: cgrid

Fields Selected:
customer.customer_id
customer.company_name
customer.contact_name
customer.contact_title
customer.address
customer.city
customer.region
customer.postal_code

Control Type: textbox
Header: Nummer
Control Source: customer.customer_id
Output Mask: XXXXXX
[✓] Read Only
[✓] Incremental Search

[ ] Use DBC Definitions
[✓] Overwrite Font

Use Grid Page. Check this checkbox if you want a 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. Following control types are available (For performance reasons we only offer VFP base classes for the grid):

Control Type	***Description***	***VFP Base Class***
TextBox	Text Box (Default)	TEXTBOX
Editbox	Editbox	EDITBOX
Combobox	Combobox	COMBOBOX
Checkbox	Checkbox	CHECKBOX

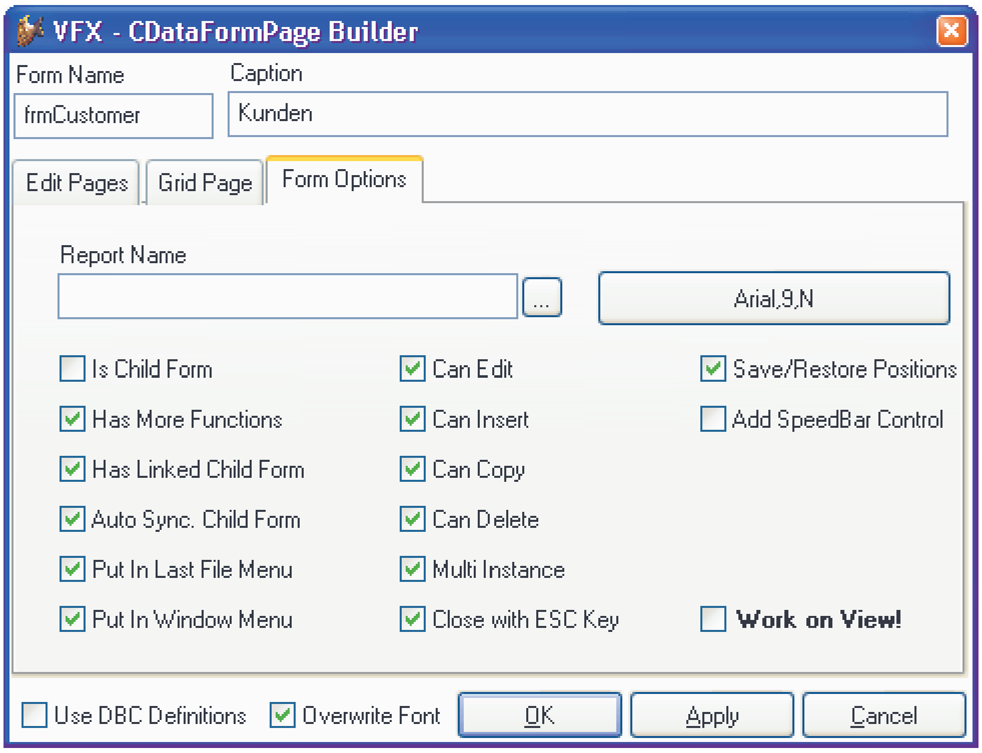
Header. Captions for the column headers of your grid. VFX Form Builder will automatically take the captions from your database container.

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

Incremental Search. Check 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 CDX index file 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)

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


VFX - CDataFormPage Builder

Form Name: frmCustomer
Caption: Kunden

Edit Pages | Grid Page | Form Options

Report Name:
Arial,9,N

Is Child Form	[✓ Can Edit]	[✓ Save/Restore Positions]
[✓ Has More Functions]	[✓ Can Insert]	Add SpeedBar Control
[✓ Has Linked Child Form]	[✓ Can Copy]	
[✓ Auto Sync. Child Form]	[✓ Can Delete]	
[✓ Put In Last File Menu]	[✓ Multi Instance]	
[✓ Put In Window Menu]	[✓ Close with ESC Key]	**Work on View!**

Use DBC Definitions [✓ Overwrite Font]

Report Name. Here you can select a report name. Whenever the user selects *print* or *preview*, this report will be selected and printed resp. 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 do not mix this up with the later described One-To-Many Form where you can have the master and the child table processed **on the same form**. Here we are talking about this scenario: Form1 -> calls Form2, whereas Form1 could be the master 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 master table in Form1.

I.e. if 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 *Advanced Form Features using the VFX Form 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 will set it up as a *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 Customer X*.

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.

For further details, refer to the topic *Advanced Form Features using the VFX Form Builder* later in this document.

Has Linked Child Form. If the form you are currently creating will have child forms which should be dynamically linked to this master 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 *lAutosynchChildform* 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 *lPutinLastFile*, 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 *lPutinWindowmenu*, 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 *lCanEdit*, which defines whether the user can edit records in the current form, or not.

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

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

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

Multi Instance. This sets the form property *lMultiInstance*. 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 to use multiinstantiation is to set the form to private data session, which is the default of all VFX forms.

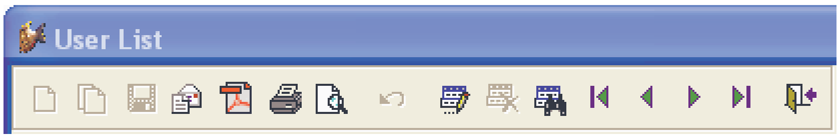
However sometimes it is convenient to disable the feature of multiinstantiation. That is why we created the property *lMultiInstance*. Just set this property to .F. and the form can only be called once.

However sometimes it is convenient to disable the feature of multiinstantiation. That is why we created the property *lMultiInstance*. Just set this property to true and the form can only be called once.

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

Save/Restore positions. This sets the form property *lSavePosition*, 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:



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 is best to use the Form Builder whenever you want to add a new field to your form.

Another big 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“.

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.

8.5. The VFX Grid Builder

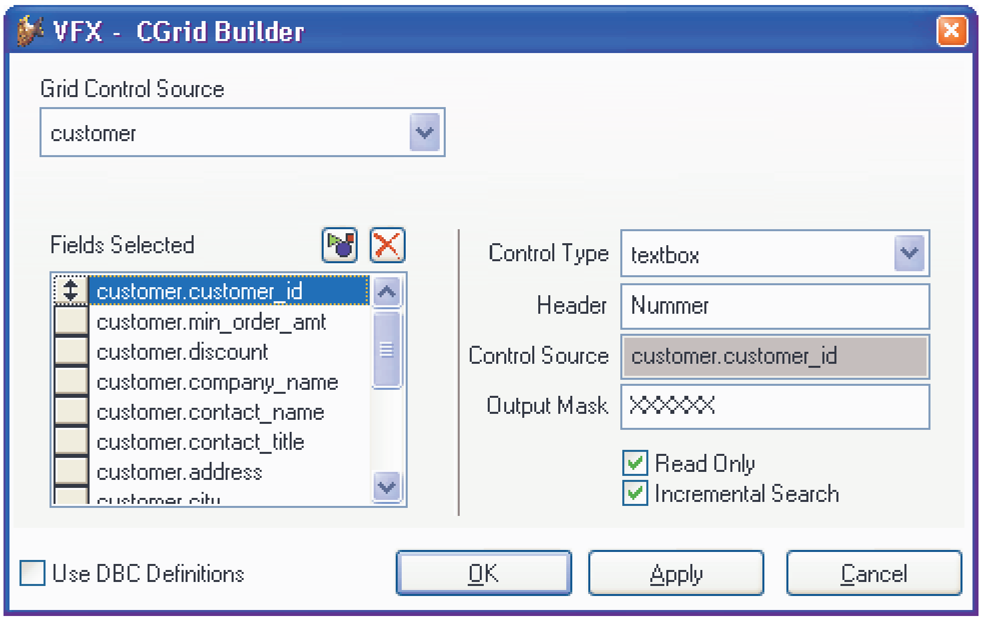
8.5.1. Call the VFX Grid Builder

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 rightclick 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:

8.5.2. The VFX Grid Builder User Interface


VFX - CGrid Builder

Grid Control Source: customer

Fields Selected:
- customer.customer_id
- customer.min_order_amt
- customer.discount
- customer.company_name
- customer.contact_name
- customer.contact_title
- customer.address
- customer.city

Control Type: textbox
Header: Nummer
Control Source: customer.customer_id
Output Mask: XXXXXX
[✓] Read Only
[✓] Incremental Search

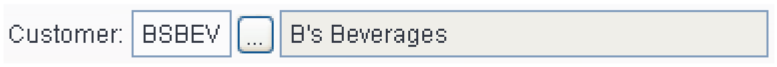
[ ] Use DBC Definitions

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 Form Builder*.

8.6. The VFX Picklist Builder

8.6.1. Result

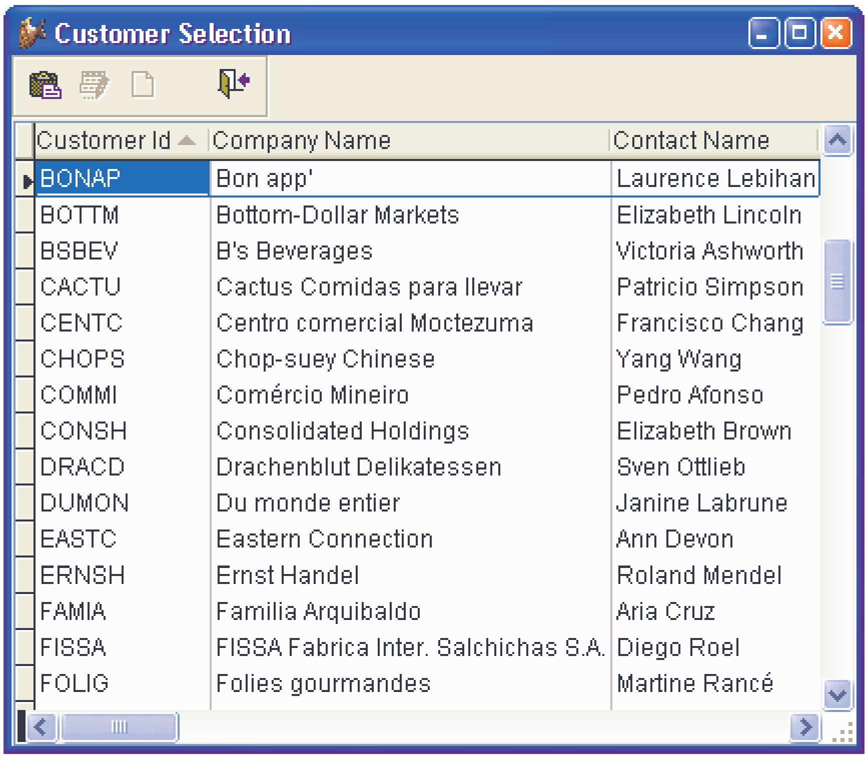
If you use a Picklist Control on a form, it might look similar to this:



Customer:	BSBEV	B's Beverages

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



Customer Id	Company Name	Contact Name
BONAP	Bon app'	Laurence Lebihan
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln
BSBEV	B's Beverages	Victoria Ashworth
CACTU	Cactus Comidas para llevar	Patricio Simpson
CENTC	Centro comercial Moctezuma	Francisco Chang
CHOPS	Chop-suey Chinese	Yang Wang
COMMI	Comércio Mineiro	Pedro Afonso
CONSH	Consolidated Holdings	Elizabeth Brown
DRACD	Drachenblut Delikatessen	Sven Ottlieb
DUMON	Du monde entier	Janine Labrune
EASTC	Eastern Connection	Ann Devon
ERNSH	Ernst Handel	Roland Mendel
FAMIA	Familia Arquibaldo	Aria Cruz
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel
FOLIG	Folies gourmandes	Martine Rancé

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 of position and grid 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 *Maintenance* 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.

8.6.2. Call the VFX Picklist Builder

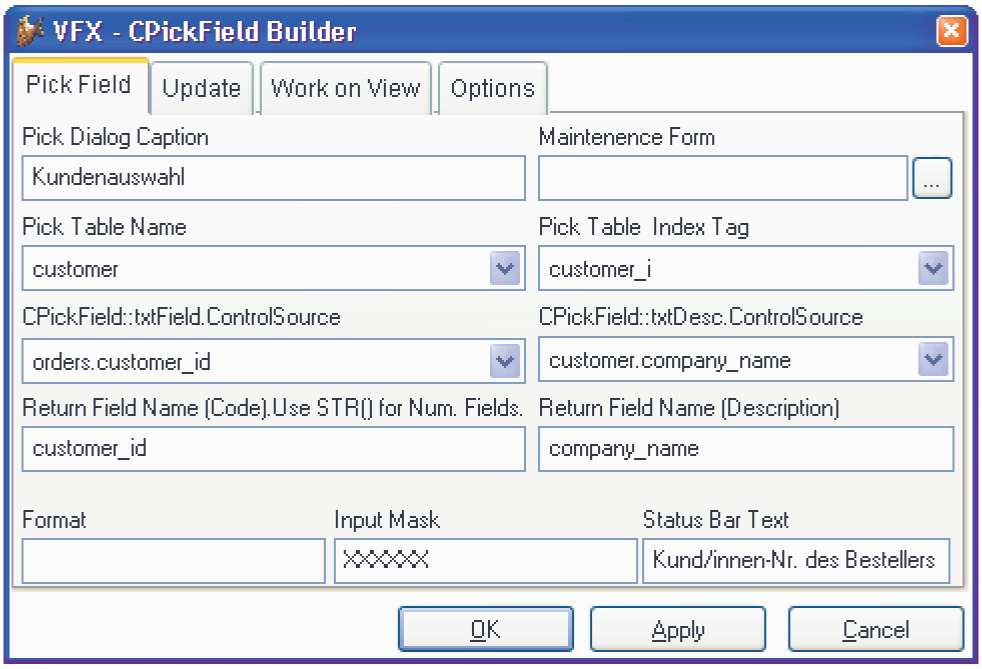
One control, which you might use quite a lot, is the Picklist. This container control offers you an easy way to add textboxes, with the ability to validate the user entry against a table or view, as well as providing an easy way to call a picklist form in which the user can locate and pick a desired record with its value. Since this picklist class has user properties, which must be defined, a VFX Picklist Builder helps you to generate the desired picklist controls easily. 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, rightclick 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+Shift+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 presents the following dialog:

8.6.3. The VFX Picklist Builder User Interface



Pick Dialog Caption: Kundenauswahl
Maintenence Form:
Pick Table Name: customer
Pick Table Index Tag: customer_i
CPickField::txtField.ControlSource: orders.customer_id
CPickField::txtDesc.ControlSource: customer.company_name
Return Field Name (Code).Use STR() for Num. Fields.: customer_id
Return Field Name (Description): company_name
Format:
Input Mask: XXXXXX
Status Bar Text: Kund/innen-Nr. des Bestellers

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

On the page *Pick Field*, the following options are available:

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 does not 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 maintenance 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 picktable 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 picktable 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.

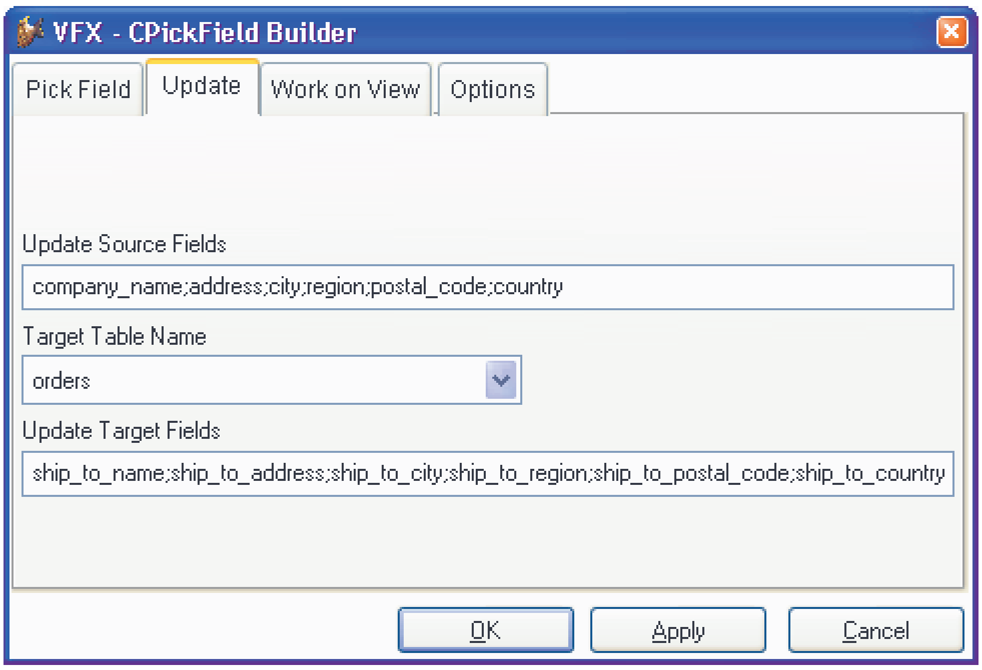
Status Bar Text. The VFX Pickfield Builder takes this property from the database container.

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 pickfield builder dialog.

Cancel. Cancels the VFX Pickfield Builder process. Any selections and entries will be lost.

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



Update Source Fields
company_name;address;city;region;postal_code;country
Target Table Name
orders
Update Target Fields
ship_to_name;ship_to_address;ship_to_city;ship_to_region;ship_to_postal_code;ship_to_country

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.

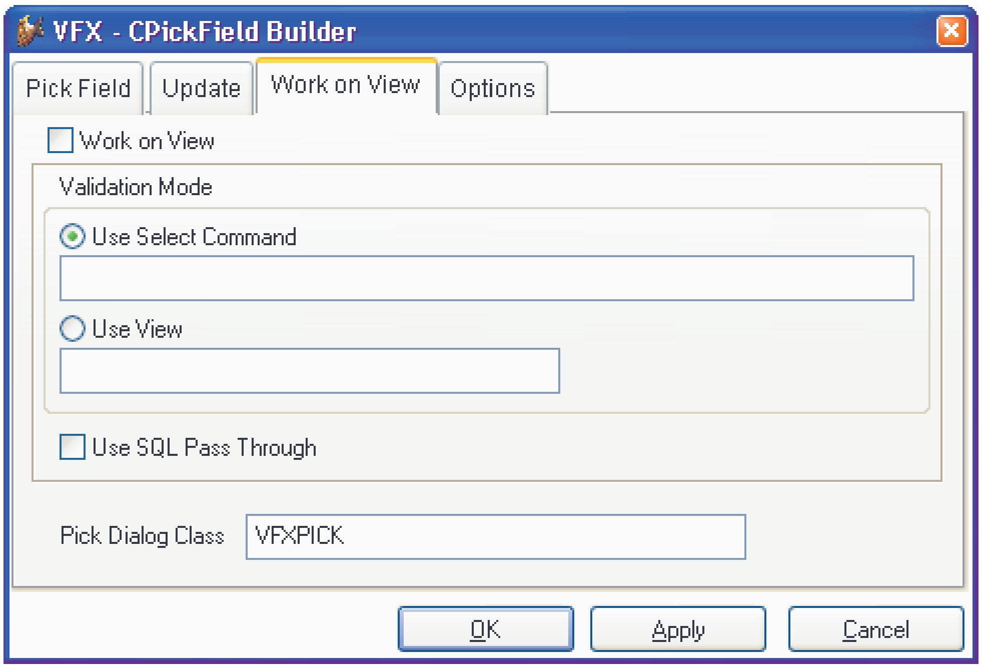
Update Target Fields: Choose the fields for updating. When you enter more than one value, you must separate them with semicolon.

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 pickfield builder dialog.

Cancel. Cancels the VFX Pickfield Builder process. Any selections and entries will be lost.

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



[ ] Work on View

Validation Mode

[✓ Use Select Command]

Use View

[ ] Use SQL Pass Through

Pick Dialog Class: VFXPICK

Work on View. If the datasource that you are picking from is a view, check 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 through SQL Pass Through will be sent to the remote datasource.

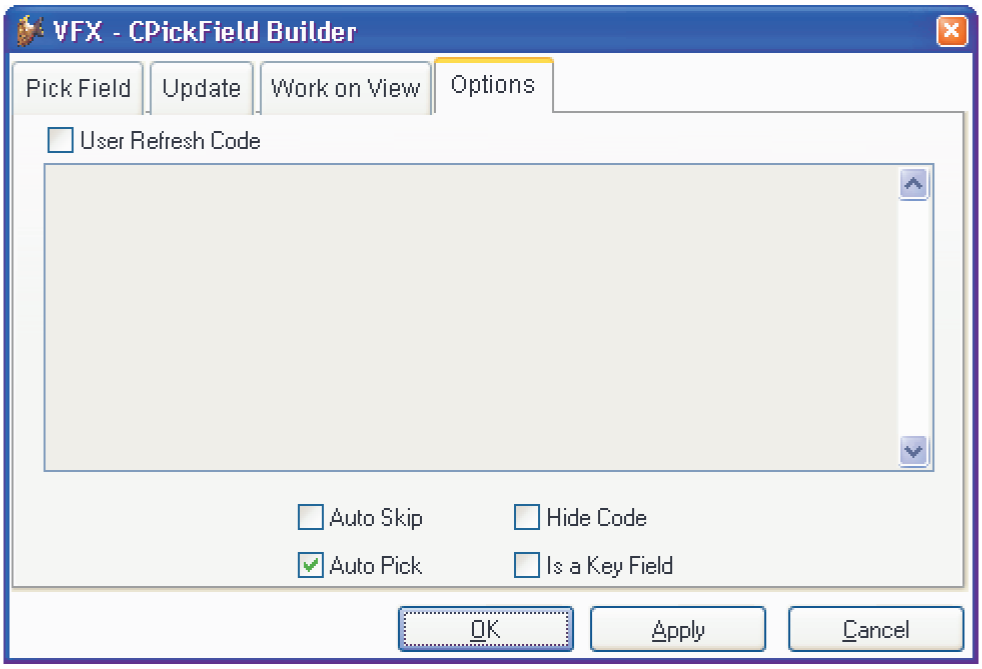
Pick Dialog Class: Here can be used your user-defined class for the selection list control. Please note that this class must be inherited from the CPickField class.

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 pickfield builder dialog.

Cancel. Cancels the VFX Pickfield Builder process. Any selections and entries will be lost.

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



[ ] User Refresh Code

[ ] Auto Skip
[✓] Auto Pick
[ ] Hide Code
[ ] Is a Key Field

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 true.

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 true.

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 *lHideCode* 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 pickfield builder dialog.

Cancel. Cancels the VFX Pickfield Builder process. Any selections and entries will be lost.

8.6.4. Test and refine your Form

Run your application, select your newly created form in the *File Open* dialog and start it with mouse click. Test it and check where your form needs further enhancements.

8.6.5. Next step

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*.

8.7. 1:n Form

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, full featured standard data manipulation form functionality, together with a grid showing the child records for the currently selected master record. VFX allows you also to have multiple children to one Master in a tab dialog. If you have many input Fields for a child table, you can also have a multipage tab dialog for the child data. This allows you to cover many 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 Master and Child Tables are linked. Let us look at a simple example:

8.7.1. Result

Please refer to the chapter *Discussion of the VFX One-To-Many Data Manipulation Form* later in this document to get an idea about the user interface of one-to-many forms created, using VFX.

8.7.2. Creating a new form

Start the VFX-form Wizard from the VFX-Menu and create a form, based on the class *cOneToMany*.

8.7.3. Setup the Database container

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 do not 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 master record and multiple child records. A good example for a master child relation is the Order (Master) and Items (Child) situation, of any order and invoicing system.

