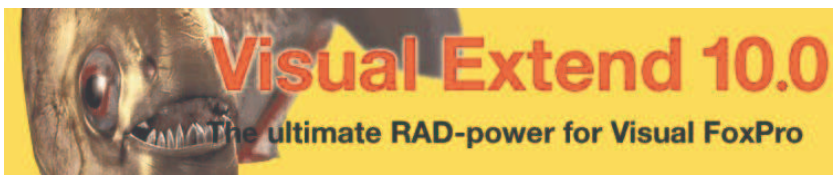
VFX 10.0 – What is new?
2. Quarter 2007

June 2007



Venelina Jordanova, Uwe Habermann

Table of content

Application activation and error reporting ViaHTTP 4
Class cVFXActivate (vfxappl.vcx) 4
Class cConnectWebService (vfxappl.vcx) 4
Class cRegistration (regservice.vcx in RegistrationWebService project) 4
Using product activation for the developed application 5
Settings in VFX – Application Builder 5
Activation Wizard 6
cRegisterProcessing class 6
cActivationWizardVfxBase class 7
Support of east asian languages 15
Config.vfx 15
Support of CSV format 15
Manage config.vfx menu 16
All VFX builders 16
VFX – Grid builder 16
VFX - Upsizing Wizard 16
VFX - Data Environment Builder 16
VFX - Audit Trigger Wizard 16
VFX - Class Switcher 16
Open dialog and XP open dialog 17
XP open dialog on Terminalserver 17
Unique fields 17
Class cDateTime 17
User management 17
Delete flag 17
Readonly flag 18
Record level security rights enhancements 18
New proprties of goProgram object (cFoxApp class) 19
Installation of Postscript printer driver 19
Data 19
Product activation 19
End user features 19
New methods of goProgram object (cFoxApp class) 19
New proprties of cBaseDataAccess class 20
Class cFormBase 20
Methods 20
Class cDataFormVFXBase 21
Properties 21
Methods 21
Class cVfxActivate 21
Properties: 21
Methods: 21
Class cExport: 21
Properties 21
Class cErrorReportDialog 21
cDocumentManagement. 22
Drag and Drop. 22
cGridMoverDialog. 22
Properties 22

Methods 22
Combobox to open forms - class cXPOpenCombo (vfxappl.vcx) 22
Properties: 22
Methods: 22
cSearchDialogBasevfxbase class 23
Help 23
lFixField for comboboxes 23
Pick Dialog 23
Run child form from child grid 24
VFX – Extended Parent/Child Builder 24
New features in cOneToMany form classes. 26
One-to-many reports 26
Child Edit Pages 27
Class cComboPicklist 28
New properties of the class: 29
Methods: 30
Class cFooterBar 30
Class cMapPoint 30
Properties 30
Methods 30

Application activation and error reporting ViaHTTP

The registration of a VFX application is now possible via HTTP besides the standard Web Service. For an application to be using HTTP to register cVFXActivation.nRegWay (appl.vcx) should be set to 13 and the HTTP URL for registration should be keyed up in the property cVFXActivation.cHTTPregisterURL (appl.vcx). These settings can also be done using VFX – Application Builder. The registration process is similar to the registration using Web Service. The processing when this registration type is selected, uses RegisterCustomerViaHTTP() method instead of RegisterCustomer()(in cConnectWebService class)
The registration via HTTP uses a Win32 Internet API functions (wininet.dll) to connect to the URL provided and sends the registration information in encoded text format.
When converting data into appropriate text format are used the functions CSVStringToCursor and CursorToCSVString.
The result returned by the registration service is stored into the property cConnectWebService.cXML. If an error occurs during registration the error message is saved in the property cConnectWebService.cLastErrorText.

Class cVFXActivate (vfxappl.vcx)

New properties:

cHTTPregisterURLServerName – Server name for HTTP application registration. The server name must not contain leading "http://" and trailing "/" in it.
cHTTPregisterURLObjectName – Object name for HTTP application registration. In this property should be keyed up the name of ASP page that receives registration data.

Example:
The HTTP Registration page is available at URL
http://vfxserver.org/vfxtestregistration/Register.asp
The values of properties should be
cHTTPregisterURLServerName = "vfxserver.org"
cHTTPregisterURLObjectName = „vfxtestregistration/Register.asp"

Class cConnectWebService (vfxappl.vcx)

New methods:

RegisterCustomerViaHTTP – Connects to the registration web page and sends customer data using HTTP. If a valid activation key is received, the application is automatically registered.

Class cRegistration (regservice.vcx in RegistrationWebService project)

Registration Web service is changed to recognize received data format automatically

New properties:

lDataInXMLFormat – Internally used. =.T. when XML format is used in received data

New methods:

CursorToString() – Converts the result data to string, using appropriate format depending on incoming data format, kept in lDataInXMLFormat property.

CursorToCSV() – Converts the specified cursor in string data using CSV format with additional structure info line
CursorToXML() – Converts the specified cursor in string data in XML format
CSVToCursor() – Creates a cursor from Data string, sent in CSV format
XMLToCursor() – Creates a cursor from Data string, sent in XML format

Note: Keep in mind that the RegistrationWebService.DLL must be build as Multi-threaded COM Server.

Using product activation for the developed application

Settings in VFX – Application Builder

The functionality can be enabled using VFX – Application Builder by marking the checkbox „Enable Product Activation“. This checkbox sets the value of the property cFoxAppl.lUseActivation. Another way to key up the application to use product activation is to set this property to .T. in design time.

At customer side, the Activation key is stored into a file. Name of this file can be entered in the field „Store activation data to“ in VFX – Application Builder. Alternatively, this can be keyed up in the property cVFXActivation.cStoreActivationData. The default value is VFX.ini

Additional setting whether files related with product activation should be hidden from end-users, can be done using the property cVFXActivation.lHideRegistrationFiles. Its value can be set in VFX – Application Builder via checkbox “Hide registration files”

End users can register and obtain the activation key using SOAP or HTTP. Developer can set default rights and key validity period, which will be used to generate an activation key, when end users register online. In this way developer can give a chance to the potential customers to test specified application functionality for defined time period. Time limited activation key can be generated only in case when the activation key type is selected to be “Windows compatible”. The trial period (in days) can be keyed up in VFX – Application Builder in „Activation key validity in days“ field or manually in the property cVFXActivation.nProductActivationBehavior. The default value is 30 days.

Setting the "Activation key type" value to “1 – VFX 9.0 activation key format” can be kept backward compatibility to the former VFX versions. These activation keys can become relatively long. Alternatively a shorter format can be also used for the activation key. With this format the activation key is always exactly 25 characters long. Each five characters are separated by a hyphen. Users already know this format of activation keys of different Microsoft products. In the VFX - Application Builder, this feature is keyed up under "Activation key type". Manually, the value of the property cVFXActivation.nProductActivationBehavior can be set to 1 for long activation keys (VFX 9.0 format) or 2 for short activation keys (MS compatible format).

