
# VFX 9.5 – What is new?

January 2006


*Uwe Habermann, Venelina Jordanova*

# Table of content

# New features for developers

## *Inheritance architecture*

### Vfxobjbase.vcx

The inheritance architecture of the VFX classes is now extended. In previous VFX versions it has been possible to affect the functionality and in the layout of base VFX classes, only using hooks. If a developer wants, for example, to use a specific font in the whole application, this could be achieved only with hooks in the *Init* event. This has the disadvantage that in the VFP designers the application has been always displayed in the standard VFX font – Arial, and the selected font has been visible only at run-time by hooks.

There is an additional inheritance layer in VFX 9.5. The base VFX classes, available in previous VFX versions in the base *Vfxobj.vcx* class library, are now placed in *Vfxobjbase.vcx* class library. To the previous class names is appended a suffix *base*.

For every class from this first-level library there is a 1:1 inherited class in the class library *Vfxobj.vcx*. Now the class library *Vfxobj.vcx* is available to the developers for their own enlargements and enhancements. Here it is possible, for instance, to change the font of a class. Then this setting will affect all controls in the whole application based on this class.

By an updating the project with the VFX - Update Project Wizard, when the class library *Vfxobjbase.vcx* exists, the class library *Vfxobj.vcx* is not updated anymore.

### VFX Form Classes

The design of all VFX forms is now renewed so that DE in forms is no more used. This means that any further changes concerning these forms will be made mainly in *Vfxform* class library and will not require forms to be updated with new VFX versions. This gives the VFX developers a chance to make their own changes in provided VFX pre-developed forms without a need to rewrite all these changes later with new VFX version coming.

## *Data Handling*

VFX applications, as well as all developer tools use SQL Server 2000 and MSDE, as well as SQL Server 2005 and SQL Server 2005 Express Edition.

### Dataenvironment settings

There are several SET operations, which need to be made yet before opening cursors in DataEnvironment. Additionally to *SetDataEnvironment()* method of *goEnvironment* object, it is now possible to make environment settings for Forms and CursorAdapter objects.

The new method *OnSetEnv* in cBaseDataAccess CursorAdapter class, checks if parent form has method *OnSetEnv* and if method has not been called yet by other CursorAdapter object, current object calls this method. If form does not have *OnSetEnv* method cursor adapter checks for object goEnvironment (from class *cEnvironment* stored in *vfxmain.prg*) and if that object exists, its method *SetDataEnvironment()* is called. In all other case hard-coded set of commands is executed for setting the environment.

Developers can add their own code to make environment settings in *cAppDataAccess* class in *appl.vcx* class library.

## The new object goPath

A new object goPath is created at run time. This object has properties containing values of the currently selected path:

*CDataDir* – Path to currently used database.

*ClientName* – Name of current client database. This is the name that end-users see in client selection dialog. This is not the physical name of the database.

*VfxPath* –Path to VFX system tables.

*ReportPath* – Path to report files.

*UpdatePath* – Path to the folder, containing new database structure, used for actualization.

*ImportPath* – This property is not used by VFX and is available for free usage.

*ExportPath* – This property is not used by VFX and is available for free usage.

When *Vfxpath.dbf* table is used for client selection, every of its fields will become a property of the object *goPath*

When *Config.vfx* is used for client selection, every field, defined in it, will become a property of the object *goPath*

In such way developers have information at run-time about currently used path settings.

## Enlargements in cBaseDataAccess class

### New properties

*cFieldsToWriteNULLWhenEmpty* – This property contains a list of fields, which values should be set to *NULL*, before saving data into database. This property is used always when *DataSourceType* for *CursorAdapter* object is different than *NATIVE*.

*cForeignKeyName* – Used for child-alias cursor adapter objects. Contains the name of field which is foreign key to the parent alias. This information is used when saving a new parent record with auto-increment primary key field. The value of this primary key field is obtained after the record is saved and is updated for all records in all child aliases.

*cForeignKeyValue* – Used for child-alias cursor adapter objects. Contains the name of field in parent alias or an expression, which is used to obtain key values for newly saved parent record. The value of this expression is evaluated and saved in the field, specified in *cForeignKeyName* property of the object.

*cWhereClause* – Here developers can write a form-specific where clause. Its value is concatenated at run time with the value of SelectCmd property of the CA object before executing cursor fill. This ensures that value of SelectCmd property does not need to be changed in class instance and any future change in the class will be automatically inherited in the instance in form's DE.

*lWriteNULLWhenEmptyForDBC* – When the value of this property is .T., the empty date and datetime data fields, listed in *cFieldsToWriteNULLWhenEmpty* property, will be replaced with NULL even when working with local DBC. The purpose of this is to ensure same application behavior regardless of database backend. The default value for this property is .F. This property is only considered when DataSourceType for CursorAdapter is *NATIVE*.

### New Method

*OnSetEnv*- This method is called during instantiation.

The new method *OnSetEnv* in cBaseDataAccess CursorAdapter class, checks if parent form has method *OnSetEnv* and if method has not been called yet by other CursorAdapter object, current object calls this method. If form does not have *OnSetEnv* method cursor adapter checks for object goEnvironment (from class *cEnvironment* stored in vfxmain.prg) and if that object exists, its method *SetDataEnvironment()* is called. In all other case hard-coded set of commands is executed for setting the environment.

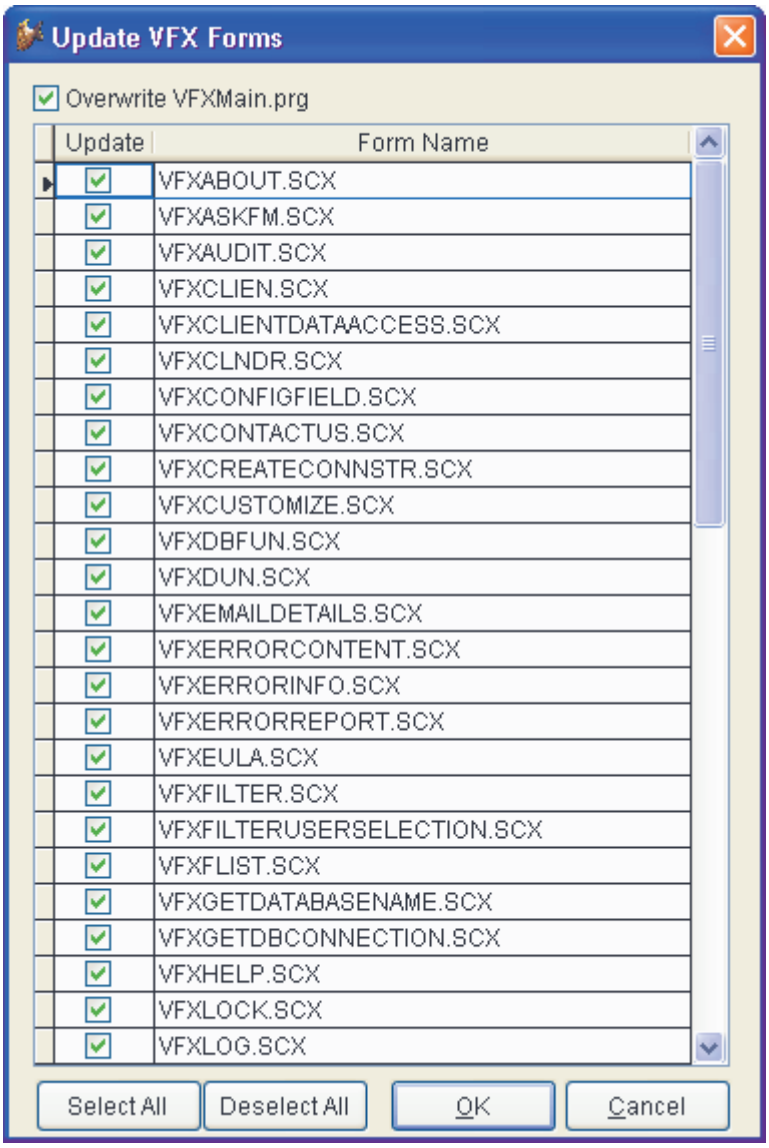
## *Builder and Wizards*

### VFX – Task Pane

There is a new button in toolbar of VFX – Task Pane. It starts VFX – Update Project Wizard for the selected project.

## VFX – Update Project Wizard

Before the updating, will be created archive of the project including all subfolders. The name of archive file consists of project name, current datetime in ANSI format and VFX version of the project before updating.
A new dialog with a selection list containing all VFX forms is added to the builder.




In this dialog, developer selects which VFX predefined forms should be replaced with new versions, coming with VFX. Developers, who had made changes in VFX predefined forms, can select these forms not to be overwritten. Selected forms list is saved per project and used in future version updates.

## VFX – Application Builder

All new properties of *cFoxAppl* and *cAppUpdateEngine* classes can be edited very easy in VFX – Application Builder.

Tooltips are added for all edit controls in the VFX - Application Builder, displaying the class and property names which are keyed up with every control. Tooltip is shown in format *ClassName.PropertyName*.

Additionally developers have search functionality based on label captions in the builder.
Of course all goProgram new properties as well as properties of cAppUpdateEngine class are available for setting here.

## VFX – Upsizing Wizard

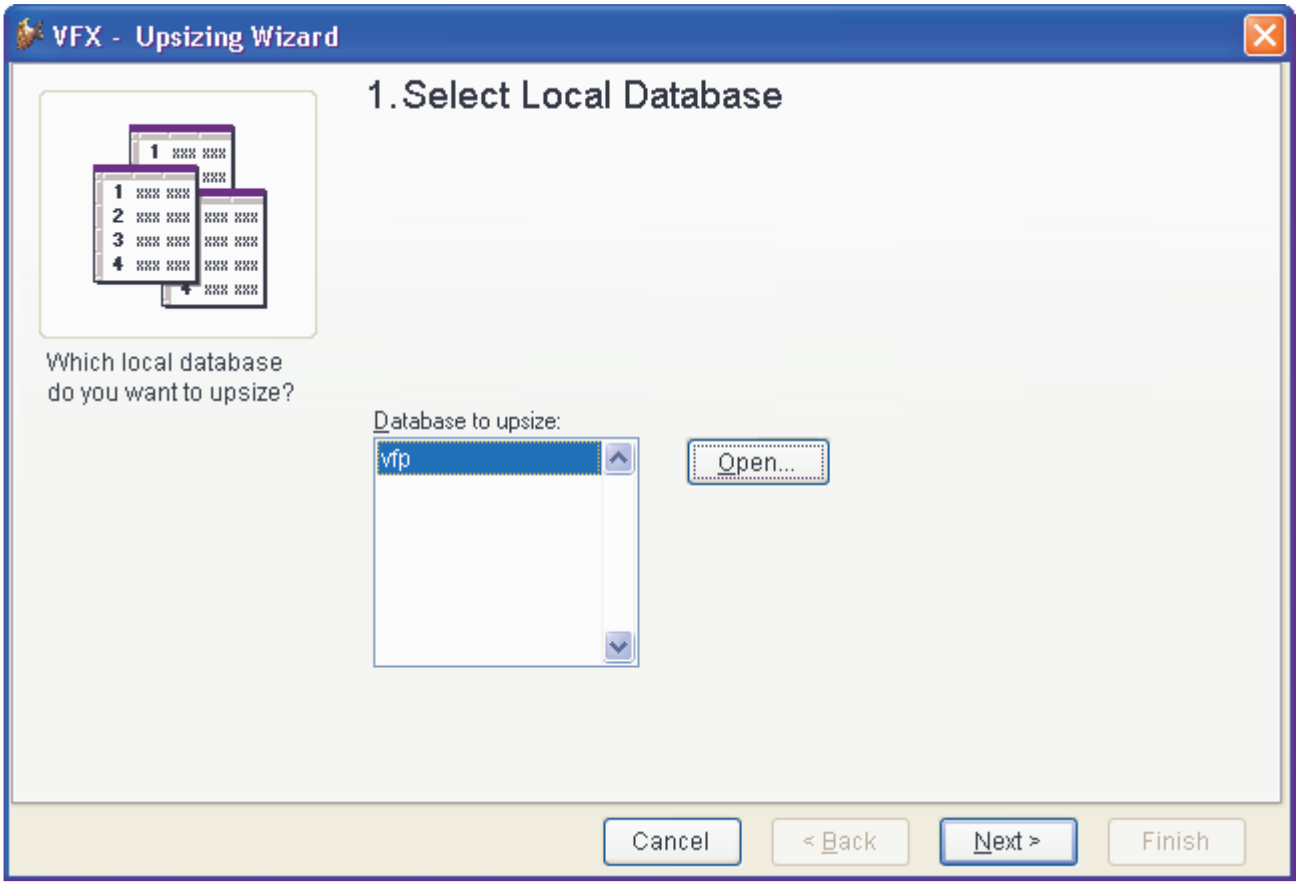
The VFX – Upsizing Wizard is a tool to export an existing VFP database to SQL Server or Oracle. The structure of new created SQL database is as similar as possible to the original VFP database. The data is uploaded to the newly created database. While upsizing, views that have SQL Server compatible syntax are also created.

The VFX – Upsizing Wizard has the full functionality of VFP – Upsizing Wizard and additional enhancements giving the developers chance to control more detailed the upsizing process. With the VFX – Upsizing Wizard ID values for autoincrement fields are upsized correctly. Data upsizing is made without codepage conflicts and all fields' names which are reserved SQL words can be upsized. The wizard takes case about empty date or datetime values and uploads these values as *NULL*. And finally, VFX – Upsizing Wizard ID does not require DSN to be used and establishes its own connection to the remote database.

The VFX – Upsizing Wizard leads the developer through these steps:

### 1. Select Local Database

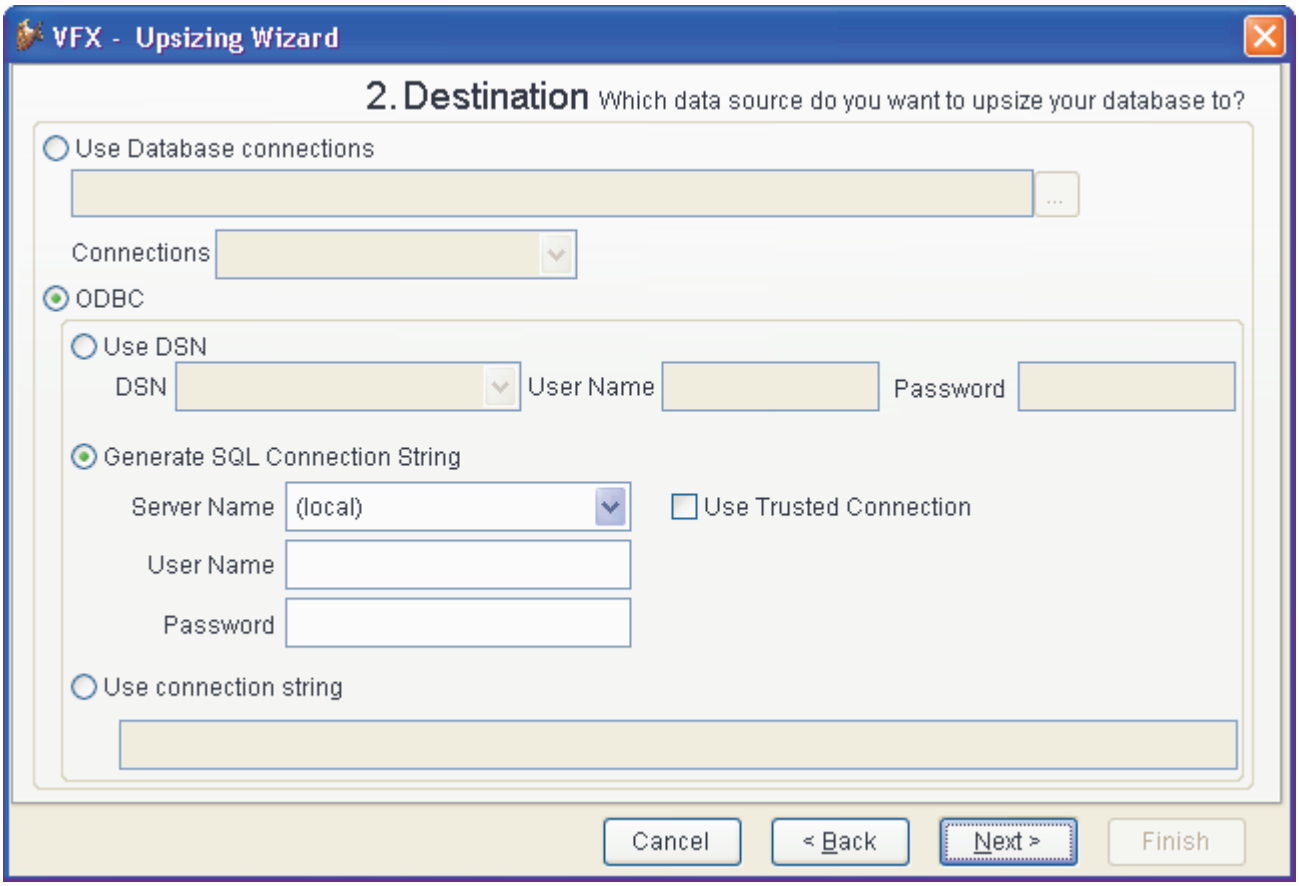
On first step is selected the local database that will be upsized to SQL Server.




A list of the currently open databases is displayed for selection. It is also possible to open a database that is not currently open using the *Open* button.

## 2. Destination

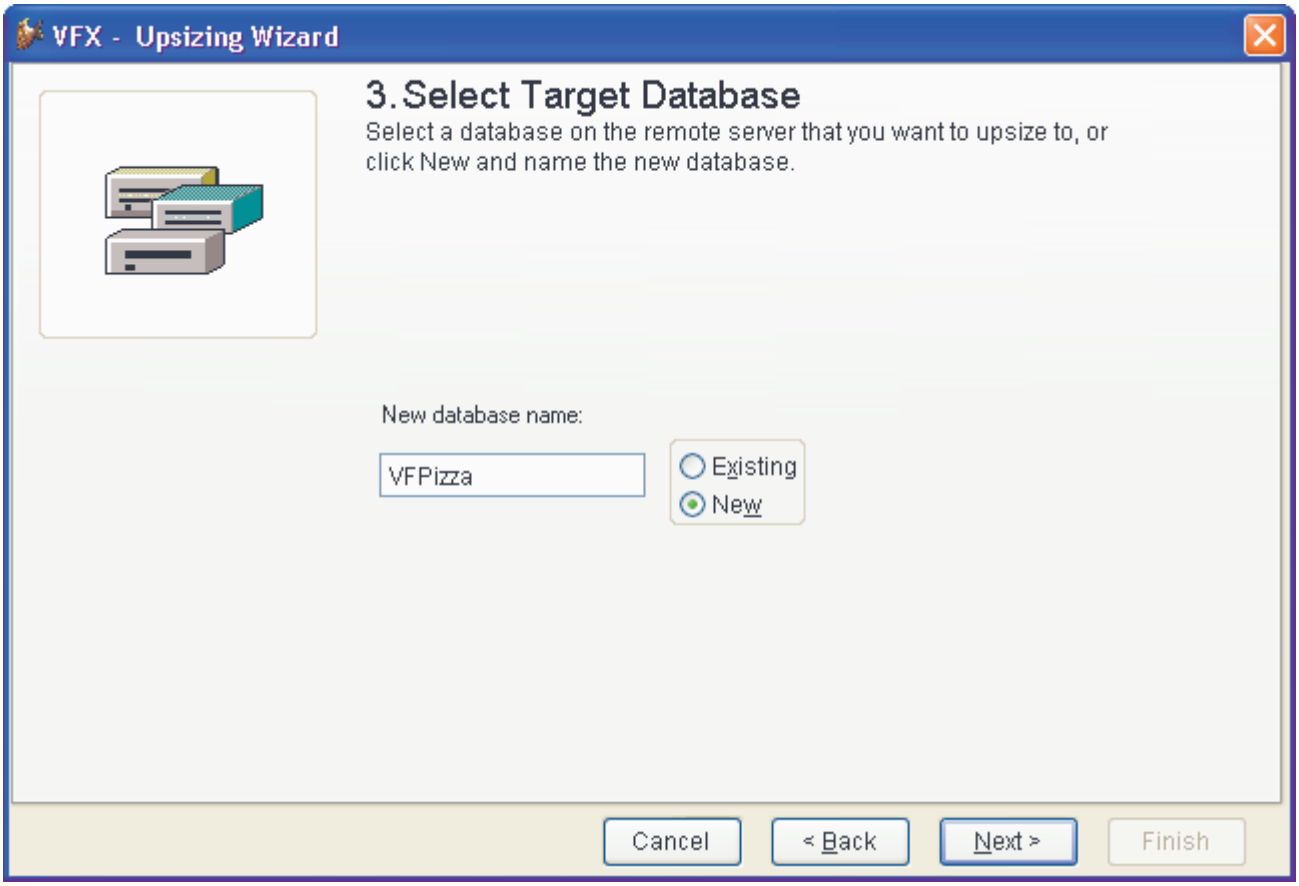
On this step, developer has to specify a way to connect to the SQL Server. It is possible to use an existing connection from the database or to use an existing DNS.




It is also possible to define connection parameters and to connect to remote SQL Server database without having predefined connection. By default can be used locally installed SQL Server. In case when the entered username and password, do not allow establishing a connection to the server, the wizard automatically tries to establish trusted connection using windows login data.

## 3. Select Target Database

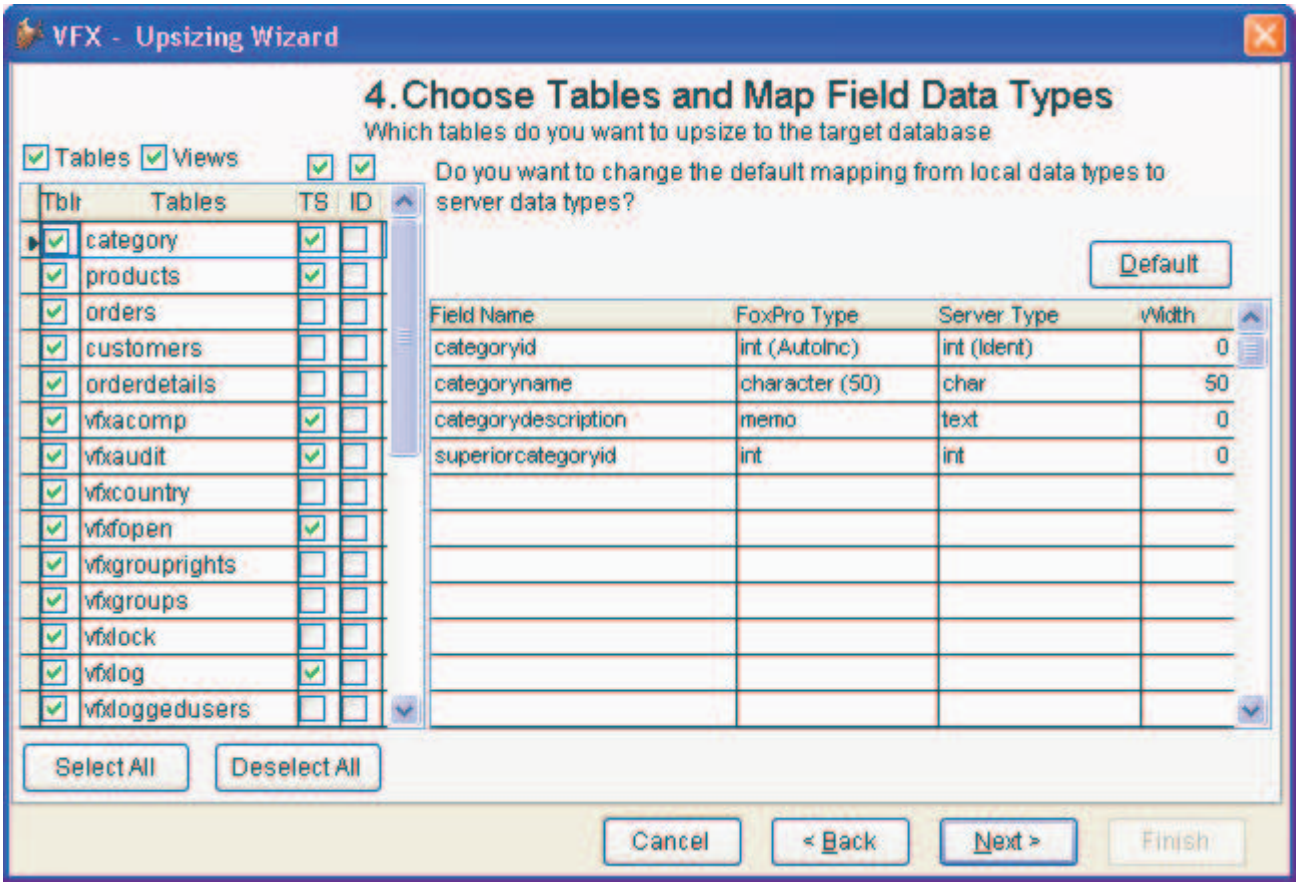
Here can be entered name of the database that should be created. It is also possible to select an existing database where data to be uploaded. When the option *New* is selected, the name of the database should be entered. The name must conform to naming rules of the database engine you are upsizing too.