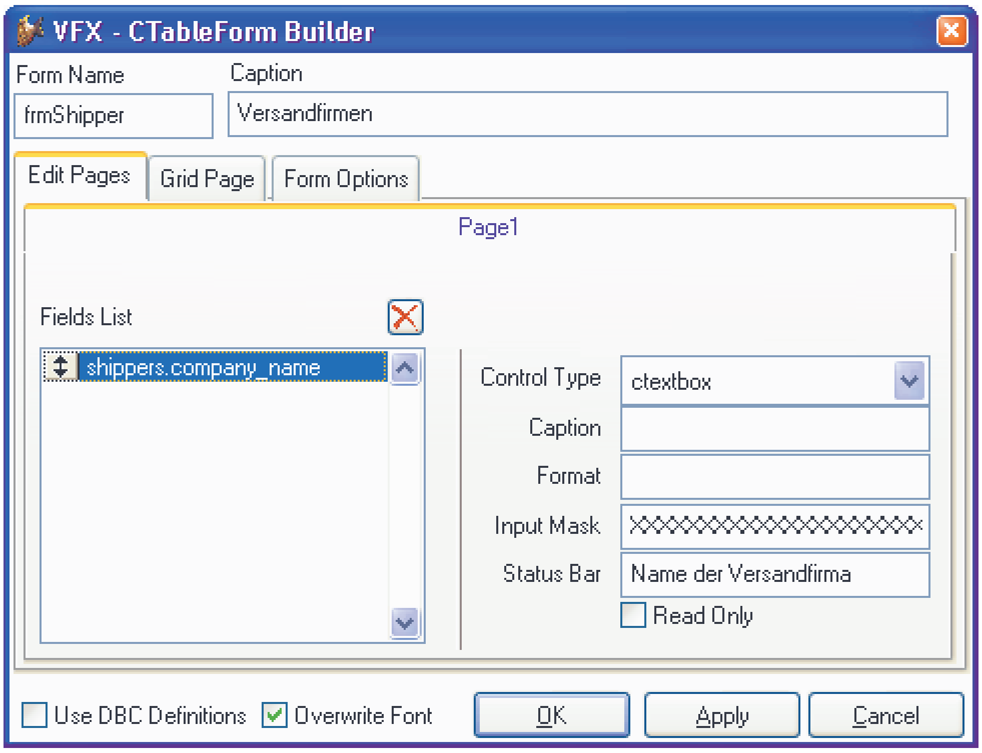
NOTE: If you do not want to assure the referential integrity (RI) manually using VFX methods like *OnPostDelete()*, it is a good practice, to generate the RI code in the database container before you generate one-to-many forms. If you do not do this, you will have to write manually the code for the deletion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

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 master table to the child table (in the case of multiple child tables which all depend from the same master table you might have more than one child table, each linked with the master table through a relation) you can generate one-to-many forms as easy as standard VFX data manipulation forms.

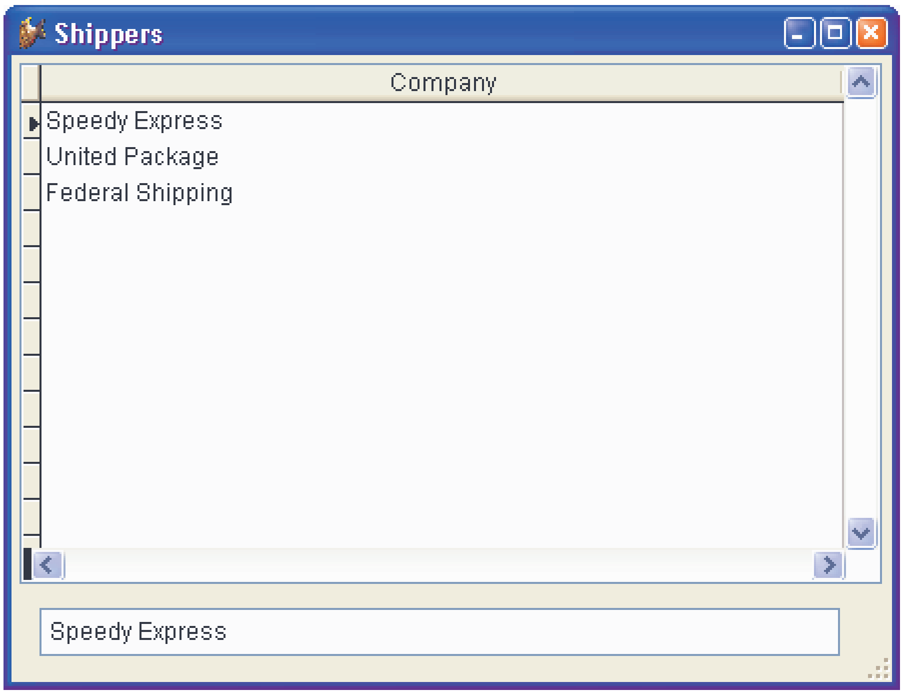
IMPORTANT: Remember to define the initially selected alias, order property, and the one-to-many relation from the master to the child in the data environment, otherwise, your form may not run as you expect!

8.8. The VFX – CTableForm Builder



Form Name: frmShipper
Caption: Versandfirmen
Fields List: shippers.company_name
Control Type: ctextbox
Caption:
Format:
Input Mask: XXXXXXXXXXXXXXXXXXXXXX
Status Bar: Name der Versandfirma
☐ Read Only
☐ Use DBC Definitions
☑ Overwrite Font

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. It is suitable therefore in particular for forms with only few input fields.



Company
Speedy Express
United Package
Federal Shipping

Speedy Express

8.9. The VFX One-To-Many Form Builder

8.9.1. Call the VFX One-To-Many Form Builder

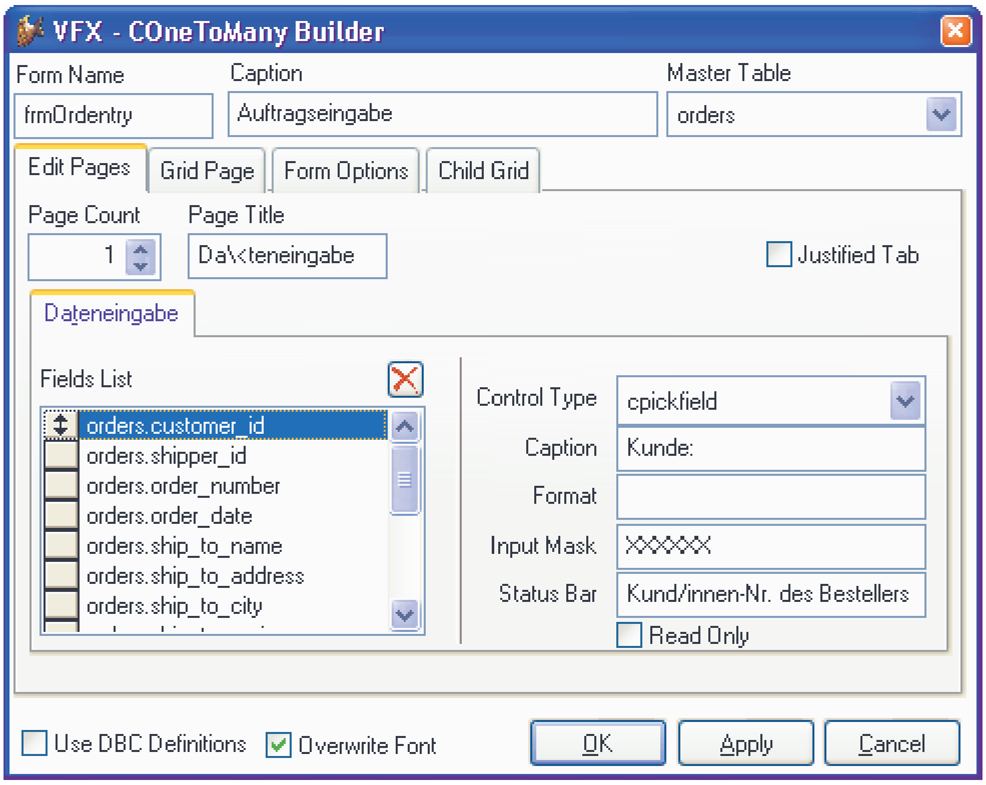
To call the VFX One-To-Many Form Builder, put the mouse on the white background of the new

COneToMany based form, click the right mouse button and select Builder.

NOTE: If you receive a message indicating that there is no builder available for the currently selected object, or if the standard Visual FoxPro Builder appears, make sure that you that you selected the form object and not another object on the form. One common mistake is that you select the PageFrame container instead the form itself. Check the object in the property sheet if you are not sure whether you have selected the Form Object.

8.9.2. Call the VFX One-To-Many Form Builder

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



Form Name: frmOrdentry
Caption: Auftragseingabe
Master Table: orders

Page Count: 1
Page Title: Da\<teneingabe
Justified Tab: [ ]

orders.customer_id
orders.shipper_id
orders.order_number
orders.order_date
orders.ship_to_name
orders.ship_to_address
orders.ship_to_city

Control Type: cpickfield
Caption: Kunde:
Format:
Input Mask: XXXXXX
Status Bar: Kund/innen-Nr. des Bestellers
Read Only: [ ]

Use DBC Definitions: [ ]
Overwrite Font: [✓]

First, you select the following options:

Form Name. See description in the chapter *The VFX Form Builder*.

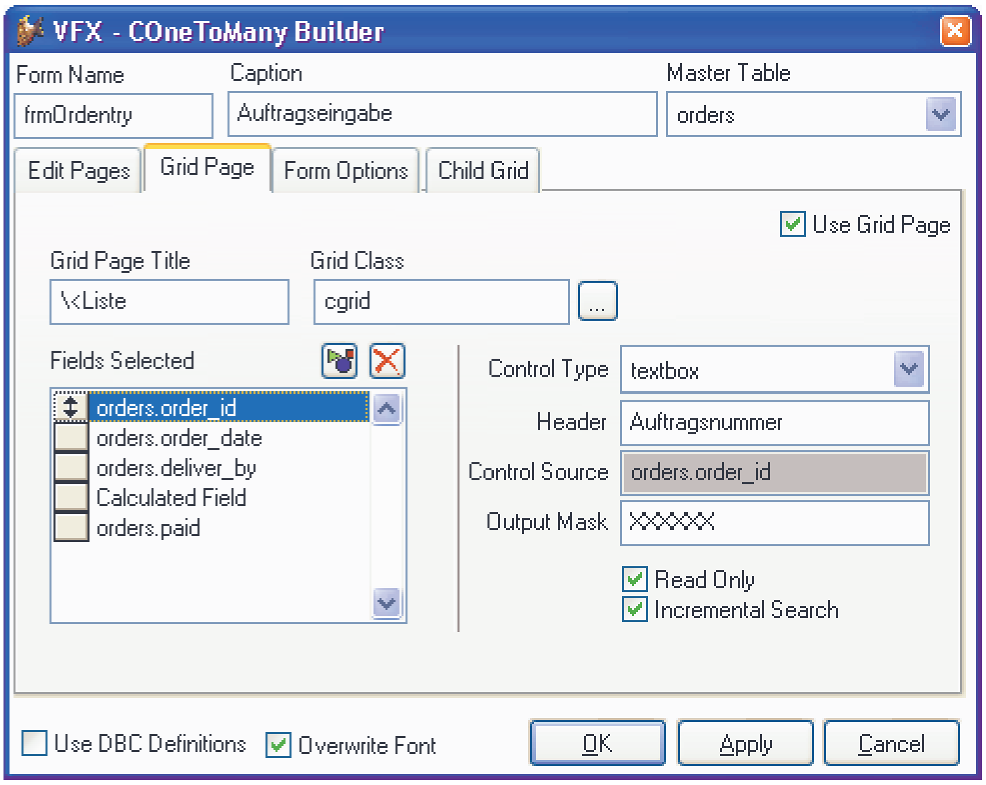
Caption. See description in the chapter *The VFX Form Builder*.

Master Table. Table name of the master table or view.

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

On the page named *Edit Pages*, you see the same user interface like in the VFX Form Builder, described earlier in this documentation. Here you define the edit pages for the master (also called parent) table.

On the page named *Parent Grid Page*, you see the same user interface like in the VFX Form Builder, described earlier in this documentation. Here you define the Grid Page of the master (parent) table:


VFX - COneToMany Builder

Form Name: frmOrdentry
Caption: Auftragseingabe
Master Table: orders

Edit Pages | Grid Page | Form Options | Child Grid

[✓] Use Grid Page

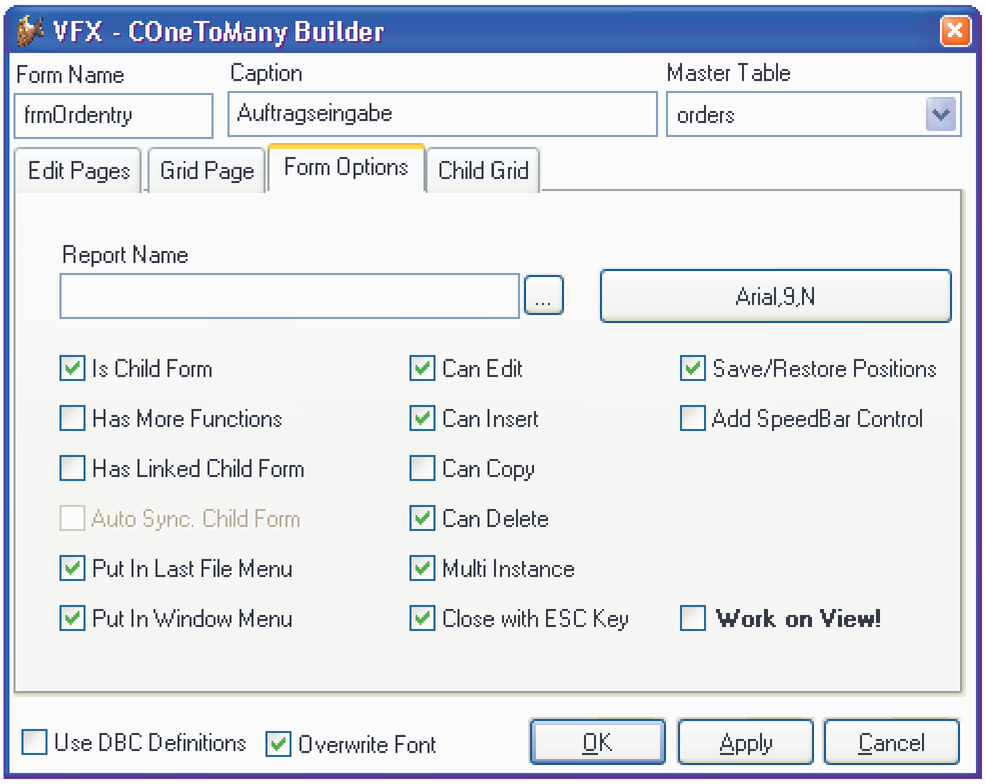
Grid Page Title: \<Liste
Grid Class: cgrid

Fields Selected:
orders.order_id
orders.order_date
orders.deliver_by
Calculated Field
orders.paid

Control Type: textbox
Header: Auftragsnummer
Control Source: orders.order_id
Output Mask: XXXXXX
[✓] Read Only
[✓] Incremental Search

[ ] Use DBC Definitions [✓] Overwrite Font
OK Apply Cancel

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:


VFX - COneToMany Builder

Form Name: frmOrdentry
Caption: Auftragseingabe
Master Table: orders

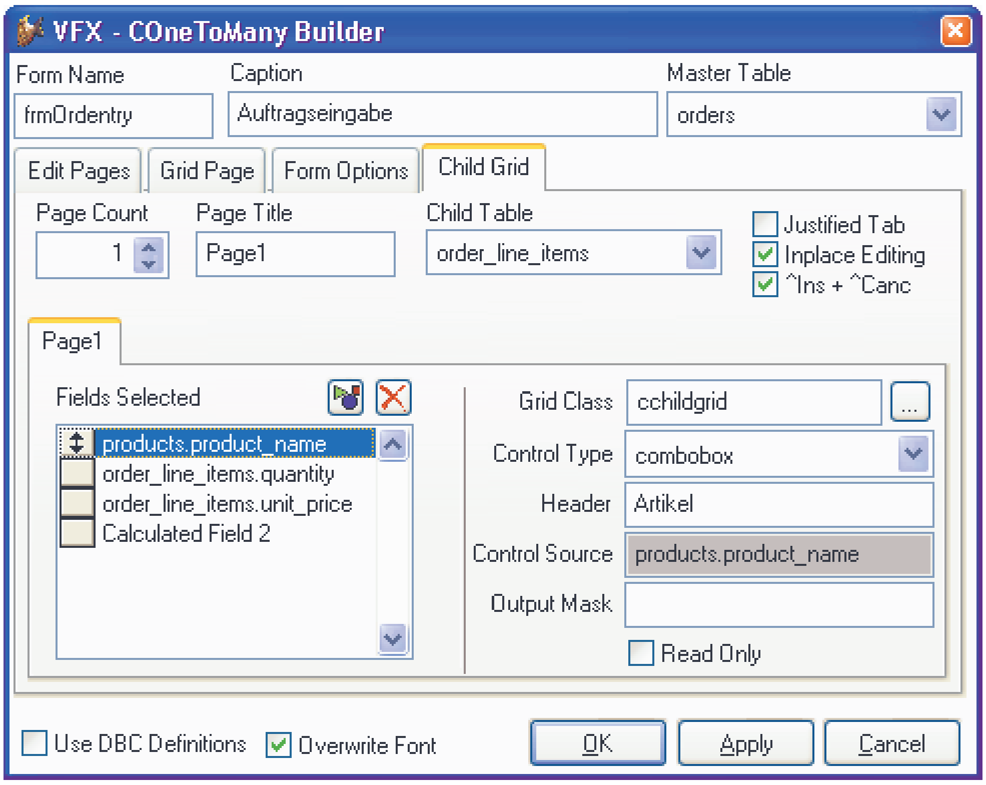
Edit Pages | Grid Page | Form Options | Child Grid

Report Name:
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
[ ] **Work on View!**

[ ] Use DBC Definitions [✓] Overwrite Font
OK Apply Cancel

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



Form Name	frmOrdentry
Caption	Auftragseingabe
Master Table	orders
Page Count	1
Page Title	Page1
Child Table	order_line_items
Justified Tab	☐
Inplace Editing	✓
^Ins + ^Canc	✓
Grid Class	cchildgrid
Control Type	combobox
Header	Artikel
Control Source	products.product_name
Output Mask	
Read Only	☐
Use DBC Definitions	☐
Overwrite Font	✓

Fields Selected
- products.product_name
- order_line_items.quantity
- order_line_items.unit_price
- Calculated Field 2

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 do not set this property, your form will not work correctly.

Justified Tab. Check this option, if your tabs should be justified, otherwise they will be variable in the width.

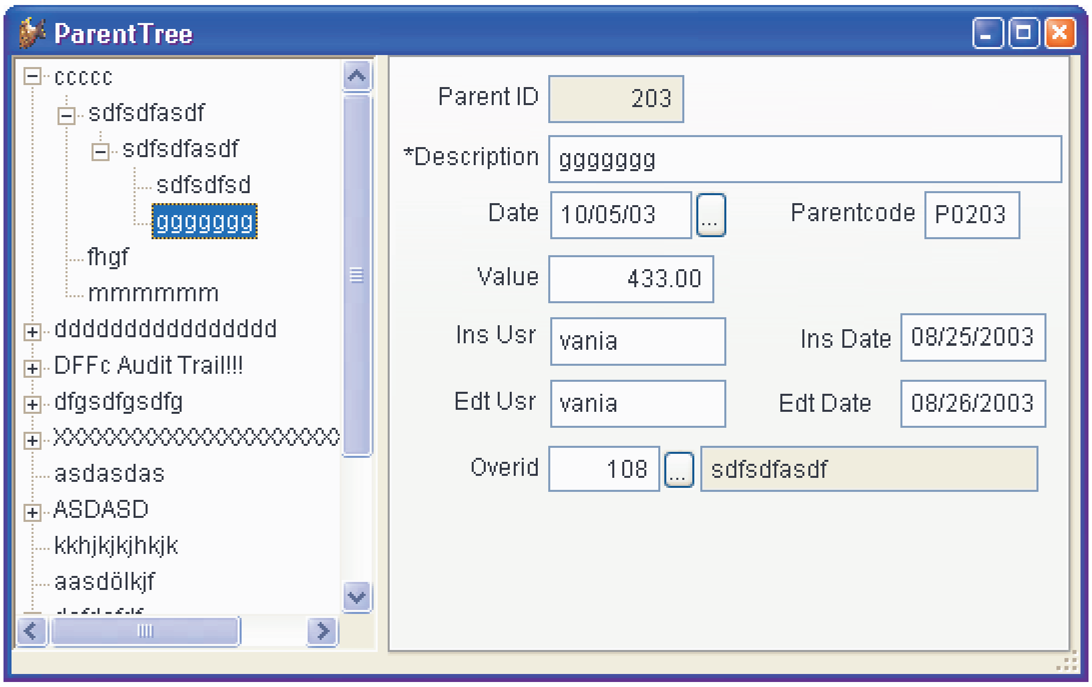
Inplace Editing. Set this option, if you want to enter data in your child grid, which is usually what you want.

^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 Form Builder on the grid page.

8.10. cTreeViewForm Class

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

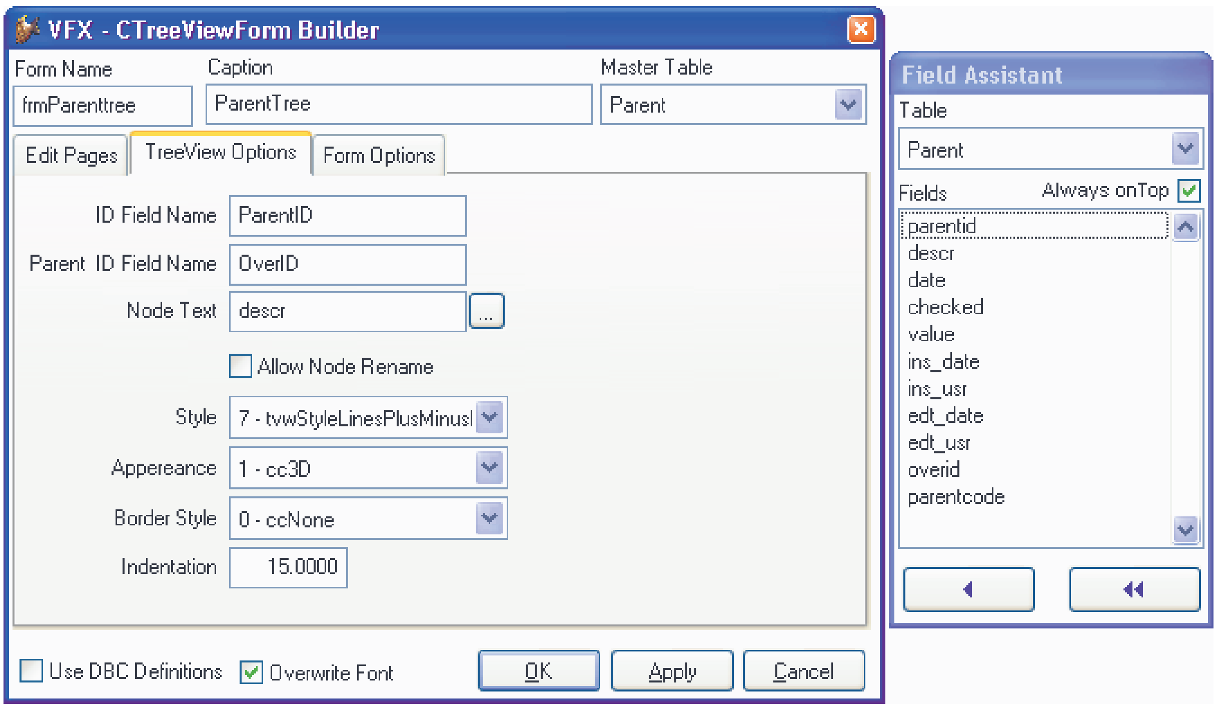

ParentTree

- ccccc
  - sdfsdfasdf
    - sdfsdfasdf
      - sdfsdfsd
      - ggggggg
  - fhgf
  - mmmmmmm
- dddddddddddddddd
- DFFc Audit Trail!!!
- dfgsdfgsdfg
- XXXXXXXXXXXXXXXXXXXX
- asdasdas
- ASDASD
- kkhjkjkjhkjk
- aasdölkjf

Parent ID	203		
*Description	ggggggg		
Date	10/05/03	Parentcode	P0203
Value	433.00		
Ins Usr	vania	Ins Date	08/25/2003
Edt Usr	vania	Edt Date	08/26/2003
Overid	108		sdfsdfasdf

The class is based on cTreeView (vfxappl.vcx) and cDataFormPage (vfxform.vcx). It combines the functionality of cDataFormPage and the advanced data presentation in a hierarchical tree structure. When a particular node in the TreeView is clicked, the record pointer is moved to the corresponding data row in the main table. The user can view and modify the data on the right part in the form.

With the VFX – CtreeViewForm Builder you can rapidly create and tune a form based on cTreeViewForm class and set all necessary properties.