If time limited activation keys should be generated, the check box Time limited activation key“ must be marked. This check box is only enabled if short activation keys (MS compatible format) are used. Manually, this can be keyed up by setting the value of property cVFXActivation.lUseTimeLimitedActivationKey.

In the textbox „Start day of activation key “ can be entered the start date, starting from which, time limited activation keys will be generated. The default is the 1/1/2007. This value can be keyed up manually in the property cVFXActivation.dStartActivationDate.

Developers can select to use tolerance to the hardware computer parameters, used to identify the customer computer. The activation key is still valid, even after end-user changes some hardware details and values used in application activation are already not same, as at the time when activation key has been generated. The number of parameters that user is allowed to change is keyed up in the textbox “Number of changes accepted when using hardware parameters tolerance”
Manually, this can be keyed up in the property cVFXActivation.nHardwareParametersTolerance.
Initial values of hardware parameters are stored in a file, which name can be keyed up in the property cVFXActivation.cStoreHardwareParameters or in the VFX – Application builder in the textbox “Hardware parameters file”. Default value is “vfx.hrd“

The form that will be started when the application is not registered is keyed up in the property cVFXActivation.cRegisterFormName. In the VFX – Application builder, this setting is made in the textbox “Name of the register form”.
To run the VFX 9.5 registration form, fill “vfxRegister” in this property. The property should b set to „vfxactivationwizard“, in order to be started the new activation wizard.

Activation Wizard

In order to make it easier for the user to register and activate the Application, VFX offers a new activation Wizard. The wizard functionality is developed in the class *cActivationWizardVfxBase* of *cFormBase* library and has an instance *cActivationWizard* in *cForm*. The class is based on *cWizard* class. Also a new form, *vfxActivationWizard*, is created, based on *cActivationWizard* class.

In *cVfxActivate* class of *vfxAppl* library, is added a new property *cRegisterFormName*, which keeps the name of the register form to be run. The default value is “*vfxRegister*” (current behavior). The application will use the new Activation Wizard if this property is set to “*vfxActivationWizard*”. The developer can create his own registration form and set it as registration form of the Application, as well.

cRegisterProcessing class

For easier usage and supporting the old register form and new activation wizard, the main functionality of registration processing is encapsulated into a new class *cRegisterProcessing* in *vfxAppl* library.

Properties:

cListNotSetProperties – List of properties that are not set. Internally set. Used to display missing settings in *cActivationWizardVfxBase* form;

lRegEmailNotAvailable – Internally set. If .T. displays a message that the registration e-mail is not set or diasables email option in *cActivationWizardVfxBase* form;

lRegPropertiesNotSet – Internally set to .T. properties, required to process registration are not set.

Methods:

CheckRegistrationProperties – Checks if all needed properties for registration are set;

CheckRequiredFields – Checks if all requred fields (marked with *) are filled;

CreateRegInfoTable – Creates RegInfo cursor;

ErrorMessage – Occurs when the Valid event returns false (.F.), and provides a means to display an error message;

ErrorMessage_SendPassword – Displays an error message when an error arose while sending password;

GetApplicationVersion – Returns the application version;
GetCaption – returns the caption of a given label;
GetEmailText – Composes the email text of registration email;
PrepareCustomerData – Prepares customer data to be sent;
PrepareRequestReplacementKeyData – Prepares request replacement key data to be sent;
PrepareSendPasswordData – Prepares the password data to be sent;
ProcessActKey –Gets the activation key, returned from web service or registration page and validates it;
RegisterOnlineErrorMsg – Displays error message when register online failed;
RunWithoutRegistration – Called when Run Without Registration is selected;
SaveToFile – Saves the registration information to file (name of the file is kept in cParamfile property);
SendEmail – Sends an email with registration information;
SendPasswordViaEmail – Calls the web service method to request password via email;
SendRegInfo – Sending registration information to Web Service or Registration web page;
TryAutomaticRegistration – Tries to register the application automatically, when XAK file exists in application folder;
ValidActKey – Validates the activation key;
ValidateData – Validates the entered registration data. Called by *CheckRequiredFields*.

cActivationWizardVfxBase class

Properties:

cParamFile – Name of the file to store registration data.
cRegEmail – Email to send registration data to;
cRegistrationResultCaption – Internally used. Used as caption to display the result from the activation;
cRegistrationResultMessage – Internally used. Used as message to display the result from the activation;
cServiceMethodName – Contains name of Web service method to be called;
cServiceName – Contents Service Name when connecting to Web Service;
cWSAcivationKeyFieldName – Name of field containing Activation Key;
cWSDL – Contents URL to Web service WSDL;
cWSResultAddInfoFieldName – name of the field in the XML or CSV string, returned from Webservice or registration page, containing additional information;
cWSResultStatusFieldName – name of the field in the XML or CSV string, returned from Webservice or registration page, containing operation status;
INIFileName – Used internally. Stores path and filename of the INI file used to store activation info;
nRegWay – Controls the way in which registration data is sent to the developer;
oActivation – Holds a reference to goActivation as far as it is still not a global object during Init method of the class;