A list of databases, available on selected server, is displayed, when the option *Existing* is selected. Select the database into which you want VFX – Upsizing Wizard to copy Visual FoxPro tables when you upsize.

## 4. Choose Tables and Map Field Data Types

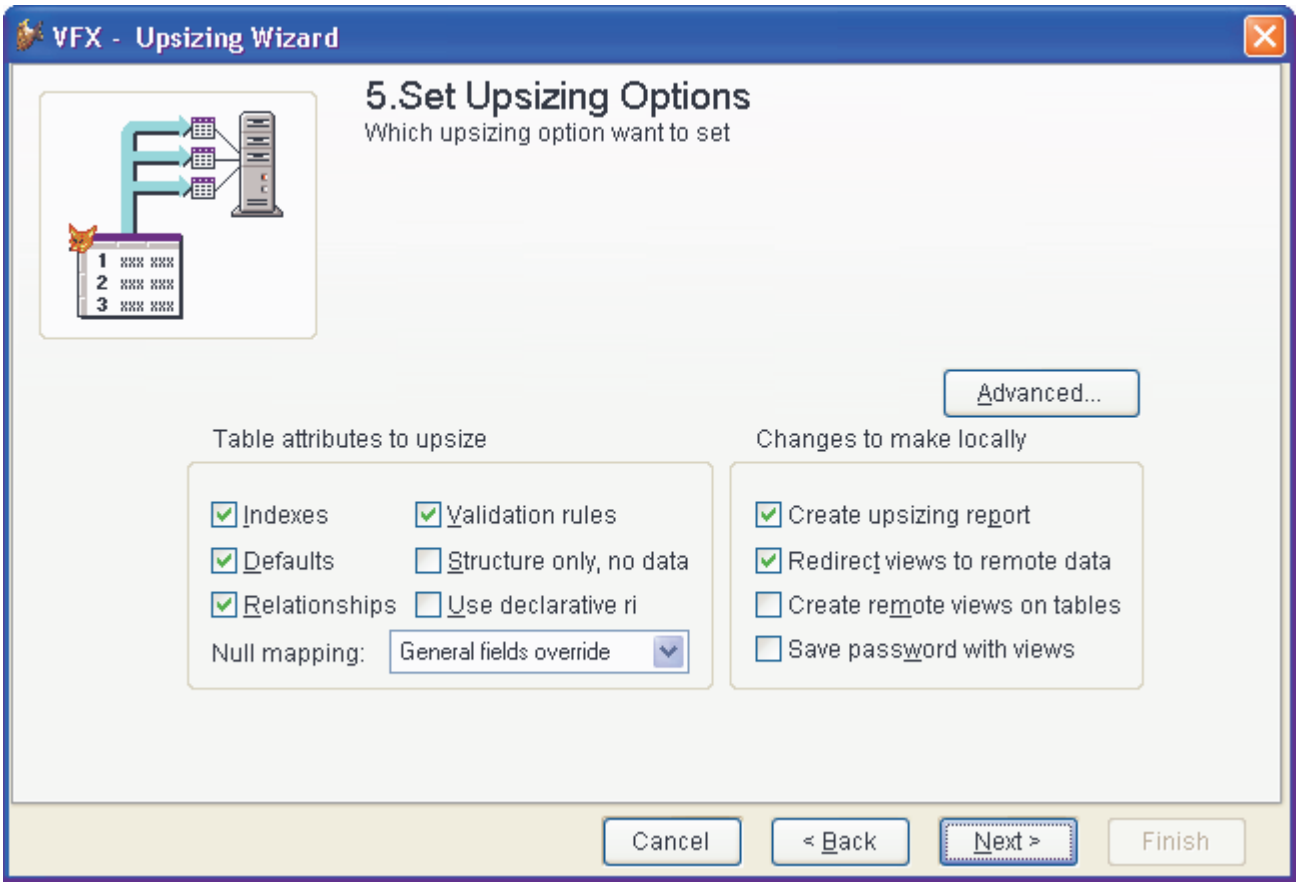
In this step, you can select the Visual FoxPro tables and/or Views you want to export to SQL Server. By default all tables are selected o be upsized. For every table can be added timestamp (TS column in tables grid) or Identity field (ID column in tables grid). VFX – Upsizing Wizard automatically suggests timestamp to be added for tables, which contain memo fields. For tables which do not have primary key is suggested to be added Identity field. If necessary, these settings can be changed for every table.




For every selected table, structure of VFP table as well as field data types that will be created in SQL Server database is displayed. For every field from the table is suggested corresponding server type. Server data types for fields can be changes if necessary.

## 5. Set Upsizing Options

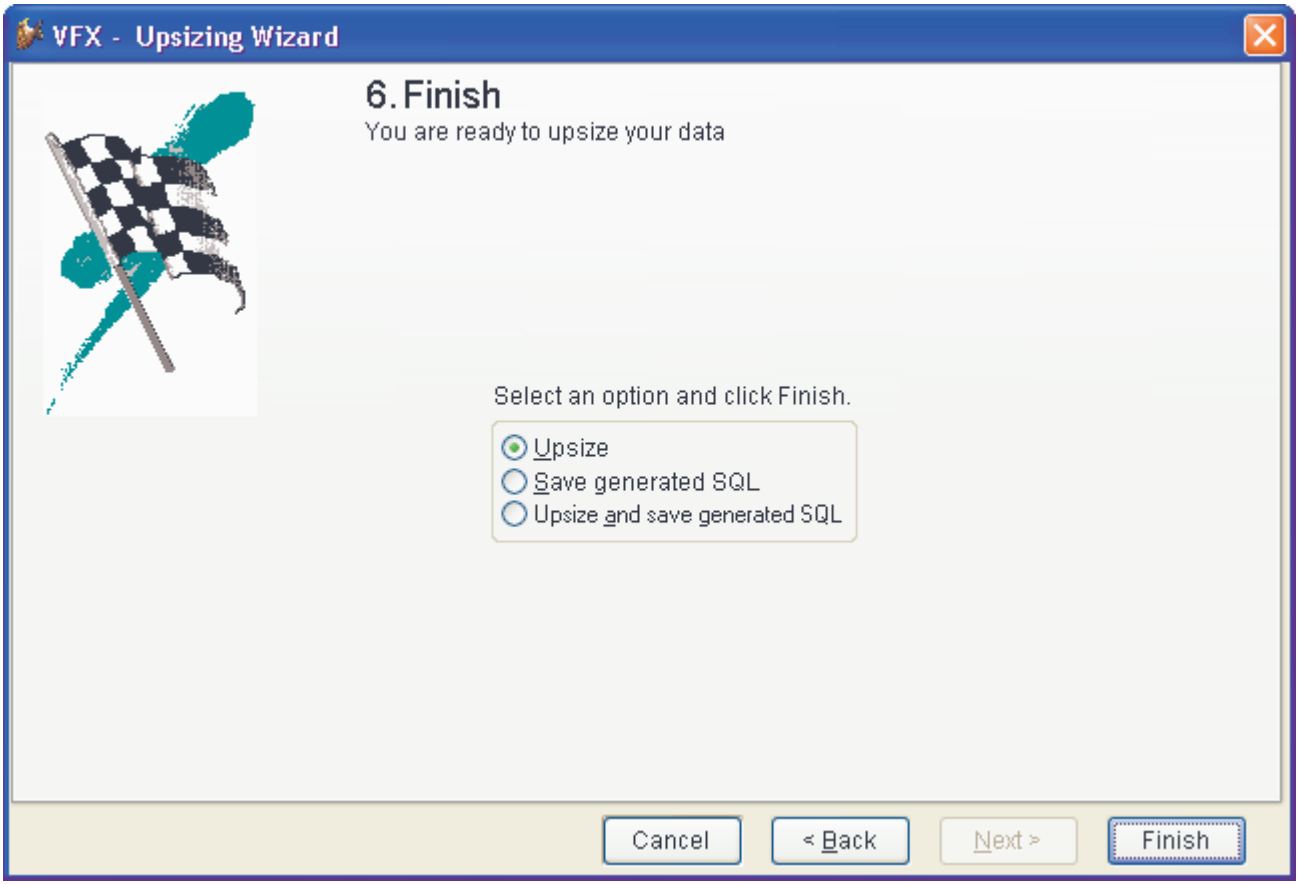
By default, the VFX – Upsizing wizard exports a table's structure and its data. Along with field names and data types, you can also export indexes, defaults, relationships (referential integrity constraints) and validation rules. In the combobox *Null mapping* can be selected the fields that can accept .NULL. This option helps to ensure that insert and update commands against remote data succeed.




In this step can also be selected if an upsizing report should be created. The upsizing report is a new VFP project which includes tables, holding information how the process went. In reports can also be seen problems, raised during upsizing.

## 6. Finish

On this step you can select how upsizing process will work.




You can select one of these options:

**Upsize** – Executes upsizing according selected options.

**Save generated SQL**– Generates SQL Script necessary to upsize your database and then stop without beginning to create databases and tables on your remote server.

**Upsize and save generated SQL**– Executes upsizing according selected options and additionally generates SQL Script that can be used to upsize your database later.

It's a good idea to create a backup copy of your database before upsizing. During upsizing the tables and local views in the VFP database are renamed, in order to create remote views to the SQL database with same names

VFX – Upsizing Wizard allows uploading to SQL Server database tables which contain empty date or datetime fields. So far the value of 1.1.1900 has been saved in those fields. From now on, if fields allow NULL values, the value send in the SQL Server database will be NULL, in case of empty date or datetime.

## VFX – CursorAdapter Wizard

CursorAdapter classes, generated by VFX – CursorAdapter Wizard can now be created on views, saved in a DBC or in a SQL Server database.

The wizard can be run for separate tables or views.

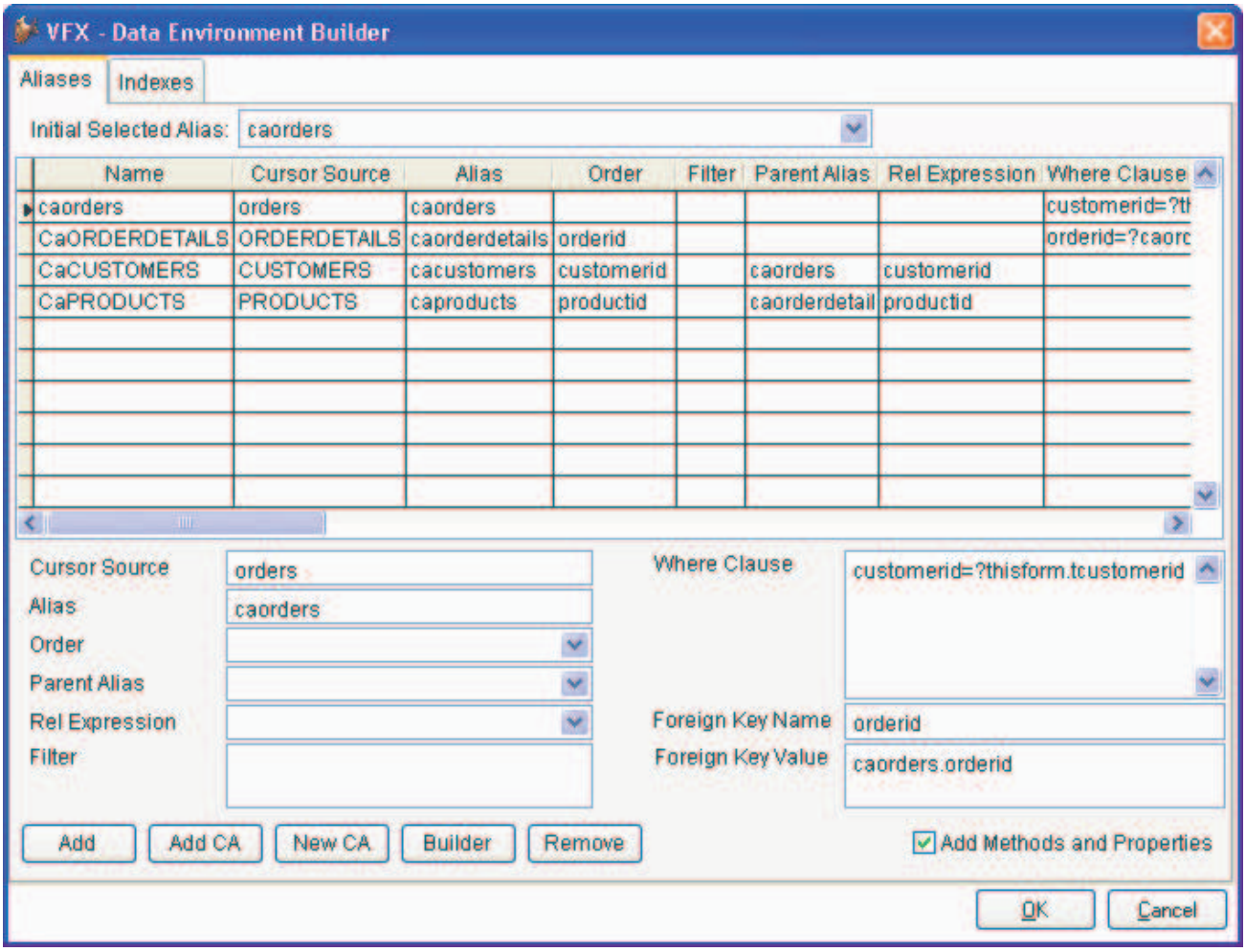
For every field from every table can be specified type conversion. All appropriate VFP data types can be selected from a Combobox.

It is now possible from VFX – CursorAdapter Wizard to call VFX – ConnectionString Wizard for generating connection string.

VFX – CursorAdapter Wizard fills automatically a row in *Config.vfx* containing connection parameters used for generation CursorAdapter classes.

## VFX – Data Environment Builder

The VFX - Data Environment Builder can now be used as a standalone builder on *Dataenvironment* classes. Yet in VFX 9.0 *Dataenvironment* classes were supported for forms. Now the VFX - Data Environment Builder can be used in VFX 9.5 for editing of these standalone classes.




In lower part of DE Builder window you can see and edit all data for currently selected alias in grid. This gives developers the chance to see better content of properties, especially those which are long strings like Filter, WhereClause, ForeignKeyValue etc.

There are also several new columns in VFX - Dataenvironment Builder:

*Where Clause* – Content of this column is stored into *cWhereClause* property of cursor adapter object. Its value will be used when working with CursorAdapter objects. In this column developers can write a form-specific where clause. Its value is concatenated at run time with the value of SelectCmd property of the CA object before executing cursor fill. This ensures that value of SelectCmd property does not need to be changed in class instance and any future change in the class will be automatically inherited in the instance in form's DE. This has the advantage that by any structural changes, CursorAdapter classes can be recreated, and the where clause will not be lost by inheriting new SelectCmd value.

In VFP databases for primary key can be used field of data type *Integer (AutoInc)* and *Integer Identity* data type can be used in SQL Server databases for primary key fields.

When working with CursorAdapter classes, and primary key is generated by the database, in a 1:n scenario the generated primary key for the parent record must be saved as a foreign key in the child records. Additionally, it is necessary that first the parent record is saved and after that to read the primary key value from the database. VFX automatically saves the foreign key value in all records of the child alias. For this purpose, it is necessary in CursorAdapter object for the child alias to have information which is the foreign key. Yet in VFX 9.0, in cBaseDataAccess CursorAdapter class, there were properties *cForeignKeyName* and *cForeignKeyValue*. In VFX 9.5 the values of these properties are automatically determined and suggested and can be edited in the VFX - Data Environment Builder.

*Foreign Key Name* –Used for child alias CursorAdapter. Here should be written the name of field which is foreign key to the parent alias. This information is used when saving a new parent record with auto-increment primary key field. The value of this primary key field is obtained after the parent record is saved and is updated for all records in child alias. Content of this column is stored into *cForeignKeyName* property of cursor adapter object.

*Foreign Key Value* – Write here the name of field in parent alias (including alias name) or an expression, used to obtain key values for newly saved parent record. The value of this expression is evaluated and saved in the field, specified in *Foreign Key Name* field for current alias. Content of this column is stored into *cForeignKeyValue* property of cursor adapter object.

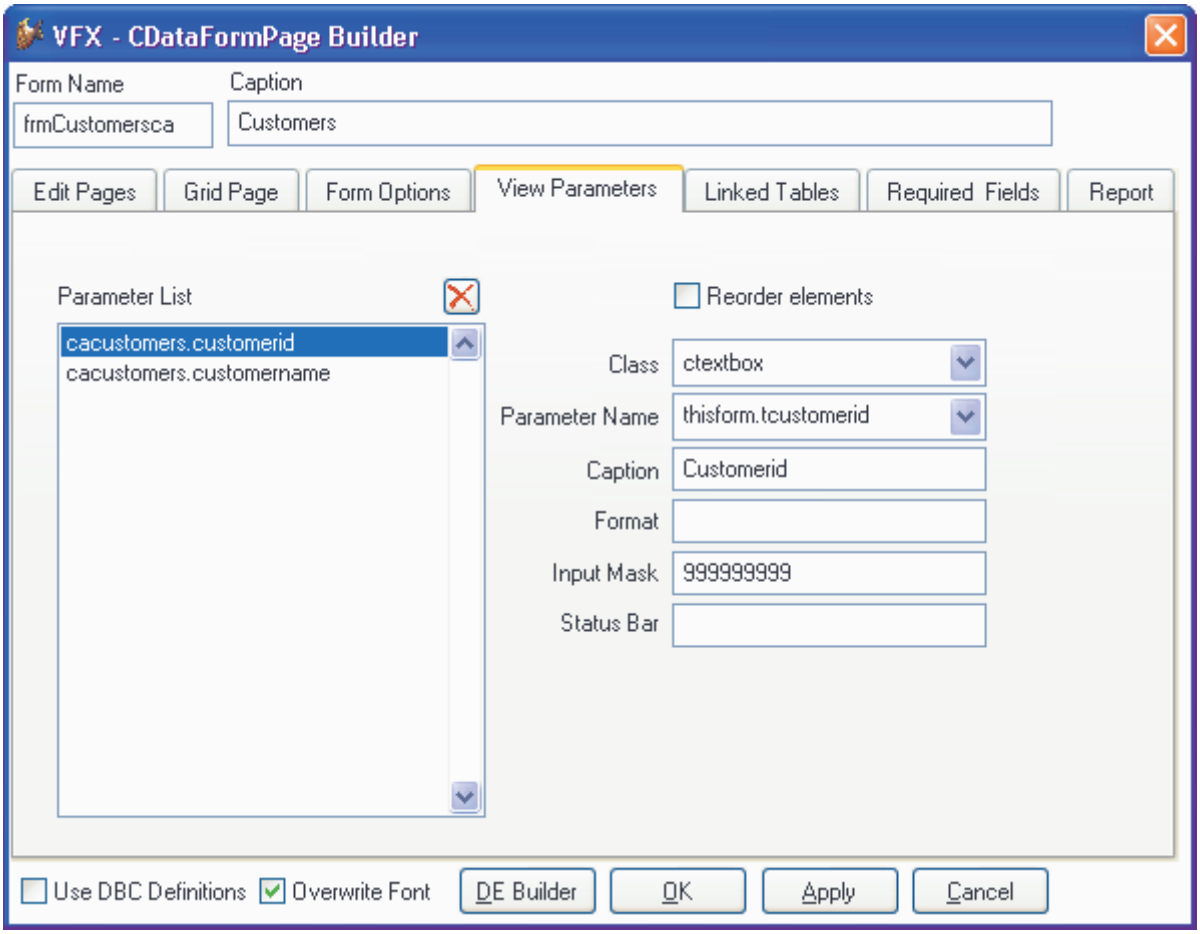
If you use your own *CursorAdapter* classes that are not based on *cAppDataAccess* class, but you want to be able to use new Dataenvironment Builder features, you can simply mark *Add Methods and Properties* checkbox and properties to keep defined values for *Where Clause, Foreign Key Name* and *Foreign Key Value* will be created if they do not exist.

## VFX – Form Builder

When the Builder in started from the VFP - form designer, the VFX - Form Builder starts directly. The VFX - Data Environment Builder can be open by clicking on the button *DE Builder* or by running the builder directly from the DataEnvironment of the VFP - form designer.

Now, in VFX 9.5 form builders, on *View Parameters* page, for every parameter developer can select object class to be used.

Developers can also select among parameters, used in *cWhereClause* property of CursorAdapter objects available in DataEnvironment. They are available in a drop-down combobox, along with parameter names that has already been defined.

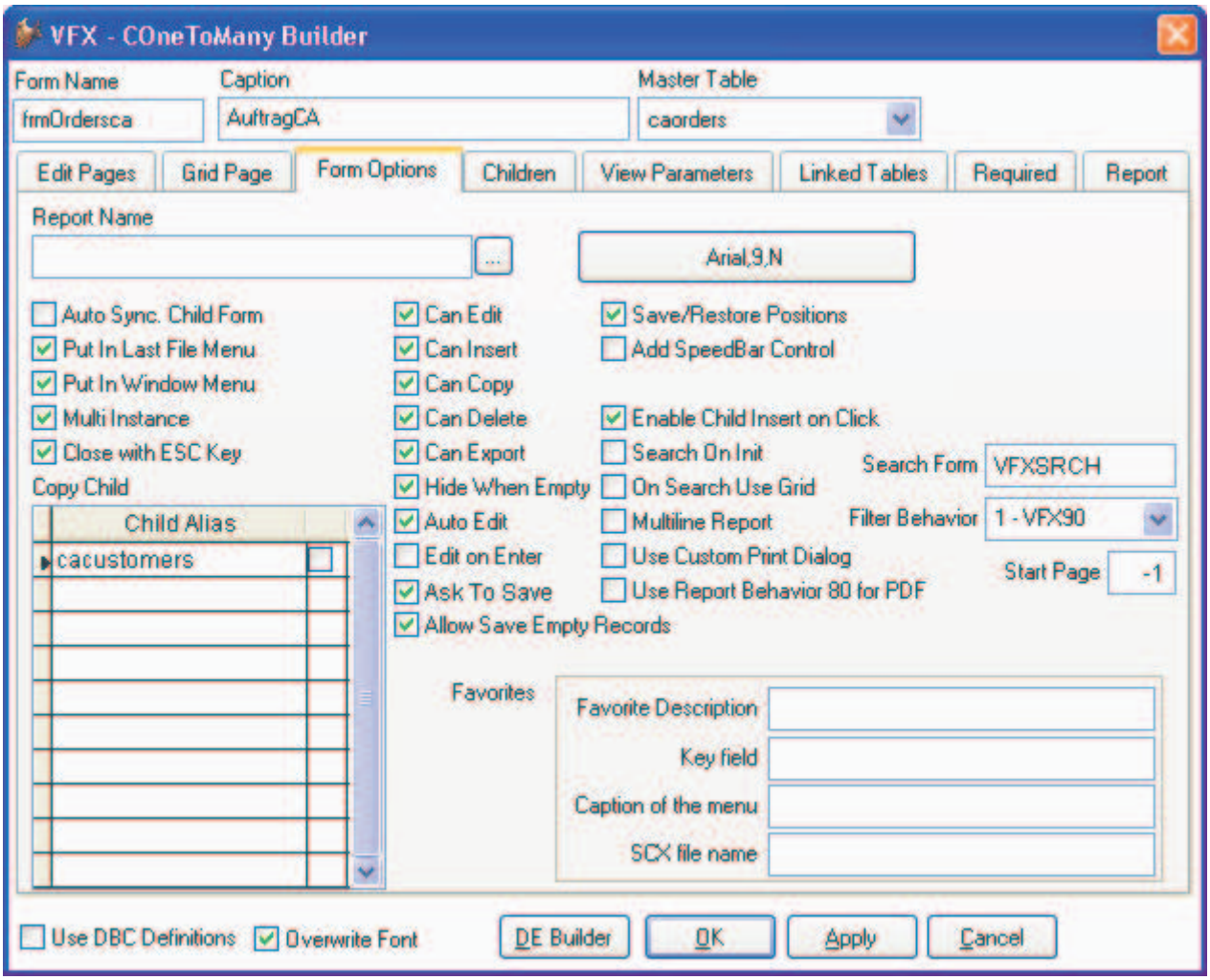



The settings in the VFX - Form Builders *Is Child form* and *Has linked child form* are no more required and are redundant. In VFX 9.5 every form can work without special setting as Parent- and as a Child form.