VFX - CTreeViewForm Builder

Form Name	Caption	Master Table
frmParenttree	ParentTree	Parent

ID Field Name	ParentID
Parent ID Field Name	OverID
Node Text	descr
Allow Node Rename	☐
Style	7 - tvwStyleLinesPlusMinusI
Appereance	1 - cc3D
Border Style	0 - ccNone
Indentation	15.0000

☐ Use DBC Definitions ☑ Overwrite Font

Field Assistant

Table: Parent

Fields — Always onTop ☑

parentid
descr
date
checked
value
ins_date
ins_usr
edt_date
edt_usr
overid
parentcode

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.

8.10.1. Data binding settings for 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 being updated according new description entered in the TreeView control.

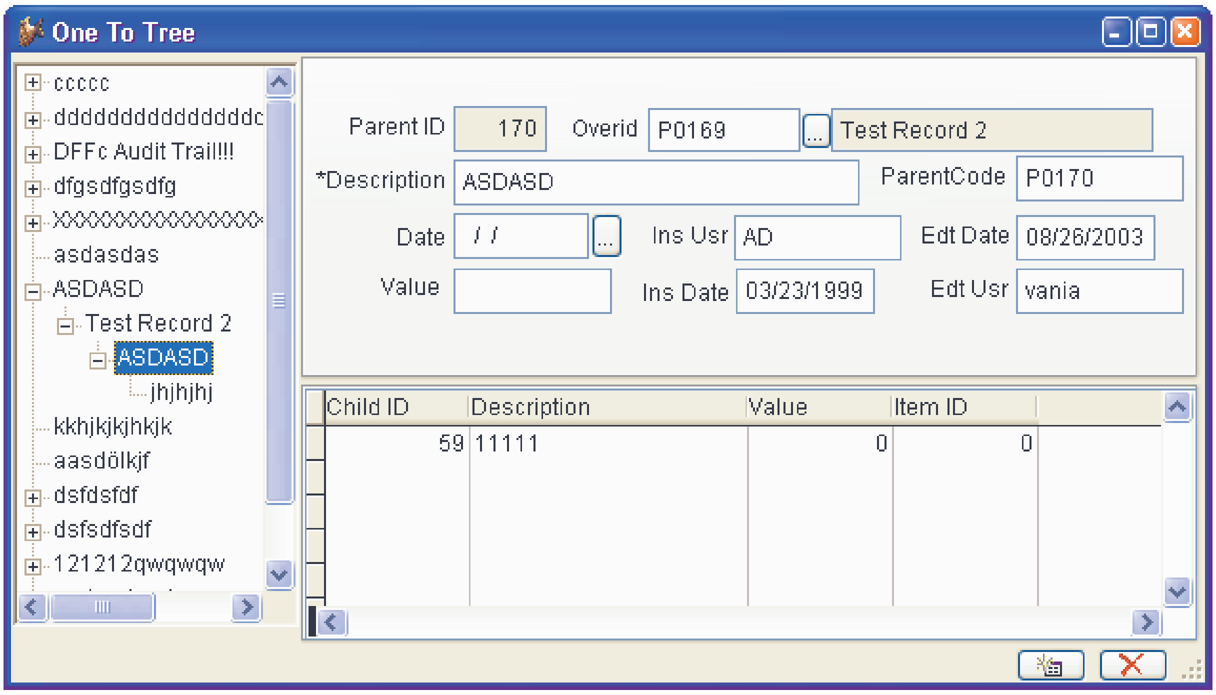
8.10.2. Layout-Settings of the TreeView Control

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. cTreeViewOneToMany Class

Main purpose of this class is to represent the data contained in a particular table (main table) in a tree structure along with the powerful functionality that cOneToMany class gives the developers. The TreeView structure gives the end-user complete overview of the hierarchical data relations contained in the particular table.

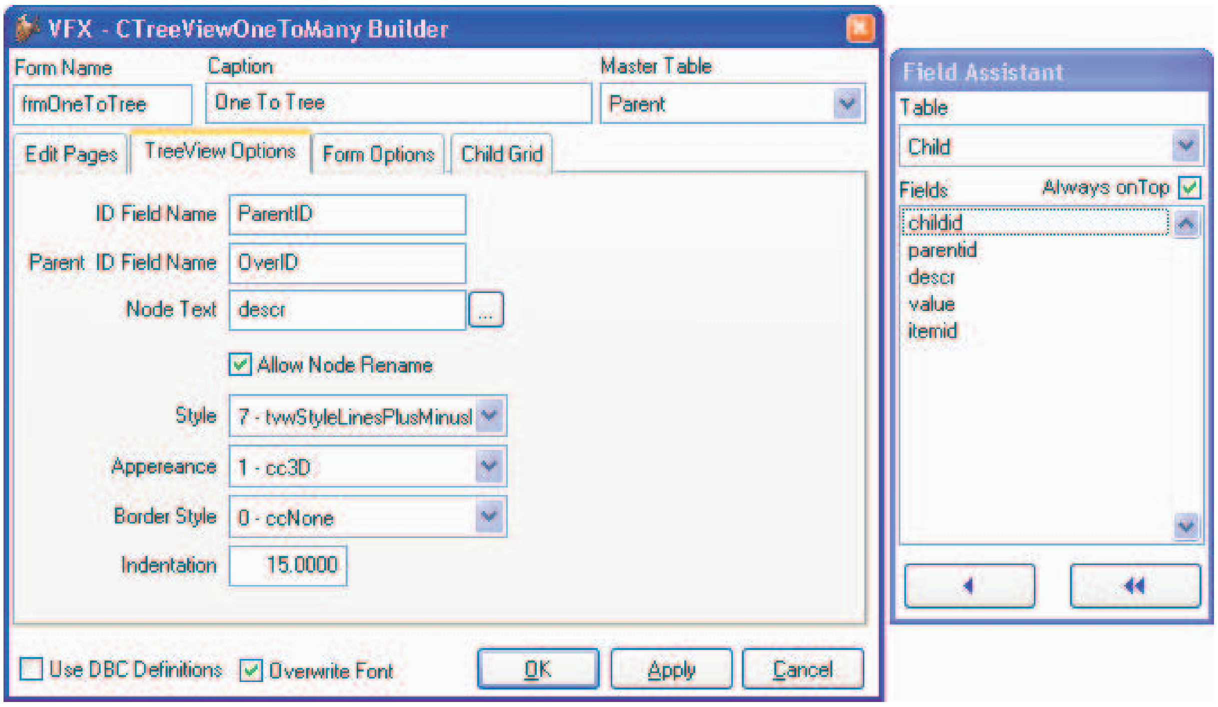


Parent ID	170	Overid	P0169	Test Record 2	
*Description	ASDASD			ParentCode	P0170
Date	/ /	Ins Usr	AD	Edt Date	08/26/2003
Value		Ins Date	03/23/1999	Edt Usr	vania

Child ID	Description	Value	Item ID
59	11111	0	0

This class is based on class cOneToMany (vfxform.vcx) and contains a TreeView element 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.


VFX - CTreeViewOneToMany Builder

Form Name	Caption	Master Table
frmOneToTree	One To Tree	Parent

Edit Pages | TreeView Options | Form Options | Child Grid

ID Field Name	ParentID
Parent ID Field Name	OverID
Node Text	descr
	☑ Allow Node Rename
Style	7 - tvwStyleLinesPlusMinusl
Appereance	1 - cc3D
Border Style	0 - ccNone
Indentation	15.0000

☐ Use DBC Definitions ☑ Overwrite Font

Field Assistant
Table: Child
Fields: childid, parentid, descr, value, itemid
Always onTop ☑

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

8.11.1. Data binding settings for the TreeView-Control

IDFieldName - Here you should be entered the name of the primary key field of your main table

ParentIDFieldName - This property contains 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 being updated according new description entered in the TreeView control.

8.11.2. 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

8.12. The VFX Child Grid Builder

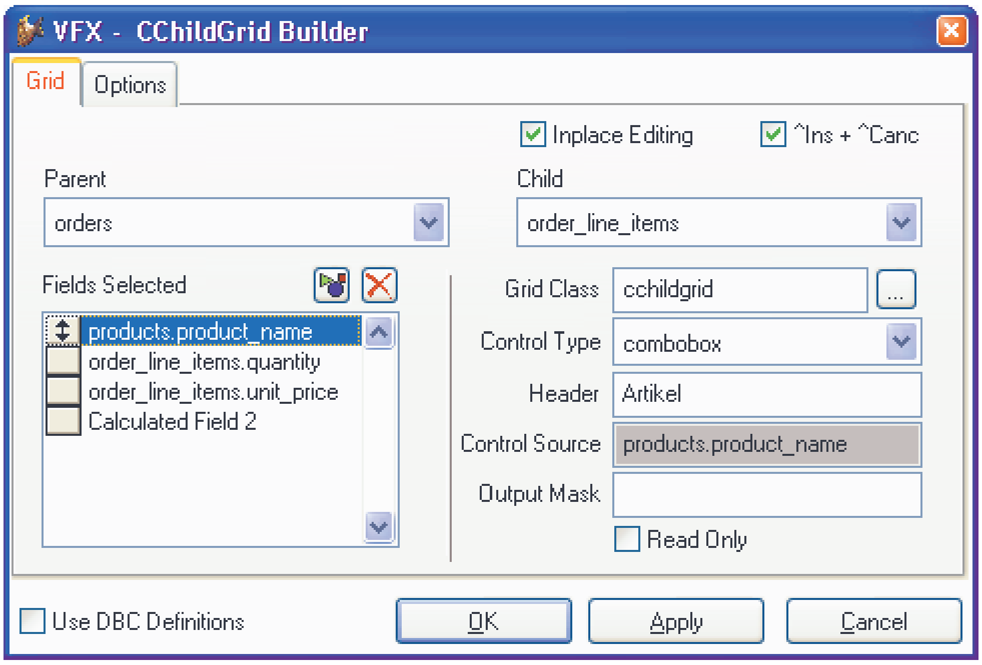
The Child Grid Builder, allows you to enhance the functionality of your child grids. Use this Builder either to customize some fields or to edit the method code *OnPostInsert()*, which will fire whenever a new child record has been inserted. It is similar to the standard VFX data manipulation form, where 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 <ChildTable.ChildLinkField> WITH <Master.MasterField>
```

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



☑ Inplace Editing ☑ ^Ins + ^Canc

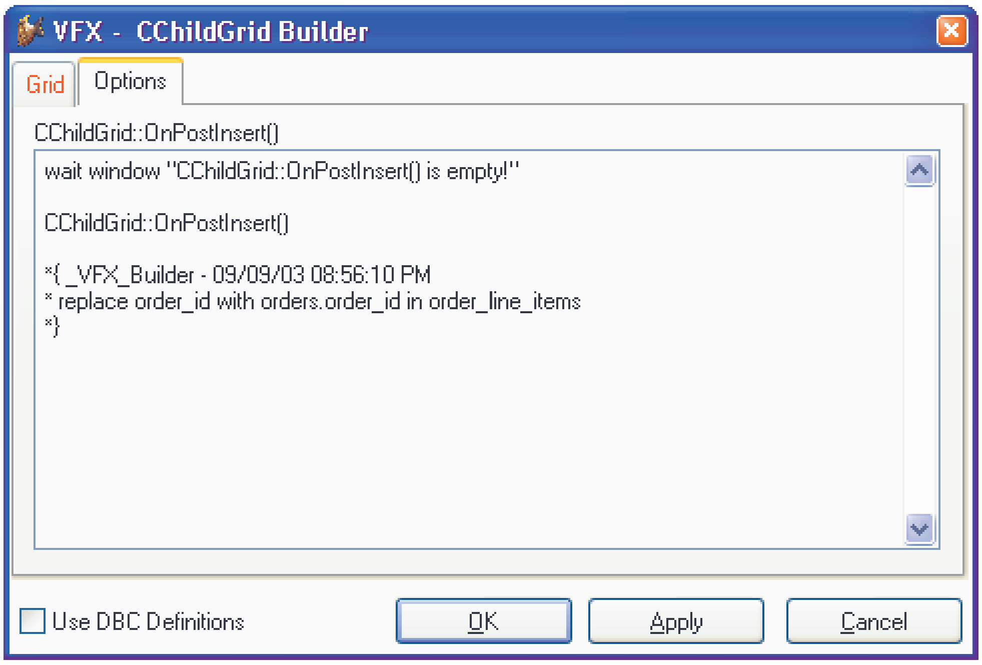
Parent: orders
Child: order_line_items

Fields Selected:
- products.product_name
- order_line_items.quantity
- order_line_items.unit_price
- Calculated Field 2

Grid Class: cchildgrid
Control Type: combobox
Header: Artikel
Control Source: products.product_name
Output Mask:
☐ Read Only

☐ Use DBC Definitions

On the second tab called options, you can edit the method code for the replace of the field in the child table, which must store the value of the parent field.



CChildGrid::OnPostInsert()

```
wait window "CChildGrid::OnPostInsert() is empty!"

CChildGrid::OnPostInsert()

*{ _VFX_Builder - 09/09/03 08:56:10 PM
* replace order_id with orders.order_id in order_line_items
*}
```

NOTE: 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

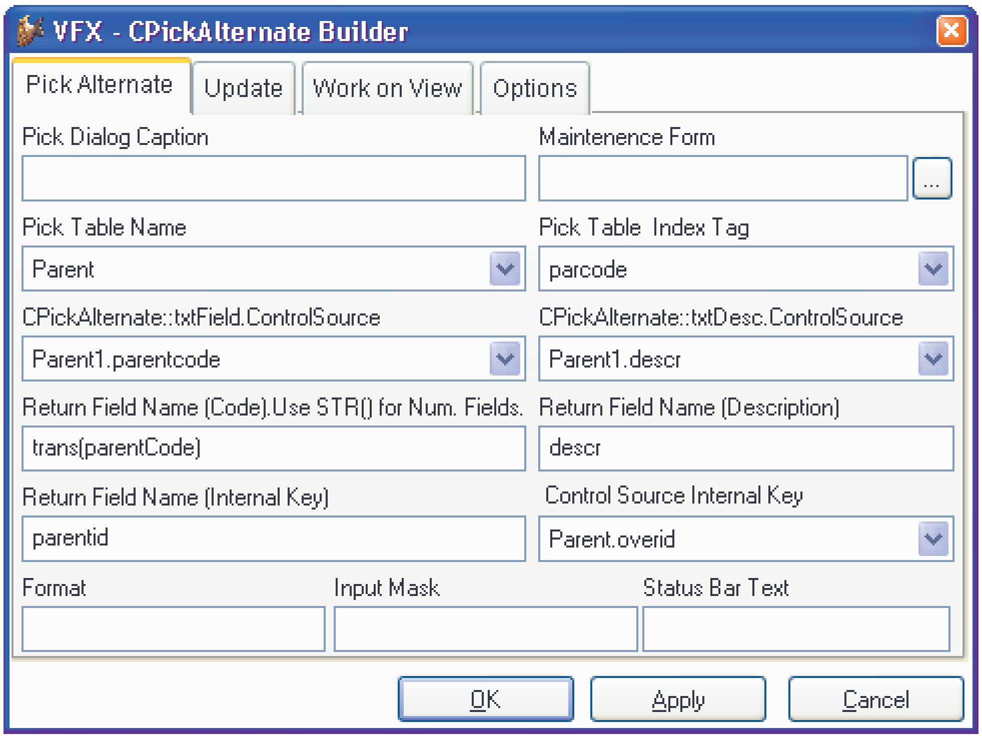
8.13. cPickAlternate class

Similar to the cPickField control, the cPickAlternate class can be used for validating user input against a table as well as to provide the call of a pick list dialog where the user can pick a 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 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 – CtreeViewForm Builder you can easy set all necessary properties.


VFX - CPickAlternate Builder

Pick Alternate | Update | Work on View | Options

Pick Dialog Caption:	Maintenence Form:
Pick Table Name: Parent	Pick Table Index Tag: parcode
CPickAlternate::txtField.ControlSource: Parent1.parentcode	CPickAlternate::txtDesc.ControlSource: Parent1.descr
Return Field Name (Code).Use STR() for Num. Fields.: trans(parentCode)	Return Field Name (Description): descr
Return Field Name (Internal Key): parentid	Control Source Internal Key: Parent.overid

Format: | Input Mask: | Status Bar Text:

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 Controlsource 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 through this field.

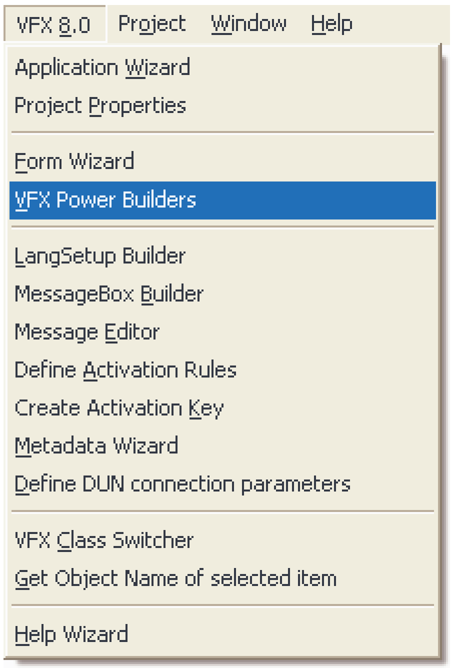
Control Source Internal Key – The name of the field from the main table, where the cPickAlternate class will store the key value. This field contains the foreign key to the Picklist table.

8.14. The VFX-CPickTextBox Builder

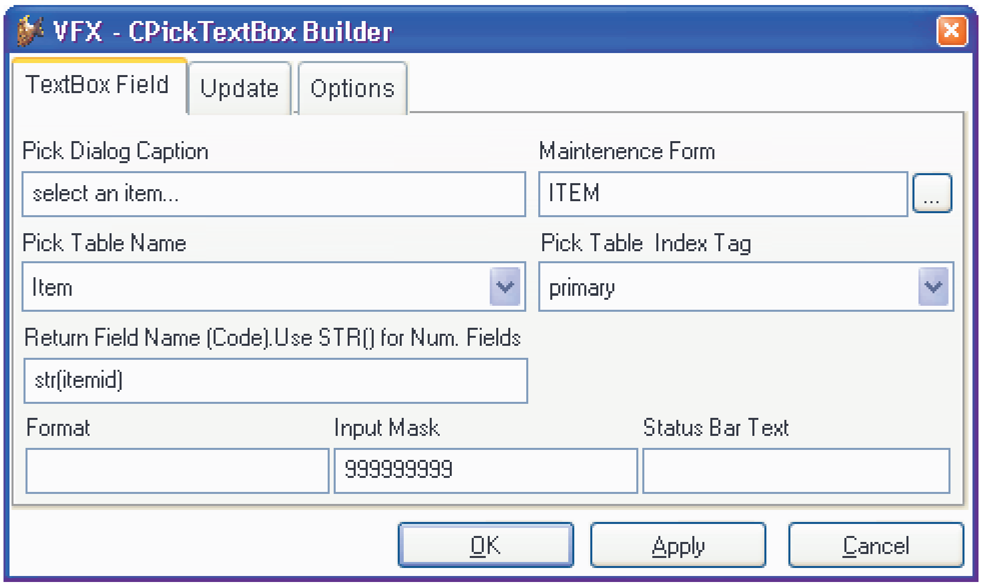
Visual Extend offers a Builder, which helps you to create full featured Picklist Controls. Picklist Controls can be used in Child Grids.

8.14.1. Call the VFX-CPickTextBox Builder

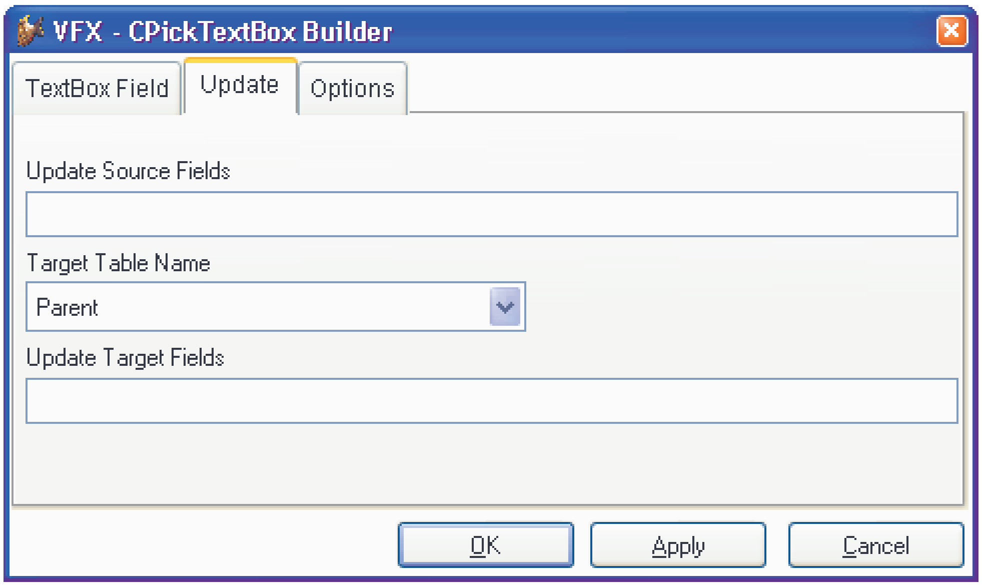
To call the VFX Pick Text Box Builder, select the column in the child grid, which should become a picklist control and select the option VFX Power Builders from the VFX Menu:



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



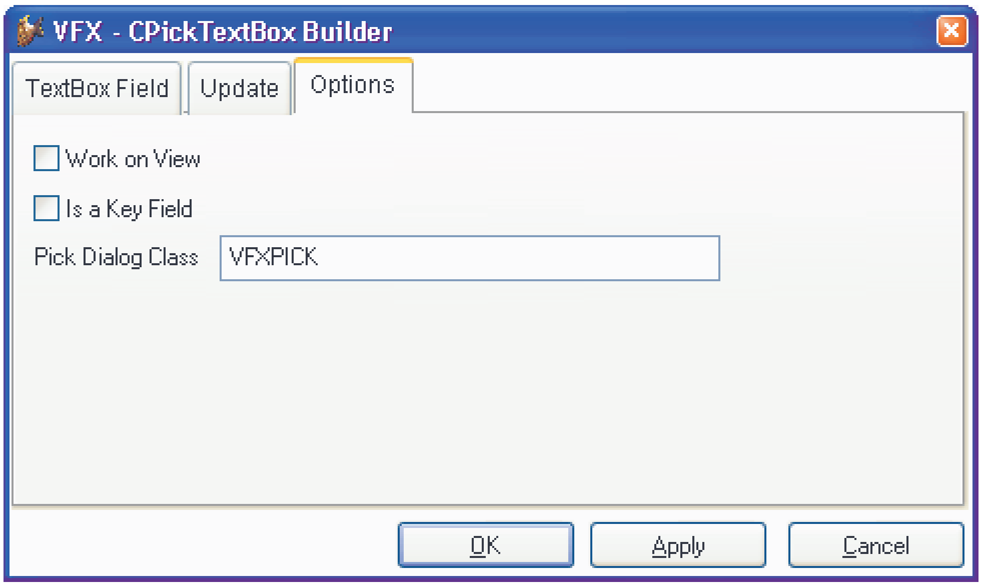
Pick Dialog Caption: select an item...
Maintenence Form: ITEM
Pick Table Name: Item
Pick Table Index Tag: primary
Return Field Name (Code).Use STR() for Num. Fields: str(itemid)
Format:
Input Mask: 999999999
Status Bar Text:



Update Source Fields

Target Table Name: Parent

Update Target Fields



☐ Work on View

☐ Is a Key Field

Pick Dialog Class: VFXPICK

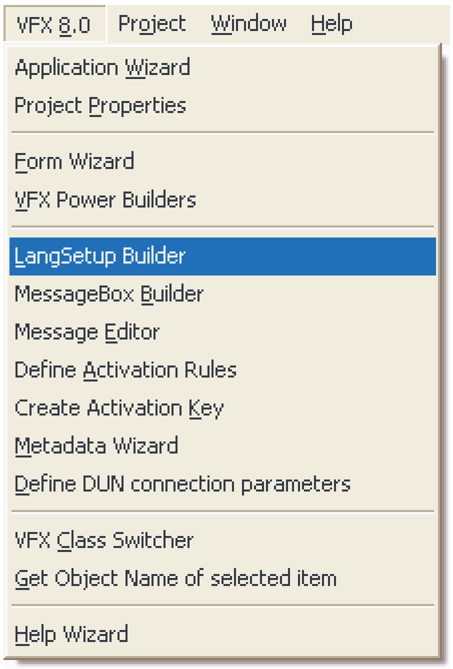
8.15. The VFX LangSetup Builder

8.15.1. Objective

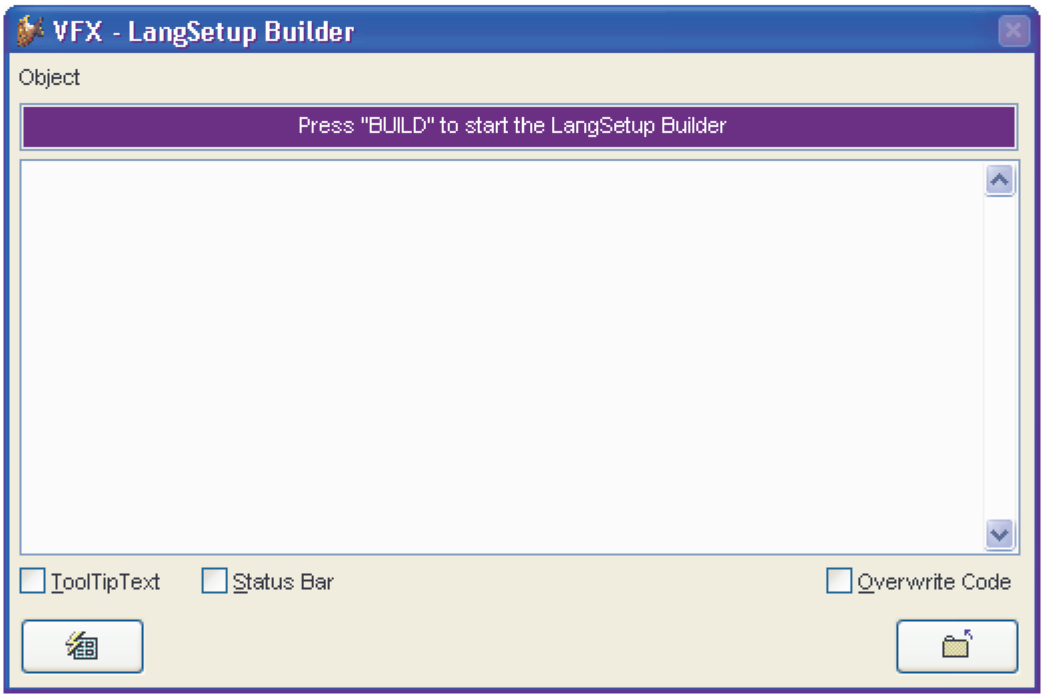
The VFX LangSetup Builder automates the creation of the needed code within the Langsetup method. You will need this, if you have to supply your application in more than one language. The goal of this builder is to extract all potential captions, tooltiptext and Status Bar messages, generating unique constants and putting them into the VFXMSG 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.

8.15.2. Call the VFX LangSetup Builder

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 in the VFX menu:

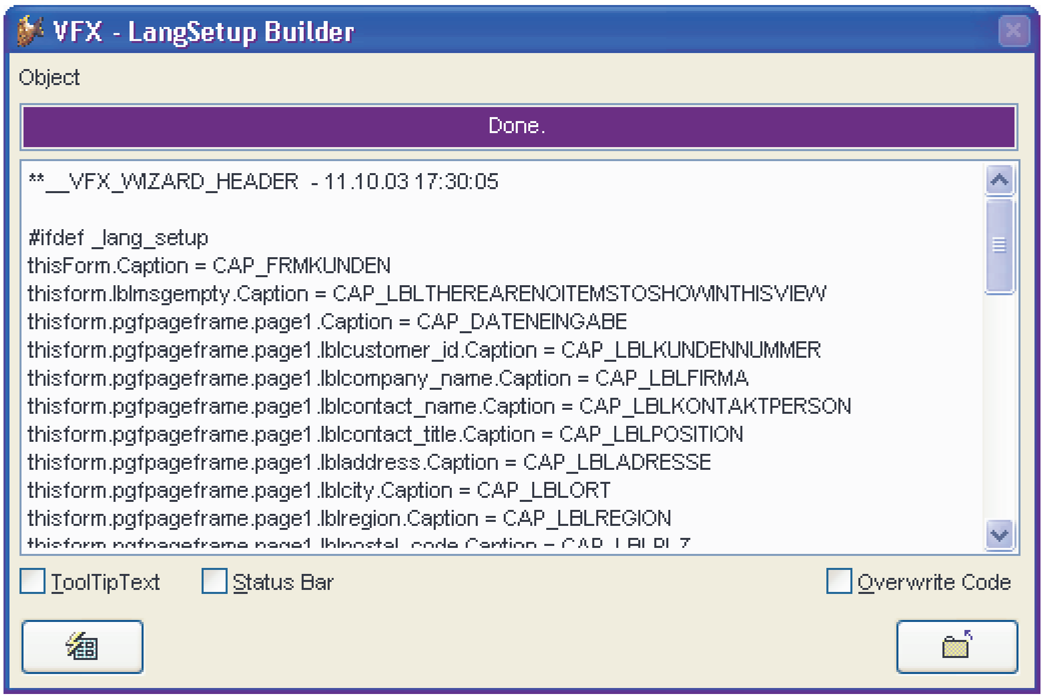


8.15.3. The LangSetup Builder user interface



Mark the Checkboxes correspondent to the desired options. Click on the button Build to start the generation.

After generation look at the code that were generated for the *LangSetup()*-Method. . If you checked 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 where you can edit and translate them into other languages.