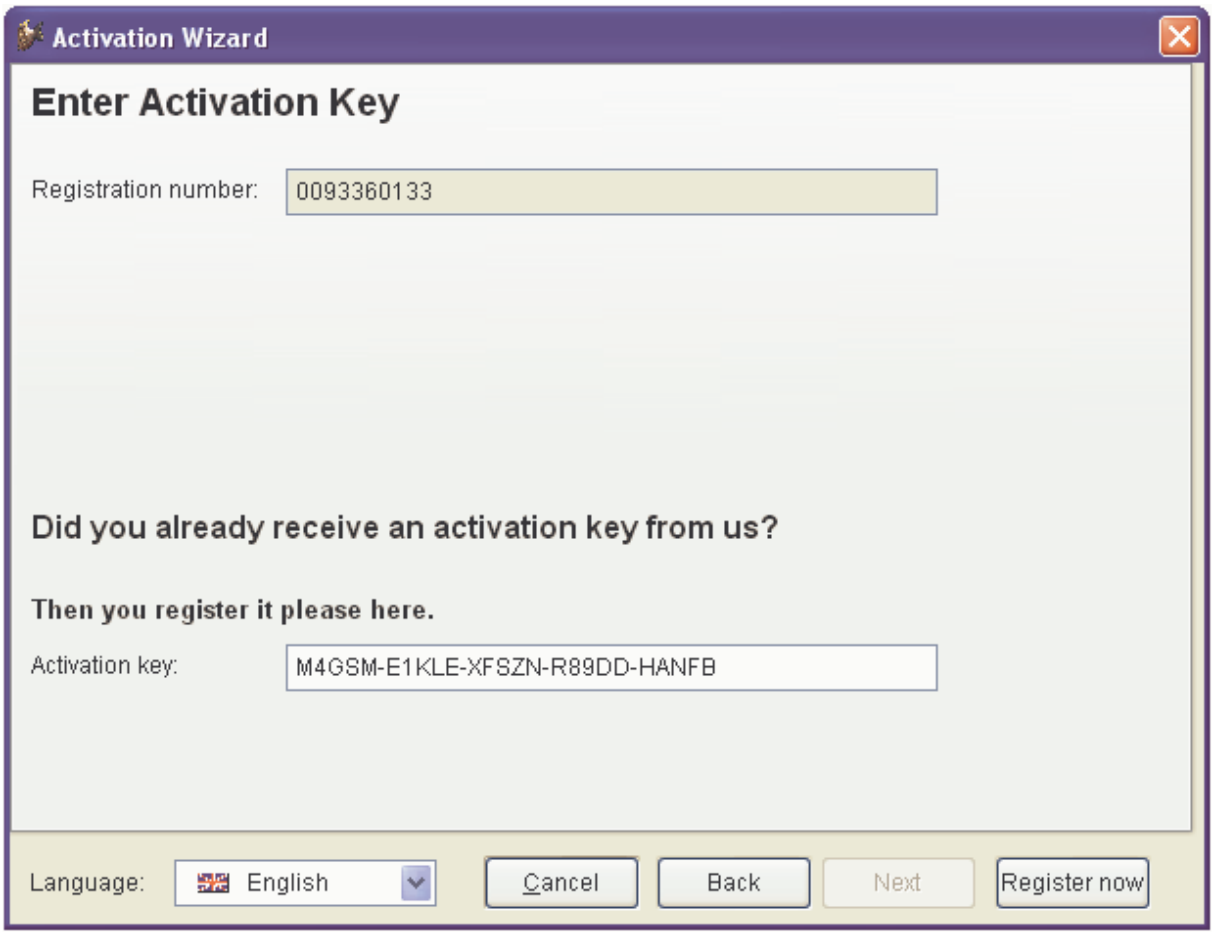
Methods:

GetMessageText – Returns message text in current language;
LoadLicenseAgreement – Reads and loads license agreement text;
LoadReasonCombo – Loads reason combo texts, run-time localized;
OnSuccessfulActivation – Executed when the activation is successful.

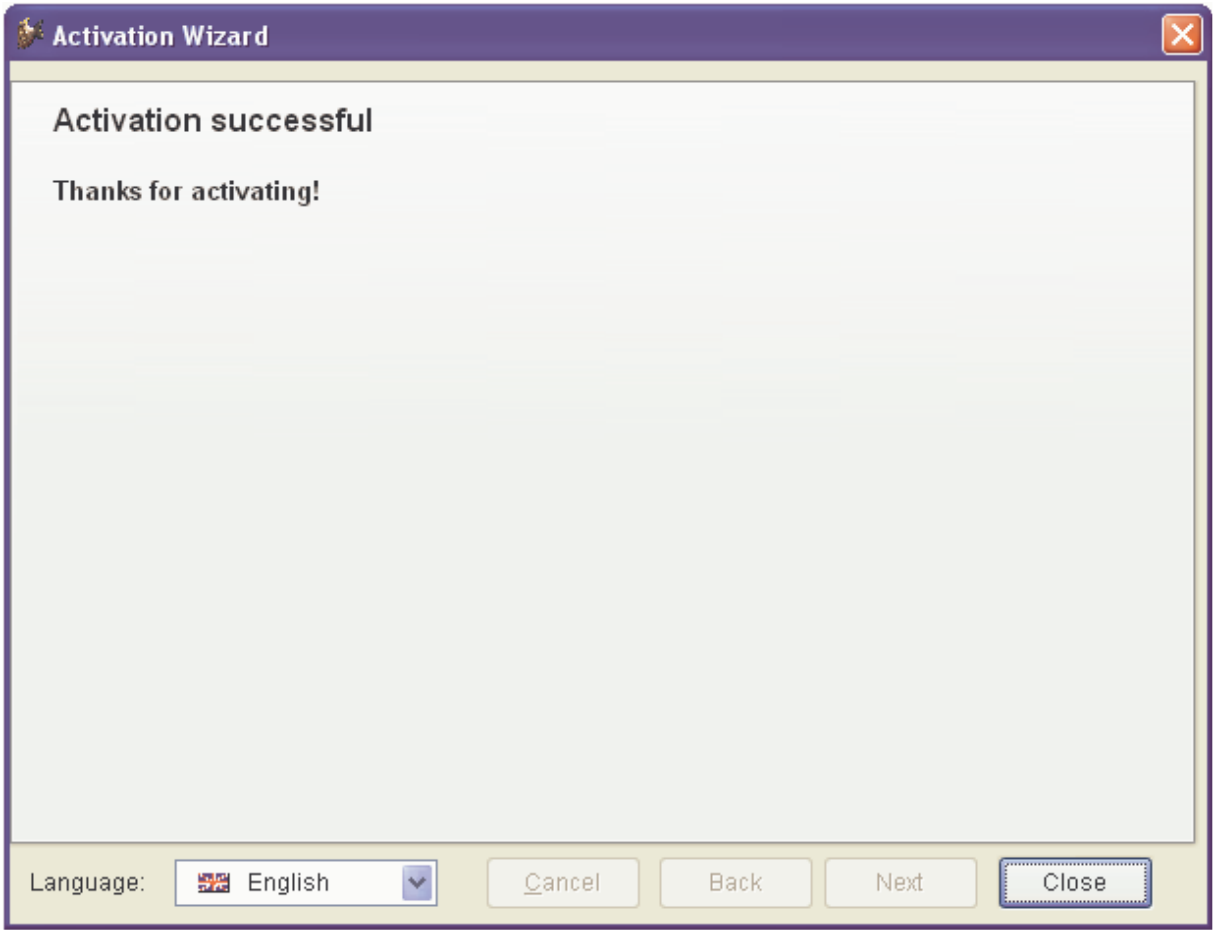
User Interface

At the screenshot below is shown the start page of the Activation Wizard. Here is places the functionality to enter the Activation Key directly, if the user has received it already. This can be done by pressing the button “Enter Activation Key”.