For forms based on CursorAdapter classes as data source, the VFX - Form Builder reads fields' properties from the underlying VFP tables, if database is available.

The VFX – Form Builder now generated localized captions for pages, added to the pageframe.

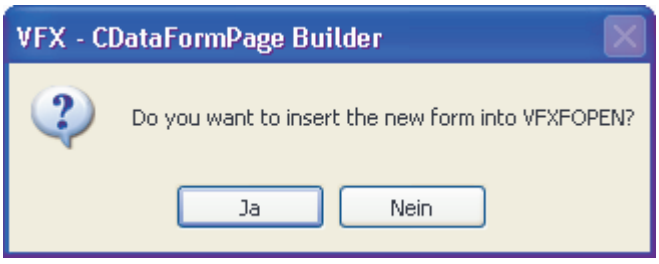
A numerous widely used and a bunch of new properties are available in form builders on Form Options page.




The checkbox *Allow Save Empty Records* controls if when saving a newly added record, data will be automatically reverted in cases when end-user did not fill any field. Selected value is saved in the property *lAllowSaveEmptyRecords*. Its default value is .T. – empty records can be saved. This ensures backward compatibility application behavior.

Here, on *Form Options* page, can also be made all settings necessary for adding a data record to Favorites menu.

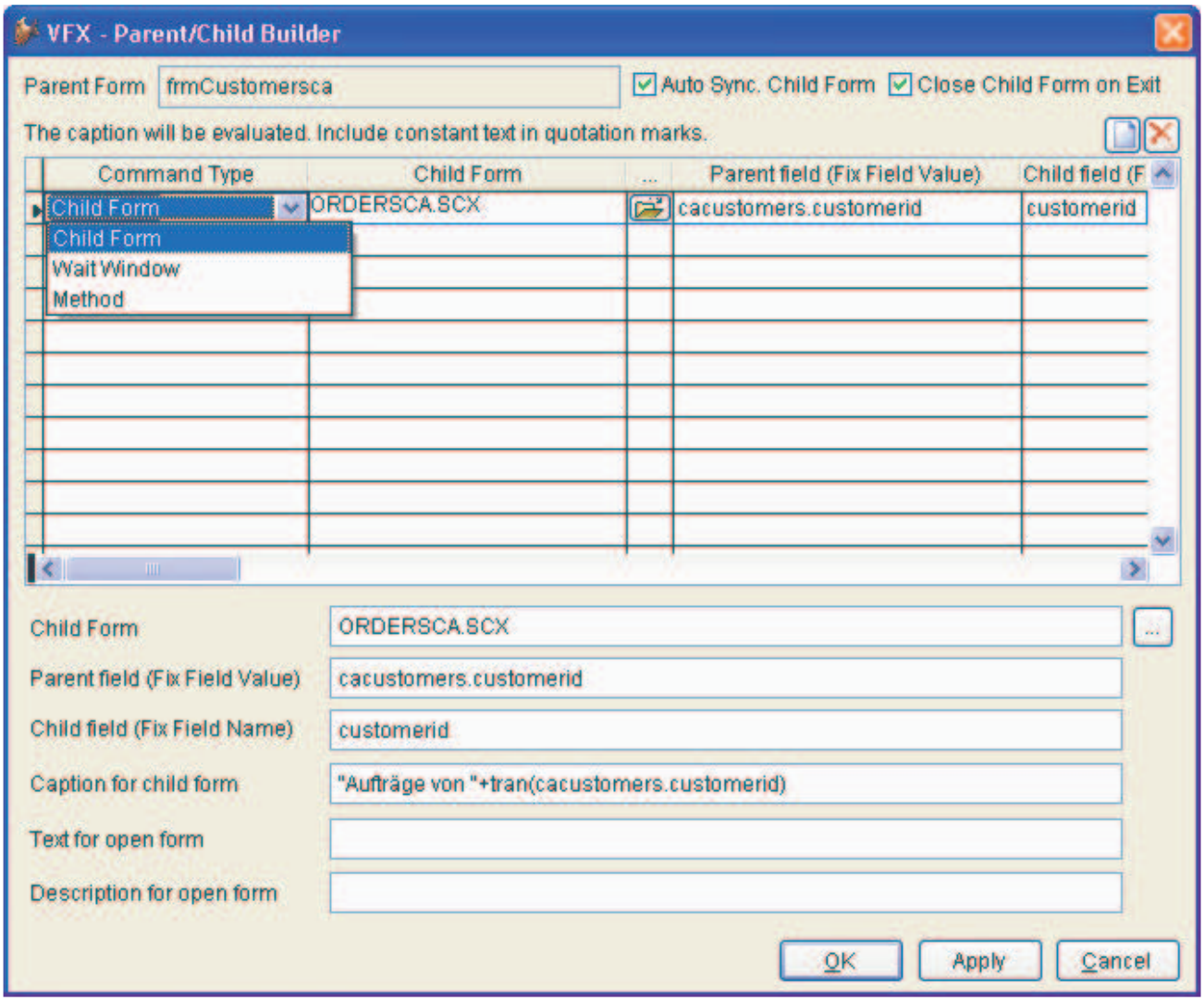
Instead of inserting every newly created form into *Vfxfopen* table, developer is asked if he wants that with a messagebox "Do you want to insert the new form into VFXFOPEN?" (Yes/No) . If the answer is No, the form is not added into *Vfxfopen.dbf* and will be available in the Open dialog.



## VFX – Parent/Child Builder

In earlier VFX versions it was possible in the *OnMore* method to call Child forms or methods of the Parent form or to execute also a *Wait Window* command. Now in VFX 9.5, all these three possibilities can be selected in the VFX - Parent/Child Builder. Developer can choose one of them in the Grid in *Command Type* column.

Additionally to the possibility to edit data in the Grid, all values can be entered alternatively also in textboxes below the Grid.



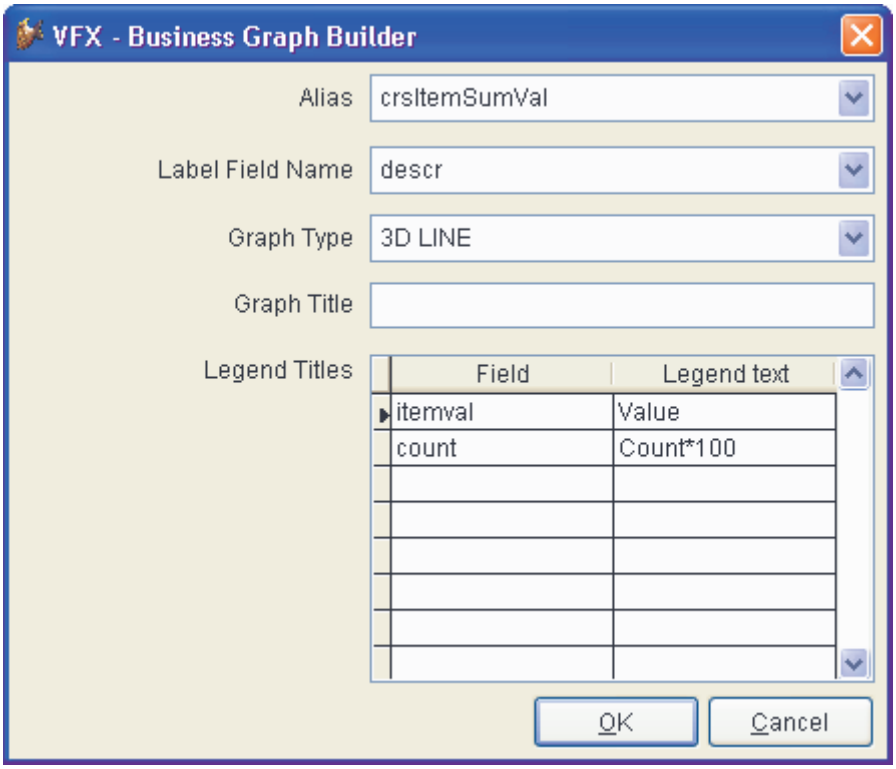

If the checkbox *Auto Sync. Child Form* is marked, all Child forms are automatically synchronized while moving the record pointer in the Parent form.

With the checkbox *Close Child form on Exit* can be set whether with the parent form closing, automatically also all open Child forms should be closed.

With child form selection, VFX – Parent/Child Builder tries to retrieve information from Parent and the Child forms and to fill the following Grid columns properly. In former VFX versions, when using cursor adapter objects for data access, it was not possible for the builder to recognize relation fields between parent and child form.
In VFX 9.5, the builder uses also the primary key field of work alias while suggesting a field to maintain relation between the two forms.

## VFX – Business Graph Builder

The VFX – Business Graph Builder helps the VFX developers easy and fast to set properties of *cBusinessGraph* object just in few minutes.




From *Alias* combobox you can select the alias which contains data series for graph drawing. The combobox lists all aliases in dataenvironment of the form. The value, selected in combobox will be stored in *cAliasName* property.

The combobox *Label Filed Name* lists all fields of the selected alias. Here you can specify the field which will be used to set data labels in the graph. The value of combobox will be stored in *cLabelField* property.

The type of the graph is selected in combobox *nGraphType*. It lists all allowed types of graph: 3D BAR, 2D BAR, 3D LINE, 2D LINE, 3D AREA, 2D AREA, 3D STEP, 2D STEP, 3D COMBINATION, 2D PIE, 2D XY. The selected value of combobox will be written in *nGraphType* property.

In the textbox *Graph Title* should be written a title for the graph. The value of textbox will be stored in *cGraphTitle* property.

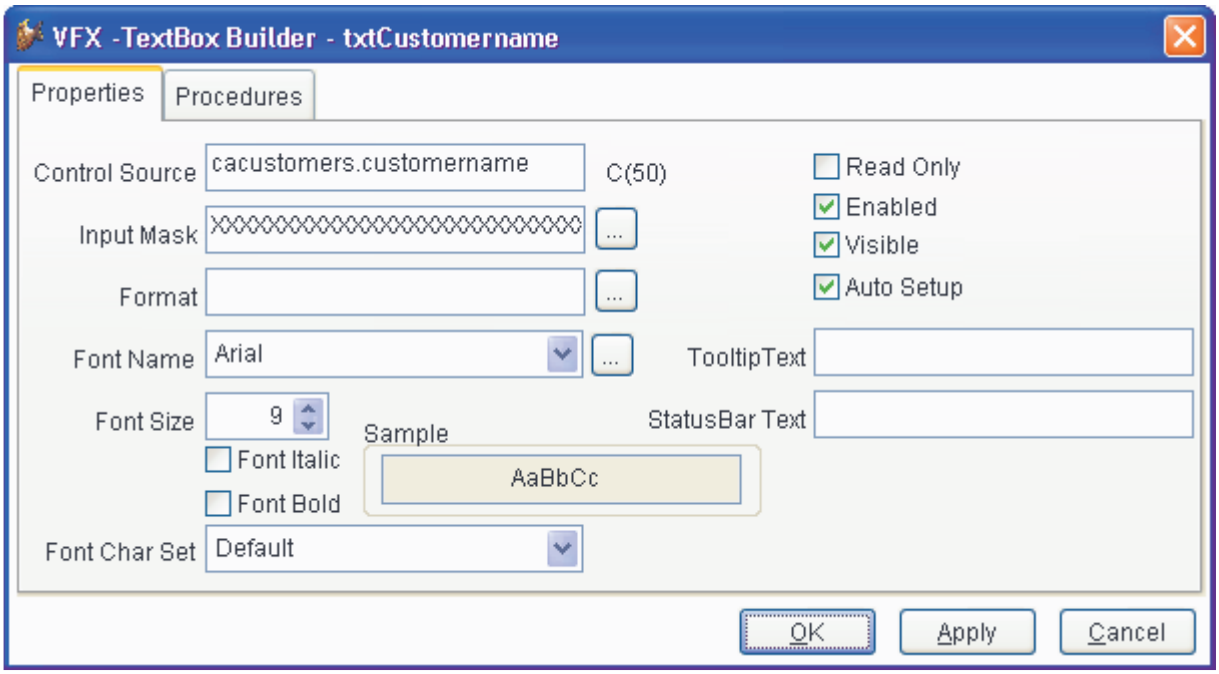
In the grid *Legend Titles* are seen all the fields from the selected alias, except the field used to set data labels in the graph. All these fields will be used in graph generation and shown in the business graph. For each field you need to provide a legend text. These texts are used to be created a comma-separated list, containing series titles. The list is saved in *cLegendTitles* property of *cBusinessGraph* class.

## VFX – TextBox Builder

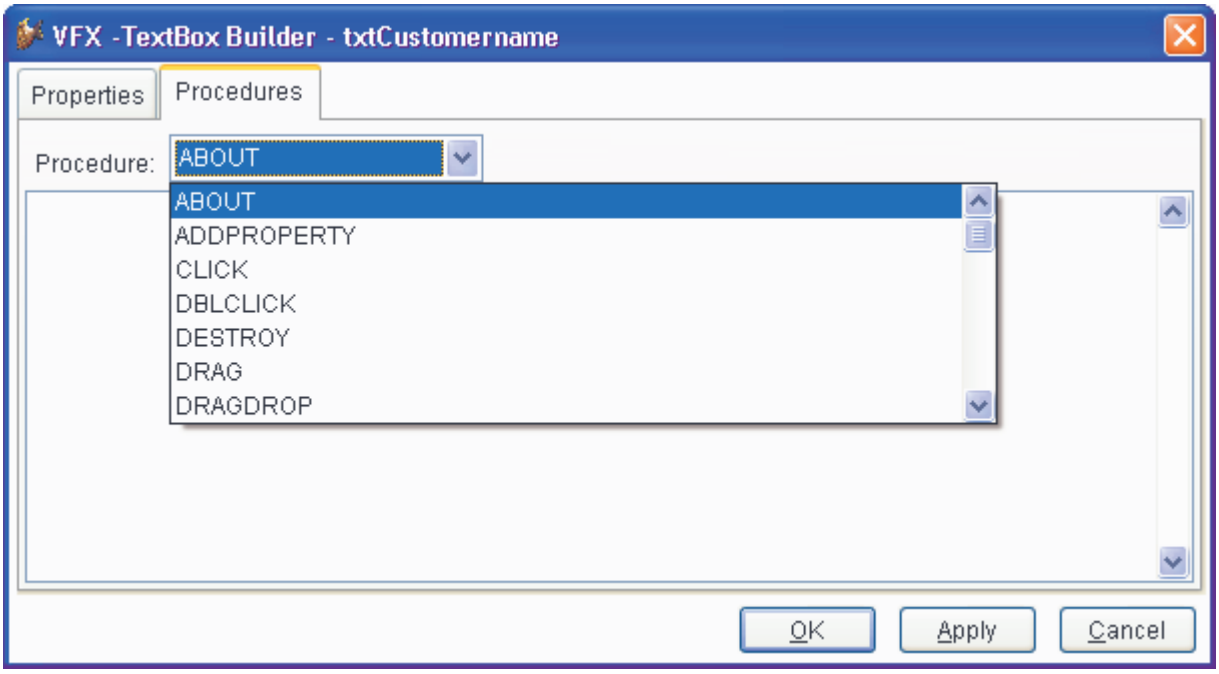
In the new VFX – TextBox Builder you can setup most used properties of all controls, inheriting the base VFP class *Textbox* and you can also edit code in all methods and events of the control.

To call the VFX – TextBox Builder, select *VFX Power Builders* bar from VFX 9.5 menu while the TextBox based control is selected in VFP form designer or right click on the control and select *Builder* from the context menu.

On page *Properties* you are able to make settings for properties and to have a look how the textbox will look like in *Sample* area.




Page *Procedures* gives you the chance to edit code of all the control's methods and events. Select the desired method or event from the Combobox and edit its code in the edit area below.

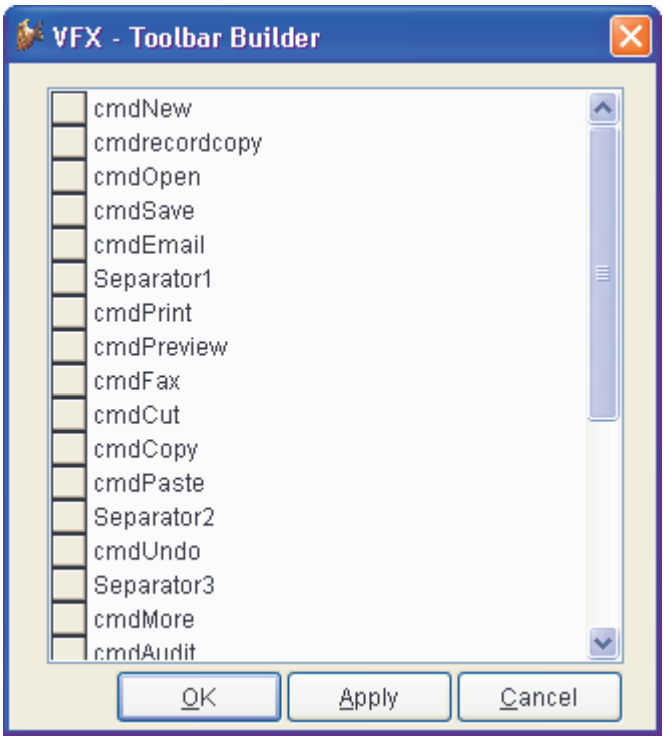


## VFX – Toolbar Builder

Another new builder in VFX is the builder for toolbars.

VFX – Toolbar Builder makes it easy to rearrange buttons and separators in your toolbars, just by moving then down and up in the builder form. The builder is used to manage objects sequence in toolbar class.

To call the VFX – Toolbar Builder, open a toolbar, right click and select Builder from the context menu.




All controls from the toolbar are listed in the same order as in the toolbar. To reorder the controls move them with the help of the mover bars. Then click *Apply* or *OK* button to view the new order in a toolbar.

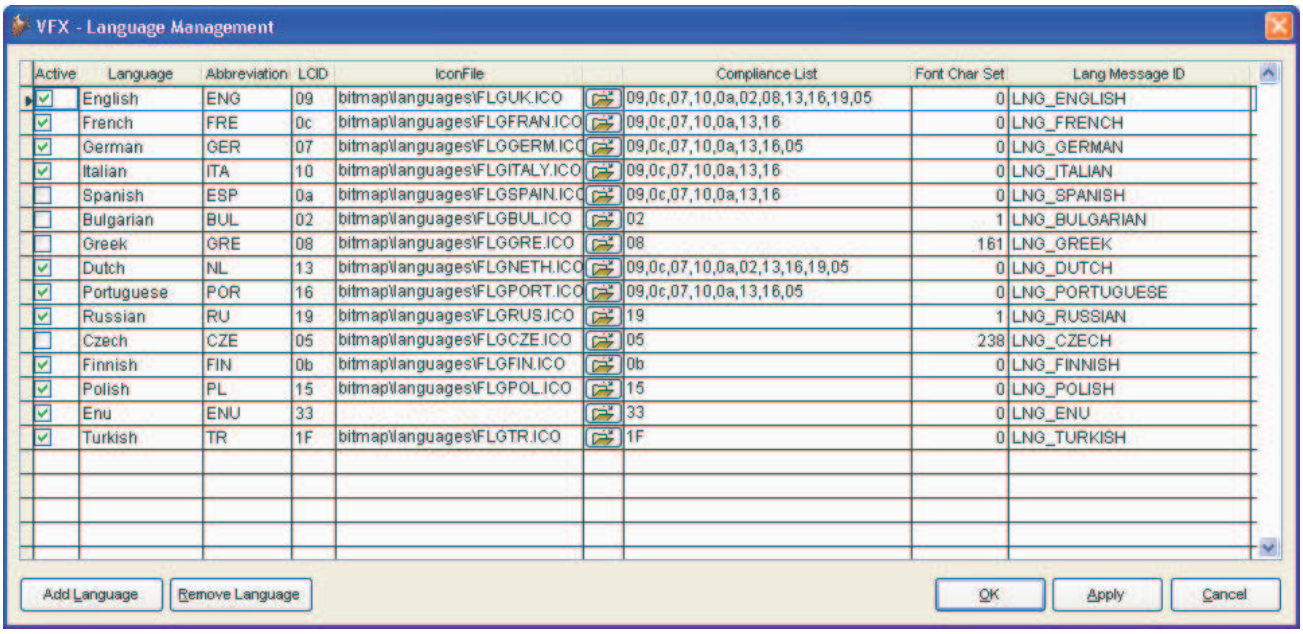
## *Localization*

According to the selected application language, date and time settings are now also localized when runtime localization is used. To enable this feature in VFX – Application Builder, you must set *Date format* to *VFX LOCALIZED*. This value is stored in cDateFormat property of application object and can be edited manually, if needed.

## VFX - Language Management Builder

This builder is used to manage the languages, used by the application when runtime localization is used. Data about used languages is saved in *VFXLanguage.dbf*.

When a new application is generated, *VFXLanguage.dbf* contains records for all predefined VFX languages which are distributed with the template application. Developers can add new language records, and can also delete added records. VFX predefined languages cannot be deleted. They can only be set to Inactive. When a predefined language is set to Inactive, at run time it will not be visible in a language combobox in login form and on main toolbar.




In column *Language* is entered a short description, which is shown in the language combobox in login form and on main toolbar.

In column *Abbreviation* is entered the abbreviation used for getting messages and captions from *vfxMsg.dbf* in selected language. It corresponds to the name of column in *vfxMsg.dbf*.

In column *LCID* is entered the locale identifier for the language. This is Windows defined value and represents regional settings.

In column *IconFile* is entered the name and relative path of the icon file used to display language flag. The icon is shown in a language combobox in login form and on main toolbar.

In column *Compliance List* is entered a comma separated list of locale identifiers of languages compliant with the current language. This list contains values, corresponding to regional settings for which the particular language can be displayed properly.

In column *Lang Message Id* is entered the name of the constant from *VfxMsg* table used for runtime localization of language combobox in login form and on main toolbar.

The settings described above cannot be changed for VFX Languages. They can only be entered and modified for newly added languages. Adding of a new language is possible from the *Add Language* button. A language added in that way will be visible in a combobox in login form and on main toolbar. Adding a new language adds also a column and a constant in a *VFXMsg* table. The last step needed to get the application localized in a new language is to translate all message texts in *VFXMsg* table.

To be able to use a newly added language, all texts in the table *Vfxmsg. dbf* must be translated into this new language.

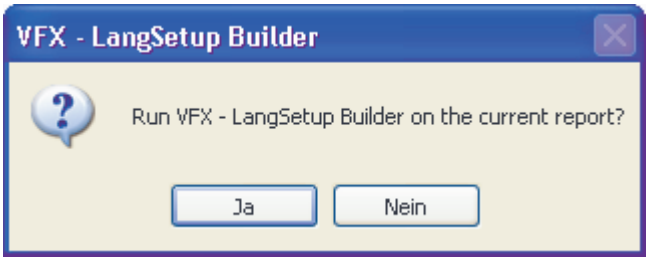
### Executing LangSetup method

When the *LangSetup* method at form level is called, all objects contained in the form are automatically searched for existence of *LangSetup* method. Container objects are searched recursively. Thus the *LangSetup* method is executed for all objects on the form.

### VFX - LangSetup Builder

In VFX 9.5 VFX – LangSetup Builder can be used to localize not only forms of developed application, but also the reports and texts for *XPStyle open dialog*.

To run VFX – LangSetup Builder on a report, the report must be open in design mode. Now you can select *LangSetup Builder* from the VFX menu. A message box will be displayed:




Answering Yes runs the VFX – LangSetup Builder on the currently open report. When the LangSetup Builder finishes all the labels in the report will contain text constants for captions which will be replaced with the corresponding translation of the label by the reportlistener when the report is rendered. Captions of labels in the report file are replaced with '^' char followed by *Message_ID* of the text in *vfxmsg.dbf*. At run time the VFX reportlistener class replaces these texts with corresponding localized texts.

If there is no form or report open when VFX – LangSetup Builder is run, the builder will localize the table *vfxfopen.dbf* .

## *Project Hook*

Since VFX 9.0 the global variables created based on fields in *Vfxusr* and *Vfxsys* tables are replaced by two global objects *goUsers* and *goSystem* and to keep backward compatibility a new include file *Vfxglobal.h* was included in VFX projects. In this file have been mapped former global variables with corresponding properties of the new global objects. Now, in VFX 9.5 this file is automatically updated with every project build, in such a way that it contains also fields that developer had added to the both *Vfxusr* and *Vfxsys* tables.