```
**__VFX_WIZARD_HEADER - 11.10.03 17:30:05

#ifdef _lang_setup
thisForm.Caption = CAP_FRMKUNDEN
thisform.lblmsgempty.Caption = CAP_LBLTHEREARENOITEMSTOSHOWINTHISVIEW
thisform.pgfpageframe.page1.Caption = CAP_DATENEINGABE
thisform.pgfpageframe.page1.lblcustomer_id.Caption = CAP_LBLKUNDENNUMMER
thisform.pgfpageframe.page1.lblcompany_name.Caption = CAP_LBLFIRMA
thisform.pgfpageframe.page1.lblcontact_name.Caption = CAP_LBLKONTAKTPERSON
thisform.pgfpageframe.page1.lblcontact_title.Caption = CAP_LBLPOSITION
thisform.pgfpageframe.page1.lbladdress.Caption = CAP_LBLADRESSE
thisform.pgfpageframe.page1.lblcity.Caption = CAP_LBLORT
thisform.pgfpageframe.page1.lblregion.Caption = CAP_LBLREGION
```

Note, that the constants will automatically be inserted into the VFXMSG table, if you mark *Overwrite Code* checkbox.

8.15.4. Define _Lang_Setup

In the Include File ***VFX.H*** the _Lang_Setup constant 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.
```

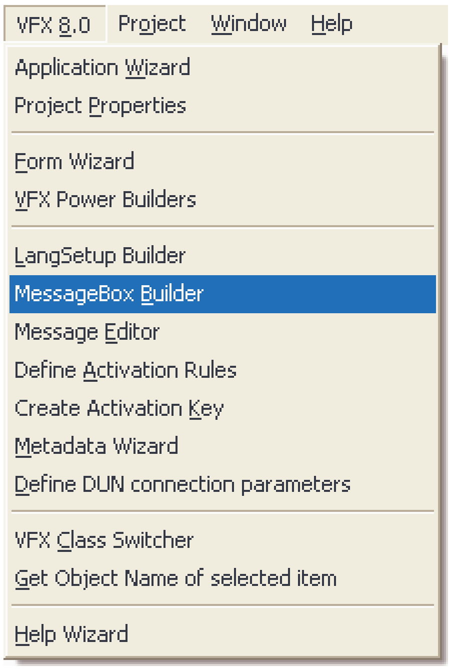
8.16. The VFX Messagebox Builder

8.16.1. Objective

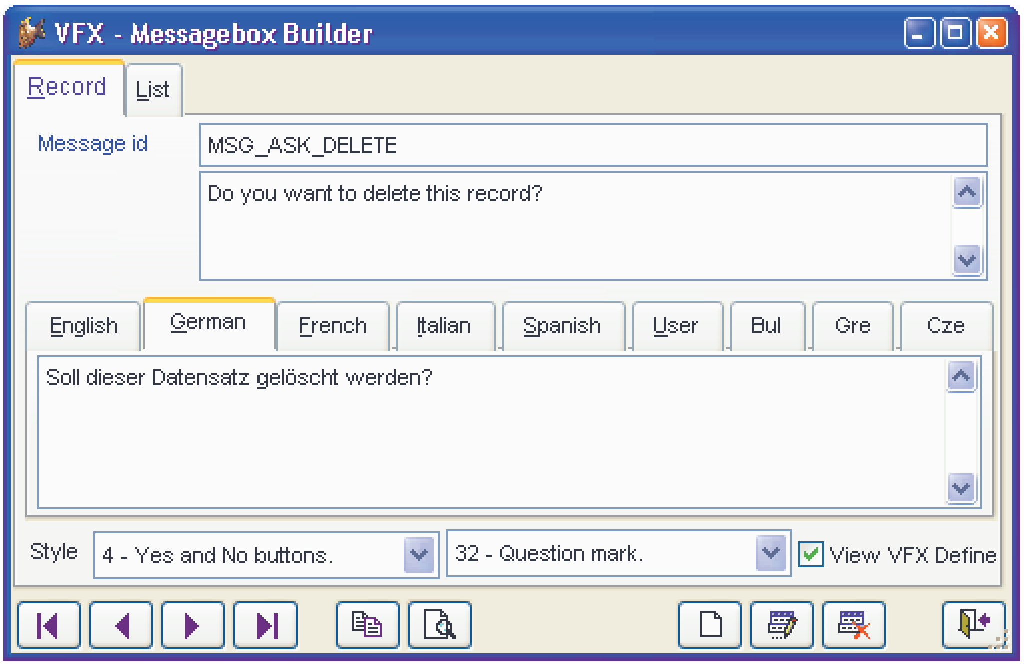
The VFX Messagebox Builder is a handy utility to create messageboxes (and WAIT WINDOWs) on the fly while coding. The messagebox builder does not only assist you in the generation of the line of code for the messagebox but greatly reduces the work to provide the captions in the VFX message table from where you can edit and translate them into the different languages. The VFX message editor is described in details later in this manual.

8.16.2. Call the Messagebox Builder

To call the VFX Messagebox Builder, select the following option in the VFX Menu:



8.16.3. The VFX - Messagebox Builder User Interface



Message id: MSG_ASK_DELETE

Do you want to delete this record?

Soll dieser Datensatz gelöscht werden?

Style: 4 - Yes and No buttons. | 32 - Question mark. | ✓ View VFX Define

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.

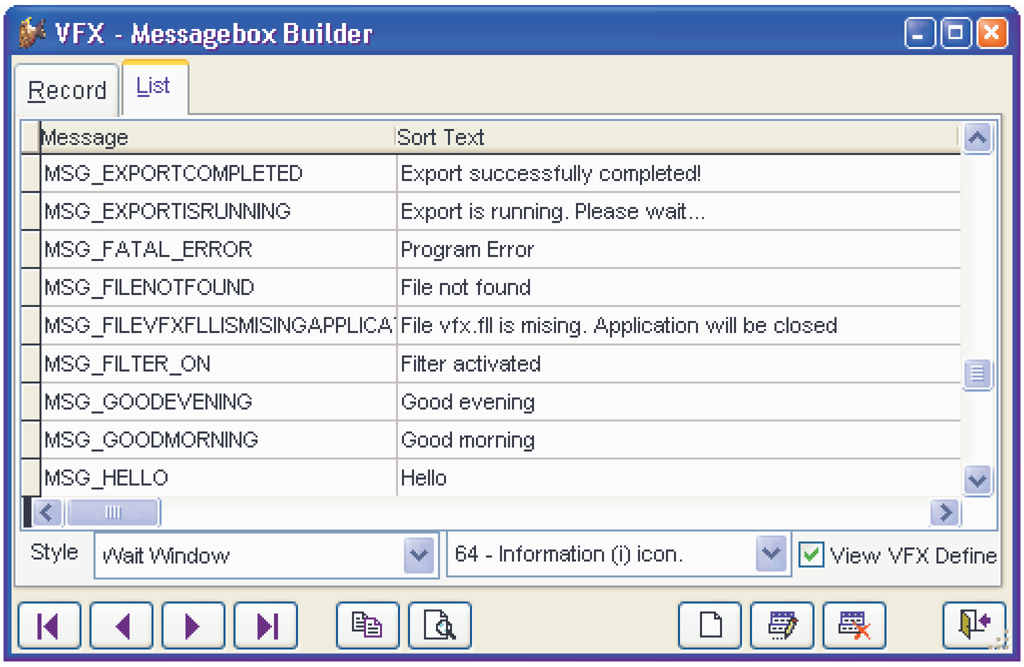
In the row *Style*, select desired type of the Messagebox. You can choose among different icons and buttons.

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 puts a data record in the table *VFXMSG.DBF*.

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



Message	Sort Text
MSG_EXPORTCOMPLETED	Export successfully completed!
MSG_EXPORTISRUNNING	Export is running. Please wait...
MSG_FATAL_ERROR	Program Error
MSG_FILENOTFOUND	File not found
MSG_FILEVFXFLLISMISINGAPPLICA	File vfx.fll is mising. Application will be closed
MSG_FILTER_ON	Filter activated
MSG_GOODEVENING	Good evening
MSG_GOODMORNING	Good morning
MSG_HELLO	Hello

Style: Wait Window; 64 - Information (i) icon.; [✓] View VFX Define

Tip: Also, if you do not create multilingual applications, you can use the messagebox builder.

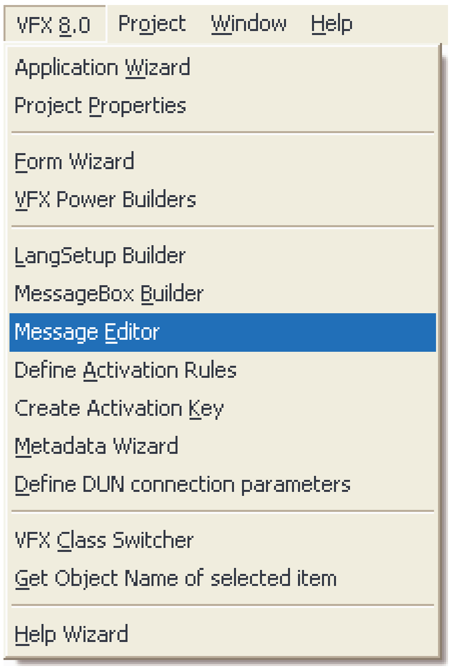
8.17. The VFX - Message Editor

8.17.1. Objective

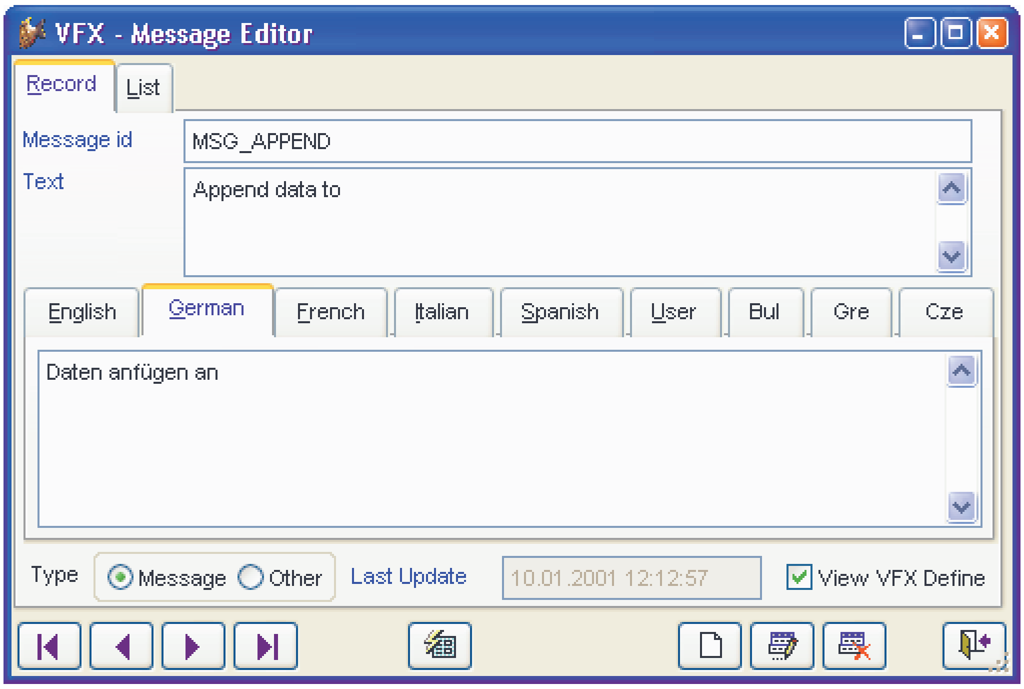
The VFX Message Editor is the central place to manage and translate all messages and other language specific text elements such as captions, tooltip texts and status bar messages. From within the VFX - Message Editor you can create all the needed include files (*USERTEXT.H* and *USERMSG.H*).

8.17.2. Call the VFX - Message Editor

To call the VFX - Message Editor, select the following option in the VFX Menu:



8.17.3. The VFX - Message Editor user interface


VFX - Message Editor

Record | List

Message id: MSG_APPEND

Text: Append data to

English | German | French | Italian | Spanish | User | Bul | Gre | Cze

Daten anfügen an

Type: [✓ Message] Other

Last Update: 10.01.2001 12:12:57

[✓] View VFX Define

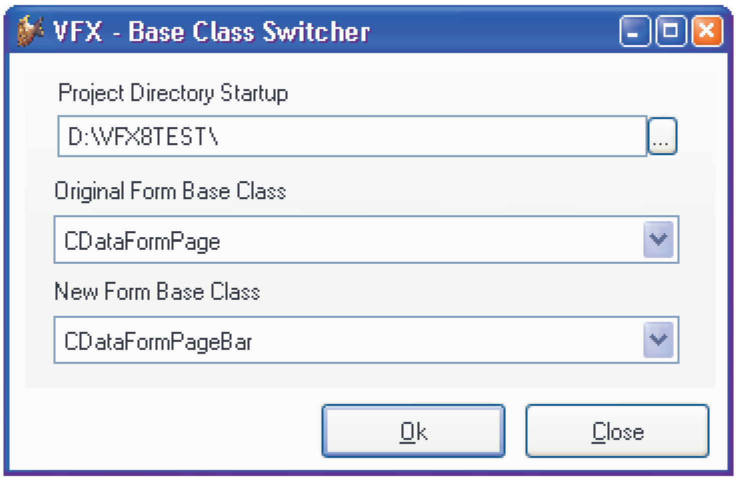
Click on the button *Make Include File* to create an include file in the language selected in the Pageframe. The Include files are saved in a folder with the name of the respective language under the Include folder of your project.

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.

8.18. The 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 removed them again. As particularly helpful this tools handles the actualization of existing VFX 3-Projekte. In VFX 3 each form had a border a with button at the lower edge. In VFX 8.0, you can use instead a correct symbol border.

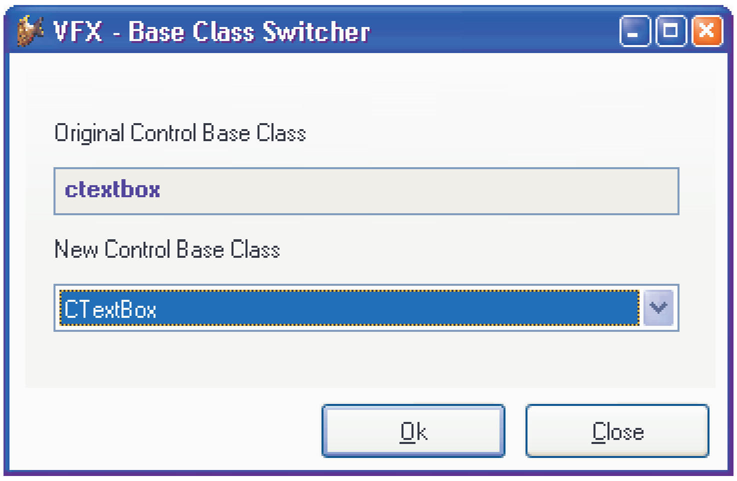

VFX - Base Class Switcher

Project Directory Startup: D:\VFX8TEST\

Original Form Base Class: CDataFormPage

New Form Base Class: CDataFormPageBar

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.

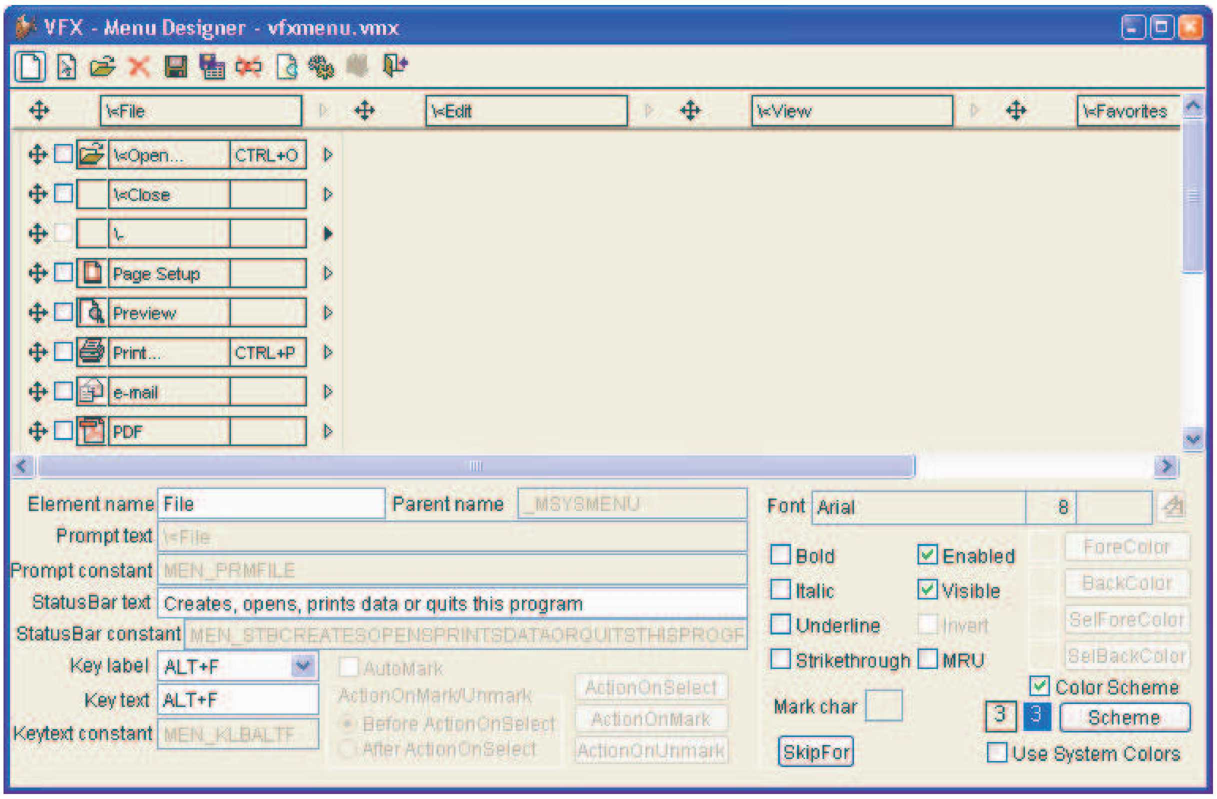

VFX - Base Class Switcher

Original Control Base Class: ctextbox

New Control Base Class: CTextBox

8.19. 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.