When the button “*Enter Activation Key*” is pressed, the user is directed to the next step:

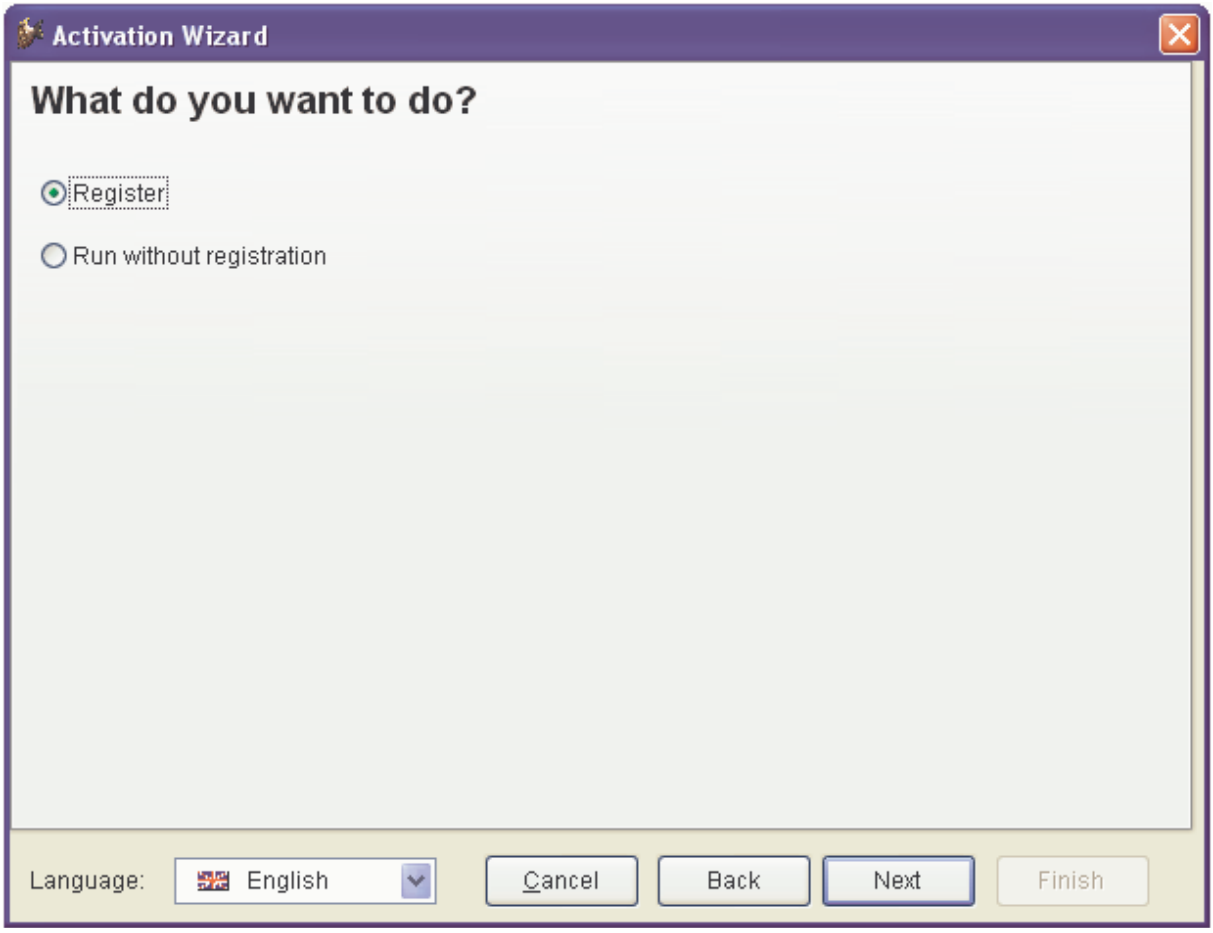


Here the wizard expects an Activation key to be entered. After the activation key is entered and the button "*Register now*" is pressed, the key value is validated. If the key is valid the Application is activated and used sees the last page step of the wizard.

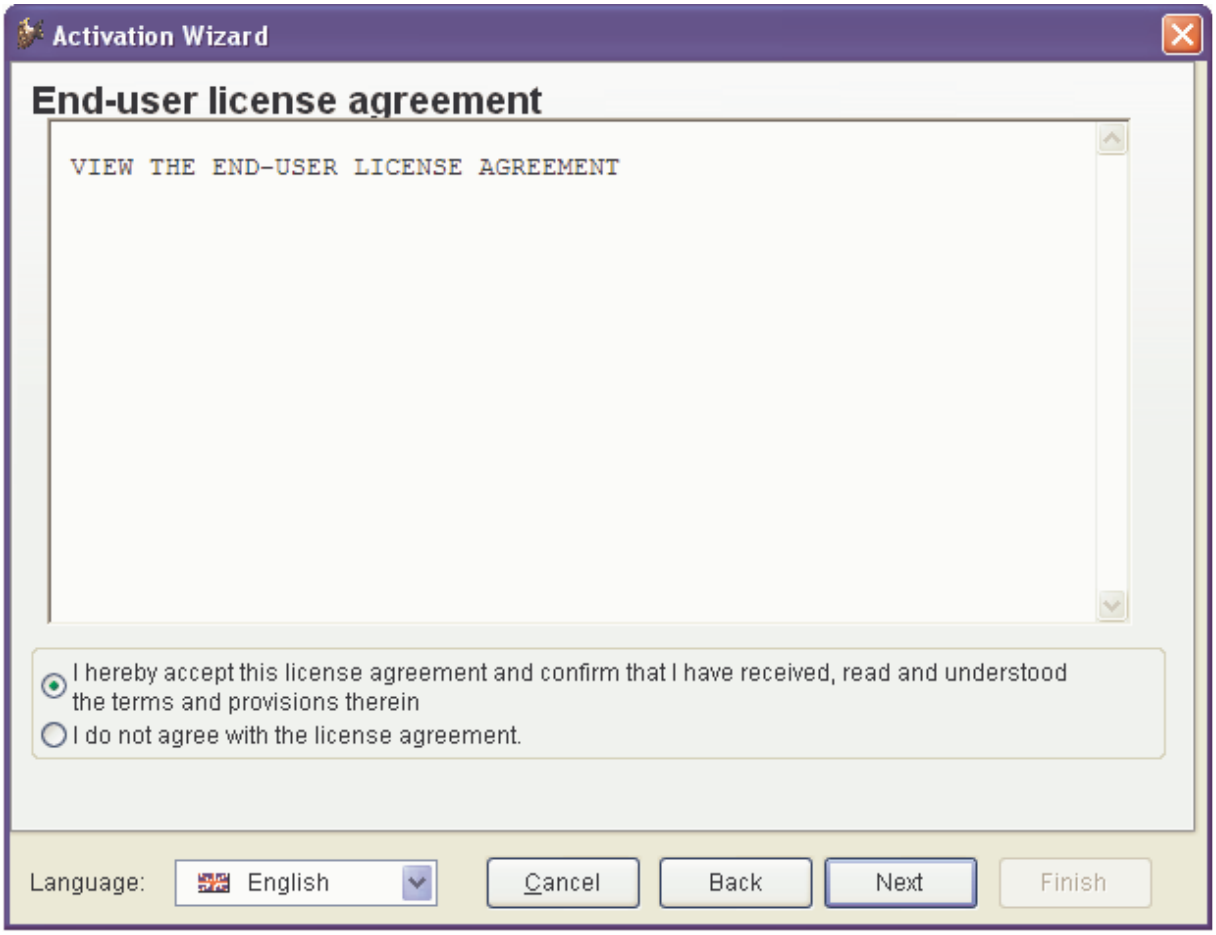


This is the last page of Activation Wizard. Here are displayed all messages that in former register form are seen as Message box. The button "*Close*", release the wizard.