## *Product activation*

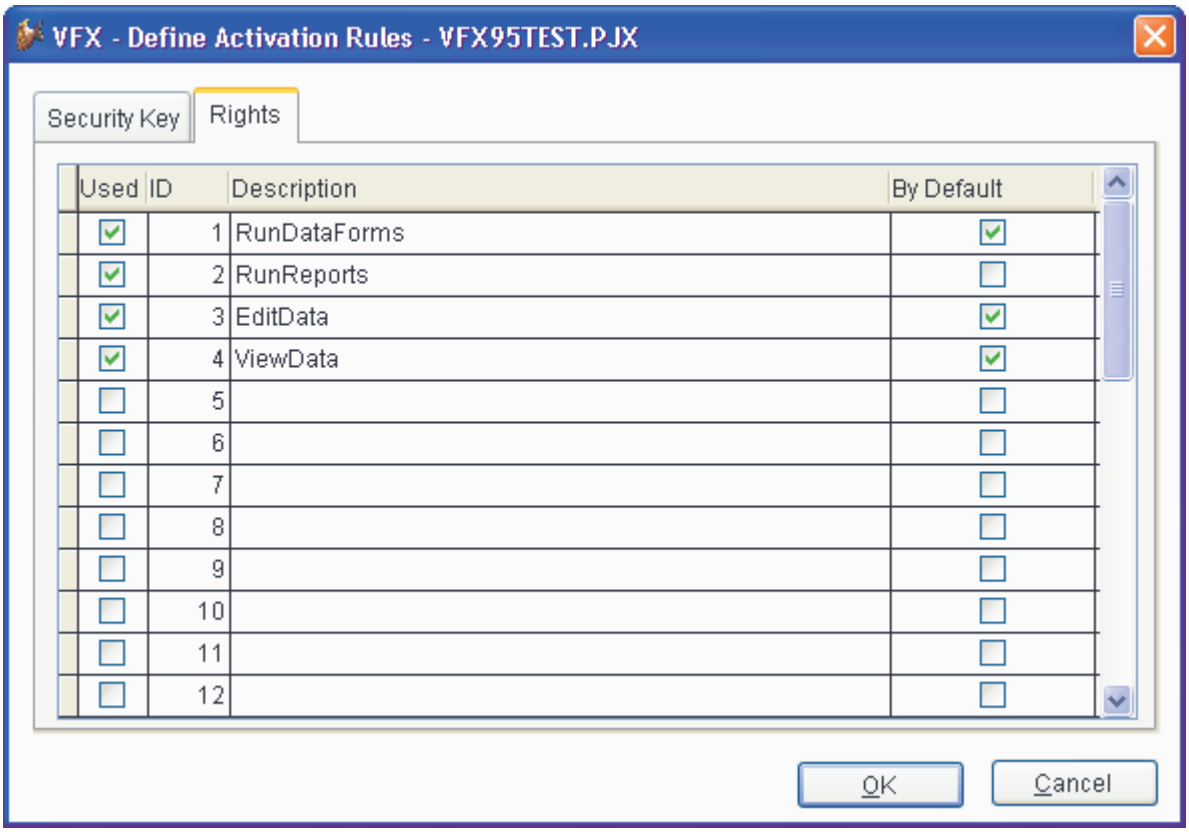
VFX 9.5 – Applications can be protected against unauthorized use by product activation. The product activation works in such a way as it is described in the user's handbook. Here are explained only the extensions in VFX 9.5.

Now, for the product activation integrated in VFX, is available a web service. Likewise the VFX activation, now using a web service, users can get an activation key for an application.

For managing customer data and the generated activation keys, a new application *VFX – Customer Management* is available.

## Defining Activation Rules

First, for the application must be set activation rules. This is done in the dialog *VFX - Define Activation Rules*. The definition of the rules is done exactly in such a way, as in VFX 9.0. The activation rules are stored encrypted in the class library *Appl.vcx* in the class *cVfxActivation* in *cActPattern* property.




To define rights in the wizard, first must be marked the checkbox in the first column. Then a name for the right should be entered in the column *Description*. At run-time the application will create a property of the *SecurityRights* object with this name.

After that a default value if this right will be grant by default, should be entered in column *By Default*. The Default value will be used for new customers, when they are edited in *VFX – Customer Management* application.

The registration number is a numeric value, 10 digits long. The user must give the registration number to the developer or send it by e-mail. The developer creates a new in record for this user the VFX – Customer Management application and enters here the registration number.

## Generating an Activation key

The 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.

Activation key for VFX 9.5 – Applications is generated in VFX – Customer Management application. The VFX – Customer Management is a separate project, delivered with VFX. From the VFX – Customer Management project can be compiled Exe file. Thus, the management of the customer data and generation of activation keys can be made on a PC independent of developing computer.

In order the VFX – Customer Management application to be able to generate activation key, the activation rules must be available to it. However, the activation rules are stored in the actual application in the class library *Appl.vcx*.

By generating the activation key the VFX – Customer Management application uses a registration DLL. This registration DLL holds information about activation rules. The project for building a registration DLL is under the project folder of the actual application and gets copied from the VFX - Application Wizard in every new project. The name of the registration project is: *"Register" + <application name>*

While compiling the registration DLL the activation rules from *Appl.vcx* class library of the actual application are read through the project Hook and are stored in the class *cRegDll* in *cActPatternName* property. In such way the registration DLL contains the activation rules. Thereby, the VFX – Customer Management is independent of a particular application. The VFX – Customer Management application can use registration DLLs for different applications and thus can be used for more than developed application. The registration DLL can be built from the VFX menu, using *Activation, Build register DLL* menu bar.

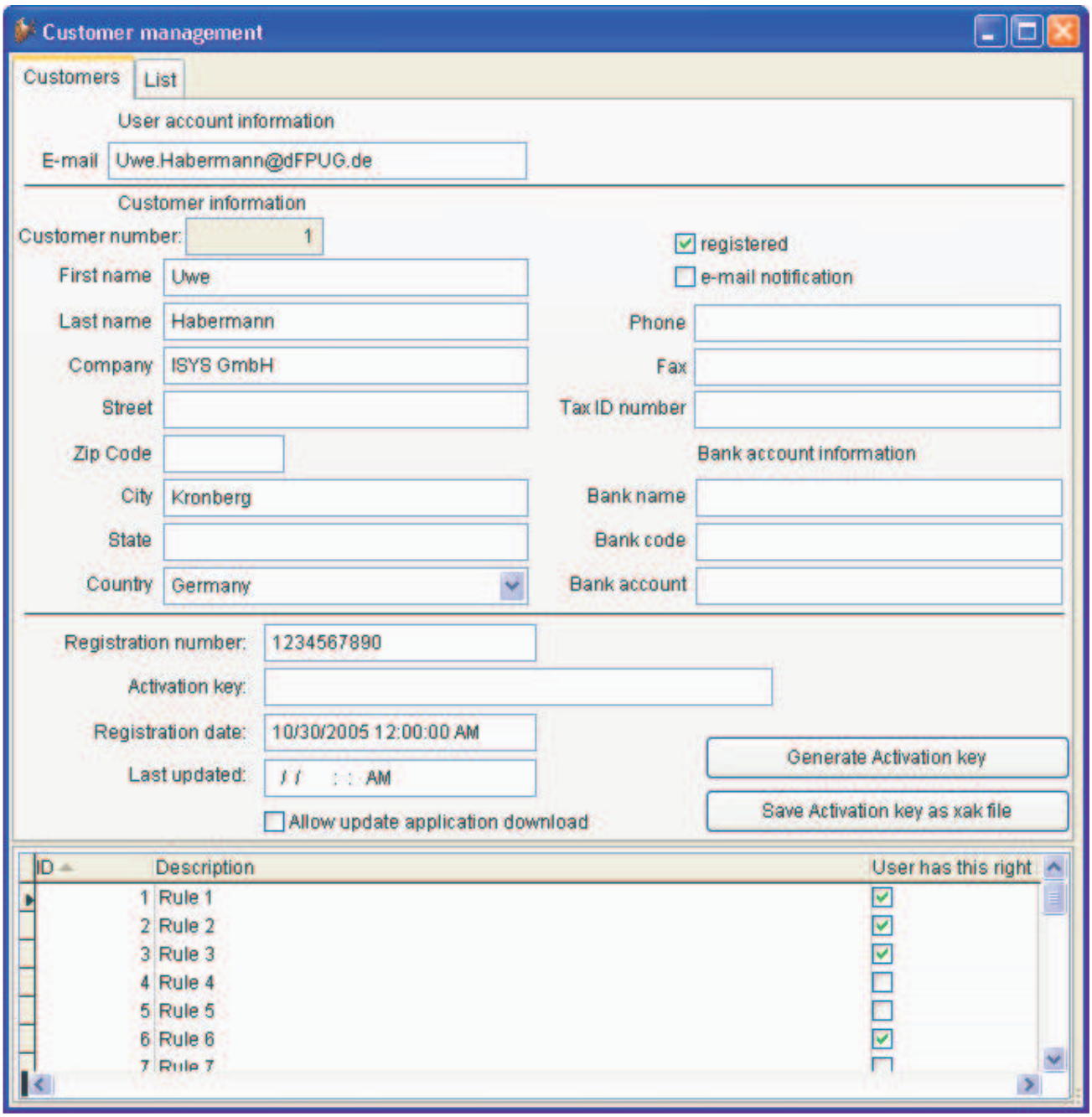
There is a method *GenerateActKey* in the registration DLL. Parameters of this method are the registration number, a string containing the rights to be granted and the path to the file *VFXGenActKey.app*. The return value is the generated activation key.

The registration DLL in turn, calls a function of the application *VFXGenActKey.app*. *VFXGenActKey.app* is delivered with VFX and is placed in the VFX – Customer Management project folder. *VFXGenActKey.app* contains the algorithm which constructs an activation key using the activation rules. The source code *of VFXGenActKey.app* is not delivered with VFX. The developers, who would have the source code available, could generate activation key for VFX applications of other developers.

The VFX – Customer Management application accesses the customer database *using Config.vfx* file. The customer database can be either a VFP database or a SQL Server database. The data access works exactly as it works in other VFX applications. For VFX – Customer Management application was added one more column to the file *Config.vfx*. The column *RegDllName* contains the name of the registration DLL to be used for generating activation keys.

## VFX – Customer Management

This application consists of two forms: Customer management and Application Versions.



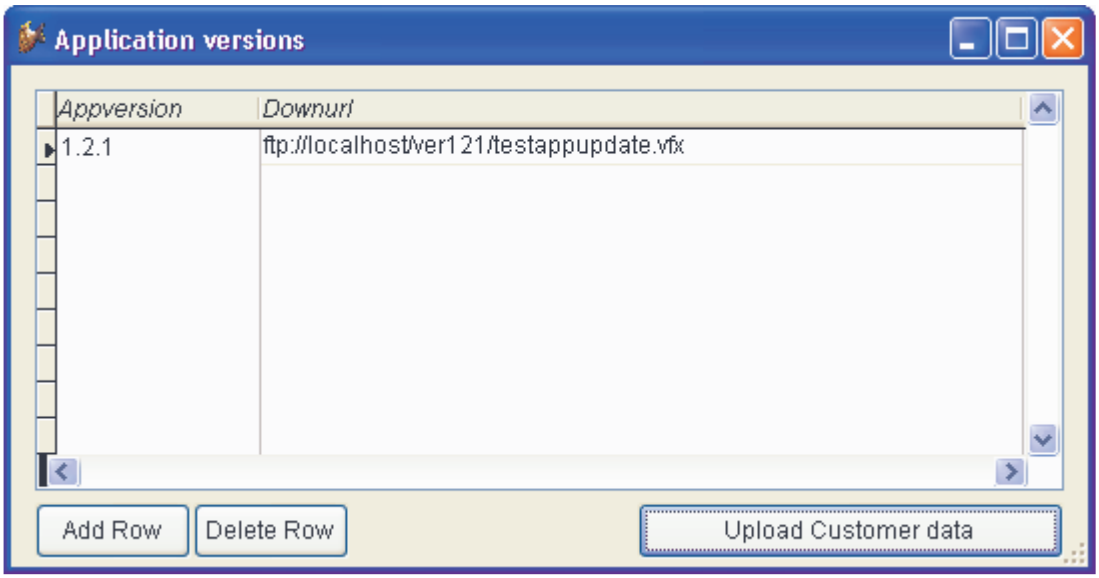

In the form *Customer Management* can be added and deleted customers' data. For every user is kept the installation key and granted user rights. When necessary rights can be changed and easy and fast to create and send a new activation key to end user.

*Generate activation key* button invokes the generation of the activation key, according rights granted to users. Generated activation key is saved back in Customers data.

Using the button *Save activation Key as xak file* the Generated Activation key is stored in file named *<Projectname>.xak* in current folder. The Activation key or the file must be sent to the user, to be used for application activation. When the application starts at end user side, this file is automatically read and the activation key contained in it is used to activate the application.

From the customer management application can be sent emails with the XAK file as attachment. These emails are sent if the registration type *(cvfxActivation.nRegWay)* of the application is different than 10 (online). When the registration type is 10, it is expected that the user gets the activation key through a web service. In this case the user gets email with instructions how to obtain the activation key. The subject and text of the generated email can be edited in the form *Manage E-Mail Texts*. The E-mail text is processed with *Textmerge* VFP function and can contain and can contain any valid expressions like fields from the customers table. In this way can be generated and sent personalized emails.

The form *Application Versions* is used to maintain the *versions* of the application and the *download URL* to update it. New versions of the application can be provided on Internet server. If a customer wants to update his application, the application downloads the file *UpdateCustomers.vfx* first. In this file there are listed the registration numbers of customers entitled to actualize their application. If the user is authorized to update the application, the file *UpdateVersions.vfx* is downloaded. In this file are listed download links to the available application versions. The download links to files *UpdateCustomers.vfx* and *UpdateVersions.vfx* are stored in the application and can be set with the VFX – Application Builder.




The two files *UpdateCustomers.vfx* and *UpdateVersions.vfx* are generated by clicking on the button *Upload Customer Data*.

In order to upload these files on the internet server, VFX – Customer Management applications, needs login parameters. Login data (username and password) are kept in the registration DLL project in *Regdll.vcx* class library in *cRegDll* class.

### Properties

*cFtpUrl* – Domain name of the FTP server.

*cFtpDir* – Folder on the FTP server, where *UpdateCustomers.vfx* and *UpdateVersions.vfx* files should be upload.

*cPort* – Port, used to connect to the FTP-Server. By default for FTP upload is used port 21.

*cUserName* – Username for FTP access.

*cPassword* – Password for FTP access.

## Registration Web Service

Along with the VFX – Customer Management project is delivered the project *RegistrationWebService.pjx* in the subfolder *RegistrationWebService* under the VFX – Customer Management folder. From this project can be compiled a COM server, which can be installed on Internet information server as a web service. Applications can send the end-users registration data to this web of services. The web service connects to customers database, saves received data and can send an activation key back to the application.

The web service uses the file *Config.vfx* for the data access. Here must be set the same database which is also used from the VFX – Customer Management application. In the simplest case the web can use service in the same folder where the VFX – Customer Management application is installed and thus, use the same *Config.vfx* file.

If the web service should run on remote internet server, the customer database can be upsized on a SQL server with the help of VFX - Upsizing Wizard. The internet web services, as well as the VFX – Customer Management can access this SQL server database.

In order the application to be activated using web service, must be made a few settings must be in the application in the form *VfxRegister.scx*:

*cWSDL* – Contains the URL of the WSDL file. This file is generated when the web service is registered with the SOAP Toolkit on the internet server.

*cServiceName* – Contains the name of Web Service.

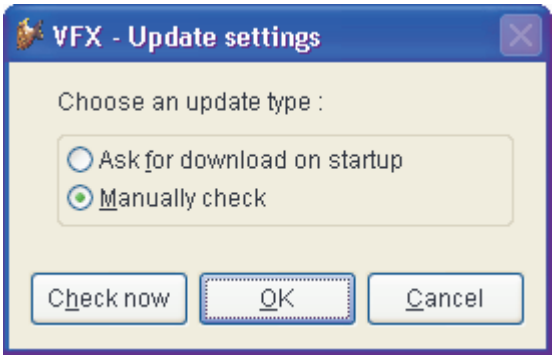
*cServiceMethodName* – Contains the name of used web service method. Normally this is the method *GenerateActKey*.

If a customer registers through the web service, the registration data will be transferred to the web service in XML format. The web service checks in the customer database for a record with the same E-mail address as well as the same application name and the same version. If this record is found, the activation key saved for this user will be sent back to the application. With this, the application is automatically activated. If such record is not found, the registration information is saved in the customer database.

The registration web service contains also the method *ReceiveErrorInfo*. This method can receive error messages of end-user applications. Using it, users have the chance to send to the web service data, which are stored locally in the error protocol table *Vfxlog*.

## *VFX Update*

It is worth to update VFX regularly, so that always the actual state is used. VFX 9.5 can automatically check for update. This can be set in the dialog *VFX – Update Settings*. If the option *Ask for download on startup* is selected, VFX checks if a new build is available every day on first start. If a new build is found, developer is asked whether the new Build should be downloaded and installed. The check is not performed, if no internet connection exists. With the button *Check for updates now* the check for new build can be started at any time manually.




### The class cUpdate

This class provides the dialog for keying up the update settings. With the *Check now* button can immediately be checked if updates are available. The class is used by VFX internally as well as in final applications.

*tlUpdateApp* - Initial parameter.

.T. – Update should be run for the end-user application

.F. – Update should be run for VFX.

### Methods

*WriteData()*

Saves update settings into table *Vfxsys*.

## The class cUpdateEngine

This class is used to execute application update. According the settings, made in update dialog, it checks if a new version is available by downloading a file containing the latest version number. When a new version is available, depending on settings, the class downloads and runs the updates. For this purpose an object of *cDownload* class is used.

## Properties

*cIniURL* – URL of the file containing the latest version number available for download.

*cVersion* – Contains the current version number of the installed application.

*dLastChecked* – Holds the date when last has been checked for updates. This value is retrieved from *Vfxsys* table.

*nManualCheck* – Specifies how the update was started. Internally used.
0 – Automatic check for updates on startup of application
1 – Manual check for updates has been run from menu.

*nUpdateType* – Selected type of update. This value is read from *Vfxsys* table.
1 – Automatically download and install updates if available. The check is performed one once a day on startup.
2 – Automatically download updates if available and after that asks if the updates should be installed. The check is performed once a day on startup.
3 – Checks if new updates are available for download. If there is a new build, the user is asked if updates should be downloaded and installed. The check is performed one once a day on startup.
4 – Manual check. No automatic checks for updates are performed. Update can be run from Help menu.

## Methods

*CheckUpdType()*
Retrieves version number of installed application, Update type and date of last check for updates. The retrieved values are stored in properties *cVersion, nUpdateType* and *dLastChecked*. Returns .T. if all values are successfully retrieved. In case of error, .F. will be returned.

*DoUpdate(tnUpdateType)*
*tnUpdateType = nUpdateType*
Calls *UpdateApp*(*tnUpdateType)* (in *Vfxfunc.prg*)

*NewVersion()*
Validates *cIniURL* and downloads the file. Return values :
0 – No new version is available
1 – A new version is available
2 – An error occurred.

*SetNewCheckDate(tdDate)*

*tdDate* – Date to be saved as last date, when check for new update has been performed
Updates the date of last updates check in Vfxsys table (field *lastcheckd*) with *tdDate*.

*StartUpdate(nManualCheck)*

*nManualCheck* – Sets the type of check for updates that is performed.
0/.F. – Automatic update. Sets class property *nManualCheck* = 0
1/.T. – Manual update. Sets class property *nManualCheck* = 1
This is the main method of this class. It starts the actual update. However, before it is called the property *cIniURL* should be set, otherwise update will not be performed. Methods *CheckUpdType()* and *NewVersion()* are consequently called. The method *SetNewCheckDate(tdDate)*is called to set date when last check has been made. If there is a new version available *DoUpdate(tnUpdateType)* method is called. If the check is performed manually, the method *DoUpdate(tnUpdateType)* is called with no other pre-checks.

An instance of the class *cUpdateEngine* is available in *Appl.vcx – cAppUpdateEngine*, where developers can make their own setting for application update in property cIniURL. This property can also be set using VFX – Application Builder.

## *AFX Support*

There are two new methods in *cFoxAppl* class, designated to help you easy to port your VFX application to a web application, using AFX wizard. These are *VFXMessageBox* and *VFXWaitWindow* methods. They receive same parameters, as you would pass to VFP MESSAGEBOX() function or to WAIT WINDOW command.

### Methods

*VFXMessageBox(eMessagetext, nDialogboxtype, cTitlebartext, nTimeout)*

*eMessagetext* – Text to be shown in messagebox. If this parameter is missing or if it is of wrong type, an empty string is displayed.
*nDialogboxtype* – Messagebox type. This parameter specifies buttons and icons for the displayed Messagebox. If missing or wrong type is passed, defaults to 0.
*cTitlebartext* – Text displayed in title bar of message box. If this parameter is missing or if it is of wrong type, an empty string is displayed.
*nTimeout* – Time interval in milliseconds for displaying the messagebox. If this parameter is missing or if it is of wrong type, message box is displayed till a button is pressed.

This method works in same way as the standard Visual FoxPro command MESSAGEBOX(). If the application is run as a server in quiet mode no message box is visualized, and the value of the default button is returned, otherwise the value corresponding to the pressed button is returned.

*VFXWaitWindow(tcMessageText, tnRow, tnColumn, tlNowait, tlClear, tlNoclear, tnTimeout)*

*tcMessageText* – Text to be shown in wait window. Defaults to empty string.

*tnRow* – Number of row to position the wait window. Works together with *tnColumn*.

*tnColumn* – Number of column to position the wait window. Works together with *tnRow*.

*tlNowait* – .T. to continue program execution after displaying wait window, .F. to stop and wait for an action (Click or KeyPress). If wrong type is passed or omitted defaults to .F.

*tlClear* – .T. if all previous wait windows should be released. If wrong type is passed or omitted defaults to .F.

*tlNoclear* – .T. if wait window should remain in the screen until *WAIT CLEAR* is issued. If wrong type is passed or omitted defaults to .F.

*tnTimeout* – Time in seconds during which the wait window is displayed. If 0 wait window is not cleared until WAIT CLEAR is issued. If the wrong type is passed or omitted, defaults to 0.

This method does the same as the standard Visual FoxPro command *WAIT WINDOW*. If the application is run as a server in quiet mode no wait window is visualized.

## *Help*

In the new help section of the VFX menu can be opened the user manual and various useful information about VFX as well as links to online resources.

### User manual

From *VFX Help* menu entry, can be opened *User manual* and *What's New* documents in German and English language. All these documents are files in PDF format. In order to open them, is needed to have the Adobe® Acrobat Reader installed.

If one of the documents to be displayed is not installed, the appropriate document is automatically downloaded.

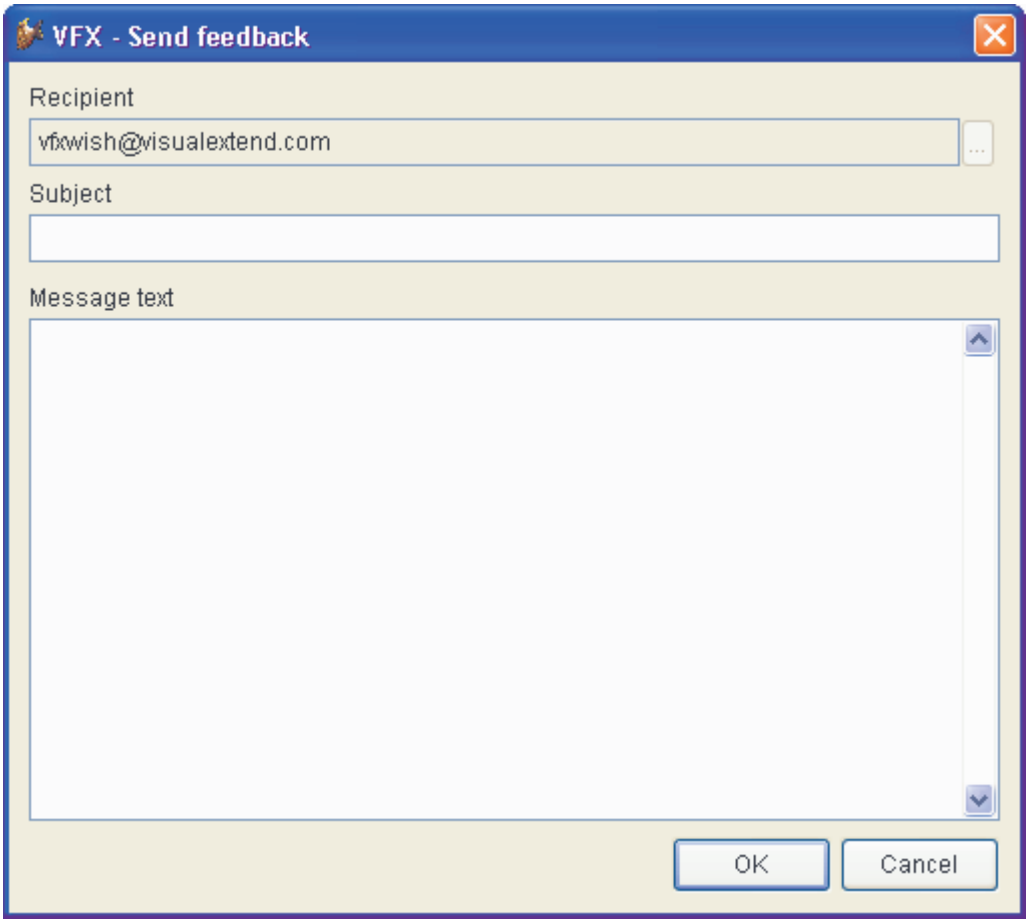
### Visual Extend Online

Links to the following sections of the VFX website are available under the *VFX Help->Visual Extend Online* menu entry.