\<File \<Edit \<View \<Favorites

\<Open...	CTRL+O
\<Close	
\-	
Page Setup	
Preview	
Print...	CTRL+P
e-mail	
PDF	

Element name: File
Parent name: _MSYSMENU
Prompt text: \<File
Prompt constant: MEN_PRMFILE
StatusBar text: Creates, opens, prints data or quits this program
StatusBar constant: MEN_STBCREATESOPENSPRINTSDATAORQUITSTHISPROGF
Key label: ALT+F
Key text: ALT+F
Keytext constant: MEN_KLBALTF
AutoMark
ActionOnMark/Unmark: Before ActionOnSelect / After ActionOnSelect
Font: Arial 8
Bold [ ] Italic [ ] Underline [ ] Strikethrough [ ]
Enabled [✓] Visible [✓] Invert [ ] MRU [ ]
Color Scheme [✓] 3 3 Scheme
Use System Colors [ ]
Mark char

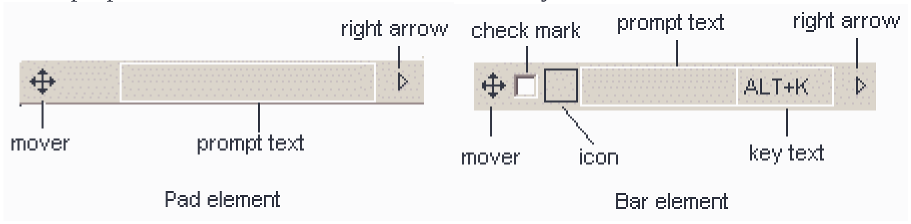
A menu contained in a VFX project can be opened with the VMD directly from the VFP-Project Manager. Alternatively, menus can be opened also from the VMD using the Open button in the toolbar or using the appropriate menu option. In the Opening Dialog, you can choose between the menu types mnx and vmx.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

The menu can be edited visually. You 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.

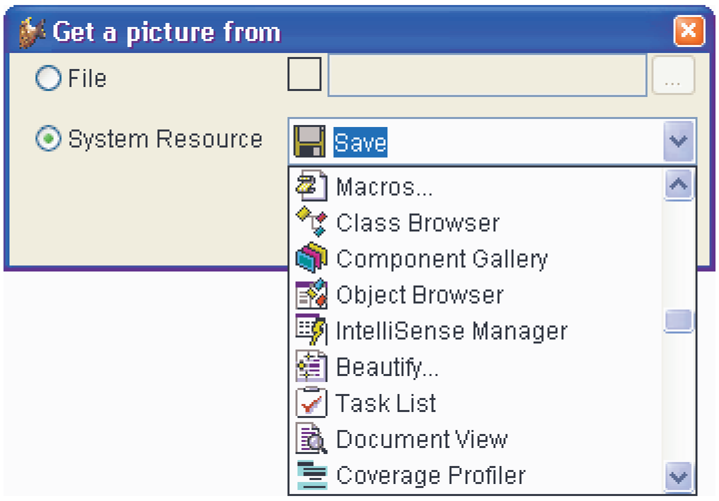
- *Check mark*

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 or 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 cannot 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.

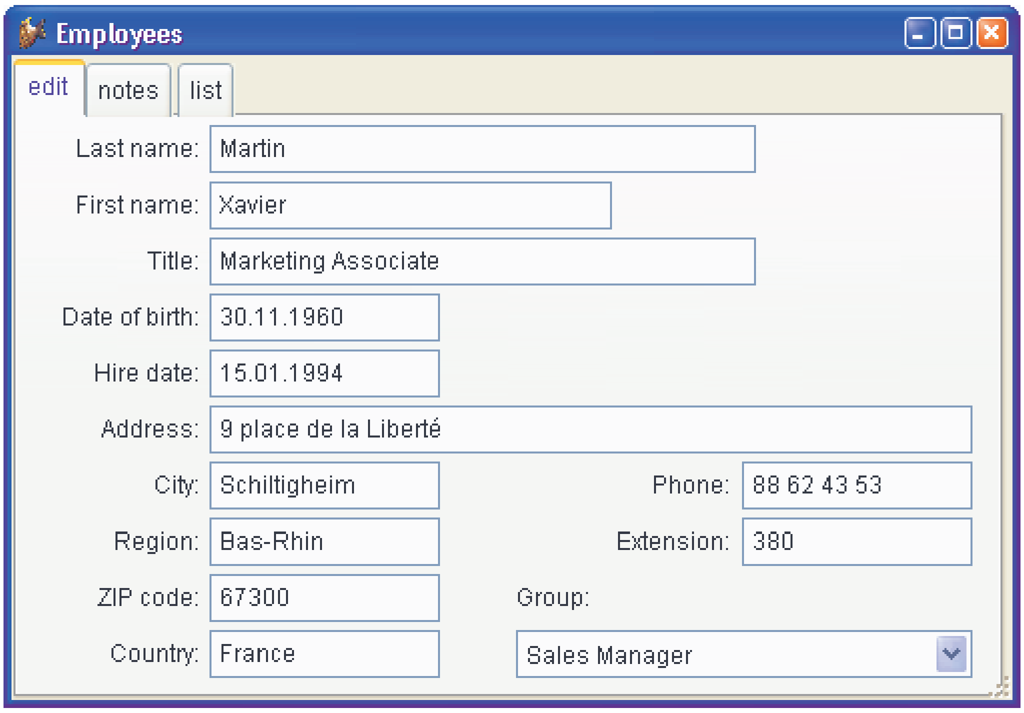
At any time can be displayed a preview for the menu, using *Preview* button in the toolbar or by choosing the option *Preview* from VMD-menu.

9. Features of the created Form

The form that is created with the VFX Form Builder has many 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 individually for every particular user. When a user closes a form and then opens it again, the form is displayed exactly same as when it was closed.

9.1. 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:

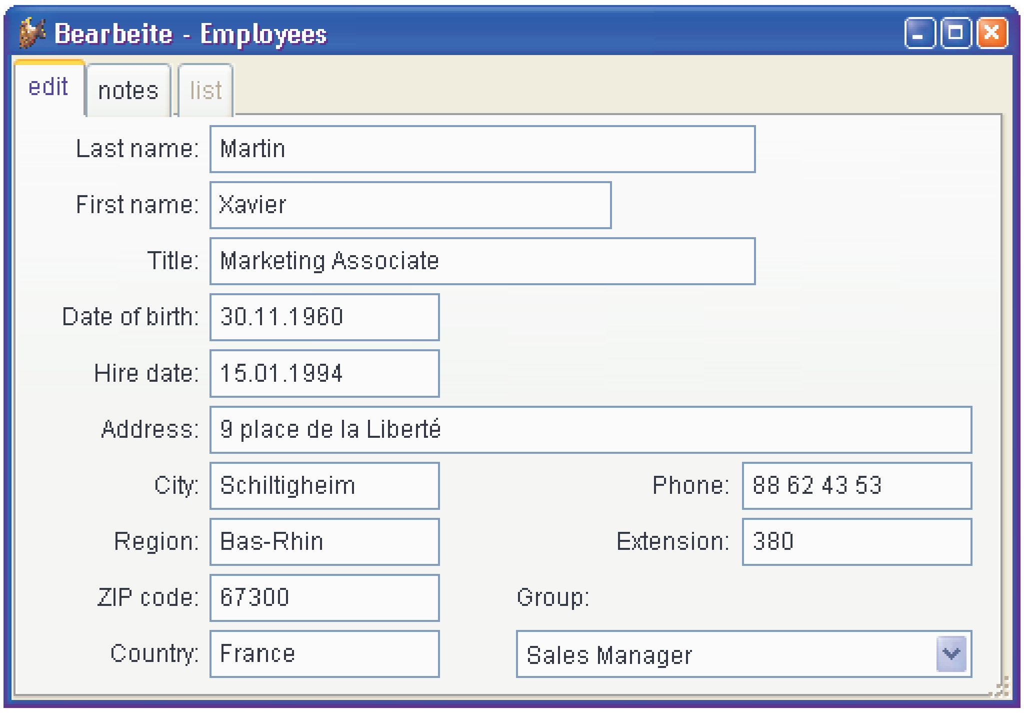


Last name:	Martin
First name:	Xavier
Title:	Marketing Associate
Date of birth:	30.11.1960
Hire date:	15.01.1994
Address:	9 place de la Liberté
City:	Schiltigheim
Phone:	88 62 43 53
Region:	Bas-Rhin
Extension:	380
ZIP code:	67300
Country:	France
Group:	Sales Manager

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 massive 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:

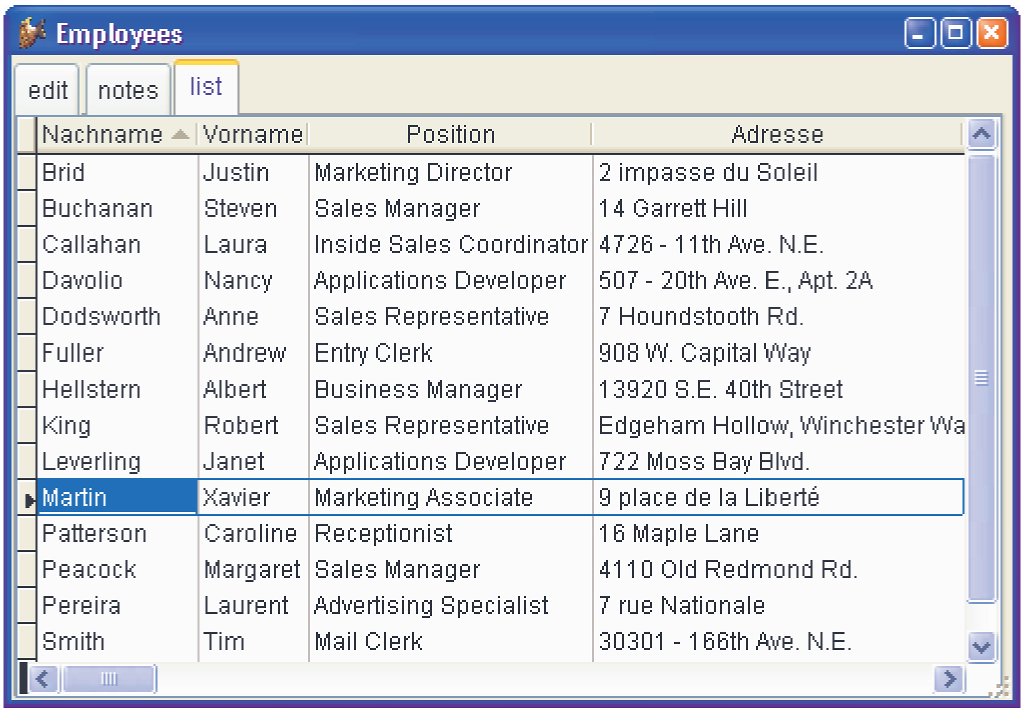


Last name:	Martin		
First name:	Xavier		
Title:	Marketing Associate		
Date of birth:	30.11.1960		
Hire date:	15.01.1994		
Address:	9 place de la Liberté		
City:	Schiltigheim	Phone:	88 62 43 53
Region:	Bas-Rhin	Extension:	380
ZIP code:	67300	Group:	
Country:	France		Sales Manager

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

9.2. The VFX Power Grid

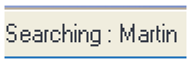
In all columns in a grid, an incremental searching is enabled by default. In addition, doubleclicking on any column header is a grid automatically sorts it by the correspondent column. If there is no existing index for the column, a temporary index will be created by VFX automatically. When it is necessary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



Nachname	Vorname	Position	Adresse
Brid	Justin	Marketing Director	2 impasse du Soleil
Buchanan	Steven	Sales Manager	14 Garrett Hill
Callahan	Laura	Inside Sales Coordinator	4726 - 11th Ave. N.E.
Davolio	Nancy	Applications Developer	507 - 20th Ave. E., Apt. 2A
Dodsworth	Anne	Sales Representative	7 Houndstooth Rd.
Fuller	Andrew	Entry Clerk	908 W. Capital Way
Hellstern	Albert	Business Manager	13920 S.E. 40th Street
King	Robert	Sales Representative	Edgeham Hollow, Winchester Wa
Leverling	Janet	Applications Developer	722 Moss Bay Blvd.
Martin	Xavier	Marketing Associate	9 place de la Liberté
Patterson	Caroline	Receptionist	16 Maple Lane
Peacock	Margaret	Sales Manager	4110 Old Redmond Rd.
Pereira	Laurent	Advertising Specialist	7 rue Nationale
Smith	Tim	Mail Clerk	30301 - 166th Ave. N.E.

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.



Searching : Martin

9.2.1. Incremental Search

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.

9.2.2. Sort using Doubleclicking on the Grid Column Header

You can sort any column by doubleclicking on the column header. Alternate you can toggle the sort order by doubleclicking again. If an index key exists, VFX will automatically use it, otherwise a temporary index will be created which will be deleted when the user closes the form.

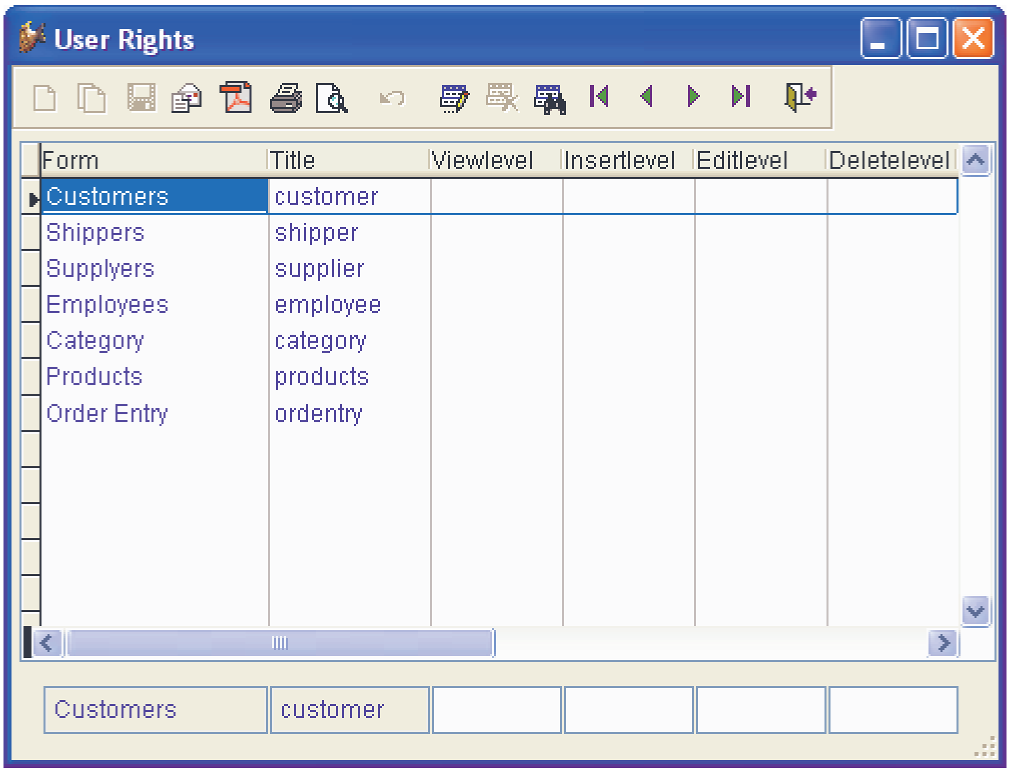
9.2.3. Show current sort order in the column header

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

9.3. *Form based on the class CTableForm*

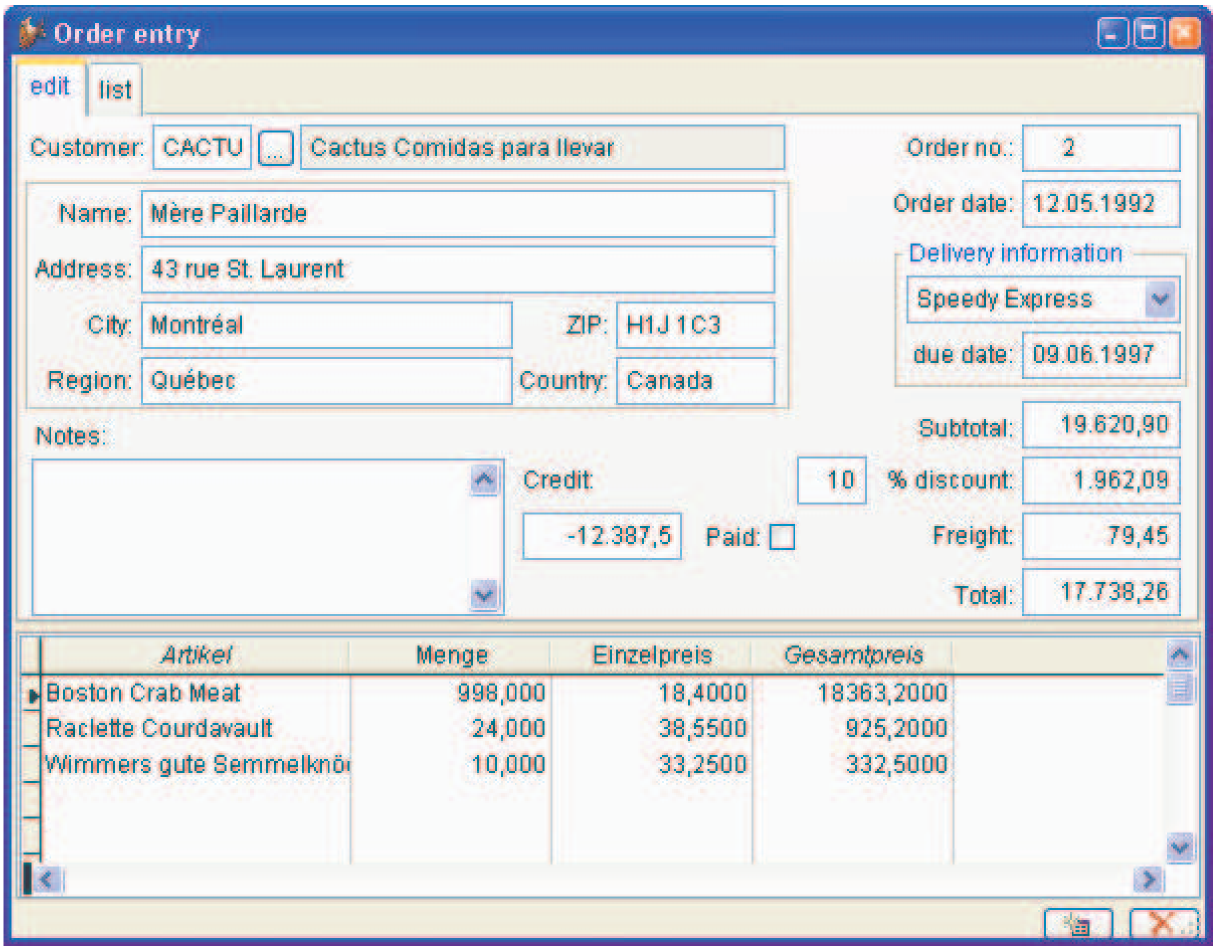
In the forms based on the class CtableForm, the search grid and other controls are placed next to each other or one under another in a container. A typical CTableForm based form is the Administration of the user rights.



Form	Title	Viewlevel	Insertlevel	Editlevel	Deletelevel
Customers	customer				
Shippers	shipper				
Supplyers	supplier				
Employees	employee				
Category	category				
Products	products				
Order Entry	ordentry				

Customers	customer				

9.4. Discussion of the VFX One-To-Many Data Manipulation Form



Customer:	CACTU — Cactus Comidas para llevar
Order no.:	2
Name:	Mère Paillarde
Order date:	12.05.1992
Address:	43 rue St. Laurent
City:	Montréal
ZIP:	H1J 1C3
Region:	Québec
Country:	Canada
Delivery information	Speedy Express
due date:	09.06.1997
Notes:	
Subtotal:	19.620,90
Credit:	-12.387,5
% discount:	10 — 1.962,09
Paid:	☐
Freight:	79,45
Total:	17.738,26

Artikel	Menge	Einzelpreis	Gesamtpreis
Boston Crab Meat	998,000	18,4000	18363,2000
Raclette Courdavault	24,000	38,5500	925,2000
Wimmers gute Semmelknö	10,000	33,2500	332,5000

9.4.1. Editing the Master table

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.

9.4.2. Editing the Child 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.

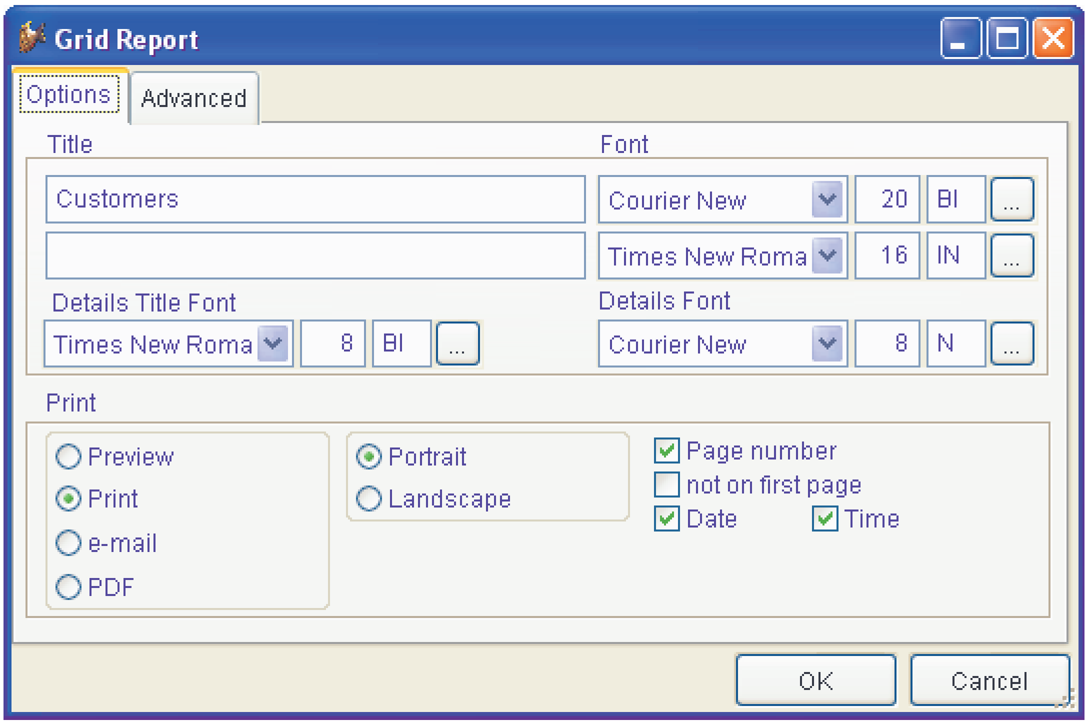
9.4.3. Picklist within the Child Grid

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.

9.5. 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.


Grid Report

Options | Advanced

Title: Customers

Font: Courier New, 20, BI

Times New Roma, 16, IN

Details Title Font: Times New Roma, 8, BI

Details Font: Courier New, 8, N

Print

Preview [✓ Print] e-mail PDF

[✓ Portrait] Landscape

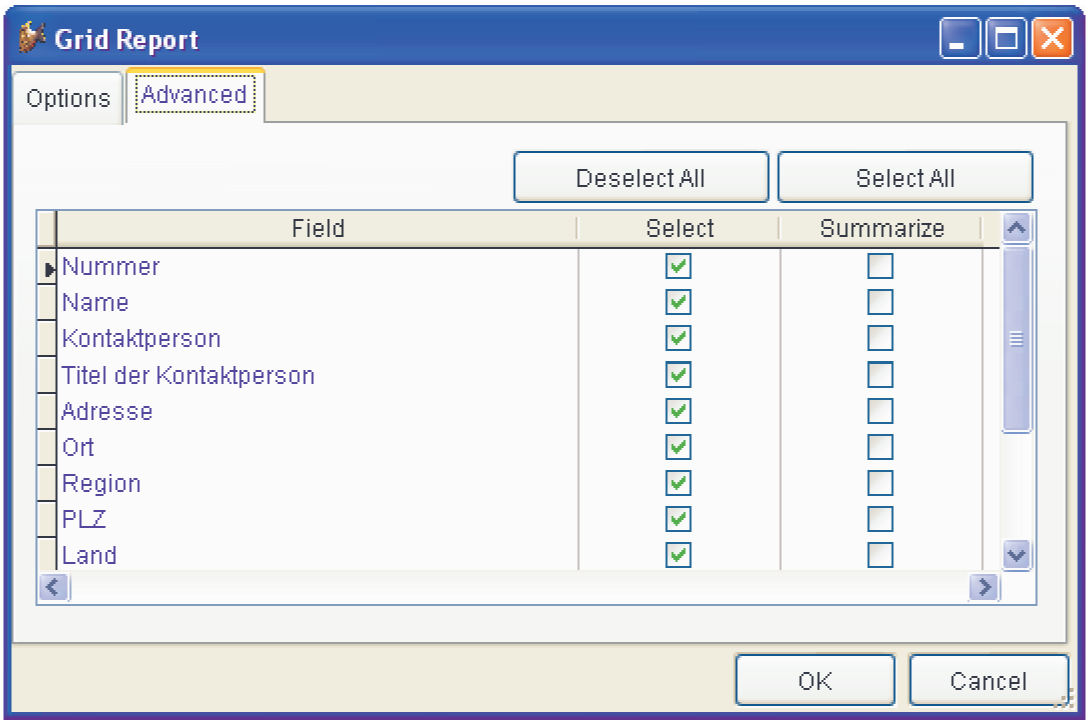
✓ Page number

not on first page

✓ Date ✓ Time

OK Cancel

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.