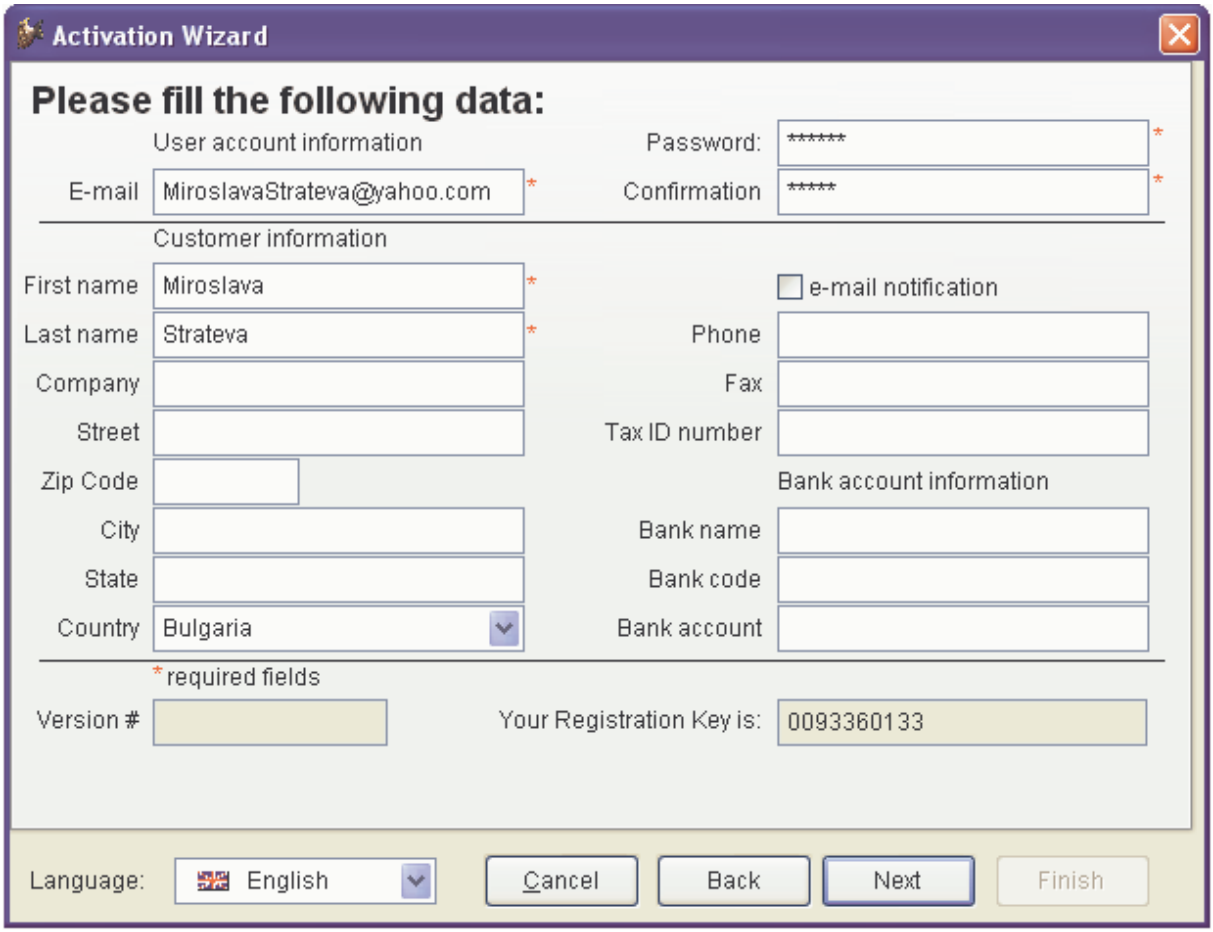
If on the first step is pressed the button "*Next*" (the case when the Activation key is not known), there comes the second step of the Wizard, where user has to select the way to be used for registering.



In this step user can select among the options to continue runing the Application without registration or to proceed next steps and register. The option "*Run without registration*" will lead to exiting the wizard and application will not be activated. In other case of selcting "*Register*" option, user is directed to next step.