- VFX Home page (available in four different languages - German, English, French and Spanish)
- VFX web directory (start page for the Visual Extend Portals)
- Forum (you can read and post in online forums)
- Newsletters (receive the news about VFX)
- DevCon (visit the developers conference)
- Online shop (buy VFX, books, VFP and other)

## Send us an Email!

Send us suggestions, opinions, comments and anything you think is worth sharing to the VFX team by filling out the form, opened by selecting *VFX Help->Send user feedback.*

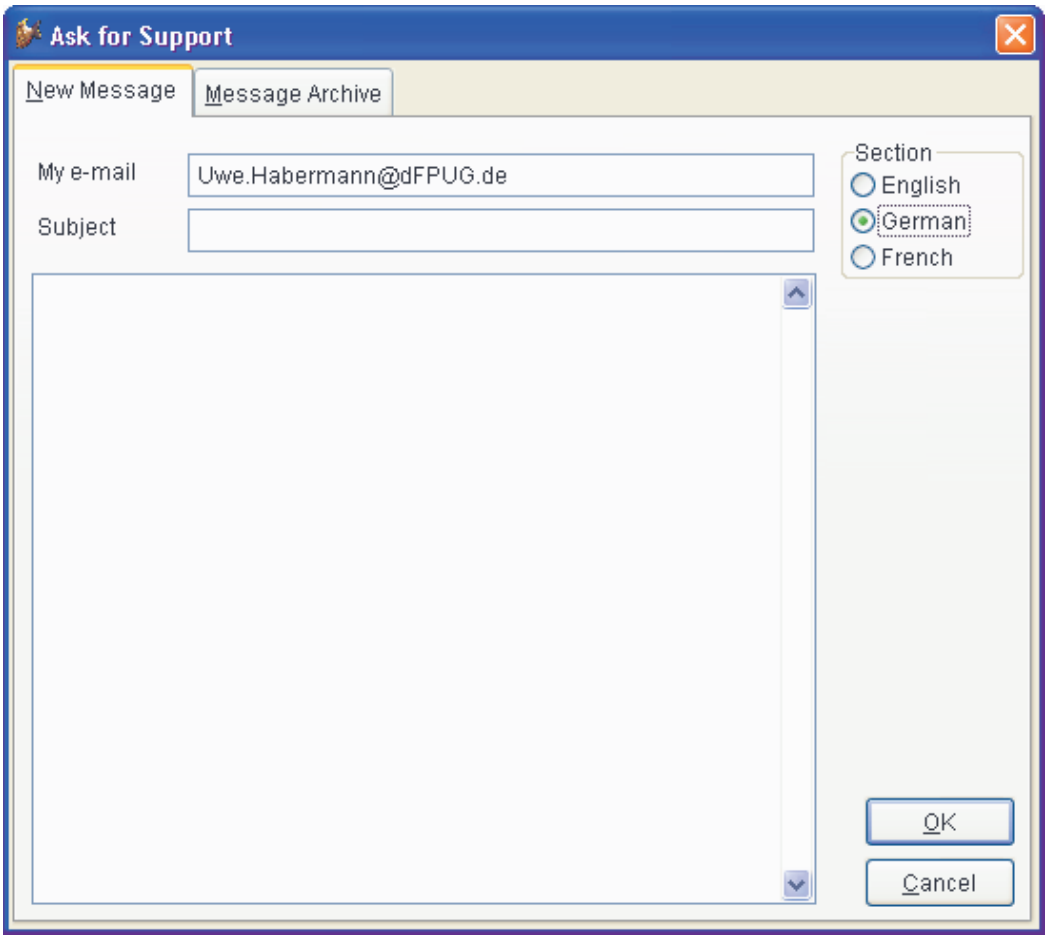


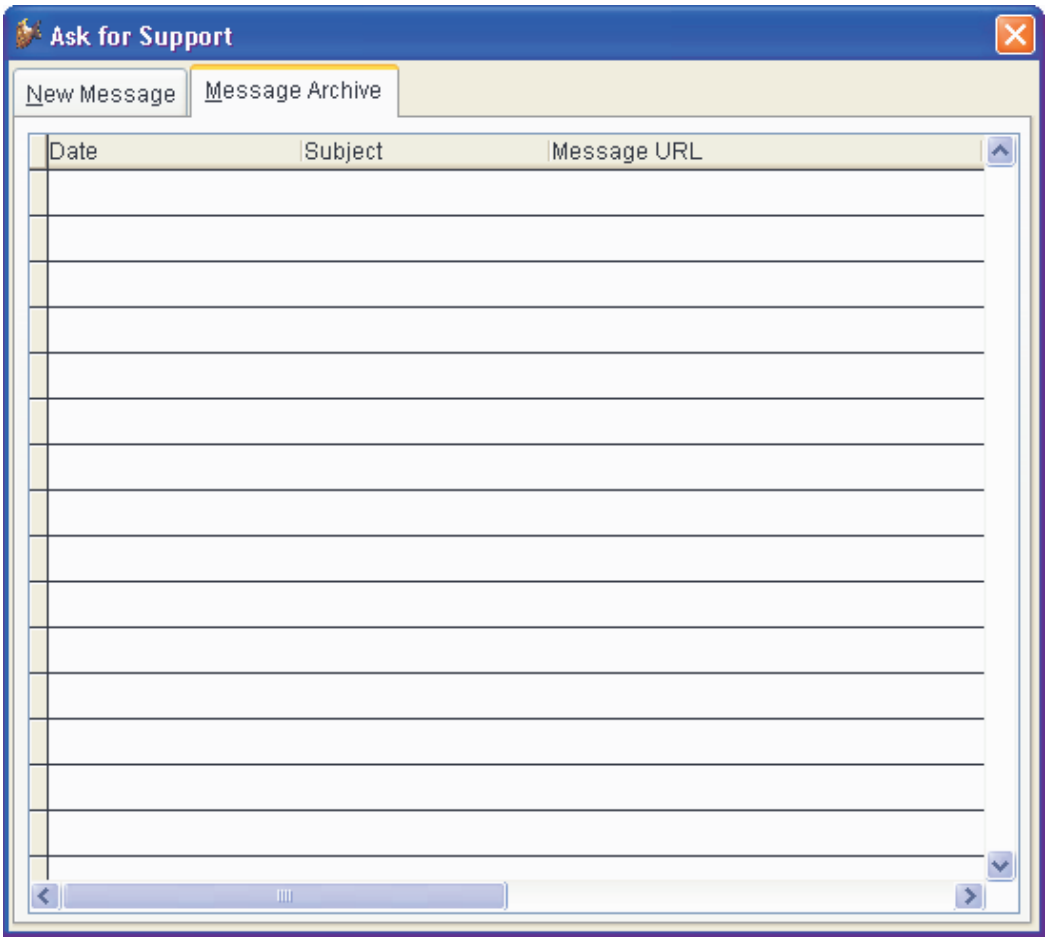

## How to reach us

Another web page with helpful links to teach us is opened on selecting *VFX Help->How to reach us*.

## Ask the VFX Forum for Support

There are two possible ways to post a VFX support question in the forum. You can go directly to the forum on the dFPUG website with menu entry *VFX Help, Visual Extend Online, Forum*. The start page of the forum is opened. Here you can be read messages on-line and also post new messages. The other way is to send a forum message from within VFX by selecting menu entry *VFX Help->Ask for Support.* A dialog, where you can write your message, opens. To be able to send messages, you need to have internet connection. The news written on this way remains stored and can be reviewed on the page *Message Archive* of the dialog.


Ask for Support
New Message
Message Archive
My e-mail
Uwe.Habermann@dFPUG.de
Subject
Section
English
German
French
OK
Cancel


Ask for Support
New Message
Message Archive
Date
Subject
Message URL

## About

Information about current Visual Extend version and registration information is accessible from *VFX Help -> About Visual extend*.

## *New properties of the Application object*

*lHideRegistrationFiles* – When value of this property is set to .T., the registration files *VFX.ini* and *FirstInstall.txt* will be created with hidden file attribute set. .T. is default value.

*lInstallClickyes* – This property specifies if *ClickYes* application should be automatically installed (if missing) when email will be send using *Outlook*. The application *ClickYes* suppresses security questions from *Outlook*. Default value is .T.

*lTableprompt* – Disables/enables display of Open dialog box when no table(or view) is open. The value, set in this property is used for *SET TABLEPROMPT* command. The Visual FoxPro commands that need a table open in the current work area, display file open dialog, if there is no open table. When this property is set to .F., file open dialog will not be invoked at run time in case when no table is open in the selected work are. Default value is .T.

*lVarcharmapping* – This property controls character data expression mapping to varchar field type in query result sets. Its value is used in *SET VARCHARMAPPING* command. If its value is set to .T., data types with variable field length from the remote data source are mapped to the VFP field type *Varchar*. The default value is *.F.* This corresponds to the VFP setting *SET VARCHARMAPPING OFF*

*nAllowSaveEmptyRecords* – Allow Save Empty Records. This property specifies if empty records saving in all forms is allowed. If its value is set to *2*, in all forms of the application users are not allowed to save empty records. When its value is set to *1*, users can save empty records in all application forms. The value *0* specifies that this feature is keyed up per form with the form property *lAllowSaveEmptyRecords*. The default value is *0*.

*nComboBoxListEntriesLimit* – Number of items shown in drop-down list of comboboxes. Default value is 15

*nReprocess* – Number of attempts to lock a file or record after an unsuccessful locking attempt. Default value is 0

*nUseReportbehavior80forPDFOutput* – If report is saved as a PDF file with *SET REPORTBEHAVIOR 90* setting, it is not possible to copy texts from the PDF file, because it is generated as an image. With this property it is enabled to generate PDF files using *SET REPORTBEHAVIOR 80* setting, so that users can copy text. If the value of this property is *0*, the form setting, made in the form's property *lUseReportbehavior80forPDFOutput* controls this behavior. If the value is set to *1*, *SET REPORTBEHAVIOR 80* is used in all forms on PDF files generation. If the value of this property is set to *2*, the current SET REPORTBEHAVIOR for all forms remains in effect.

## *Other developer enhancements*

VFX Registering dialog is already localized.

In VFX – Messagebox Builder and in VFX – Message Editor can be edited with using appropriate Unicode settings.

In VFX – Form Builders is available for selection a large list of VFX classes.

The VFX – Form Builder uses *cPickDate* VFX class as a default for fields of date data type. In it, the end-user has a chance to select the desired date from a calendar control.

In the VFX - Class Switcher in the selection dialog can now be selected any class library and class from the active project. The classes and class libraries is alphabetically sorted. If developers attempt to select a non suitable class, a warning message is displayed and the process is canceled.

In the VFX menu under *Project* menu entry, is now possible to create backup archive of the active project. File name is generated from the project file name, followed by a time stamp.

The Debug menu for the development environment can be switched on in the VFX - Application Builder. Alternatively the property *lDebugMode* of the class cFoxAppl can be manually set to .T. in the class library *Appl.vcx*.

There is a new function in *Vfxfunc.prg – GetNewGUID()* that can be used to generated global unique identifier. This function uses the API function CoCreateGuid() declared in OLE32.DLL and returns a string containing a globally unique ID (GUID) comprised of 32 hexadecimal digits and 4 delimiters, so the total length is 36 characters. The generated value can be used as unique identifier (ID) in your tables when it is expected to merge data from multiple tables.

Now when updating child records data in OneToMany forms, the fields holding the user name (*ins_usr, edt_usr*) and the timestamp (*ins_date, edt_date*) also filled automatically with data by VFX.

There is an event hook for *Destroy* event in the class *cFormbase*. With it is possible to insert your own processing on form closing.

# New properties for end-users

## *Required user rights to run the application*

The application created with VFX 9.5 conforms designed for Windows XP guidelines. The application can be started by users with standard user rights. According Windows design guidelines, executable files (Exe file, Vfx.fll) are placed from the installation under *C:\Profram files* folder. All other data files as well as files being created by the application, should be stored under *Documents and Settings* folder. The application automatically creates a folder C:*Documents and Settings\All users\Application Data\<Company name>\<Product name>*.
If current user is not allowed to save or create files in this folder, a subfolder is created in his profile: C:*Documents and Settings\<Current user>\Application Data\<Company name>\<Product name>*.

At run time files *VFXPath.dbf* or *Config.vfx* are searched in following sequence:

- Installation application folder (Exe folder) (for backward compatibility)
- C:*Documents and Settings\All users\Application Data\<Company name>\<Product name>*
- C:*Documents and Settings\<Current user>\Application Data\<Company name>\<Product name>*.

When on of these files has to be created, the application will attempt to create it in the folder C:*Documents and Settings\All users\Application Data\<Company name>\<Product name>*. When current user is not allowed to create file in *All users* folder, the file is created in the folder C:*Documents and Settings\<Current user>\Application Data\<Company name>\<Product name>*.

When working with remote database, the free table *VFXAComp.dbf*, used by VFP for storing *Autocomplete* values, will be created also following the strategy, described above.

## *New Icons*

Many new Icons were designed for end users and give a clearly improved appearance to the applications. Numerous new Icons have been also integrated into the VFX – Builder to improve the intuitive usage for developers.

## *Data access*

### Manage data access dialog

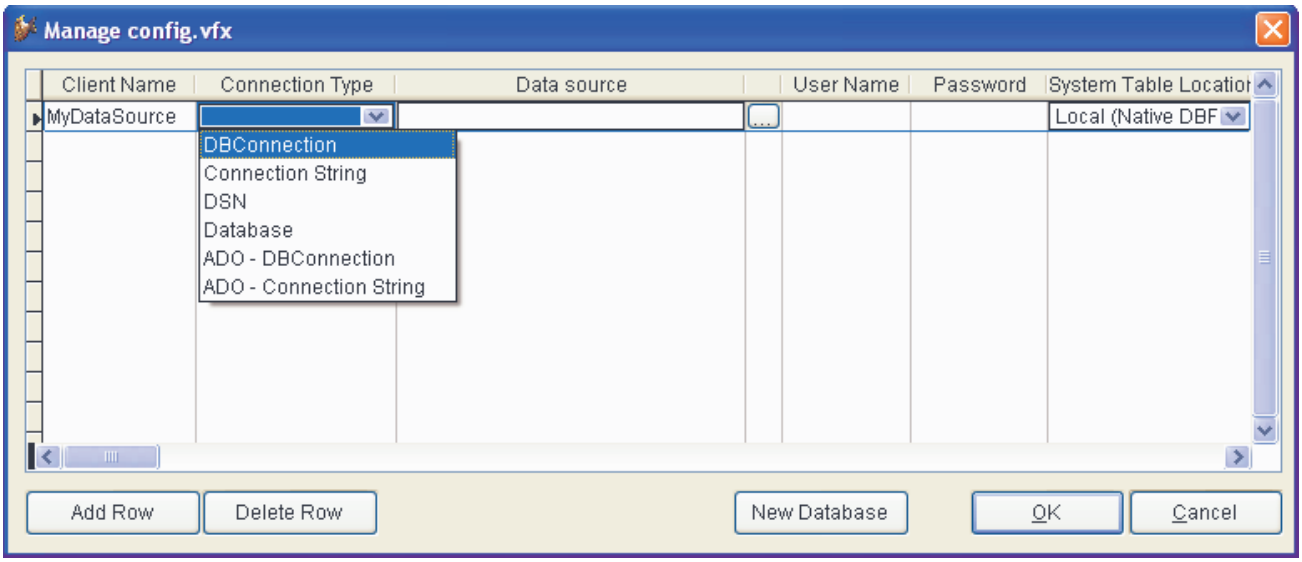
Additionally to the data access ways, used in VFX 9.0, now VFX 9.5 provides a new functionality to use OLD-DB connection to your data. The functionality of *cConnectionMgr* class is extended to support OLD-DB connections.

In *Manage Config.vfx* form it is possible to select among the two new ADO type connections:

ADO – DBConnection: This type will use a connection stored in a DBC.

ADO – Connection String: A connection string for the OLE-DB Provider.

These two ADO connection types are similar to the ODBC types *DBConnection* and *Connection String*.



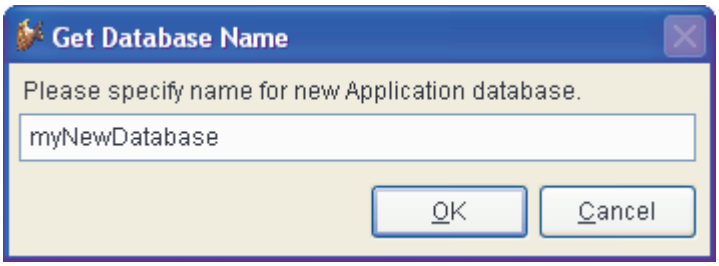

Now the VFX tables can reside alternatively also in a separate remote database and are available in same way as the main database. All connecting types are also available for the database containing the VFX tables.

### Creating a new database at customer side

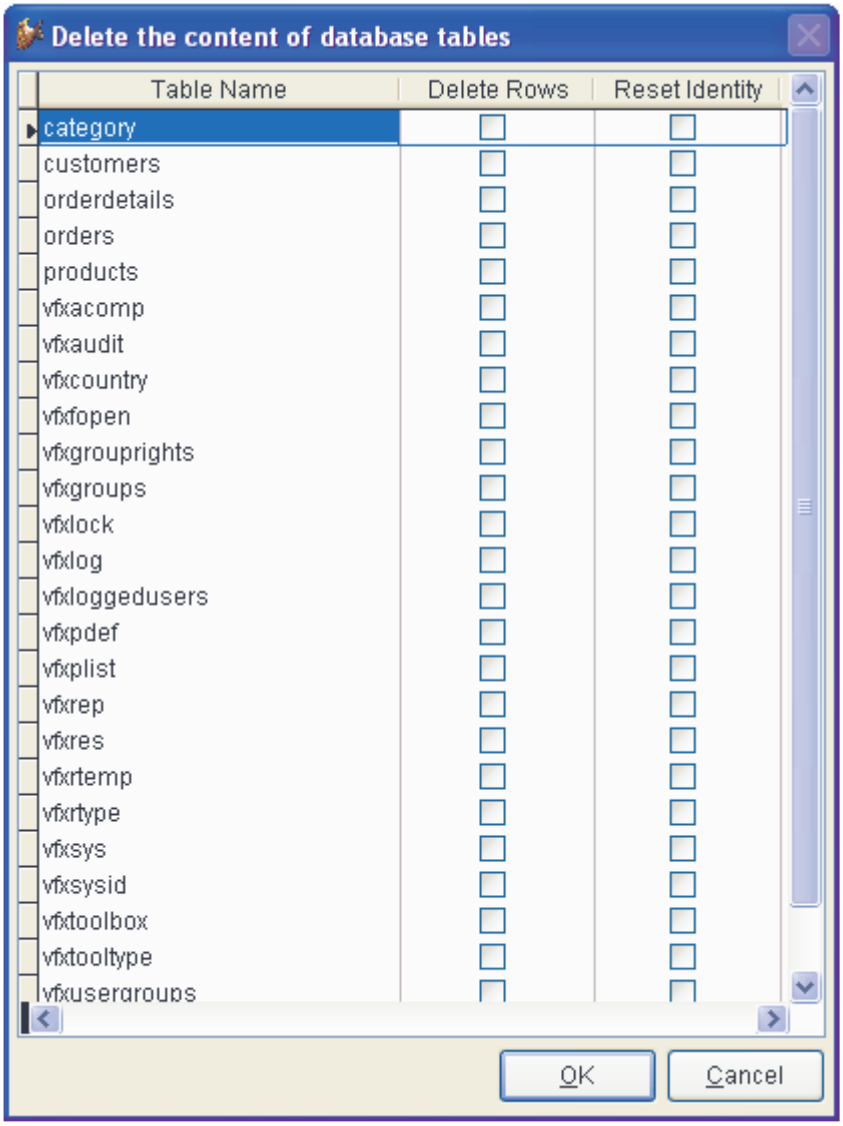
In applications, developed with VFX 9.5 it is now easy to create a new client database. At run time a new button *New Database* is available in *Manage config.vfx* dialog. Users are able to select one of existing databases as a basis for new database and click on *New Database* button.

When the selected database is VFP database, a select folder dialog is invoked and the used should specify the folder, where the new database will be created.

In case when any remote *Connection Type* is selected, a dialog window asks user to specify name of the Database to be created




In next invoked dialog, user is asked which tables are to be cleared (delete all records) for the newly created database. It is also possible to specify if Autoinc (Identity) values should be reset for every particular table.



## *New features in One-to-many form classes*

The data records of child grids in one-to-many forms can now be copied in other applications with Drag & Drop. This option can be set in VFX – COneToMany Builder and in VFX – CChildgrid Builder

In VFX – OneToMany Builder and in VFX – CchildGrid Builder, but every column in the grid can be specified in it will be included in the data copied OLE Drag & Drop.

To every column in a Child-Grid on an OneToMany form can be calculated a summary value. This option can be set in the *VFX – COneToMany Builder* and in the *VFX – Cchildgrid Builder*. If a sum should be calculated, a label with the column caption as well as a text box with the sum value is added below the child grid.

The date as well as the time, if necessary, and the user name are also saved with child records if the correspondent fields exist in the child table.

### OnChildRequery

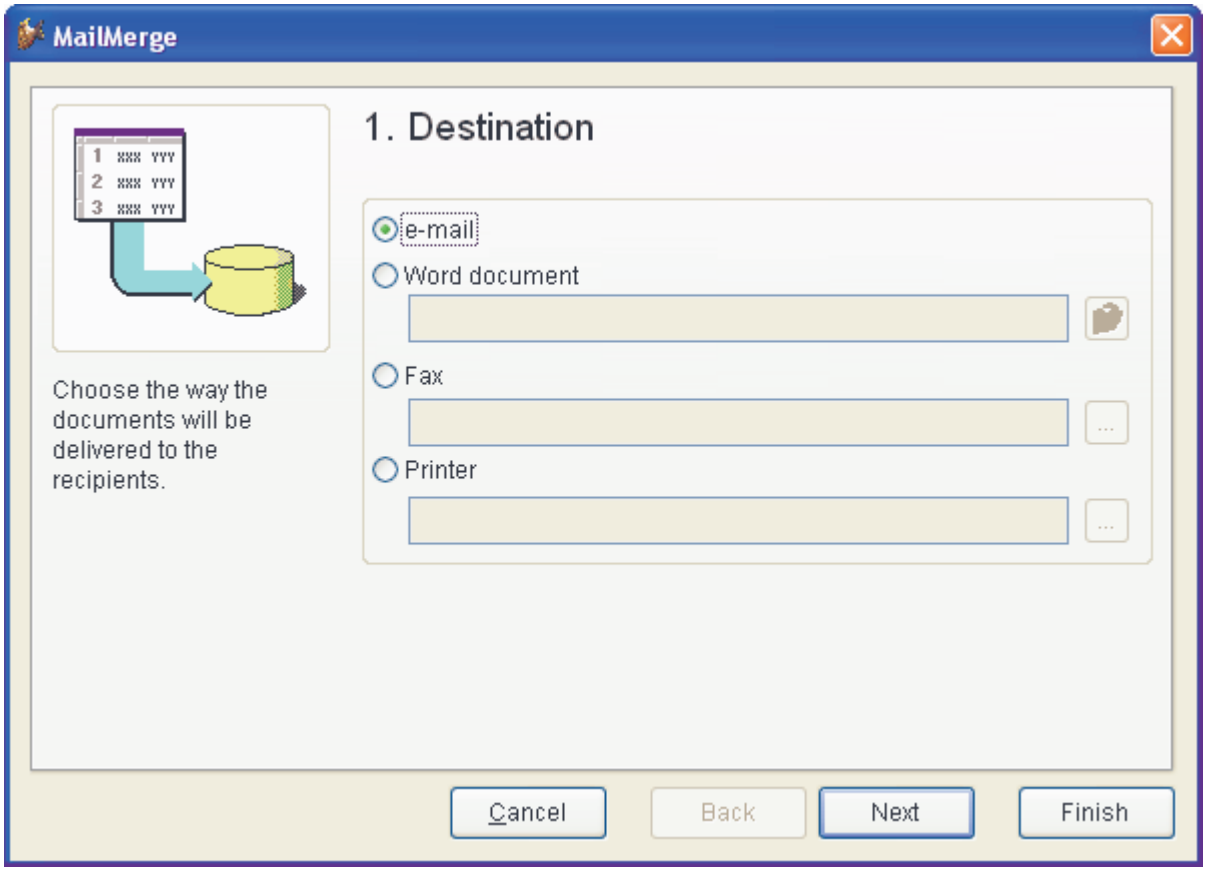
In earlier VFX versions code it has been necessary to write code in the method *OnChildRequery* of OneToMany forms in order to refresh the Child data when they are based on views or CursorAdapter classes.