Grid Report

Options | Advanced

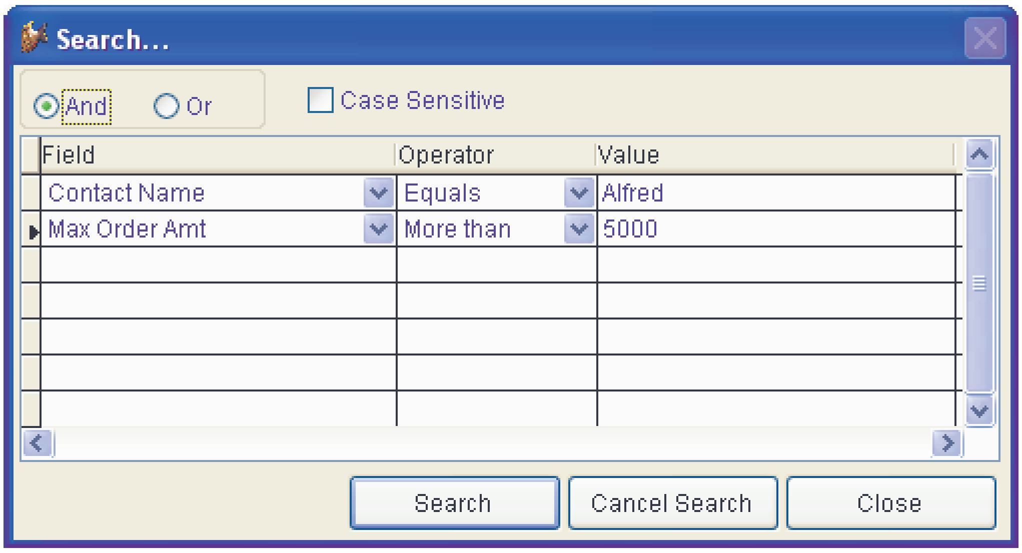
Deselect All Select All

Field	Select	Summarize
Nummer	✓	☐
Name	✓	☐
Kontaktperson	✓	☐
Titel der Kontaktperson	✓	☐
Adresse	✓	☐
Ort	✓	☐
Region	✓	☐
PLZ	✓	☐
Land	✓	☐

OK Cancel

9.6. 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.



[✓ And] Or

Case Sensitive

Field	Operator	Value
Contact Name	Equals	Alfred
Max Order Amt	More than	5000

Now you can combine many different search criteria. The search criteria will be saved per user and form. In addition, are available to be used on the following application executions.

10. Application Protection using Activation Keys

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 hanged in Vfxmain.prg. The property *goProgramm.lUseActivation* must be set to .T. in order to enable product activation. When the property *goProgramm.lUseActivation* 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.

10.1. 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. In addition, 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.

10.2. 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 properties could have 3 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 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 "First Install file" 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 "First Install file" will be deleted. If an unfair user tries to re-activate the application by deleting the INI file, application will be terminated, if "First Install file" 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)..

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*.

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. The developer uses this installation key in the *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 *cVFXAcvtivation* (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 *cVFXAcvtivation* (Appl.vcx). The default value is *FirstInstall.txt*

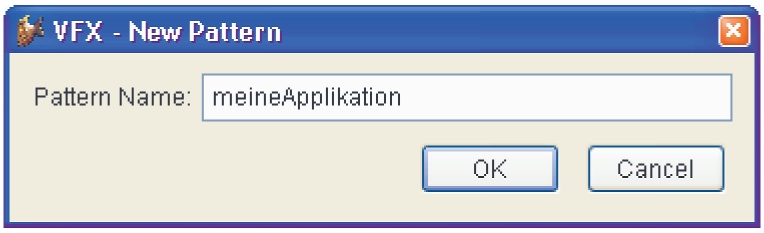
If the file *FirstInstall.txt* will be used, it must be deployed along with the applications' installation files. The installation application must store this file in the Windows folder and the activation object will erase this file when the application is started for first time. At this time, the installation 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*.

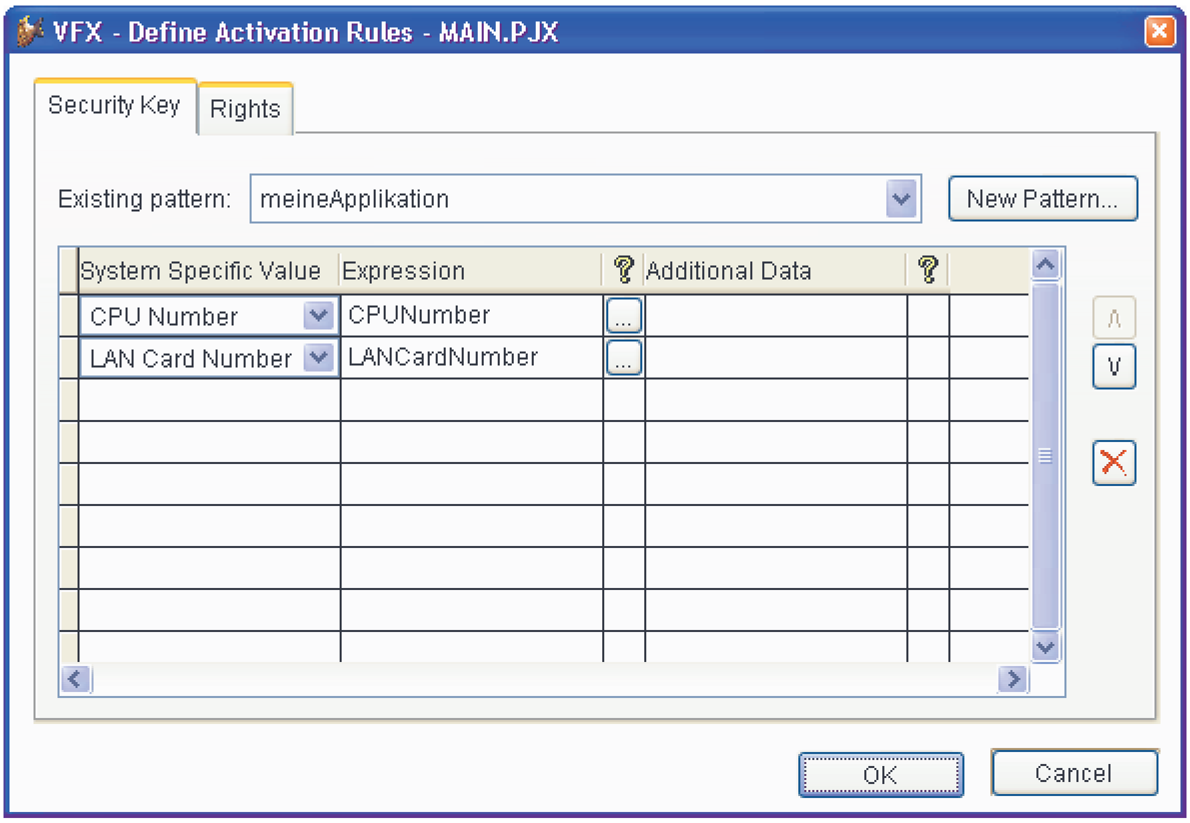
10.3. Defining the activation Rules

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.



Pattern Name: meineApplikation

After entering a name for the rule, *Define Activation Rules* Wizard is started.



Existing pattern: meineApplikation

System Specific Value	Expression		Additional Data	
CPU Number	CPUNumber	...		
LAN Card Number	LANCardNumber	...		

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 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 should 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 is automatically filled in the second column of the Grid. To use only the last 4 digits, you have 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“

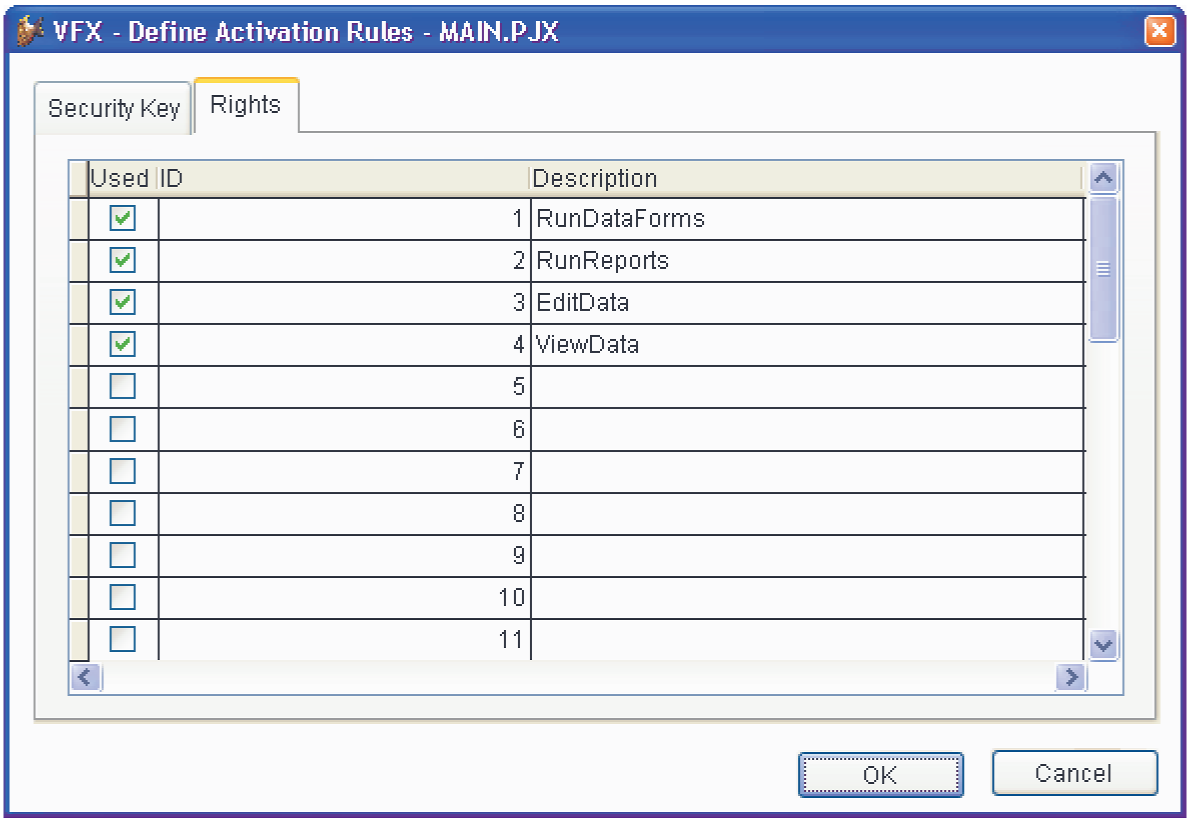
On the user’s PC will be created an Installation key from the activation rule. All parameters, included in the Activation rule will be used. If just one of these parameters of the PC changes, the application registration becomes invalid and the user need to obtain a new activation key, corresponding 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.


VFX - Define Activation Rules - MAIN.PJX

Used	ID	Description
✓	1	RunDataForms
✓	2	RunReports
✓	3	EditData
✓	4	ViewData
☐	5	
☐	6	
☐	7	
☐	8	
☐	9	
☐	10	
☐	11	

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.

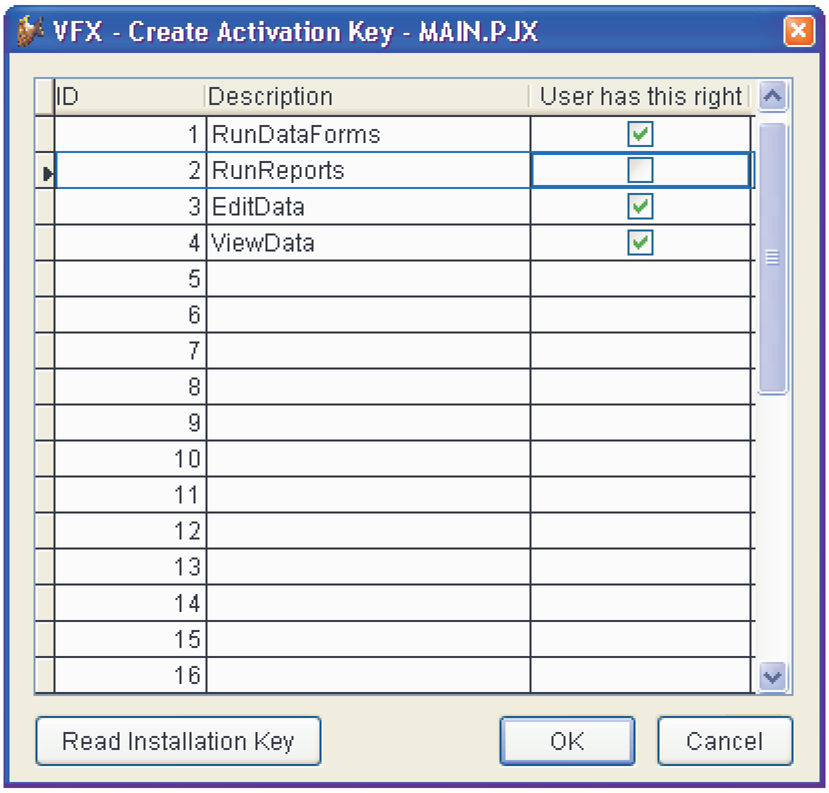
To define a right in the wizard, first must be marked the checkbox in the first column. Then a name for the right should be entered. At run-time, the application will create a property of the *SecurityRights* object with this name.
Because of this, the names should conform to VFP rules for variables 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.

10.4. Activation key generation

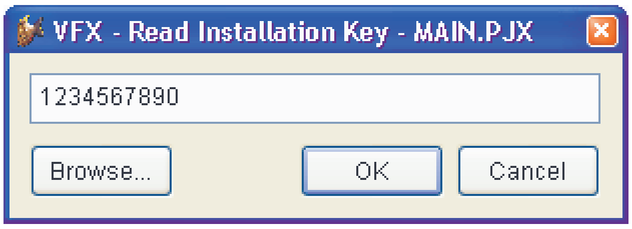
When the user sends his installation key, must be generated an Activation key. This Activation key holds information for the application, whether the user is allowed to perform a particular action. For every action must be chosen correspondent user right.

When in the VFX 8.0 menu is chosen the option “Create Activation Key”, appears the Dialog with the user rights, defined for the active project.


VFX - Create Activation Key - MAIN.PJX

ID	Description	User has this right
1	RunDataForms	✓
2	RunReports	☐
3	EditData	✓
4	ViewData	✓
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

With the button „Read Installation Key“ is opened a dialog, where the Installation key of the user is entered. The Installation key can be inserted using the clipboard or can be read from a file.


VFX - Read Installation Key - MAIN.PJX

1234567890

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*, either the generated Installation key is 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*.

10.5. 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*.

11. Advanced development techniques

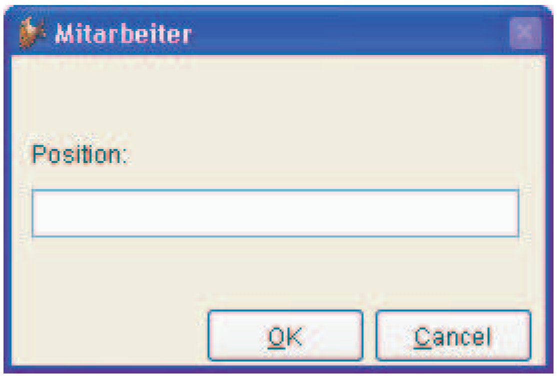
11.1. *Forms based on views*

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. If the Datasource of a form should be View, on the page Options in the VFX-form builder the option ”Work on View” must be checked. With this VFX knows that through the Datasource it handles a view. 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.

11.1.1. Entering the view parameter – CAskViewArg

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 *lworkonview* 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 written 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
```

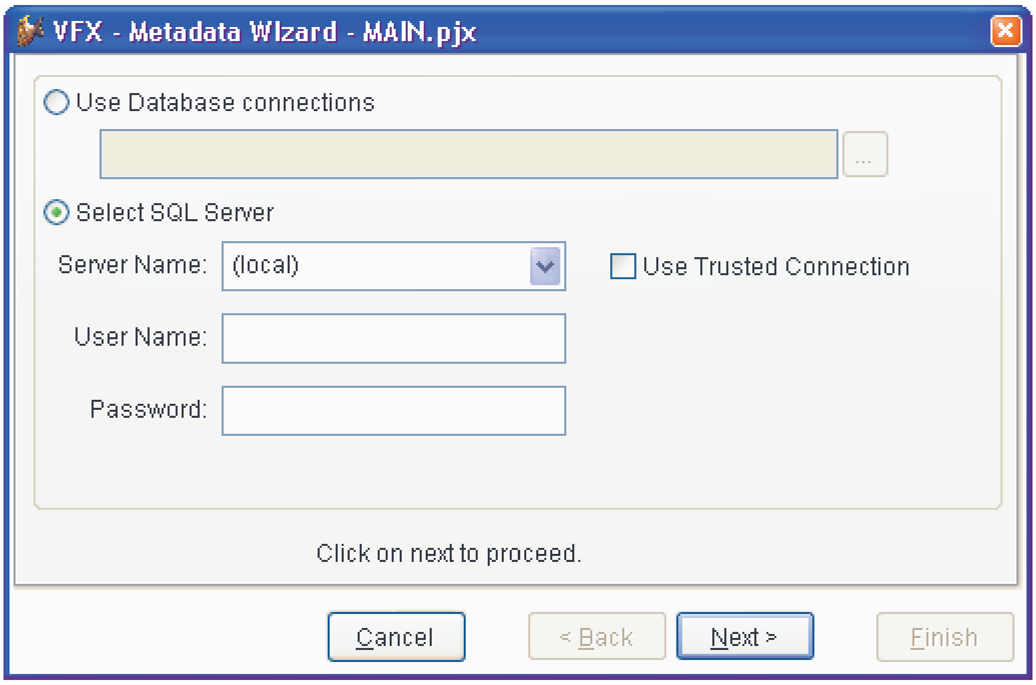
It is also possible the form for input of the view parameters to be called again at run-time. If the call is made from a control, for example from the Click Event of a button, the call must look in this way:

```
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.

11.2. *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.


VFX - Metadata WIzard - MAIN.pjx

( ) Use Database connections

[✓ Select SQL Server]

Server Name: (local)

[ ] Use Trusted Connection

User Name:

Password:

Click on next to proceed.

11.3. Delayed Instantiation

The load time of a form essentially depends on the number of controls, which must be loaded with the form. 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', '< ClassName >')
```

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 individual controls must be changed. Before the conversion to Delayed Instantiation, it could refer to a text box for example like this:

```
Thisform.pgfPageframe.Page1.txtMyTextbox
```

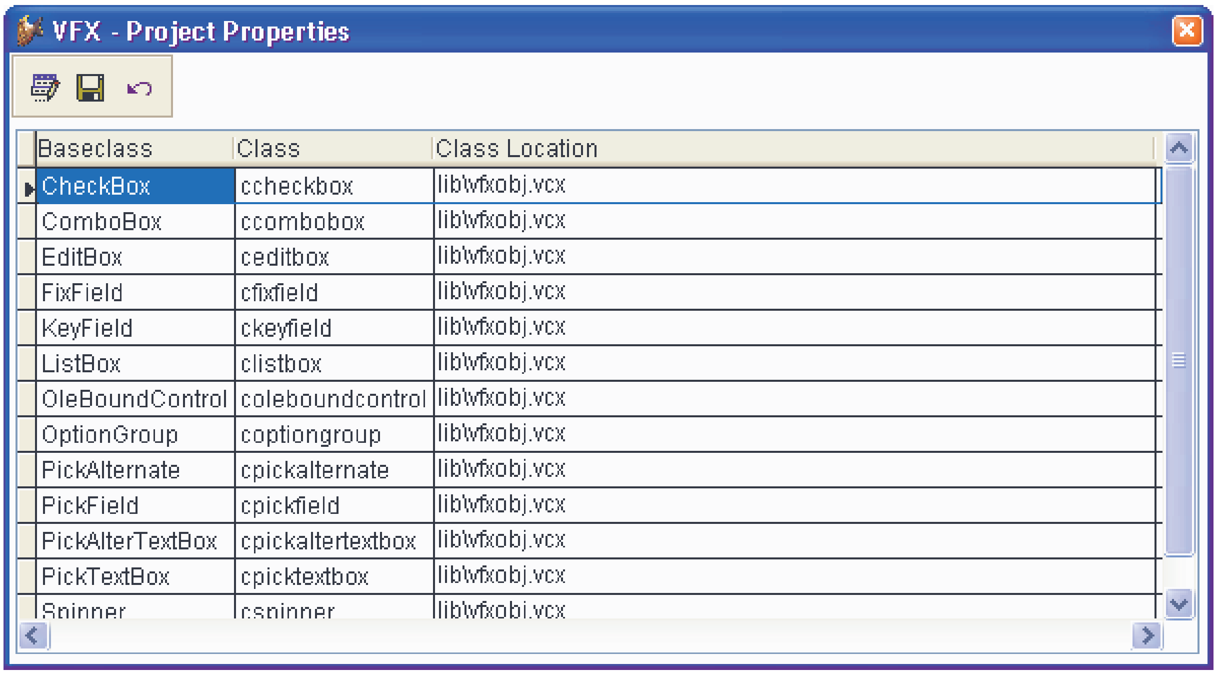
After the conversion to Delayed Instantiation, the reference looks in this way:

```
Thisform.pgfPageframe.Page1.x. txtMyTextbox
```

The x here is the name of the container, in which are the controls of the page.

11.4. 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 individual 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.



Baseclass	Class	Class Location
CheckBox	ccheckbox	lib\vfxobj.vcx
ComboBox	ccombobox	lib\vfxobj.vcx
EditBox	ceditbox	lib\vfxobj.vcx
FixField	cfixfield	lib\vfxobj.vcx
KeyField	ckeyfield	lib\vfxobj.vcx
ListBox	clistbox	lib\vfxobj.vcx
OleBoundControl	coleboundcontrol	lib\vfxobj.vcx
OptionGroup	coptiongroup	lib\vfxobj.vcx
PickAlternate	cpickalternate	lib\vfxobj.vcx
PickField	cpickfield	lib\vfxobj.vcx
PickAlterTextBox	cpickaltertextbox	lib\vfxobj.vcx
PickTextBox	cpicktextbox	lib\vfxobj.vcx
Spinner	cspinner	lib\vfxobj.vcx

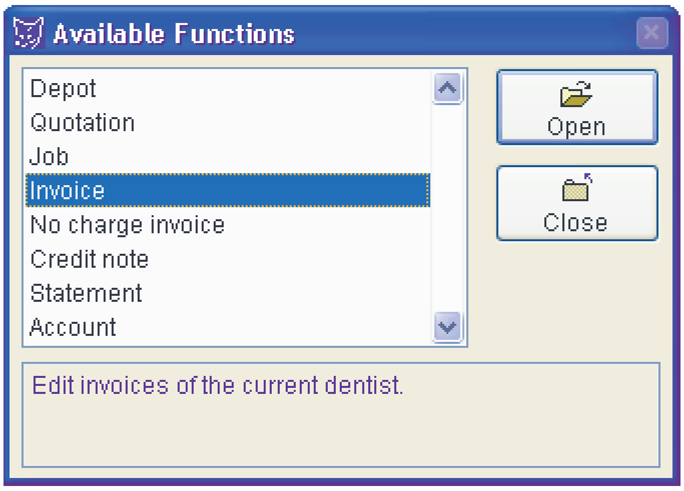
11.5. Important VFX – Methods

11.5.1. 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.

11.5.2. 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.



Depot
Quotation
Job
Invoice
No charge invoice
Credit note
Statement
Account

Edit invoices of the current dentist.

In the Onmore-method at run-time will be shown a Dialog, in which the user can select the Child-form that will be called.

11.5.3. 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.

11.5.4. 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.

11.6. VFX primary key generation

It is possible you not want to show the primary key of some tables to the users. However, 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 record new data. 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.

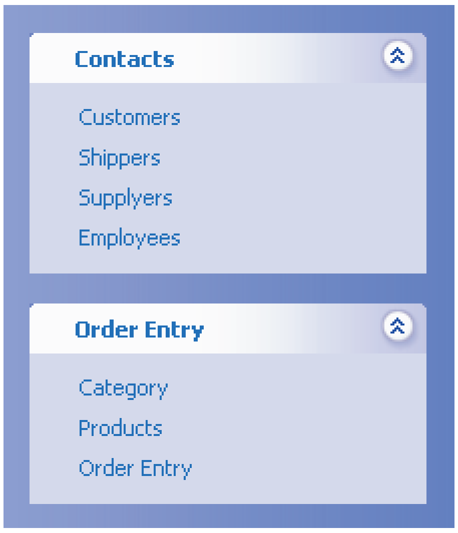
```
DODEFAULT()
REPLACE comp_id WITH GetNewId('CUSTOMER') IN customer
```

The last generated key value is stored in the table VFXSYSID.

11.7. 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 in 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.

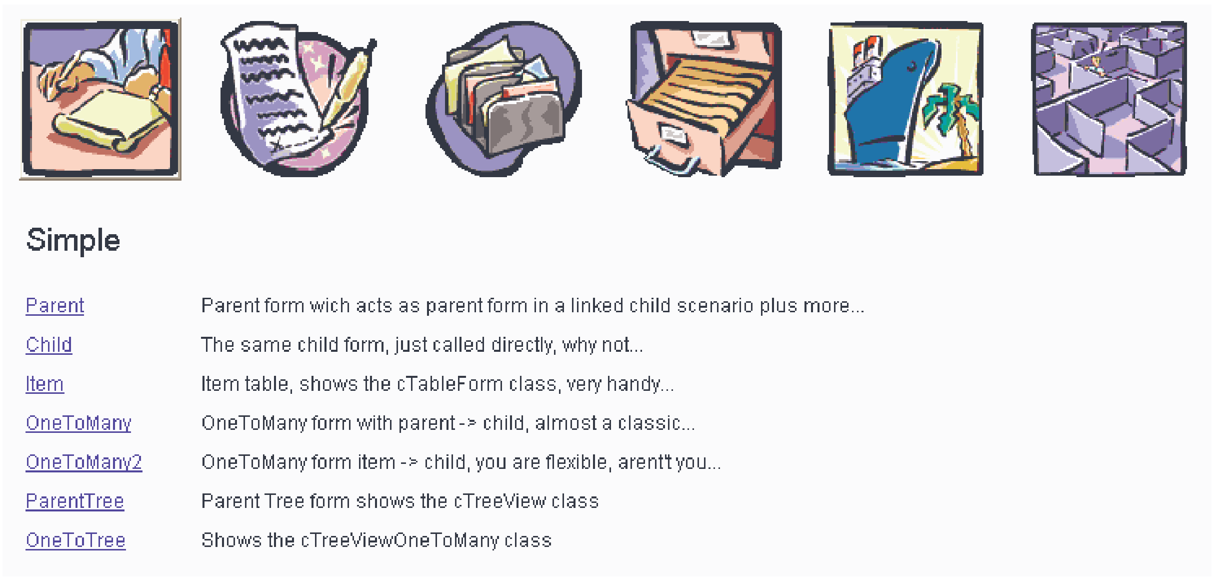
The File/Open-Dialog uses the table VFXFOPEN.DBF. For each form, the VFX-Form-Builder adds automatically a data record in the table VFXFOPEN.DBF. Here is the structure of the table VFXFOPEN.DBF:

VFXFOpen-Field	*Description*	*Example*
ObjectID	This field is used, when the Open-Dialog Vfxfopen.scx is used. For this the property goprogram.lxpopenstyle must be set to .F. Usually the VFX Open-Dialog has two pages. (Tip: 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 Vfxopen.scx.) 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
ObjectNo	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
GroupCap	This field is used when the Open-Dialog Vfxxpopen.scx is used. For this purpose, the property goprogram.lxpopenstyle 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
Title	Enter the title, which appears in the list window.	Customers
Descr	Enter a description text, which will be shown, when the user selects this entry.	Address list
Form	Enter the names of the called forms.	ADRE
Parameter	If you want to pass parameters to the form, you can enter them here.	
Viewlevel	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)
NewLevel	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)
EditLevel	The user level, which is required in order to be able to edit the data records	1 (only administrators can edit the data records)
Eraselevel	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)

11.8. 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.



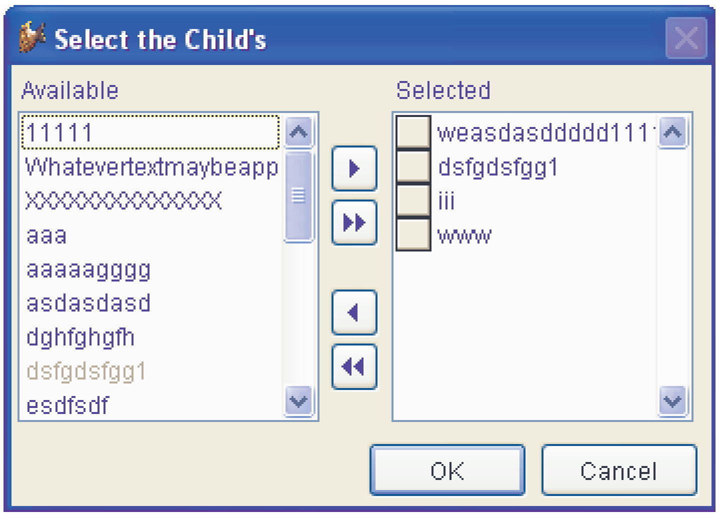
Simple

Parent	Parent form wich 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, arent'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.

11.9. Using the VFX Mover dialogs

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.



Available: 11111, Whatevertextmaybeapp, XXXXXXXXXXXXXX, aaa, aaaaagggg, asdasdasd, dghfghgfh, dsfgdsfgg1, esdfsdf

Selected: weasdasddddd111, dsfgdsfgg1, iii, www

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.
```

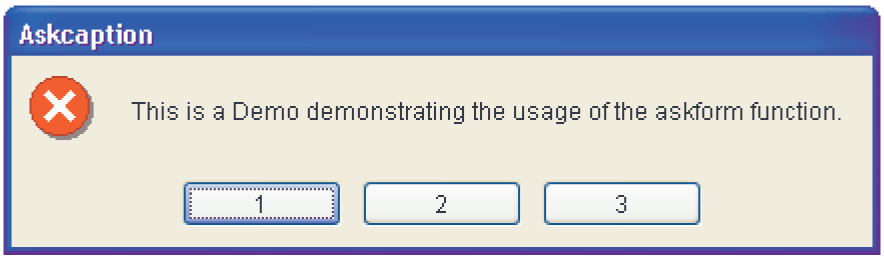
```
*--pass the array of all available items
*  here you can also pass a second parameter if you want to define, which
*  elements from the array must appear as already selected
loMover.cntMover.SetData(@laSource)
*--show the mover dialog
loMover.Show()

*--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 Library including all properties and methods, please read in the VFX technical reference.

11.10. 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.


Askcaption

This is a Demo demonstrating the usage of the askform function.

An example for using of the function *Askform()* is in the form *Parent.scx* in the demo application VFX80Test.

11.11. IDX Know How

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 Vfxmain.prg:

```
lremakeidxafterclear = .T. && set the Index again after data
                                                            manipulation.
```

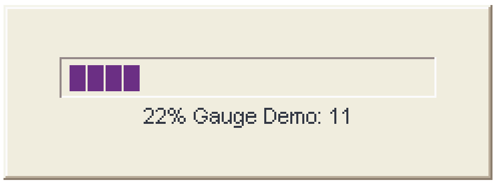
If no transactions in the used tables are invoked in a form and any code called from it, and no RI code is placed, you can change Vfxmain.prg not to delete temporary index files during the data manipulation:

```
lnoclearidxonedit = .t.    && do not clear Index when editing data.
lnoclearidxoninsert = .t.  && do not clear Index when inserting data
                                                               rows.
lnoclearidxondelete = .t.  && do not clear Index when deleting data
                                                               rows.
```

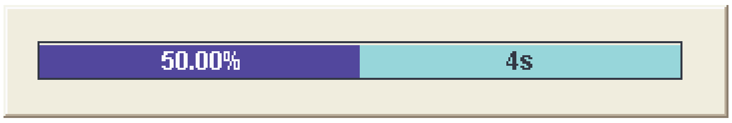
Temporary index files are deleted every time when a form. is closed.

11.12. Progress bar

VFX offers 2 possibilities to indicate the progress of lengthy operation. 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.

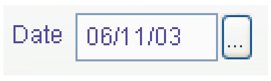


Examples for the use of both progress indicators are in the form Parent.scx of the demo application VFX80Test.

11.13. Date selection

11.13.1. cPickDate class

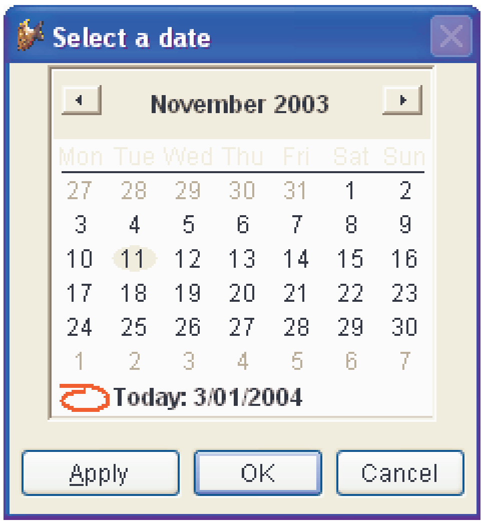
The class *cPickDate* contains a text box for entering a date as well as a button for calling a calendar.



In the Textbox for selection of a date are available the following hotkeys:

+	Next day
-	Previous day
H, h	Today
B, b	The first day (beginning) of the current month
L, l	The last day of the current month
A, a	New Year 's day
E, e	End of the year
V, v	Previous month
N, n	Next month

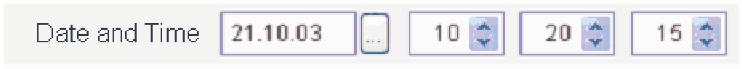
For the calendar is used the Microsoft MonthView ActiveX control. When creating a Setup, this ActiveX control (Mscomct2.ocx) must be included also into the Setup. VFP 8 provides a Merge module for this.



Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

11.13.2. cDatetime class

Additionally the class cDatetime can be used for entering Datetime values.



This class contains a *cPickDate* control for input of the date. 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 the type datetime.

11.14. Choice of reports

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 individual user only.

An example how it can be used is in the form Reports.scx in the demo application VFX80Test.

11.15. The Microsoft Agents

The agents are nice characters, which loosen up the use of VFX applications.

In VFX80Test, the form Agent.scx shows simple examples for the possible usage.

11.16. Linked Child 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 Formclass can serve as Parent form. In addition, the Child form can be based on any VFX Formclass.

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.

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 the form property *lChildForm* to T..

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

11.16.1. Creating a form that calls a child form

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.

In the section about the VFX Form Builder we discussed the Checkbox named *Has More Functions*. If you mark this Checkbox, VFX Form Builder generates the following code in the *OnMore()* method of the form:

```
lparameters tnPassThrough

local lcCalledBy, lcFixFieldValue, lcCaption,;
         lcFixFieldName, lcFilterExpr

lcCalledBy      = ""
lcFixFieldValue = ""
lcCaption       = ""
lcFixFieldName  = ""
lcFilterExpr    = ""

local laFunct[1,5]

laFunct[1,1] = "<Function Title>"
laFunct[1,2] = "<Function Desctiption>"
laFunct[1,3] = "W"            && W - Wait Window, F - Form to run, M – Method of this form
laFunct[1,4] = "<FormName>"
laFunct[1,5] = lcCalledBy      + ";" +;
                         lcFixFieldValue + ";" +;
                         lcCaption       + ";" +;
                         lcFixFieldName  + ";" +;
                         lcFilterExpr

if alen(laFunct,1) = 1
       tnPassThrough = 1
endif

if empty(tnPassThrough)
       do form VFXMORE with laFunct, this
else
       do form VFXMORE with laFunct, tnPassThrough,this
endif

goProgram.RefreshWindowMenu()
```

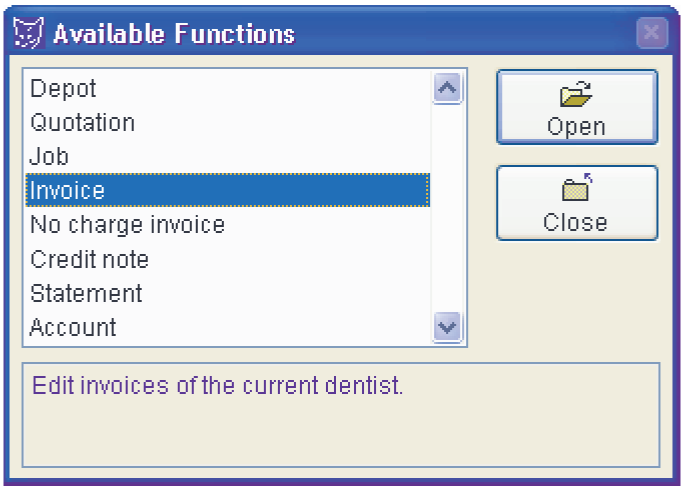
This code can look in such a way, when you adapt it to your needs:

```
lparameters tnPassThrough

local lcCalledBy, lcFixFieldValue, lcCaption,;
         lcFixFieldName, lcFilterExpr

lcCalledBy      = "PARENT"
lcFixFieldValue = PARENTID
lcCaption       = "Child records for " + trim(parent.descr)
lcFixFieldName  = "PARENTID"
lcFilterExpr    = "PARENTID='"+parentid+"'"

local laFunct[1,5]

laFunct[1,1] = "Child Records"
laFunct[1,2] = "Child Records for selected parent"
laFunct[1,3] = "F"            && W - Wait Window, F - Form to run, M – Method of this form
laFunct[1,4] = "CHILD"
laFunct[1,5] = lcCalledBy      + ";" +;
                         lcFixFieldValue + ";" +;
                         lcCaption       + ";" +;
                         lcFixFieldName  + ";" +;
                         lcFilterExpr

if alen(laFunct,1) = 1
       tnPassThrough = 1
endif

if empty(tnPassThrough)
       do form VFXMORE with laFunct, this
else
       do form VFXMORE with laFunct, tnPassThrough,this
endif

goProgram.RefreshWindowMenu()
```

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

- He can press the function key *F6*.
- He can select option *Other...* in the *Edit* menu.
- He 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:



Edit invoices of the current dentist.

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.

11.16.2. Creating a Child-form

The counterpart of a form, which calls another form, is the called form. As described in a preceding chapter, there can be different reasons, for which a form is called by another form.

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, VFX-forms store 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 = 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 = "<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 can 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.Page1.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;1234568890;Contacts for company DEAG;CONT_COMP_ID;UPPER(CONT_COMP_ID)= '1234568890'"
```

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 us see the form properties, which hold the information from the passed parameter character string *tcArg*:

VFX-Form property	***Description***	***Example***
cCalledBy	This character string indicates, from which form this form was called.	COMP
cFixFieldValue	The value of the field from the main table (**primary key in the main table**).	1234568890
cFixFieldName	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 (**foreign keys in the Child table**).	CONT_COMP_ID
cFilterExpr	The (at best) Rushmore optimized Filter expression, in order to show the data records that correspond the criteria of the main 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, the filter expression does not need to be passed.

11.17. The VFX-Resource tables

VFX applications use a VFX Resource Table (on a per user basis) to store all information about forms the user has called since the last initialization. This information is used to reset the size of the Form, the Grid layout, as well as the current Sort Order.

Here are the settings stored in the VFX Resource Table.

Setting	Description	Remarks
Position and size of the form.	The user sees the forms always appearing exactly the same way he left it.	Personal Form Setting. **Note** that this also applies to the picklists the user called!
Any layout changes applied to a grid.	The user sees the grid of any form exactly the way he left it. This applies to column width, as well as position and regardless of the type of column. (Also calculated fields work, not like within the FoxPro 2.x Resource File).	Personal Grid Setting. **Note** that this applies also to the picklists the user called and in the advanced one-to-many form with multiple child grids!
Actual sort within a Data manipulation form.	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 create unique named Index Files in the root directory of the application and deletes them when the user closes the form. **Note** that this applies also to the picklists the user called!
Position and state of the form toolbar.	If you have form-specific toolbars, the user sees them always in the same, position and docking state they left.	
Form Toolbar Hide	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 *View/ Toolbars* Menu Option.	

You can initialize 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.

VFX applications do NOT rely on FoxPro Resource Tables, like *FOXUSER.DBF/CDX*, they use their own VFX Resource Tables, which is the free table named *VFXRES.DBF/CDX*.

11.18. User specific Settings

VFX creates for every field in the *VFXUSR.DBF* table a public variable with the prefix *gu_* and handles automatically the save & restore for these values.

Assume you have a field called *Test* in the *VFXUSR* table. In that case, you will see a public variable called *gu_test*, which will pick up the value from the field *Test*, whenever the user logs on VFX writes the content of the public variable *gu_test* back to the table *VFXUSR* whenever the user logs off.

This way, the only thing you have to do to support user-specific settings, is add a field in the table *VFXUSR*.

11.19. The Include Files

Since include files play an important role in VFX Application Development, let us understand what types of Include Files VFX uses:

Include File	Included by	Location	Language specific	Content/Description
VFX.H	VFXMAIN.PRG	\VFX50\INCLUDE\	No	Sets the _DEBUG_MODE, _LANG_SETUP, _DBCX and other core constant and includes the other Include Files
FOXPRO.H	VFX.H	Visual FoxPro HomeDir	No	Standard FoxPro Definitions
VFXDEF.H	VFX.H	\VFX50\INCLUDE\	Yes	Sets the ID_LANGUAGE constant and defines other not language specific VFX constants
VFXTXT.H	VFX.H	\VFX50\INCLUDE\	Yes	Language Specific Captions and Tooltip Text used in VFX User Interface
VFXMSG.H	VFX.H	\VFX50\INCLUDE\	Yes	Language Specific Messages used in VFX User Interface
VFXOFFCE.H	VFX.H	\VFX50\INCLUDE\	No	Used in the Office Classes Word, Excel and Outlook
USERDEF.H	VFX.H	\VFX50\INCLUDE\	No	Language independent Constants used in your own Application
USERTXT.H	VFX.H	\VFX50\INCLUDE\	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.
USERMSG.H	VFX.H	\VFX50\INCLUDE\	Yes	Language Specific Text for messages used in your own

				Application. Optionally generated by the VFX message editor, when type MESSAGE has been selected.

The application wizard generates most of the constants automatically when you generate a new application. You have to make changes in some of the VFX Include Files if you want to change the debugging mode or the current language.

11.19.1. Define _Debug_Mode

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 do not 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.
…
```

11.19.2. Define ID_Language

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.

11.19.3. Define _Lang_Setup

In the Include-file *VFX.H* defined the constant *_LANG_SETUP*, which controls whether the code in the *LangSetup()*-Method will be executed or not. In the *LangSetup()*-Method will be checked if this constant exists. Only if the constant exists, the code written in the *LangSetup()*-Method will be executed. The purpose of this is the speed optimization in all forms.

```
…
#DEFINE _LANG_SETUP           .T.
…
```

11.19.4. Compile your Application after changes in your Include Files

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. You should include the file *VFX.H* into your forms, if you use constants in your forms

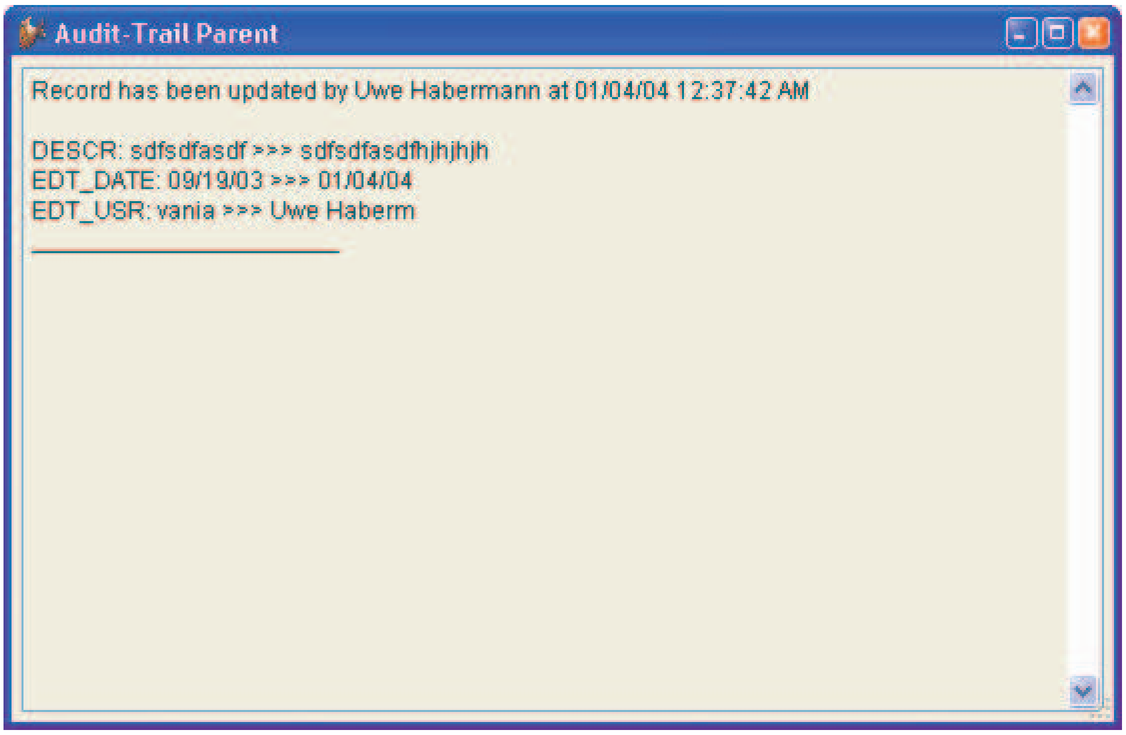
11.20. Data manipulation tracking

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 put 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()
```



Record has been updated by Uwe Habermann at 01/04/04 12:37:42 AM

DESCR: sdfsdfasdf >>> sdfsdfasdfhjhjhjh
EDT_DATE: 09/19/03 >>> 01/04/04
EDT_USR: vania >>> Uwe Haberm

On a button in the standard Toolbar can be viewed the activity log for the current data record.

11.21. 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.

If you like, you can also switch OLE drag & drop for particular controls. By default this feature is switched off and sung this line

```
nOLEenableDrag=1  && 0 use form setting (default), 1 enable, 2 disable
```

in Vfxmain.prg, it can be switched on.

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 this line

```
nPageOLEdragdrop=1  && 0 use form setting (default), 1 enable, 2
disable
```

in Vfxmain.prg.

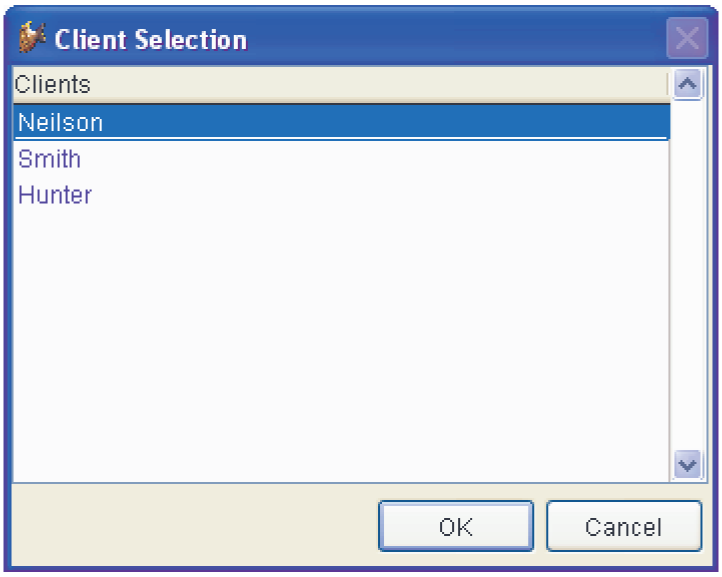
11.22. Multi-Client-Support

Usually VFX-Application works with just one database, as it was set in the VFX – Application Wizard. If you need, you can build client-processing feature. Therefore, in Vfxmain.prg, set the data path to an empty string.

```
cdatadir = ’’
```

When the data path is empty, at run-time, VFX-Application searches the table Vfxpath.dbf. This table must be in same folder where the executable program file is located. 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.