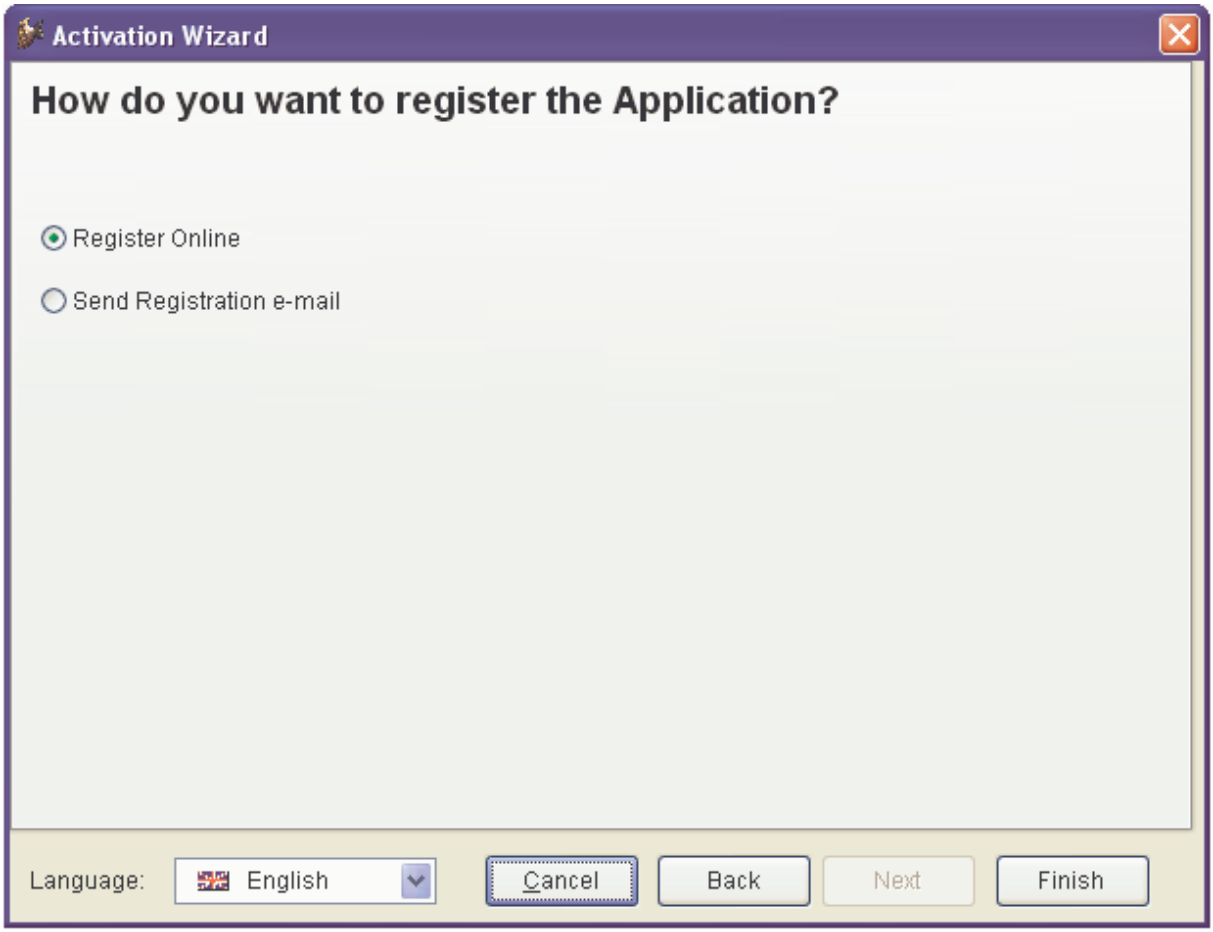


In this step is displayed the End-user license agreement. The wizard will not continue ahead unless the user accepts the agreements.



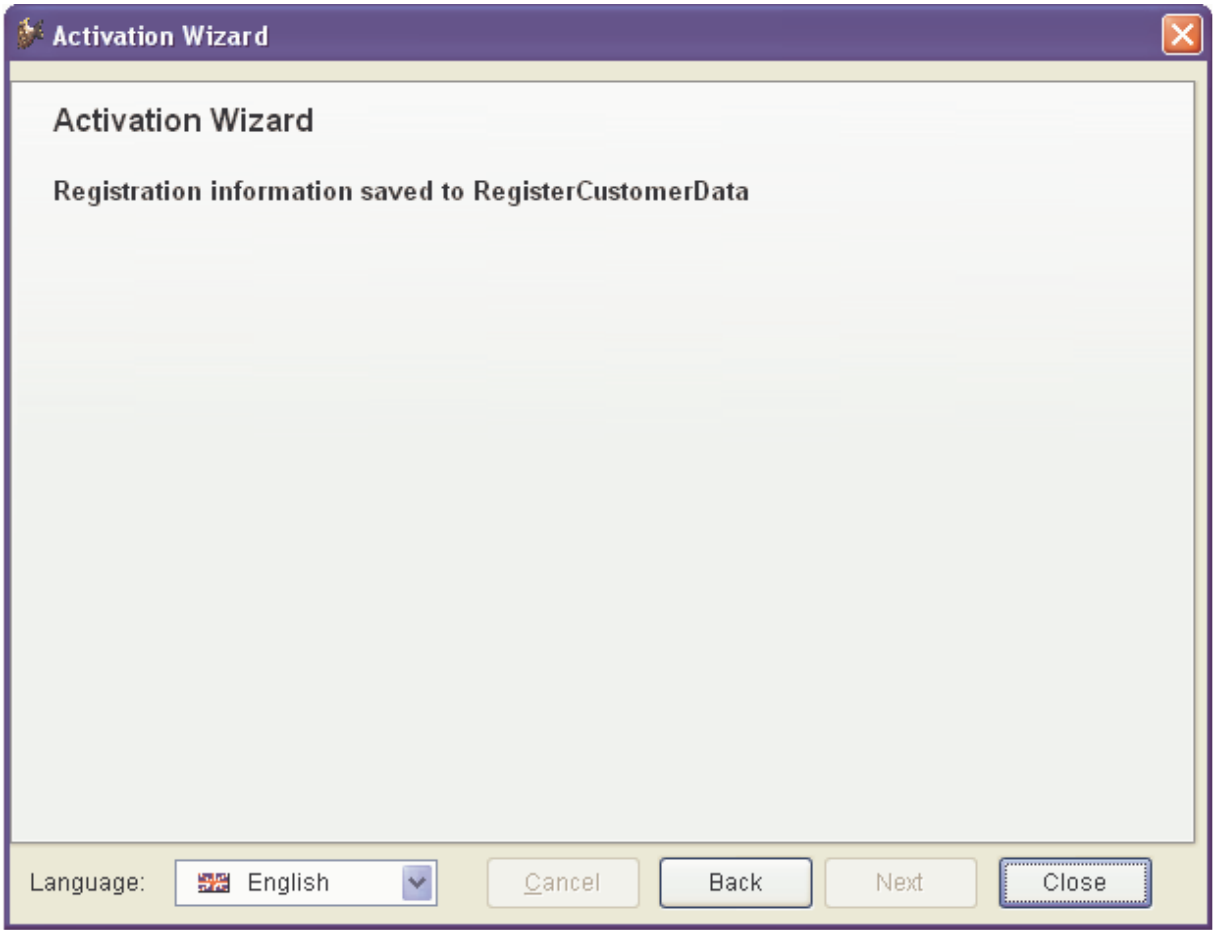
On this step user has to enter registration information. The wizard will continue execution only when all required fields are filled. In the E-mail text box must be entered a valid E-mail address. The password must be at least 5 characters long and the password and confirmation fields must have same values.

The next step depends on the selected way to send the registration data to the developer. The available options depend on application settings made by the developer. If registration will not be performed online, a message, explaining what will e done is displayed. If the registration data will be sent online, using Web Service or HTTP the following step will be:



The option "*Send Registration e-mail*" is displayed disabled, if the e-mail, to send registration data to, is not set up (i.e. when property *cRegEmail* is empty).
If the selected method for sending the registration data is set to 11 (the registration data will be saved in a file) or to 12 (the registration data will be send by e-mail), in this step will be shown only a message.
Now, when use clicks "*Finish*" button the registration process starts.

The last wizard page shows the final message with the result of the registration process. When user registers first time this message is:



The “Close” button will release the wizard.

The wizard form is localized and the language can be changed at any time. The wizard is started in the language which is default for the computer.
In any time, the user can exit the wizard by pressing the button “*Cancel*”. This will close the Wizard and the application will run without activation.

Support of east asian languages

VFX 10.0 supports the languages traditional and smplified Chinese as well as Japanese. This feature requires a WIndows version which supports DBCS character sets.

Config.vfx

Support of CSV format

The file Config.vfx can now be stored as XML or CSV. This allows to run VFX applications in environments where MSXML is not available and cannot be installed.
This behaviour is controlled by goProgram.nConfigVfxFormat property. The property determines the format in which config.vfx is saved. When the file is read its format is detected automatically. The file is always encrypted with the password for config.vfx from goProgram.cConfigPassword.