In VFX 9.5 this is no more necessary. While moving the record pointer in the Parent part of the form all Child aliases are automatically checked. If a Child alias is based on a view or a CursorAdapter class, the data are automatically updated. For views is issued *REFRESH()* command. If the alias is based on CursorAdapter object, its *CursorRefresh()* method is called.

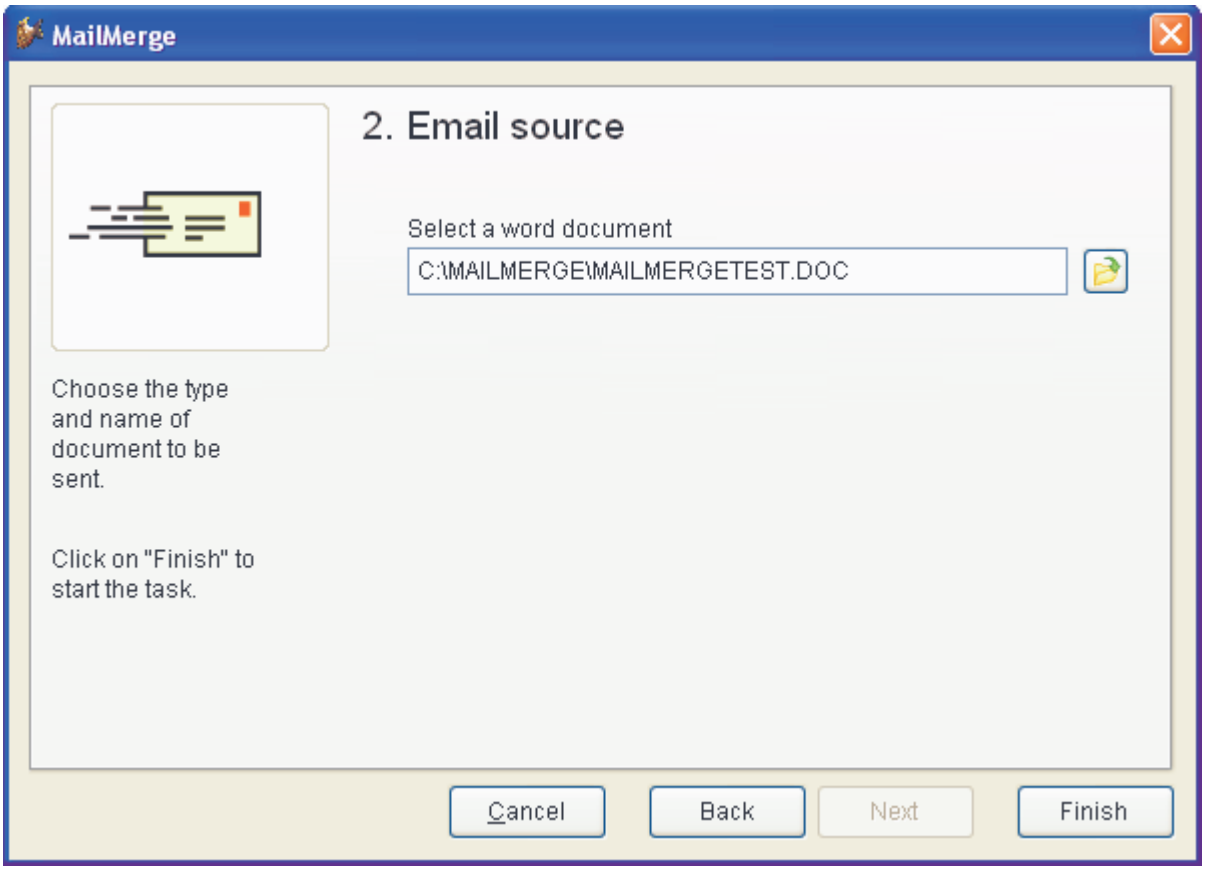
## *Mail Merge documents*

This new end-user Mail Merge wizard, gives the amazing chance to users to use their data as a Mail Merge data source and to generate Mail Merge output. As body for generated documents can be used MS Word Mail Merge document, text file or simply some text, entered by user in the wizard. The result can be saved as a word document, printed, faxed or sent as an e-mail. User is guided by the wizard through a few very intuitive steps.

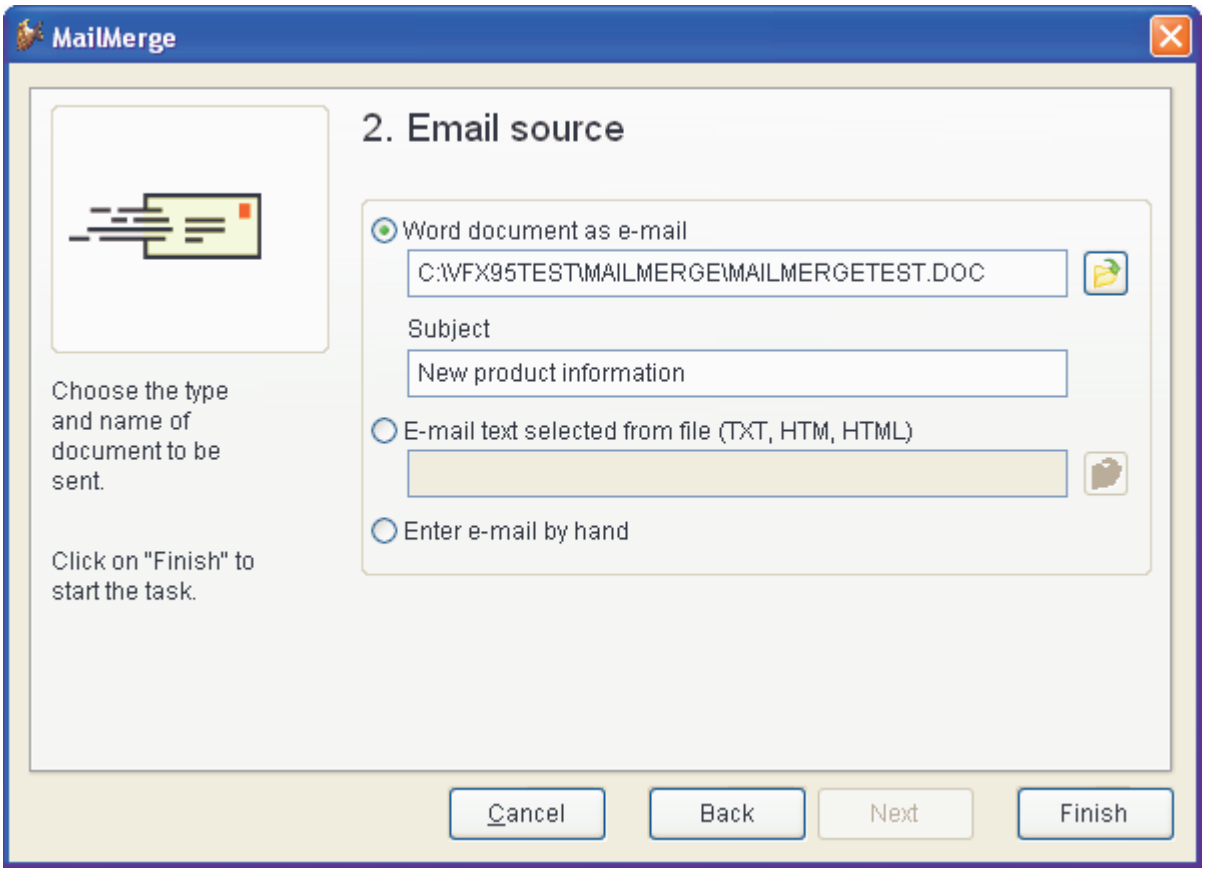
In the first step the user is asked to specify desired output type.



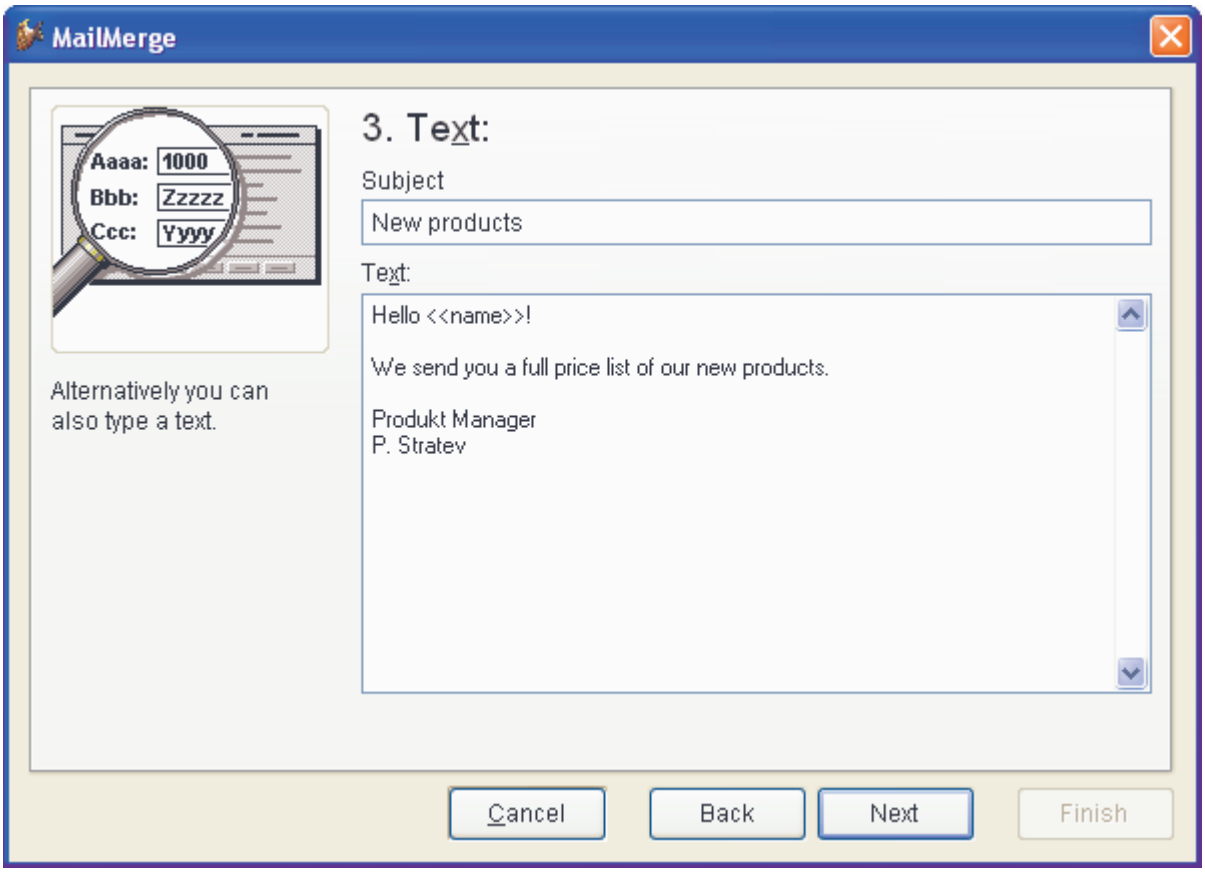
On next step is selected the source for generated documents. If in Step 1 Document, Fax or Printer is selected, at Step 2 user should specify the file path and name of a mail merge word document.



If E-mail is selected in Step 1, at Step 2 user will be asked to select among three possible sources for the e-mail body.



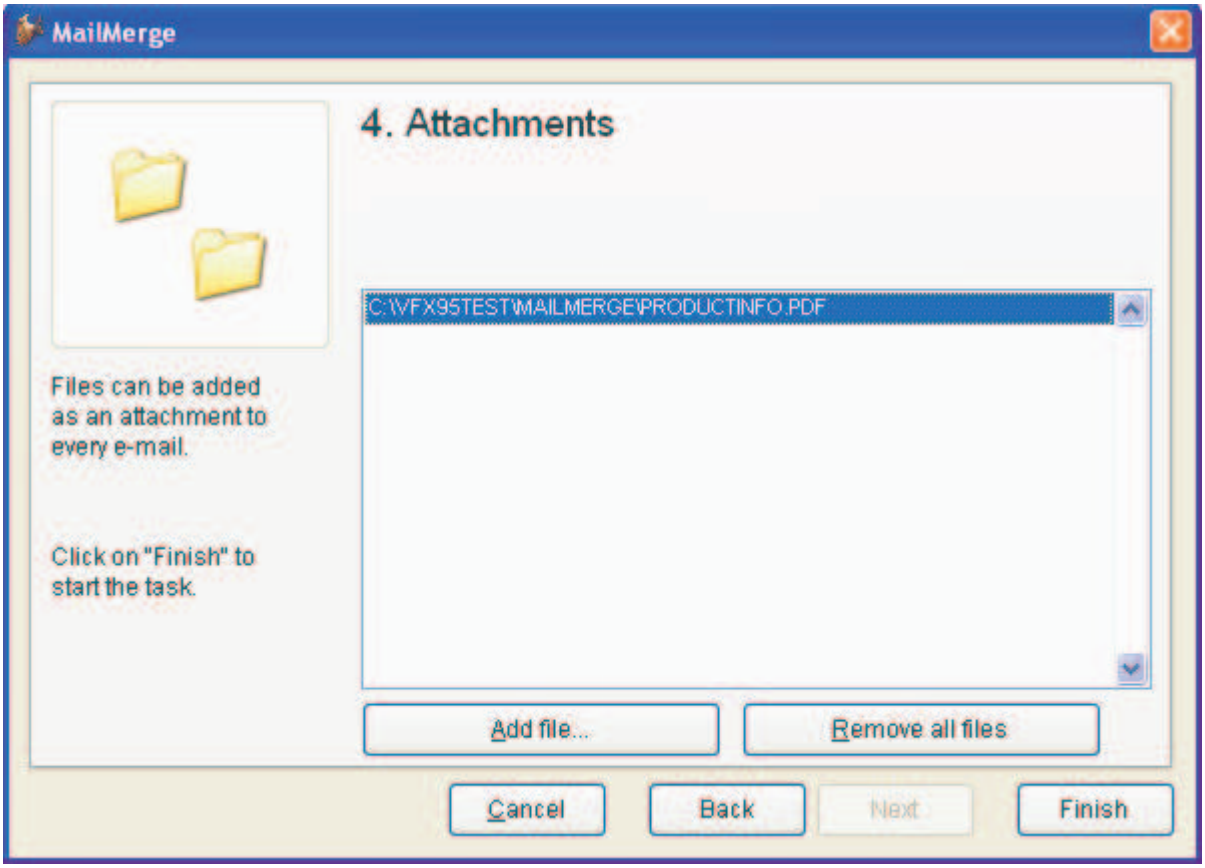
If as document source is selected text, entered by hand, on third step user should write the text for document body.




If a text file is selected to be used as source, its content is displayed here and can be edited in necessary.

In this text, similar to the Word documents, can be used also data fields. The names of variables that user wants to use in the text, should be enclosed in special characters. By default the textmerge characters “<<” and “>>” are used. Developer can change this be setting *cLeftDelim* and *cRightDelim* properties of *cMailMerge* class.

Next step is used to give the users chance to add additional attachments to generated emails.




By clicking on the button Finish, the mail merge documents will be generated according selected options. At last step of the wizard, after the requested action is completed, user is informed for the result.

## cMailMerge class

This is a form class placed in *VfxForm.vcx* class library

With this class the end users have the chance at run-time to build mail-merge documents using data kept in the application. Main features are:

1. Email
   Generating bulk emails. The email text can come from a Word document or a text file or be entered also by hand in an *Editbox*. When the email text is generated, it is also possible additionally to attach to the email as many files as necessary.
2. Word documents
   Generation of a Word mail merge output, based on a Word mail merge document. The Word mail merge document can be edited in Word as needed.
3. Fax
   Bulk sending faxes based on a Word mail merge document.
4. Merge and Send documents to Printer.
   Printings letters based on a Word mail merge document.

In order to use the class, it is needed to have a table with all fields used in merged documents or texts and also fields for e-mail address, cc and bcc address, as well as faxNumber.

## Properties

*cDataSource* – Contains the name of DataSource for the Word Document. It is used from word or wizard to merge given texts. It is expected all fields used in merged documents or texts to be available.

*cMailAddressFieldName* – Contains field name for e-mail address. It is expected this to be a field from *cDataSource* table. It is used for obtaining recipient address when sending emails.

*cCCFieldName* – Contains field name for CC e-mails addresses. It is expected this to be a field from *cDataSource* table. It is used when sending emails for filling cc recipients address.

*cBCCFieldName* – Contains field name for BCC e-mails addresses. It is expected this to be a field from *cDataSource* table. It is used when sending emails for filling bcc recipients address.

*cFaxNumberFieldName* – Contains field name for fax numbers. It is expected this to be a field from *cDataSource* table. It is used to obtain the recipient fax number when sending fax.

*cLeftDelim* – Left delimiter to use for the text merge search. By default it is "<<". This string must be used as left delimiters in text files or texts to be merged.

*cRightDelim* – Right delimiter to use for the text merge search. By default it is ">>". This string must be used as right delimiters in text files or texts to be merged.

*cMergeText* – For internal use only. Used to hold the text that will be merged.

*nEmailsSent* – For internal use only. Used to count sent emails, prepared documents, sent faxes or printed sheets number and to display this value on last wizard step.

*nPreviousPageNum* – For internal use only. Used to keep subsequent number of displayed wizard step.

## Methods

*LoadFileContent* – Loads the content of file that is selected in Step 2 *Source*, in field *Text*, available for edition in Step3 – *Text*.

*SendMails* – Performs the process of generating merge documents, according to selected options. It calls one of the following two methods: *SendThroughMapi* or *SendThroughOleWord*.

*SendThroughMapi* – Merges the text and sends emails using the standard VFX class *cEmail*.

*SendThroughOleWord* – It's used when the merged text is a word document. An automation instance of MSWord is created and its features are used to send merged document by e-mail, to make new word document or to send it to fax or to printer.

## *Reporting*

In VFX 9.5 applications the default setting for report output is *Set Reportbehavior 90*. Thus can be used all new features of VFP 9 report designer and the report engine. With the property *nReportBehavior* of the application object the Reportbehavior can be set to 80 if necessary. This property can be also be set in the VFX – Application Builder.

Now all generated at run time reports, based on Grids can be generated with multiline details band. The property *nMultiLineReport* of the application object controls this behavior. When its value is set to 0, the value of property *lMultiLineReport* at form level will control if multiline reports can be generated. When this value is set to 1, multiline reports forced for all forms of the application. With the value of 2 the behavior is set similar to previous VFX versions. Reports get printed in one line, up to the right margin.

### Modify reports

End users have the chance to edit report files themselves. For this, the file *Vfxmodifyreport.exe* must be in the same folder like the application. In addition, the current user must have right to modify report. This right can be set by administrators users (having user level = 1). It is set individually per user in the User list dialog, as well as per user group.

The start of the application *Vfxmodifyreport.exe* from Windows Explorer is not possible. In this way is prevented an unauthorized use of this application.

The application for the modifying report files is a separate stand-alone application, by security reasons. When started, this file receives as a parameter the currently used language of the application and therefore starts localized. Please, pay attention, that some of the dialogs come from VFP development environment and thus are displayed into language of the VFP environment.

The developers who want to run this application programmatically can pass as a second parameter the name of the report file. When the application is run from the menu *Tools*, an open dialog is invoked, for selecting a report file.

Reports are edited, with the clause *PROTECTED* of the *MODIFY REPORT* command. Thereby all protection features of Reportbehavior 90 can be used.

### ReportOutput and ReportPreview

Source code of *ReportOutput* and *ReportPreview* VFP applications is included into the source code of VFX. Of course, as all parts of VFX framework, it is adapted to use both runtime localizations or design time localization.

ReportOutput application includes all available VFX report listener classes. Additionally a new report listener is added for creation PDF output files.

In the report preview, a context menu is available, on right mouse click. It gives the users the chance to print the report output, save it in a file or send as an email.

An advanced print dialog is available for printing reports. In this dialog can be selected pages to be printed. Depending on the printer driver, it also possible to print multiple report pages on a sheet. When printing more than one copy, the pages sort order can be selected.

## PDF Report Listener

This new ReportListener class was added in addition to the ReportListeners already available in VFX 9.0. With this ReportListener class it is possible to save report output to PDF format. The installation of Ghostscript is no more required.

The files, required by PDF-ReportListener have a substantial size and, hence, are not enclosed in VFX applications. These files are included in the project *PDFOutput.pjx*. The APP file built from this project can be placed by installation in the Exe folder of the application and then it will be used automatically. Alternatively the file *PDFOutput.app* can be automatically downloaded when required in a Zip archive from internet. The download link is hold in the table *Vfxsys.dbf* in *Install_GS* field. Usually this file is downloaded from the Visual Extend web page.

```
D: http://files.visualextend.com/files95/PDFOutput.Zip
```

---

**Tip:** in previous VFX versions in the field *Vfxsys.Install_GS* was written the installation script for GhostScript. If it is desired to use the download option for *PDFOutput.app*, the script download it should be written on this field manually.

---

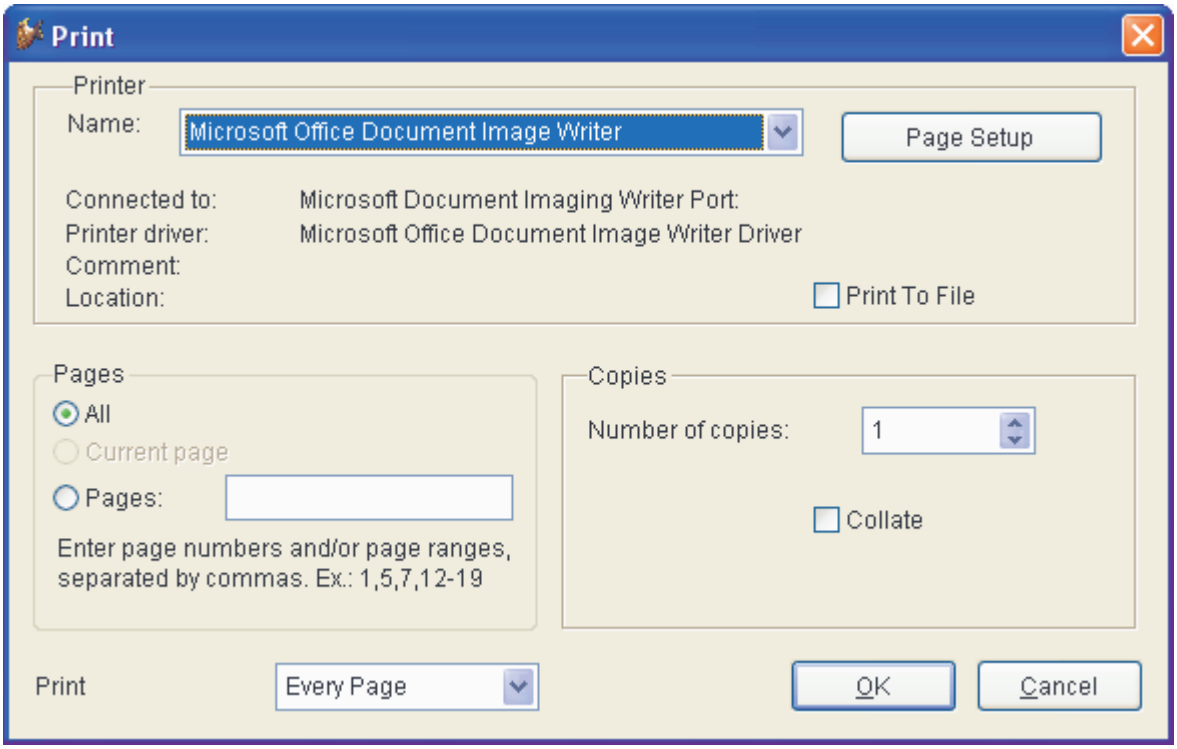
The installation of GhostScript is not required for VFX 9.5 applications.