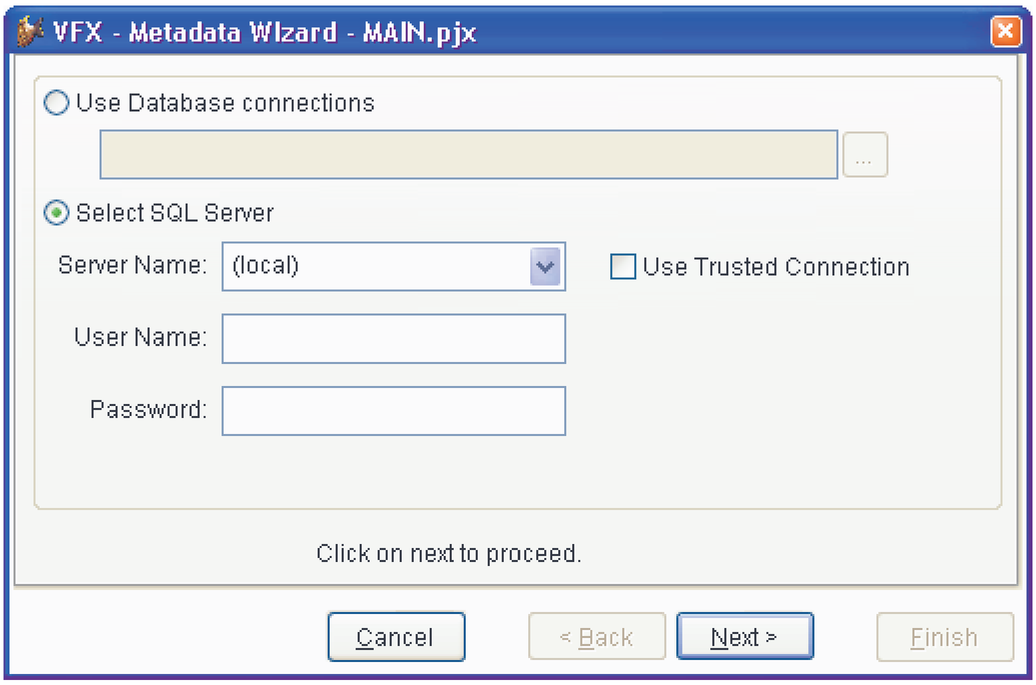
11.23. Updating the user's database

11.23.1. VFP-Databases usage

VFX gives the opportunity automatically to update the database at customers. 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.

11.23.2. SQL Server-Databases usage

The Metadata Wizard helps you to prepare meta-data definition of your currently used database structure. The Metadata can be used for database update at customer's side.


VFX - Metadata Wizard - MAIN.pjx

Use Database connections

Select SQL Server

Server Name: (local)

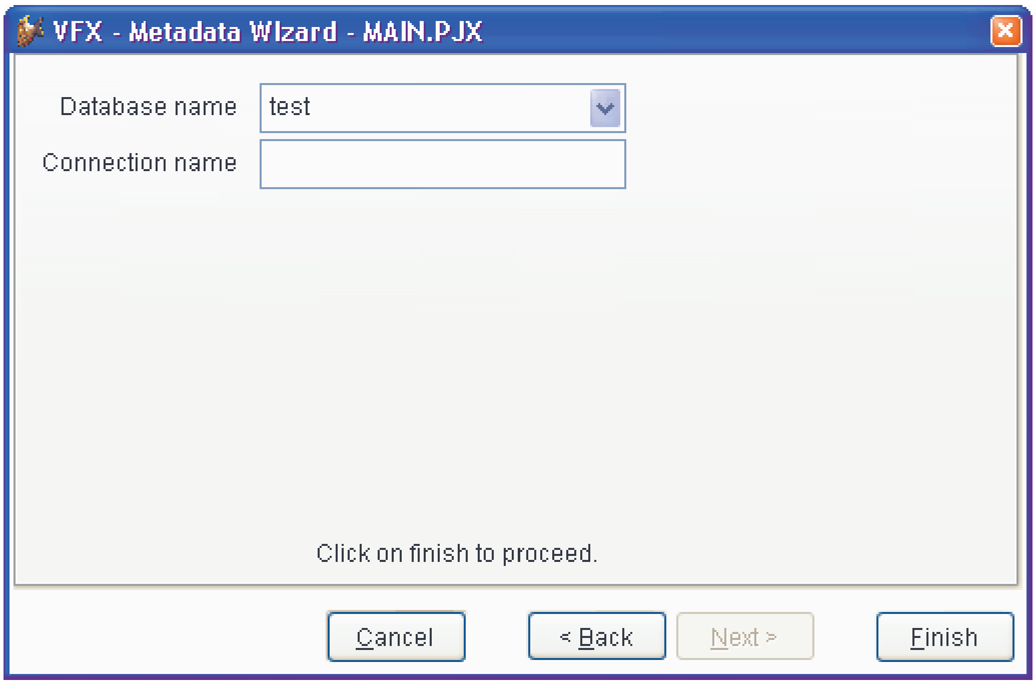
Use Trusted Connection

User Name:

Password:

Click on next to proceed.

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.


VFX - Metadata Wizard - MAIN.PJX

Database name: test

Connection name:

Click on finish to proceed.

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 is deployed to end-users the data structure update will use it and again scanning existing connections will update user’s database structures.

11.24. Hooks

An elegant way to intervene in the sequence of functions of VFX methods, without having to change the classes, is the use of Hooks.

The concept of the Hooks was extended in VFX 8.0. 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 8.0 the return value of the method of the Hook can be additionally controlled.

Hooks are stored in the file Vfxhook.prg. Using Hooks can be enabled in VFX-Application Manager or in Vfxmain.prg with following line:

```
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
```

11.25. 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, 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 had 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 child tables 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. Do not 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 after save"!

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!!! Do not expect a German version if the include files are in English.

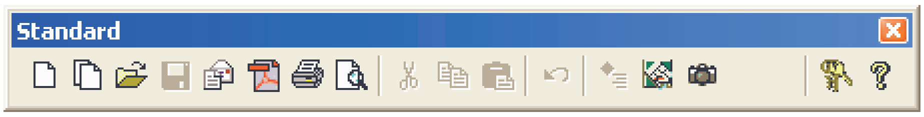
11.26. Use the Main Toolbar you like

It is 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:

CAppBar and *CAppNavBar*.

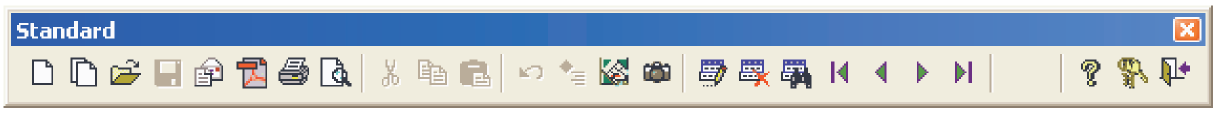
The first is the standard toolbar and the second is one which you may use if you do not want to have the navigation and other buttons on the form.

CAppBar:



CAppBar 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.

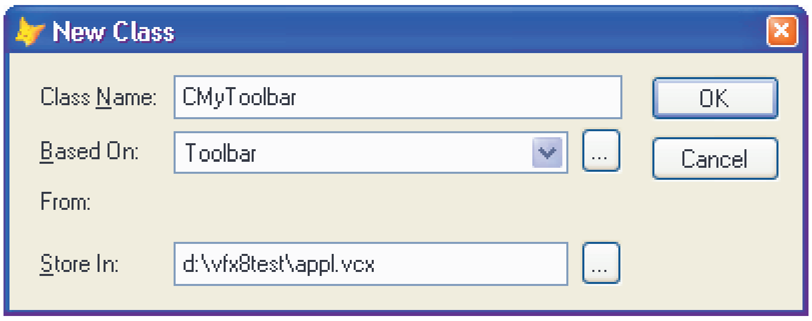
To switch from one main toolbar to another is as simple as switching one property in the application class in *VFXMAIN.PRG*:

```
define class CApplication as CFoxApp

        …
        …
        ****************************************************************
        ** CAppToolBar - Tooolbar without Navigation Buttons
        ** CAppNavBar  - Tooolbar with    Navigation Buttons

        cMainToolBar = "CAppNavBar"
        …
        …
        …
```

11.27. Create your own Toolbar Class

You could use the *CAppBar* or *CAppNavBar* toolbar class, to develop most of your office compatible applications. However, of course, you can create also other toolbars. All you have to do is create a new class, which inherits from the *CToolbar* class or even from the *CAppBar* or *CAppNavBar* class. Here is described how:

Select *New* while in the project manager on the class tab or on any class library object, you will see the following Dialog:


New Class

Class Name: CMyToolbar

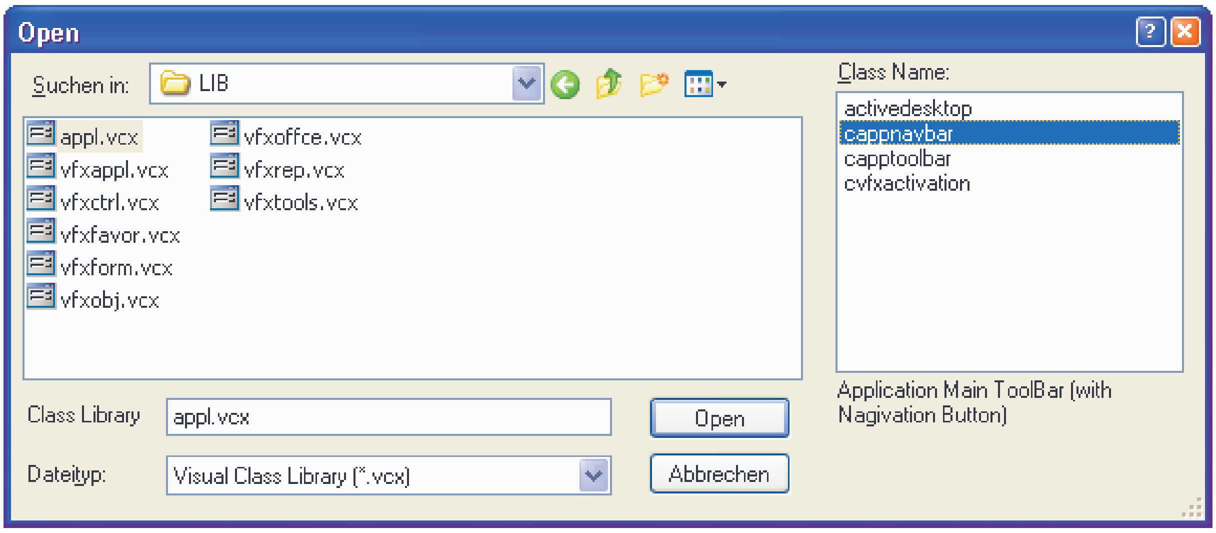
Based On: Toolbar

From:

Store In: d:\vfx8test\appl.vcx

Class Name: Enter the name of the new class. Assume we call it *CMyToolbar*.

Based On: Click the three-dot button, and in the following Open Dialog, select the Class *CAppbar* (or *CAppNavBar*) from the VFX Class Library *APPL.VCX*.


Open

Suchen in: LIB

appl.vcx, vfxappl.vcx, vfxctrl.vcx, vfxfavor.vcx, vfxform.vcx, vfxobj.vcx, vfxoffce.vcx, vfxrep.vcx, vfxtools.vcx

Class Name: activedesktop, cappnavbar, capptoolbar, cvfxactivation

Application Main ToolBar (with Nagivation Button)

Class Library: appl.vcx

Dateityp: Visual Class Library (*.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 three dot button (GetFile Dialog).

11.28. Modify your Toolbar Class

Now, you have to modify your toolbar class. Do this using the Visual Class Designer.

11.28.1. 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.

11.28.2. Add a Custom Icon

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 form. In this case, we place the code

```
goProgram.RunForm("CUSTOMER")
```

into the *Click()* Event.

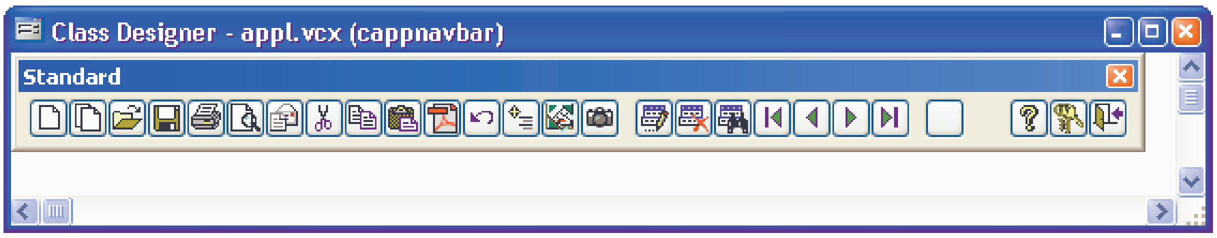
Picture: Select a *BMP* or *ICO* file to be used on your toolbar command button.

NOTE: Make sure to put 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 is 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.

11.28.3. Sample Application-specific Toolbar



11.29. Toolbars for Forms

It appeared to be very practical to be possible to assign forms their own toolbar. The toolbar should be based on the class *ctoolbar* and should be stored in the class library Appl.vcx. The name of the toolbar should be entered in the property ctoolbarclass of the form.

VFX initializes the toolbar along with the form. The toolbar is visible as long as the form is active.

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 symbol border 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

11.30. Properties of the class cApplication

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 cFoxapp from the classlibrary Vfxappl.vcx. The values of the properties can be set in Vfxmain.prg under the line DEFINE CLASS capplication AS cFoxapp. In the same way, here can be changed or overwritten methods of the class. In particular, the properties of the Application object that are 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. In this case, the records inserted in the table *Vfxpath.dbf* will obtain the correspondent data path.

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 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*

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 name 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 be entered 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 name 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 splash screen. 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 splash screen. Change of this value is necessary only if a Splash screen 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. The default value is *Untitled*.

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 *CAppBavBar* 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.

lNoClearIdxOnInsert – 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.

lRelogonQuit – Controls the behavior of the application when a 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.

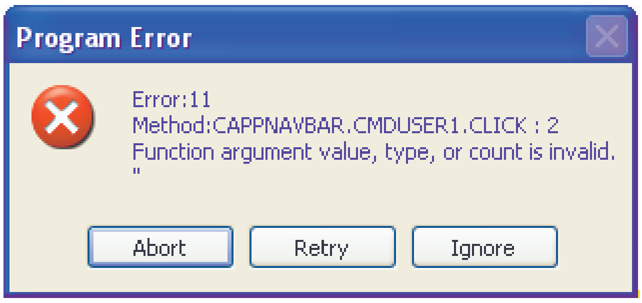
lRemakeIdxAfterClear – When the value of this property was set to .T., the temporarily created index files will be automatically recreated after the save procedure finishes. Compare also with the properties lNoClearIdxOnDelete, lNoClearIdxOnEdit, lNoClearIdxOnInsert. The default value is .F.

In VFX 8,0 of the application object class was extended by a number of new pro0perties for controlling the behavior of the application in case of an error, for implementing the product activation and for the Postscript printer driver installation, which is needed for the creation of pdf files.

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

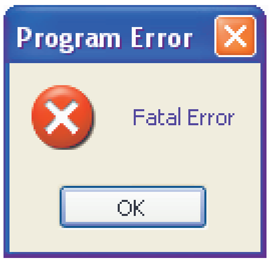
0 – Run-time error will be ignored

1 – An error message will be displayed (Default).


Program Error

```
Error:11
Method:CAPPNAVBAR.CMDUSER1.CLICK : 2
Function argument value, type, or count is invalid.
"
```

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


Program Error

Fatal Error

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 cal stack information (Default).

2 – Complete, detailed error information.

PSPrinterToInstall – This property holds 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.

lUseActivation – 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 vfxmain.prg. The default value is .F., the product activation will not be implemented.

lActivationType – 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 is new project is created. Later its value can be changed in vfxmain.prg. The default value is .F., the existence of the file „FirstInstall.txt" will not be checked.

11.31. 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 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.

11.31.1. 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

11.31.2. Methods

ExecMacro(vcMacro, lnNoRun)

vcMacro – Macro-language script to be executed

lnNoRun – When this property is set to .T. the downloaded file will not be run

11.31.3. 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 lnNoRun parameter is .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 open 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 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.

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

The listed keycodes will be Put 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

11.31.4. Example

Explanations for the Installation of the Ghostscript:

D: ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/AFPL/gs800/gs800w32.exe
Downloads the file gs811w32.exe from the Internet and finally runs it.

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 keycode “Enter” to the active window. This will start the files extraction.

C: 60; .F.; .F.; .F.; AFPL Ghostscript Setup

Waits until the window with the caption “AFPL Ghostscript Setup” appears.

K: 43

Sends the keycode “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.

11.32. 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.

11.32.1. 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

11.32.2. Methods

AddAttachment(tsAlias, tcFileName, tcReport, tcFor)

Fills the CreatePDF object information about attachment files. A PDF file according passed parameters will be automatically created. An optional expression can be passed as parameter. This expression will be used to filter data, included in the report

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 in *LastErrorNo* a *LastErrorText* properties of the class.

11.33. 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 reports in PDF format.

11.33.1. 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

oEmail_Attachment – This property will be used internally. It contains a collection of added attachments.

11.33.2. 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 prepare PDF file.

tcFor **–E**xpression, used as FOR clause, to filter exported to PDF records.

Send_Email_Report(tcEmail, tcSubject, tcText)

Send 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. In addition, the files, which you have previously created and passed to the AddAttachment method without report name and alias name parameters will be attached.

11.34. 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 10th of September 2003 the name of the created archive will be Data20030910.ZIP.

11.34.1. Properties

OverrideFile – With this property will be defined 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

11.34.2. Methods

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 CreateZipArchive function (in VFX.FLL)

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.FLL).

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

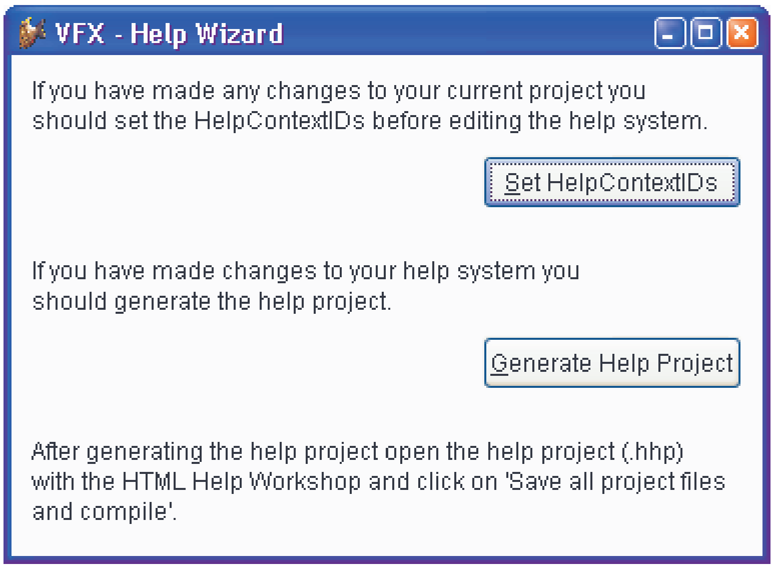
nUnZIPedFilesSize – Total size of part of archive already extracting

Return value: 0 – Cancel entire operation. 1 – Do not extract current file. 2 – Overwrite existing file.

11.35. VFX – Help Wizard

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

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


VFX - Help Wizard

If you have made any changes to your current project you should set the HelpContextIDs before editing the help system.

If you have made changes to your help system you should generate the help project.

After generating the help project open the help project (.hhp) with the HTML Help Workshop and click on 'Save all project files and compile'.

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

11.36. 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.

12. VFX.fll

The file VFX.fll contains several functions, needed for product activation, data backup as well as for Internet access. The VFX.fll must be deployed along with the application files to the customers. The functions of the VFX.fll are described in details.

12.1. 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 postscript printer drivers. An example for implementation of this function can be found in the class *cCreatePDF* in the method C*heckPSPrinter.*

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 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 C*heckPSPrinter.*

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*

cStringForEncripting – String to be encrypted.

cPassword – Password, to be used for encryption.

Return Value: Crypted string

Decrypt(cStringForDecripting ,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*

cStringForDecripting – 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 the Internet exists.

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.

12.2. 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*.

12.3. Data backup and archiving

CreateZipArchive(cPath , cFileMask, cArchiveFullPathName, cFeedBackFunction) – Creates a Zip archive file. An example for the implementation of this function can be found in the class *cArchive* in the method *CreateArchive*.

cPath – Path to the folder to be archived.

cFileMask - Names of the files to be archived.

cArchiveFullPathName – Full pathname for the created Zip archive.

cFeedBackFunction – Name of a function or a method, which will be called from *CreateZipArchive* for progress FeedBack.

cFeedBackFunction(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 the archive.

nZIPedFilesSize – Total size of files added to archive so far.

nArchiveCurrentSize – Current size of the archive.

Return Value:
0 – Cancel archiving operation.
1 – The files will be added to archive.
2 – Continue archiving operation.

ExtractZipArchive(cExtractFilesFolder, cArchiveFullPathName, cFeedBackFunction) 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.*

cExtractFilesFolder – Folder, where files extracted from archive will be stored.

cArchiveFullPathName – File pathname for the Zip archive.

cFeedBackFunction – Name of a function or a method, which will be called from *ExtractZipArchive* for progress FeedBack.

cFeedBackFunction(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 be extracted.
5 – Extraction operation completed successfully .
6 – Extraction operation did not complete successfully.

Return Value:
0 – Cancel extraction operation.
1 – Continue extraction operation.
2 – Overate existing file with the file from archive.

12.4. SQL Server

GetSQLServers(@cServersString,@cEerrorString) – 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 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.

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. Here we describe the design time localization process.

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 do not 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 languages *German, English, French, Italian, Spanish, Greek, Bulgarian* or *Czech*. If you need the Visual Extend class library in another language, you can easily extend VFX 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 than *German, English, French, Italian, Spanish, Greek, Bulgarian* or *Czech* 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* message and caption table).

√ 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 do not 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.

14. How to port an existing VFX 7.0 Application to VFX 8.0

Existing VFX 7 - applications can be ported easy to VFX 8.0. Since between VFX 7 and VFX 8.0 much changed, it is recommendable first to create a new empty project with VFX 8.0.
This new project will be created in a new folder, but obtains names and in particular the database name of the old project. All properties in the VFX - Application Wizard should be entered as for the old VFX 7 project.

In the new VFX 8.0-Project are also automatically placed the new VFX 8.0 menus, the extended Include files as well as the extended structures of the free tables. If you had extended the menu Vfxmenu of your old project, please make manual the extensions in the new menu. Only thus, remain the extended menu entries from VFX 8.0.

Look at the new *VFXMAIN.PRG* and make by hand the changes necessary for your project. Follow the documentation in the new *VFXMAIN.PRG*, in order to adapt the sample to your specific needs.

Now the database from the VFX 7-Project can be copied into the Data folder of the new project. The free VFX tables in the VFX 8.0-Project would be opened with *USE* and with *APPEND FROM* the data from the respective old VFX 7-Project table would be added. In this way, the data of the tables *Vfxfopen.dbf*, *Vfxsys.dbf*, *Vfxsysid.dbf* and *Vfxusr.dbf* will be fetched.

All forms and report would be copied from the respective folders, Form respective Report, into the new project and added manually in the VFP Project Manager. Finally still the file *Applfunc.prg* from the *Program* folder and the files *Appl.vc** from the *Lib* folder would be copied into the new project.

If you have defined your own constants into Include files, copy your Include files into the *Include* folder of the new project. Do not overwrite the Include files *Vfxmsg.h* and *Vfxtxt.h*, since they contain numerous new constants, which are needed by VFX 8.0.

After compiling of all files, the application is operational with VFX 8.0.

15. 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 helpfile. 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.

15.1. 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.com). 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)

16. 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 trough 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.

16.1. Your feedback is important for us!

Your feedback is highly appreciated! Mail us your feedback to 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 8.0 - More productive than ever!