Important: When working with CSV format, all chanarected strings are truncated to maximal length of 254

New functions:

ReadConfigVFXtoCursor(tcFileContents, tcCursorName)

CursorToConfigVFX(tnConfigVFXFormat, tcCursorName, tcFilePath, tcPass)

CSVStringToCursor(tcDataString, tcCursorName) – Converts a string in CSV format with additional structure info line into a cursor

CursorToCSVString(tcAlias) – Converts the specified cursor in string data using CSV format with additional structure info line. Converts Memo data type to C(254)

Manage config.vfx menu

The menu entry in the application menu is available only for end-users with user level = 1.

All VFX builders

In all VFX builders you can add a new row in grids with a single mouseclick now.

VFX – Grid builder

The grid builder and also form builders when creating grids, calculate the width of a column considering the width of the header as well as the field width in the column. The width is calculated by multiplying TXTWIDTH(…) by font size returned by FONTMETRIC(…)
In grid columns, used VFP base classes, get the font settings of the corresponding class in Vfxobj.vcx.
The property cfixcolumnlist of grids gets set correspondent to the readonly value of columns in grids. This behavior of the VFX builders is reentrant.

VFX - Upsizing Wizard

A new button allows NULL values in all tables and all fields (as far as possible) with a single mouse click.

VFX - Data Environment Builder

For cursoradapters there is a new checkbox “Send Updates” controls the property SendUpdates of controls the CA objects.

VFX - Audit Trigger Wizard

The wizard now only suggests tables wich have a primary key index. The grid now shows only the tables that have primary key defined. The existence of primary key is checked with DBGETPROP(<tablename>,”TABLE”,”PrimaryKey”).
At startup all settings are read from the tables in the project database.
There are two new buttons to select or unselect all tables (Select all/ Unselect all).

VFX - Class Switcher

When changing the class of a control to a container control the class switcher now assigns the controlsource to the control inside of the container.

When switching between classes and one or both are containers besides the property values of the two classes, the following properties of their base textboxes are transferred: ControlSource, InputMask, Format and StatusBarText.

Open dialog and XP open dialog

Both open dialog forms now use the same method to exclude entries based on assigned user rights. The processing is done in the function LoadVFXFopen() which loads data from vfxfopen table, according to current user level.

XP open dialog on Terminalserver

If a VFX 10.0 application runs as a client on a Windows Terminalserver the feature 'Auto Hide' is automatically switched off. The option “Auto hide XP open dialog” in Customize dialog is also disabled in this case.

Unique fields

If a user tries to save a not unique value in a unique field this field gets the properties assigned for 'required field failure properties' and a messagebos is shown.

Class cDateTime

Now, the controlsource of this class supports properties, too.
The type of the control source is determined in Init() metod of the control and the internally used property nControlSourceMode is set accordingly: 1 (Default) for field, 2 for variable/property. This property is checked when the value is saved to the control source and is executed replace or assignment(=) is executed accordingly.

User management

The new fileld useraccess in the user table allows to override individual user group settings.
In the users table Vfxusr there is a new AutoInc field for a primary key (Vfxusr.idvfxusr).
Alternatively to the existing user name there is a new field in the table Vfxusr to store the Windows login name (Vfxusr.ntlogon).

Delete flag

Instead of deleting records in VFX 10.0 applications there is a field supported which works as a delete flag. This new feature allows to synchronize databases.
Available only through the CursorAdapter class of VFX

With this new feature when a record gets deleted it is marked as deleted (and still exist in the table) and is not physically deleted. When a cursor adapter is filled, it fetches only the records whose flag is not set. This flag is a field in the table.
The application behavior is controlled by goProgram.cDel_fld property. This property keeps the name of the field in any table, which is used as deletion flag. If it’s not empty the field is used in every table where it exists. Otherwise records are deleted in the usual way.
The type of the deletion flag field in any table has to be N(1). The values stored: 0 – record exists, 1 – record has been deleted.

The field used as a deletion flag should be included in CursorSchema, SelectCmd, UpdateableFieldList and UpdateNameList properties of the CA object in order for it to be available and updatable.

PEMs used by this feature:
Properties:

cFoxApp.cDel_Fld (vfxappl.vcx) – stores the name of the field used as deletion flag in any table.

cBaseDataAccess.cDeletedFilter (vfxctrl.vcx) – Internally used to store the filter for the deleted records added to the where clause of the CA on BeforeCursorFill().

cBaseDataAccess.lDelFieldSet (vfxctrl.vcx) – Internally used to store if a deletion has occurred when cursor is updated.

Methods:

cDataFormVFXBase.SetRecordDeleted() (vfxformbase.vcx) – Called instead of the actual delete command. This method marks the record for deletion (sets the delete flag field to 1) or deletes the record, depending on the settings.

When a record is deleted by using delete flag, the actual operation that is carried out with the CA is not Delete, but Update (because a field is updated). When this is the case the property lDelFieldSet of the CA is set (on BeforeUpdate) and on AfterUpdate the record is deleted from the cursor and its deletion status restored (SETFLDSTATE(0,1) as if it has not been deleted) in order for it to be missing from the cursor, but not to trigger an update event.

SET DELETED is respected when cursor is filled. If it is OFF all records are fetched, otherwise only the ones whose deletion field is 0.

Readonly flag

New functionality allows data records to be set as readonly by setting a field in the data table.
The property *goProgram.cRdn_Fld* keeps the name of the field in any table that determines the readonly status of the record. The features works if this property is set and a field with such name exist in the table. Otherwise, nothing happens.
The new method *cDataFormVFXBase.GetReadOnlyStatus()*, is called by *OnRecordMoveRefresh()* method of *cDataFormVFXBase* class. It checks the status of the field (if it exists) for the current record and if the record is readonly, the properties *lCanEdit* and *lCanDelete* of form are set to .F.

Record level security rights enhancements

As of VFX 10 userid can be used instead of username in security tables. When used, UserID value is taken from the new field in vfxusr table – IdVfxUsr.
The new security table functionality utilizes the use of the fields for deletion status (goProgram.cDel_Fld), InsertUser (goProgram.cIns_Usr), InsertDate (goProgram.cIns_Date), EditUser (goProgram.cEdt_Usr) and EditDate (goProgram.cEdt_Date).
If any of the above fields exist in the security table, it is used in same way as these fields are used in any other table. Additionaly when a record is inserted and it exists in the security table but is marked as deleted, it is restored – its deletion status set to 0.
The code, implementing this functionality is placed in in cDataFormVFXBase class in method *CallSecurityRightsDialog()*

New proprties of goProgram object (cFoxApp class)

Installation of Postscript printer driver

lAlwaysInstallPSPrinter – (default .F.) Controls installation of Postscript printer driver. If the property is set to .T. the printer which name is keyed up in goProgram.PSPrinterToInstall is always installed to be used for PDF creation even there is already another Postscript printer driver installed.