## Advanced Print Dialog

In VFX 9.5 applications is developed a new enhanced Print Dialog, giving the end users the chance to specify output content more precisely. This functionality is based on the new class *cPrintDialog* and uses also the new class *cPrintEngine*.

## cPrintDialog class

The class *cPrintDialog* is a form class placed in *Vfxform.vcx* class library. It is an advanced print dialog form which gives end users chance to specify output content more precisely.




In this dialog form can be specified many printing properties: Printer to be used, printer page setup, print to file, print range like: selected pages, number of copies, collation etc. As a parameter, the form expects an object of class *cPrintEngine*.

## Properties

*cPageRange* - String containing the selected pages and/or page ranges separated by commas. This property will be considered only when the option *Print selected pages* (*nPagesSelectionType* = 3). Example: 1,3,5-12

*nAllOddEven* - Specifies which pages in selected range to be printed.

1 – All
2 – Odd pages
3 – Even pages

*nCollate* –Collate sequence

0 – Off – Copies will be printed one after another.
1 – On – Every page will be printed according selected number of copies.

*nNumberOfCopies* - Specifies the number of copies that should be printed

*nPagesSelectionType* - Specifies the page selection type

1 – All pages
2 – Current page
3 – Page range

*nPrintToFile* – Integer value, specifying output destination.
0 – Print to printer
1 – Report is printed to a file

*oUnderlyingObject* – Reference to the object of class *cPrintEngine* this form has been invoked by.

*Printers* – Array used for storing properties of the installed printers.

## CPrintEngine class

This class is designed for printing reports. As a parameter it receives a reference to a ReportListener object. This class has its own methods for starting a print job, filling it with pages and closing it. Depending on the property *goProgram.nCustomPrintDialog* a printing dialog is shown.

This dialog allows the user to change Printer, Printer Properties, Number of copies, Selected pages and Collation. This dialog is instantiated from the class *cPrintDialog*.

## Properties

*aPagesToPrint* – Internally used. In this property is kept an array holding the pages selected for printing. The first element of each row contains the start page of a range and the second – the end page. When a single page is selected, the value is written in both elements. If all pages are selected there is only one row in the array.

*cPageRange* – String containing the selected pages and/or page ranges separated by commas. This property will be considered only when the option *Print selected pages* (*nPagesSelectionType* = 3).

*cPrinterDriver* – Name of the printer driver of the currently selected printer.

*cPrinterName* – Name of the printer currently selected.

*cPrintFileName* – Name of the file where output should be directed to (If PrintToFile selected)

*lCustomPrintDialog* –Specifies if a custom print dialog is shown.
.T. – Custom print dialog is displayed.
.F. – Standard print dialog is displayed.

*nAOE* – Specifies which pages in selected range to be printed.
1 – All
2 – Odd pages
3 – Even pages

*nHeight* – Used internally. Height of the printable area of the currently selected printer

*nLeft* – Used internally. Left of the printable area of the currently selected printer

*nPagesSelectionType* – Specifies the page range selection type
1 – All pages
2 – Current page
3 – Page range

*nPrinterHandle* – Holds handle to the printer that will be used for printing

*nPrintToFile* – Integer value, specifying output destination.
1 – Report is printed to a file
0 – Print to printer

*nShowDialog* – Used internally to indicate if the print dialog has been shown or not

*nTop* – Used internally. Top of the printable area of the printer currently set

*nWidth* – Used internally. Width of the printable area of the printer currently set

*oRepListener* – Reference to the calling ReportListener object.

## Methods

*EndPrinterJob()*
Finalizes the print job. Returns .T. on successful finalizing. The method will return .F. if job finalizing not successful.

*GetPageProp()*
Retrieves the printing area properties Width, Height, Top and Left of the currently selected printer and stores retrieved values in *nWidth, nHeight, nTop, nLeft* properties. Returns .T. on success and .F. if not successful.

*GetPagesToPrint()*
Fills the array *aPagesToPrint* according to *nPagesSelectionType* and *cPageRange*. If an invalid range is found, it is discarded and a message is displayed. Returns .T. if execution is successful. If there has been raised an error, the value .F. will be returned.

*OpenPrintDialog()*
Opens the custom print dialog. If user pressed OK button calls *GetPagesToPrint()* and *StartPrinterJob()*
The method returns .T. if execution has been successful A value .F. is returned, if user pressed Cancel or there was an error.

*Output_Page(nPageNo)*
*nPageNo* – Number of page to be send to the printing device
The method uses *OutputPage*() method of the ReportListener object to send a page to the printing device (printer or file).

*Print_Pages()*
Reads the array *aPagesToPrint* and calls consecutively *Output_Page(nPageNo)* for every page. Finally, the method calls *EndPrinterJob()*. Returned value is .T. if execution has been successful. If an error was raised, the value .F. is returned.

*StartPrinterJob()*
Makes the printer selected in the custom print dialog, current printer for VFP and according *nPrintToFile* setting, starts a print job to the printer or to a file. Returns a handle to the print job.

## XP-Style open dialog

A new functionality is added to XP-style open dialog form. Now you can have not only forms listed in it, but also the link from the dialog can run a code. This is accomplished by leaving the *Form* field in the *Vfxfopen* system table empty and placing the code that should be executed in the *Parameters* field in that row. When a user clicks that item in the XP style open dialog, instead of opening a form the code will be executed.

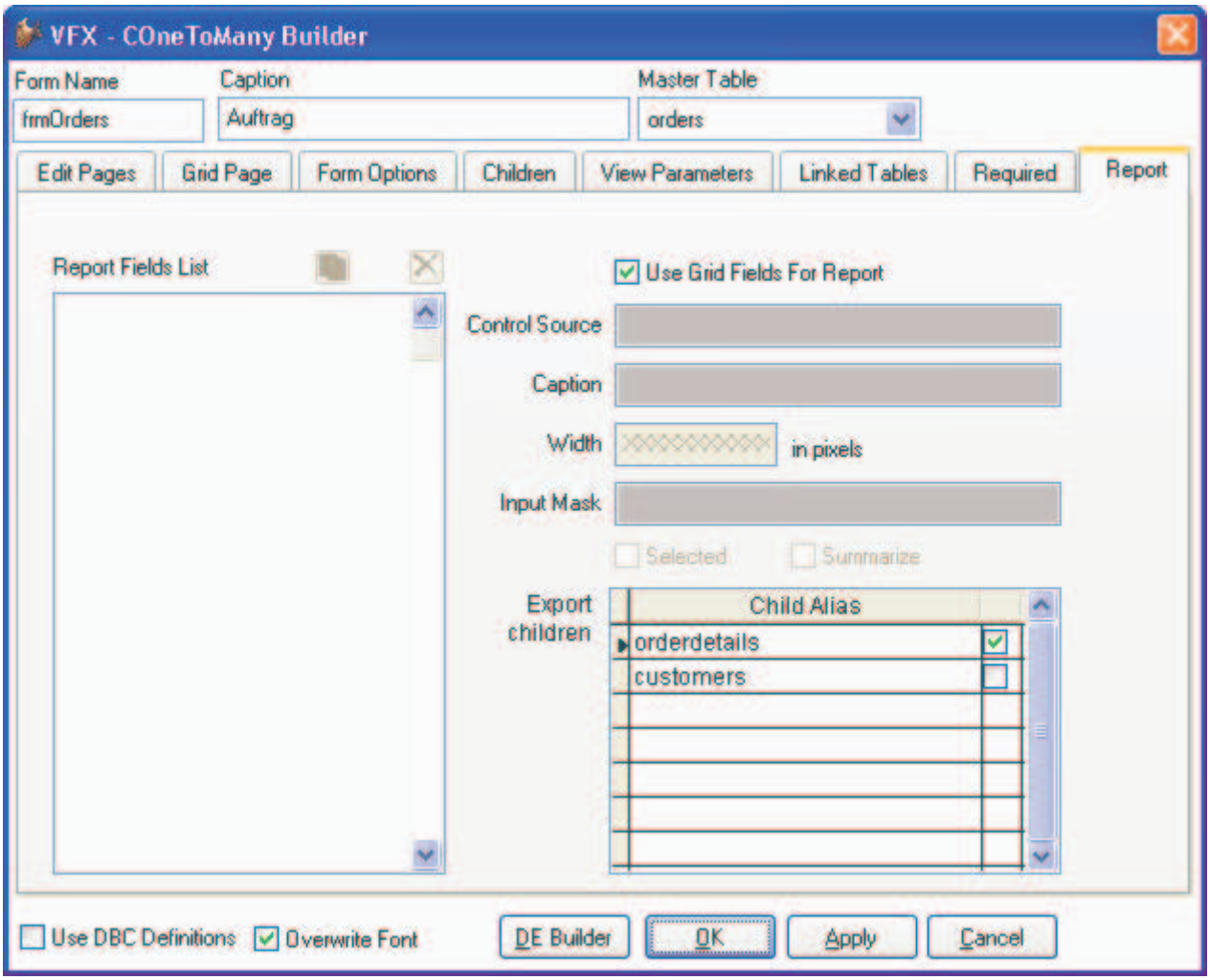
There is a new field *Iconfile* in the table *Vfxfopen*. In this field can be filled the name of an icon file. This icon will be displayed in the XP-style open dialog.

Favorites are listed in XP-style open dialog in a separate group per form. Every user can set if favorites should be added to the XP-style open dialog by making changes in his customization settings.

## Data export

In the menu of the end-user application, under File menu pad is a menu entry *Export as*. Under are available for selection the supported formats *CSV, Excel, XML* and *DBF*. These menu entries are activated if a data form is opened and is active. The selection of one of these options opens a *Save As* dialog. After selecting a file name, the data from the *Initialselectedalias* of the form are saved in a file with the selected file format. The active sort order as well as an eventually filtering is respected while exporting data. All fields are exported.

Exporting data to XML format, allows including child data into the generated XML file. This is controlled by the property *lExportWithChildren* of *cExport* class. To use this feature, parent and child aliases should be joined with a relation. All child aliases which participate in a relation to parent alias can be exported in the resultant XML file. Which child aliases to include in export result, can be specified in Report page in one-to-may builders.



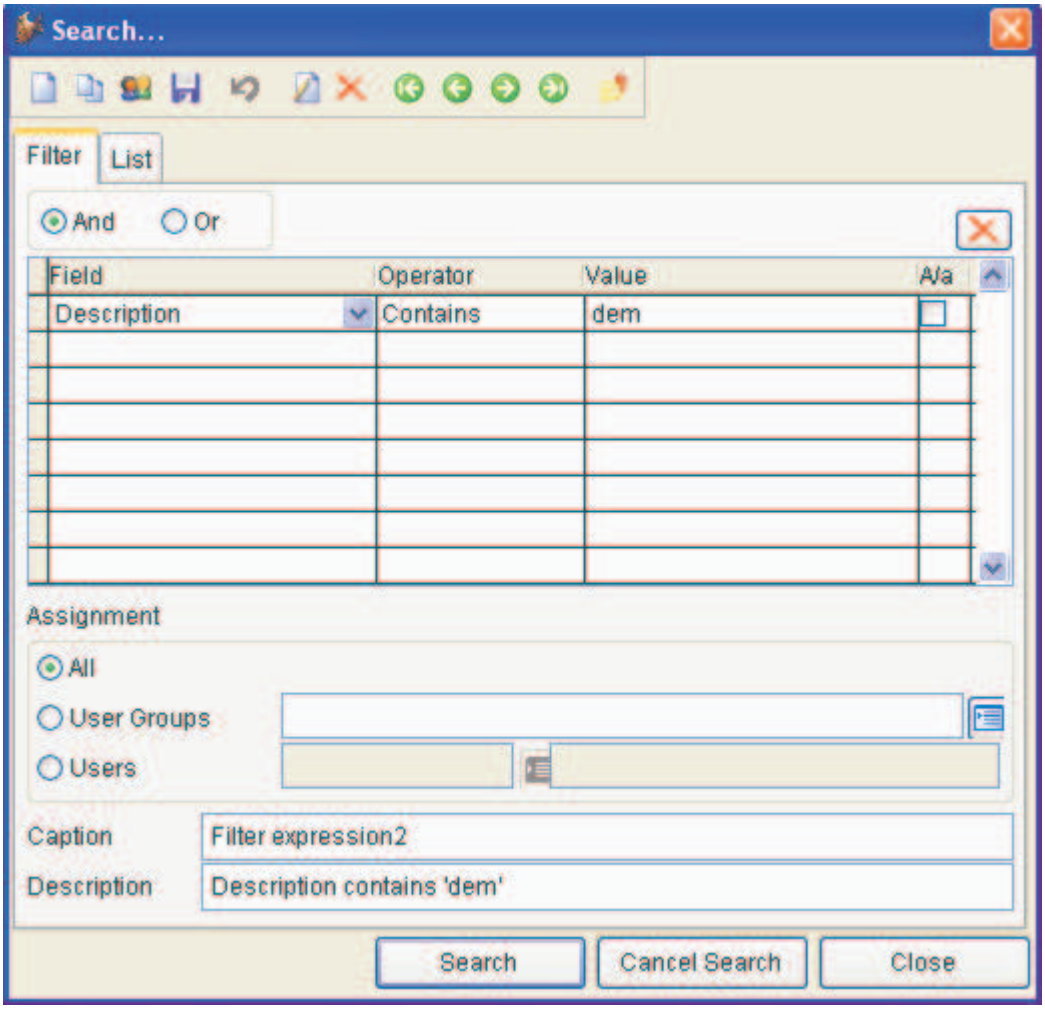

When you have defined child aliases to be included in XML export, user is asked if he/she wants to export children data.

## Search dialog

Filter definition are now able to be saved/loaded per form and per user/user group. The filter definition form is extended with the feature to manage filter definitions and to copy existing definition to other users.

The value of property *nFilterBehavior* in *cFoxAppl* class controls filter behavior which can be set to VFX90 or VFX95. It is also possible to specify different filter behavior for each form. For this, the property *nFilterBehaviour* in *cFoxAppl* should be set to 0 and the property *nFilterBehaviour* in the form should be set to the desired value. When *cFoxAppl.nFilterBehavior* is set to 1, all forms in the application are forced to use VFX 9.0 compatible filter behavior. When the value of the property is set to 2, the new VFX 9.5 filter behavior will be activated for all forms.

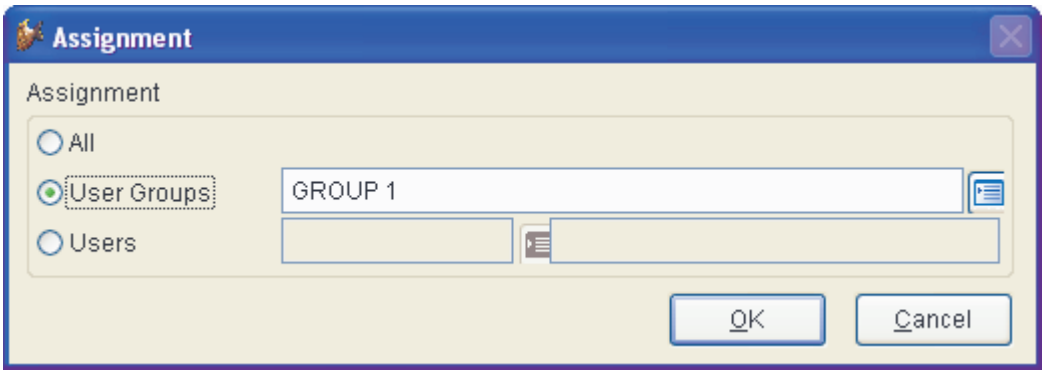
The process of setting the filter expression is the same as for VFX 9.0 behavior. Additionally every filter definition can be saved and for it can be defined rights to be used by other users or by a specific user group.




The search criteria can be assigned per user, user group or for all users. In addition the user can add a Caption and Description to each filter. It is not possible for a particular form to have more than one filter with the same name. The filter definitions are saved per form and are available to be used on later application executions.

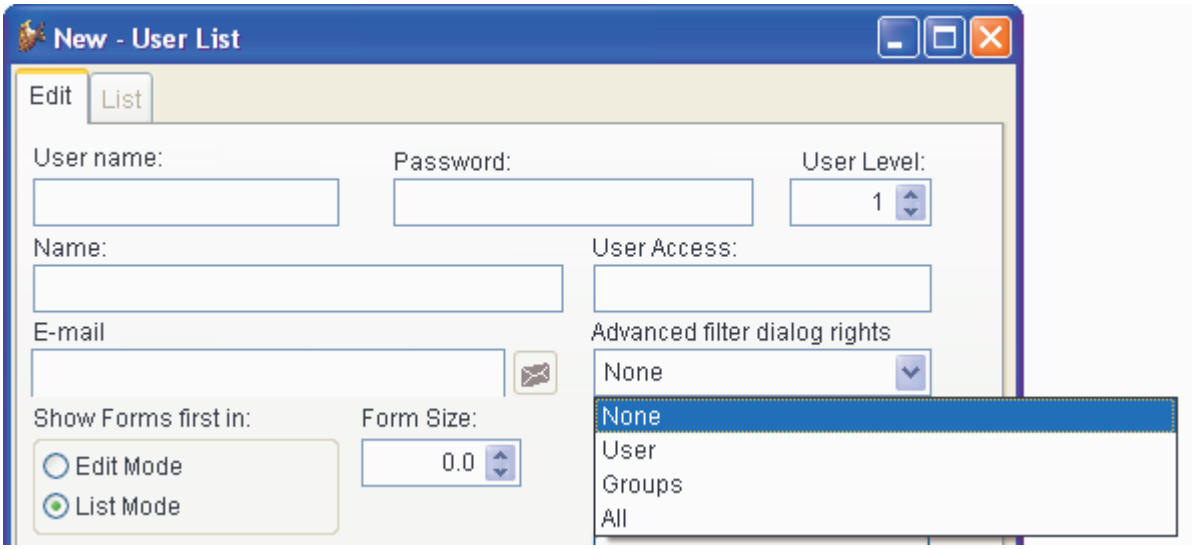
The user can use and see only search criteria which are assigned to all users, to user groups in which the user is member or to himself. For each form is checked if user is allowed to edit filter criteria. This depends on level defined in *vfxfopen.FiltrLevel* field or on the rights assigned to user groups, which this user is member of.

Users can create, copy, copy to other users, modify and delete filter criteria. When the user wants to copy current filter to other users the “Filter User Selection” dialog comes to define assignment for the new filter.




When the selected filter results in empty result set, a message will be displayed to the user and the dialog will remain open. The button *Cancel search* will delete all records of current filter definition and the dialog will remain open. A single filter definition line can be deleted with a button on the form toolbar.

Rights for end-users to define and edit filter conditions in Advanced filter dialog are controlled on user and users group level.



For every user and for every user groups can be defined Advanced search dialog right level:

*None* – User cannot save filter conditions. However, the user can use filter conditions saved by other users. This is the devault value.

*User* – User can save filter conditions only for himself. Copy for all users or for a user group is disabled.

*Groups* – User can save filter conditions for himself and for user groups.

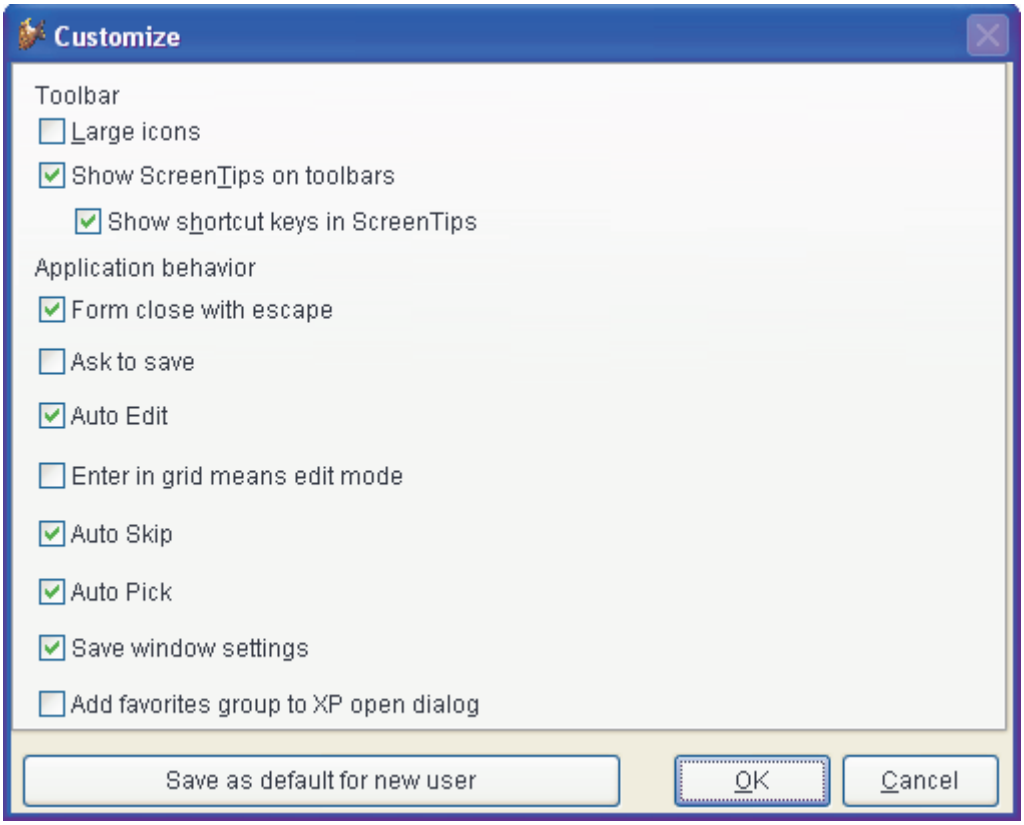
*All* – User can save filter conditions for everybody. (Current Behavior).

When the user is member of more than one user group, the highest value of this right is used.

## Customize dialog

Many properties of the application can be customized individually by the users themselves. If these settings can be changed by user, can be generally controlled by the property *lAllowUserCustomization* of the application object. If the value of this property is set to .F., the customization dialog is not accessible in the application. When this property is set to .T., Administrator users can further allow or prohibit every particular user to make his own layout settings.

User customization dialog is accessible from main menu *Tools/Customize…*




User customization is accessible only for users which have the right "*Allow Users Customization*". This right can be set only by administrator users if *goProgram.lAllowUserCustomization* is set to .T..

User customization allows users to set themselves certain properties to toolbar and application behavior.

The *Save as default for new user* button is visible only when current user is an administrator user. It saves current settings as default settings for newly created users. When a new user is logged first time, these values are used as default settings for his application environment.

The *OK* button saves and applies settings made for current user. The *Cancel* button discards all changes and closes the window.

## Carchive class

A new property *lQuietMode* is added to the class to suppress all messages displayed to the users while creating archive or extracting from archive.

*lQuietMode* – Specifies if archiving should be done in quiet mode, i.e. no on-screen messages. This property should be set manually before calling *CreateArchive()* or *ExtractFromArchive()* methods.

.T. – Quiet mode enabled (no messages or progress bar displayed)

.F. – Quiet mode off

## The new class cTextSkype

The new class *cTextSkype* is placed in the class library *Vfxctrl.vcx*. This class stands of a container with a text box and a button.


If, at run time, end-user clicks on the button, the value contained in the text box will passed as a Skype name to the application Skype. With Skype it is possibly to make telephone calls over the internet and to send messages. You can find more information about Skype on the site http://www.skype.com.

## Run-time errors handling

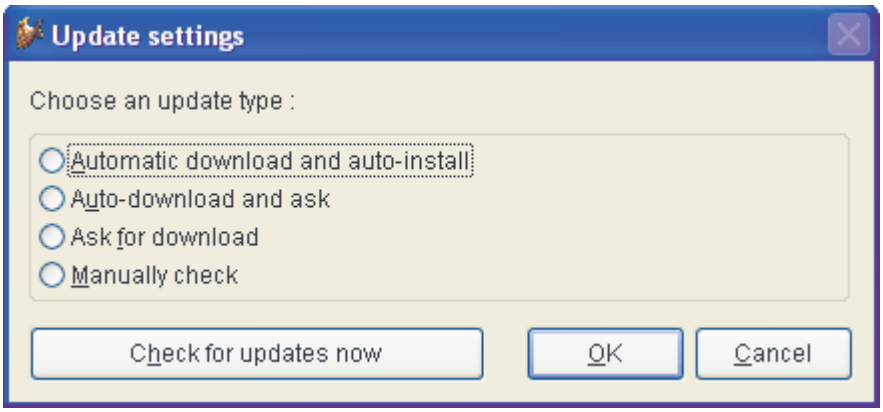
When an error occurs, *OnError* function checks if the connection to remote datasource is available. The error information is gathered in *Vfxlog* table on the server if available and possible. If there is no connection to server database, the error information is stored locally in *Vfxlog.dbf*. The *Vfxlog.dbf* table is created according Windows compatibility design guidelines.

If a VFP database is used, the error information is always stored in the local table *Vfxlog.dbf*.

When an error while saving a record occurs, the name of the work alias in which the error has been raise is saved additionally in the error log. This helps developers further to localize the problem.

## Application update

The application developed with VFX can be updated over the internet. The actualization functionality is activated if the property *lAllowUpdates* of the application object is set to *.T.* In this case the menu entries *Application update* and *Update settings* are available.




In *Update settings* dialog users can select among four different options. If one of automatic update types is selected, daily, on every first start of the application, is performed a check if new update version is available.

The update functionality in encapsulated in classes *cUpdate* and *cUpdateEngine*.

### Client database update

Database update at first start of new application version runs for all types of data sources. It is possible to run database update either in former way, providing new empty database and the metadata table in *Update* folder.

To use the new VFX 9.5 application update functionality, the Exe file must have a version number specified. You can set the VFP project automatically to increase the version number with every build of the Exe file. With every build of Exe, the Metadata table is now automatically filled by PJHook for all data sources stored in *config.vfx*. The metadata table is included in the project and thus is distributed to end-user side.

The version of the running application is now kept in *AppVersion* field in *VFXSys* table. If there are no files in update folder, when the Exe is started, its version is compared with the last number, kept in *VFXSys* table. When the Exe version is different than the number kept in *VFXSys* table, the new database update algorithm is automatically started. The client database update is performed for every database keyed up in *Config.vfx* or in *Vfxpath.dbf*. Both VFX databases and remote databases are updated. Finally, the new version is saved into *AppVersion* field in *VFXSys* table, to be checked at next start of the application

## Database repair

The database repair functionality of VFX 9.0 was significantly improved. To be able to repair a database, the structure of the correct database must be gathered. For this purpose is used the Project Hook which is set as project class for all VFX projects. On every application build is called the program *Gendbc.prg*. *Gendbc.prg* is deployed with VFP. This program reads the database container and, generates a program file describing its structure. Later, while application executes, this program file is used to create a new, empty database with the same structure. This database structure can be used for the repair of a damaged database.

The program file, generated by *Gendbc.prg* is automatically included in the project and therefore is available for execution in an Exe file. If a database contains stored procedures, they are copied by *Gendbc.prg* in a memo field of a table. This generated table is also automatically included in the project.

In most cases, a crashed DBC can be recognized immediately by the start of the Exe file and then is repaired without questions to the user.

For repairing of all other damages, repair must be started manually from the menu *Tools*, *Database*...

These are problems that can be repaired in *Database Tools* dialog:

- Crashed table headers, in particular also crashed record counters, are recovered, in any case, without data loss.
- Crashed records, for example, found duplicated primary keys, can be certainly recognized and the double records are automatically deleted. Afterwards the table can be opened and used again. It is sure that only crashed records will be removed.
- Missing DBF files can be restored. Of course, in such case the data of the concerning table is lost.
- Missing CDX files are recovered, without data loss in any case.
- Damaged CDX files can be recognized in many cases and be recovered without data loss.
- Crashed memo fields can be repaired in almost all cases. Besides, the content of memo fields of the concerning record is lost.
- Missing memo files can be restored. Of course, all memo fields of the concerning table get lost.

Tables can be repaired, while the DBC is opened by other users. Exclusive access is required only for tables being repaired. In case when the option *Entire database* or the check *Repair DBC container* is selected, is needed to have exclusive access to the database.

## *Terminal server usage*

When the application runs on a terminal server workstation, its performance is automatically optimized by turning the off-screen bitmap off with the VFP command *SYS (602*).

For further optimization of the toolbar visualization in the applications which should run on terminal server workstation, we recommend in any cases, the *Visible* property values of controls to be assigned only if this is necessary. Thus will be avoided needless executed *Refresh* events and the tooltip is displayed without blinking.

Example code for the *REFRESH()* event in *cAppToolBar* or *cAppNavBar* classes:

```
* Not recommended:
* DODEFAULT()
* This.cmdNew.Visible = .F.

* Recommended code:
DODEFAULT()
IF This.cmdNew.Visible
   This.cmdNew.Visible = .F.
ENDIF
```

## *Other end-user enhancements*

Now all user-specific interface settings, like form size, position on the screen and Grid settings can be stored and load according used screen resolution. This feature is controlled by the property *lSaveFormLayoutResolutionDependent* of the class *cFoxAppl* in the class library *Appl.vcx* Its default value is .F. The feature is enabled by setting this property to .T. This setting can also be made in the VFX – Application Builder.

The backup function from the menu constructs archive file name including the folder name, the database name as well as the current date in the ANSI format.

Navigation in TreeView forms move to the next, previous, first and last logical node in the TreeView structure. When current record does not have children or next sibling nodes, when user wants to move to next record, record pointer is moved to the next sibling node for its parent.

In VFX 9.0 the data from all grids could be pulled to other applications by Drag & Drop. In VFX 9.5 this behavior can be controlled globally and per grid. If the property *nOLEDragGrid* of the class *cFoxAppl* from the class library *Appl.vcx* is set to *1*, the data from all Grids of the application can be dragged by Drag & Drop to other applications. This was the standard behavior in VFX 9.0. If the value of this property is set to *0*, this behavior can be keyed up separately for every grid with its property *lOLEDragGrid*. The value *1* is the default value. If the value of the property *nOLEDragGrid* is set to *2*, OLE Drag & Drop is disabled for all grids of the application.

Yet in VFX 9.0, the property *lKeepLoggedUsers* of the application object has controlled if users login will be tracked. This property is available to be set in VFX – Application Builder. For application is has been possible to block users from login more than once at a time. In VFX9.5 end-user administrator can allow certain user to be able to log in the system more than once put a mark on the checkbox Allow multiple logins for this user in Tools/User List dialog.

The menu *Help* in applications is extended. Selecting the menu entry *Visits our website*, the user can visit the website whose URL is stored in the property *cCompanyWebSiteURL* of the application object. Under the menu entry *Contact us*, to the users can be displayed contact information. As contact information serves an HTML file, which has to be stored in *Vfxinternfiles. dbf* table in a record with *type = "contactus"*.

# Appendix II - Transact SQL

*by Igor Nikiforov*

The following user Defined Transact SQL character sequence functions was kindly made available by Igor Nikiforov and is supplied with VFX.

## *AT(), ATC()*

Returns the beginning numeric position of the first occurrence of a character expression within another character expression, counting from the leftmost character (including overlaps).

### Syntax

AT(@cSearchExpression, @cExpressionSearched [, @nOccurrence])
ATC(@cSearchExpression, @cExpressionSearched [, @nOccurrence])

### Parameters

@cSearchExpression nvarchar(4000) Specifies the character expression that AT( ) searches for in @cExpressionSearched.

@cExpressionSearched nvarchar(4000) Specifies the character expression @cSearchExpression searches for.

@nOccurrence smallint Specifies which occurrence (first, second, third, and so on) of @cSearchExpression is searched for in @cExpressionSearched. By default, AT() searches for the first occurrence of @cSearchExpression (@nOccurrence = 1). Including @nOccurrence lets you search for additional occurrences of @cSearchExpression in @cExpressionSearched. AT( ) returns 0 if @nOccurrence is greater than the number of times @cSearchExpression occurs in @cExpressionSearched.

### Return Values

Smallint

### Remarks

AT(),ATC() searches the second character expression for the first occurrence of the first character expression. It then returns an integer indicating the position of the first character in the character expression found. If the character expression isn't found, AT(),ATC() returns 0. The search performed by AT() is case-sensitive. The search performed by AT() is case-insensitive.

### Example

```
declare @gcString nvarchar(4000), @gcFindString nvarchar(4000)
select @gcString = 'Now is the time for all good men', @gcFindString = 'is the'
select dbo.AT(@gcFindString, @gcString, default) -- Displays 5
set @gcFindString = 'IS'
select dbo.AT(@gcFindString, @gcString, default) -- Displays 0, case-sensitive

select @gcString = 'Now is the time for all good men', @gcFindString = 'is the'
select dbo.ATC(@gcFindString, @gcString, default) -- Displays 5
set @gcFindString = 'IS'
select dbo.ATC(@gcFindString, @gcString, default) -- Displays 2, case-insensitive
```

## *RAT(), RATC()*

Returns the numeric position of the last (rightmost) occurrence of a character string within another character string (including overlaps).

### Syntax

RAT(@cSearchExpression, @cExpressionSearched [, @nOccurrence])
RATC(@cSearchExpression, @cExpressionSearched [, @nOccurrence])

### Parameters

@cSearchExpression nvarchar(4000) Specifies the character expression that RAT( ) looks for in @cExpressionSearched.
@cExpressionSearched nvarchar(4000) Specifies the character expression that RAT() searches.
@nOccurrence smallint Specifies which occurrence, starting from the right and moving left, of @cSearchExpression RAT() searches for in @cExpressionSearched. By default, RAT() searches for the last occurrence of @cSearchExpression (@nOccurrence = 1). If @nOccurrence is 2, RAT() searches for the next to last occurrence, and so on.

### Remarks

RAT(),RATC(), the reverse of the AT(),ATC() function, searches the character expression in @cExpressionSearched starting from the right and moving left, looking for the last occurrence of the string specified in @cSearchExpression.

RAT(),RATC() returns an integer indicating the position of the first character in @cSearchExpression in @cExpressionSearched. RAT(),RATC() returns 0 if @cSearchExpression isn't found in @cExpressionSearched, or if @nOccurrence is greater than the number of times @cSearchExpression occurs in @cExpressionSearched.
The search performed by RAT() is case-sensitive.
The search performed by RATC() is case-insensitive.

See Also AT(), ATC().

### Return

Values smallint

### Example

```
declare @gcString nvarchar(4000), @gcFindString nvarchar(4000)
select @gcString = 'abracadabra', @gcFindString = 'a'
select dbo.RAT(@gcFindString , @gcString, default) -- Displays 11
select dbo.RAT(@gcFindString , @gcString , 3) -- Displays 6
```

## *OCCURS(), OCCURS2()*

Returns the number of times a character expression occurs within another character expression (including overlaps).

### Syntax

OCCURS(@cSearchExpression, @cExpressionSearched)
OCCURS2(@cSearchExpression, @cExpressionSearched)

## Return

Values smallint

## Parameters

@cSearchExpression nvarchar(4000) Specifies a character expression that OCCURS() searches for within @cExpressionSearched.

@cExpressionSearched nvarchar(4000) Specifies the character expression OCCURS() searches for @cSearchExpression.

## Remarks

OCCURS() returns 0 (zero) if @cSearchExpression is not found within @cExpressionSearched.

```
select dbo.OCCURS('ABCA', 'ABCABCABCA') -- Displays 3
1 occurrence of substring 'ABCA .. BCABCA'
2 occurrence of substring 'ABC...ABCA...BCA'
3 occurrence of substring 'ABCABC...ABCA'

select dbo.OCCURS2('ABCA', 'ABCABCABCA') -- Displays 2
1 occurrence of substring 'ABCA .. BCABCA'
2 occurrence of substring 'ABCABC... ABCA'
```

## Example 1

```
declare @gcString nvarchar(4000)
select @gcString = 'abracadabra'
select dbo.OCCURS('a', @gcString ) -- Displays 5
select dbo.OCCURS('b', @gcString ) -- Displays 2
```

## Example 2

Counts the occurrences of differents characters from a string @gcCaracters in string @gcString

```
declare @gcString nvarchar(4000), @gcCaracters nvarchar(256), @i smallint, @counter
smallint
select @i = 1, @counter = 0
select @gcString = N'For the most part, the remaining four hunters leaned on the table or
lay in their bunks and left the discussion to the two antagonists. But they were supremely
interested, for every little while they ardently took sides, and sometimes all were talking at
once, till their voices surged back and forth in waves of sound like mimic thunder-rolls in the
confined space. Childish and immaterial as the topic was, the quality of their reasoning was
still more childish and immaterial. In truth, there was very little reasoning or none at all.
Their method was one of assertion, assumption, and denunciation. They proved that a seal
pup could swim or not swim at birth by stating the proposition very bellicosely and then
following it up with an attack on the opposing man''s judgment, common sense, nationality,
or past history. Rebuttal was precisely similar. I have related this in order to show the
mental calibre of the men with whom I was thrown in contact. Intellectually they were
children, inhabiting the physical forms of men.', @gcCaracters = N'abcca'
while @i <= datalength(@gcCaracters)/2
   begin
      if charindex(substring(@gcCaracters,@i,1), left(@gcCaracters, @i - 1)) = 0
         select @counter = @counter + dbo.OCCURS2(substring(@gcCaracters,@i,1),
@gcString)
      select @i = @i + 1
```

```
    end
select @counter -- Displays 92
```

## *PADL(), PADR(), PADC()*

Returns a string from an expression, padded with spaces or characters to a specified length on the left or right sides, or both.

### Syntax

PADL(@eExpression, @nResultSize [, @cPadCharacter]) -Or-
PADR(@eExpression, @nResultSize [, @cPadCharacter]) -Or-
PADC(@eExpression, @nResultSize [, @cPadCharacter])

### Parameters

@eExpression nvarchar(4000) Specifies the expression to be padded.
@nResultSize smallint Specifies the total number of characters in the expression after it is padded.
@cPadCharacter nvarchar(4000) Specifies the value to use for padding. This value is repeated as necessary to pad the expression to the specified number of characters. If you omit @cPadCharacter, spaces (ASCII character 32) are used for padding.

### Return Values

nvarchar(4000)

### Remarks

PADL() inserts padding on the left, PADR() inserts padding on the right, and PADC() inserts padding on both sides.

### Example

```
declare @gcString nvarchar(4000)
select @gcString = 'TITLE'
select dbo.PADL(@gcString, 40, default) -- Displays ' TITLE'
select dbo.PADL(@gcString, 40, '+=+') -- Displays
'+=++=++=++=++=++=++=++=++=++=++=TITLE'
select dbo.PADR(@gcString, 40, '=!!=') -- Displays 'TITLE=!!!==!!!==!!!==!!!==!!!==!!!='
select dbo.PADC(@gcString, 40, '=*=') -- Displays
'=*==*==*==*==*==*TITLE=*==*==*==*==*==*='
```

## *CHRTRAN()*

Replaces each character in a character expression that matches a character in a second character expression with the corresponding character in a third character expression.

### Syntax

CHRTRAN (@cExpressionSearched, @cSearchExpression, @cReplacementExpression)

### Parameters

@cSearchedExpression Specifies the expression in which CHRTRAN() replaces characters.
@cSearchExpression Specifies the expression containing the characters CHRTRAN() looks for in @cSearchedExpression.
@cReplacementExpression Specifies the expression containing the replacement characters.

### Return Values

nvarchar (4000)

### Remarks

If a character in cSearchExpression is found in cSearchedExpression, the character in @cSearchedExpression is replaced by a character from @cReplacementExpression that is in the same position in @cReplacementExpression as the respective character in @cSearchExpression. If @cReplacementExpression has fewer characters than @cSearchExpression, the additional characters in @cSearchExpression are deleted from @cSearchedExpression. If @cReplacementExpression has more characters than @cSearchExpression, the additional characters in @cReplacementExpression are ignored.

CHRTRAN() translates the character expression @cSearchedExpression using the translation expressions @cSearchExpression and @cReplacementExpression and returns the resulting character string.
CHRTRAN similar to the Oracle function TRANSLATE.

See Also STRFILTER()

### Example

select dbo.CHRTRAN('ABCDEF', 'ACE', 'XYZ') -- Displays 'XBYDZF'
select dbo.CHRTRAN('ABCDEF', 'ACE', 'XYZQRST') -- Displays 'XBYDZF'

## *STRTRAN()*

Searches a character expression for occurrences of a second character expression, and then replaces each occurrence with a third character expression.

### Syntax

STRTRAN (@cSearched, @cExpressionSought , [@cReplacement]
[, @nStartOccurrence] [, @nNumberOfOccurrences] [, @nFlags])

### Parameters

@cSearched Specifies the character expression that is searched.
@cExpressionSought Specifies the character expression that is searched for in @cSearched.
@cReplacement Specifies the character expression that replaces every occurrence of @cExpressionSought in @cSearched.
If you omit @cReplacement, every occurrence of @cExpressionSought is replaced with the empty string.
@nStartOccurrence Specifies which occurrence of @cExpressionSought is the first to be replaced.
For example, if @nStartOccurrence is 4, replacement begins with the fourth occurrence of @cExpressionSought in @cSearched and the first three occurrences of @cExpressionSought remain unchanged.
The occurrence where replacement begins defaults to the first occurrence of @cExpressionSought if you omit @nStartOccurrence.
@nNumberOfOccurrences Specifies the number of occurrences of @cExpressionSought to replace.
If you omit @nNumberOfOccurrences, all occurrences of @cExpressionSought, starting with the occurrence specified with @nStartOccurrence, are replaced.
@nFlags Specifies the case-sensitivity of a search according to the following values:
0 (default) Search is case-sensitive, replace is with exact @cReplacement string.
1 Search is case-insensitive, replace is with exact @cReplacement string.

2 Search is case-sensitive; replace is with the case of @cReplacement changed to match the case of the string found. The case of @cReplacement will only be changed if the string found is all uppercase, lowercase, or proper case.
3 Search is case-insensitive; replace is with the case of @cReplacement changed to match the case of the string found. The case of @cReplacement will only be changed if the string found is all uppercase, lowercase, or proper case.

### Return Values

nvarchar

### Remarks

You can specify where the replacement begins and how many replacements are made.
STRTRAN( ) returns the resulting character string.
Specify –1 for optional parameters you want to skip over if you just need to specify the @nFlags setting.

See Also replace(), CHRTRAN()

### Example

```
select dbo.STRTRAN('ABCDEF', 'ABC', 'XYZ',-1,-1,0) -- Displays XYZDEF
select dbo.STRTRAN('ABCDEF', 'ABC', default,-1,-1,0) -- Displays DEF
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', default,2,-1,0) -- Displays
ABCDEFGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', default,2,-1,1) -- Displays
ABCDEFGHJQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 1, 1) -- Displays
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 3, 1) -- Displays
ABCDEFXYZGHJXYZQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 1, 2) -- Displays
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 3, 2) -- Displays
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'xyZ', 2, 1, 2) -- Displays
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'xYz', 2, 3, 2) -- Displays
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFAbcCGHJAbcQWE', 'Aab', 'xyZ', 2, 1, 2) -- Displays
ABCDEFAbcCGHJAbcQWE
select dbo.STRTRAN('abcDEFabcGHJabcQWE', 'abc', 'xYz', 2, 3, 2) -- Displays
abcDEFxyzGHJxyzQWE
select dbo.STRTRAN('ABCDEFAbcCGHJAbcQWE', 'Aab', 'xyZ', 2, 1, 3) -- Displays
ABCDEFAbcCGHJAbcQWE
select dbo.STRTRAN('ABCDEFAbcGHJabcQWE', 'abc', 'xYz', 1, 3, 3) -- Displays
XYZDEFXyzGHJxyzQWE
```

## *STRFILTER()*

Removes all characters from a string except those specified.

### Syntax

STRFILTER(@cExpressionSearched, @cSearchExpression)

### Parameters

@cExpressionSearched Specifies the character string to search.
@cSearchExpression Specifies the characters to search for and retain in @cExpressionSearched.

### Return Values

nvarchar

### Remarks

STRFILTER( ) removes all the characters from @cExpressionSearched that are not in @cSearchExpression, then returns the characters that remain.

See Also CHRTRAN()

### Example

```
select dbo.STRFILTER('asdfghh5hh1jk6f3b7mn8m3m0m6','0123456789') -- Displays 516378306
select dbo.STRFILTER('ABCDABCDABCD', 'AB') -- Displays ABABAB
```

## *GETWORDCOUNT()*

### Syntax

GETWORDCOUNT(@cString[, @cDelimiters])

### Parameters

@cString nvarchar(4000) - Specifies the string whose words will be counted.
@cDelimiters nvarchar(256) - Optional. Specifies one or more optional characters used to separate words in @cString. The default delimiters are space, tab, carriage return, and line feed. Note that GETWORDCOUNT( ) uses each of the characters in @cDelimiters as individual delimiters, not the entire string as a single delimiter.

### Return Value

smallint

### Remarks

GETWORDCOUNT() by default assumes that words are delimited by spaces or tabs. If you specify another character as delimiter, this function ignores spaces and tabs and uses only the specified character.

See Also GETWORDNUM(), GETALLWRDS()

### Example

If you use 'To be, or not to be: that is the question:' as the target string for dbo.GETWORDCOUNT(), you can get all the following results.

```
declare @cString nvarchar(4000)
set @cString = 'To be, or not to be: that is the question:'
select dbo.GETWORDCOUNT(@cString, default) -- Displays 10 - character groups, delimited by ' '
select dbo.GETWORDCOUNT(@cString, ',') -- Displays 2 - character groups, delimited by ','
```

## *GETWORDNUM()*

Returns a specified word from a string.

### Syntax

GETWORDNUM(@cString, @nIndex[, @cDelimiters])

### Parameters

@cString nvarchar(4000) - Specifies the string to be evaluated

@nIndex smallint - Specifies the index position of the word to be returned. For example, if @nIndex is 3, GETWORDNUM( ) returns the third word (if @cString contains three or more words).

@cDelimiters nvarchar(256) - Optional. Specifies one or more optional characters used to separate words in @cString. The default delimiters are space, tab, carriage return, and line feed. Note that GetWordNum( ) uses each of the characters in @cDelimiters as individual delimiters, not the entire string as a single delimiter.

### Return Value

nvarchar(4000)

### Remarks

Returns the word at the position specified by @nIndex in the target string, @cString. If @cString contains fewer than @nIndex words, GETWORDNUM( ) returns an empty string.

See Also GETWORDCOUNT(), GETALLWORDS()

### Example

```
declare @cString nvarchar(4000)
set @cString = 'Whether "tis nobler in the mind to suffer'
select dbo.GETWORDNUM(@cString, 3, default) -- Displays 'nobler'
```

## *GETALLWORDS()*

Inserts the words from a string into the table.

### Syntax

GETALLWORDS(@cString[, @cDelimiters])

### Parameters

@cString nvarchar(4000) - Specifies the string whose words will be inserted into the table @GETALLWORDS.

@cDelimiters nvarchar(256) - Optional. Specifies one or more optional characters used to separate words in @cString.

The default delimiters are space, tab, carriage return, and line feed. Note that GETALLWORDS( ) uses each of the characters in @cDelimiters as individual delimiters, not the entire string as a single delimiter.

### Return Value

table @GETALLWORDS (WORDNUM smallint, WORD nvarchar(4000), STARTOFWORD smallint, LENGTHOFWORD smallint)

### Remarks

GETALLWORDS() by default assumes that words are delimited by spaces or tabs. If you specify another character as delimiter, this function ignores spaces and tabs and uses only the specified character.

See Also GETWORDNUM(), GETWORDCOUNT

### Example

```
declare @cString nvarchar(4000)
set @cString = 'The default delimiters are space, tab, carriage return, and line feed. If you specify another character as delimiter, this function ignores spaces and tabs and uses only the specified character.'
select * from dbo.GETALLWORDS(@cString, default)
select * from dbo.GETALLWORDS(@cString, ' ,.')
```

## *PROPER()*

Returns from a character expression a string capitalized as appropriate for proper names.

### Syntax

PROPER(@cExpression)

### Parameters

@cExpression nvarchar(4000) Specifies the character expression from which PROPER( ) returns a capitalized character string.

### Return Values

nvarchar(4000)

### Example

```
declare @gcExpr1 nvarchar(4000), @gcExpr2 nvarchar(4000)
select @gcExpr1 = 'Visual Basic.NET', @gcExpr2 = 'VISUAL BASIC.NET'
select dbo.PROPER(@gcExpr1) -- Displays 'Visual Basic.net'
select dbo.PROPER(@gcExpr2) -- Displays 'Visual Basic.net'
```

## *ARABTOROMAN()*

Returns the character Roman numeral equivalent of a specified numeric expression

### Syntax

ARABTOROMAN(@tNum)

### Parameters

@tNum number

### Return Values

varchar(15)

### Example

```
select dbo.ARABTOROMAN(3888) -- Displays MMMDCCCLXXXVII
```

## *ROMANTOARAB()*

Returns the number equivalent of a specified character Roman numeral expression

### Syntax

ROMANTOARAB(@tcRomanNumber)

### Parameters

@tcRomanNumber varchar(15) Roman number

### Return Values

Smallint

### Example

select dbo.ROMANTOARAB('MDCCCLXXXVIII') -- Displays 1888

More that 5000 developers have already downloaded my functions, I hope they are useful.

To read more about string UDFs in Transact-SQL, please visit:

http://www.universalthread.com/wconnect/wc.dll?LevelExtreme~2,54,33,27115