Data

nConfigVfxFormat – Format used to store the Config.vfx file. 0 - XML (default), 1 - CSV
nShowRetrieveMsg – Specifies if a message "Retrieving data …" will be shown while cursors fill or refresh executes. 0 – (Default) - use cursor adapter setting; 1 – show always; 2 – do not show.
cDel_Fld – Name of the field in any table to be automatically used to store the deletion status of the record. If no such field exists, records are deleted in the usual way.
cRdn_Fld – Name of the field in any table to be automatically used to store the readonly status of the record. If no such field exists, nothing happens.
lFillEditDate - ForNewRecords – When saving new records set edt_date to the same value as ins_date.
lUseGUIDKeys – If set to .T. , GUID fields are used instead of Autoinc. In form classes code generating GUID after inserting new record is executed.

Product activation

nProductActivationBehavior – 1 – VFX 9.0 activation key format; 2 – Microsoft compatible activation key format
cHTTPregisterURLServerName – Server name for HTTP registration of application
cHTTPregisterURLObjectName – Object name for HTTP registration of application. String that contains the name of the target object of the specified HTTP verb. This is generally a file name, an executable module, or a search specifier
cRegisterFormName – (default "vfxRegister") Keep a name for the register form used

End user features

nGenerateOneToManyReport – Defines whether in one-to-many forms automatic report generation will create OneToManyReport FRX. 0 - Use form setting, 1- Force to .T., 2 - Force to .F.
nEnableChildInsert – Controls the autoinsert feature of a child grid in a form. 0 - use form setting, 1 - enabled for all forms, 2 - disabled for all forms
nRecordMoveRefreshtimeout – This property allows developers to delay form refresh, after the record pointer is moved. This can decrease workload and improve performance, especially by using incremental search in search grid of the form. In the property is keyed up the interval of timer (in miliseconds) after the record pointer is moved, when the form is refreshed. The developer's code that should be executed with such delay, must be written in the method *OnRecordMoveRefresh*.
lShowNTLogonFieldInUserManagement – .F. NTLogon username field is not visible in User management dialog; .T. NtlogonUsername will be used for automatic login and available for editing in User management dialog
oXPOpenCombo – internally used. Keeps a reference to an instance of the class cXPOpenCombo, when it is used, instead of open dialog.

New methods of goProgram object (cFoxApp class)

vfxInputBox(tcInputBoxPrompt, tcInputBoxCaption, tcDefaultValue, tnTimeout, tcTimeoutValue, tcCancelValue) – Calls InputBox function and returns the result from InputBox function. Created to allow Ctrl+C, Ctrl+V and Ctrl+X to function.
Parameters are similar to InputBox function of VFP.

tcInputBoxPrompt – Specifies the Prompt displayed above the text entry box.
tcInputBoxCaption – Specifies the text to display in the title bar of the dialog box.
tcDefaultValue – Specifies a default value to display in the text entry box.
tnTimeout – Specifies a timeout value in 1/1000 seconds. Specify zero in nTimeout to prevent the dialog from timing out. This is identical to omitting nTimeout.
tcTimeoutValue – Specifies value to return if a timeout occurs. cTimeoutValue isn't returned if nTimeout is set to zero or is omitted.
tcCancelValue – Specifies a character value to return if the user exits the dialog box by choosing the *Cancel* button or pressing the Esc key.

New proprties of cBaseDataAccess class

lShowRetrieveMsg - Specifies if a message "retrieving data …" will be shown while cursors fill or refresh is executed. .T. – show a message; .F. (Default). – do not show message.

cPickWhereClause – holds where clause expression to be used when CA is instantiated by cPickDialog, to get pick list values. It usually includes This.vIDValue, instead of parameter used in the data form.
cIDFieldFullName – Alias and field which is used as a parameter value. This value is retrieved in Init method of Pick dialog from the callung form's data session and is stored nito property vIDValue to be available later, when executing into Pick dialog data session
vIDValue – Internally used. Holds the ID value to be used in where clause.

Example :
cWhereClause = "field1 = ?tablename.field2"
cIDFieldFullName = "tablename.field2"
cPickWhereClause = "field1 = ?this.vIDValue"

cDeletedFilter– Internally used to store the filter for the deleted records added to the where clause of the CA on BeforeCursorFill().
lDelFieldSet– Internally used to store if a deletion has occurred when cursor is updated.

Class cFormBase

Methods

SetAlwaysOnTopOnForms(tlSet, tlIgnoreWindowType)

tlSet – if .T., the method stores references to all open forms (excluding current), whose AlwaysOnTop property is .T. in the one dimensional array thisform.aalwaysontop[] and sets the AlwaysOnTop properties to .F..
If .F. and the array thisform.aalwaysontop[] is not empty the method sets AlwaysOnTop = .T. to the forms whose references are stored in it.

tlIgnoreWindowType – If .T. the method always executes. Otherwise it executes only if the form is modal (WindowType = 1)

This method is called on form's Init() (tlSet = .T.) and Release() (tlSet = .F.), as well as before and after the preview of a report to prevent any always on top forms (e.g. ToolBox) to appear in front of the preview window when Report Behavior is set to 80.

Class cDataFormVFXBase

Properties

aAdditionalControlsToRequery – Array keeping references to cComboPicklist objects, which Requery must be called when moving record pointer. Used when cComboPicklist.lAddCurrentValueToList = .T.
oFooterControl – Keeps a reference to topmost FooterBar container in the form.

Methods

SetRecordDeleted() – Called instead of the actual delete command. This method marks the record for deletion (sets the delete flag field to 1) or deletes the record, depending on the settings.
GetReadOnlyStatus() – Called by OnRecordMoveRefresh(). Checks the valueof the readonly field (if it exists) for the current record and if the record is readonly, the properties lCanEdit and lCanDelete of form are set to .F.
CallSecurityRightsDialog() – Inokes dialog to assign record level security rights.

Class cVfxActivate

All computer-related parameters (except constants) are saved in hardware parameters file, and included in Hardware Parameters Tolerance functionality.

Properties:

aInitialHardwareParametersValue() – An Array to store initial hardware parameters value;
cApplicationFileLocClassName() – Keeps the name of used ApplicationFileLoc class. Default value: cApplicationFileLoc;
dEndValidityDate() – Keeps the activation key validity date.

Methods:

CheckHardwareParametersValidity() – Check if the initial hardware parameters correspond to the current values;
GetActKey(tcIniFile) – Reads the INI file and returns the activation key if available;
GetFileCreationDate(tcFile) – Returns the creation date of specified file;
GetRegKeyValue(tcRegKey) – Reads and returns the value of specified registry key;
LoadInitialHardwareParametersValue() – Loads initial hardware parameters value, saved when activating application, in an array aInitialHardwareParametersValue.

Class cExport:

Properties

cExportFields – A comma separated list of fields to include in export.
If the property is not empty, the list of fields kept in it is taken into account when executing export of any type, but Export to XML. Only fields listed there are exported.

Class cErrorReportDialog

It's added the functionalityto send the error messages through HTTP, if the property *nRegWay* of *goActivation* is set to 13.
The format of sent data is changed to Text file instead of XML.

cDocumentManagement.

Drag and Drop.

The dropped Outlook folders are saved by ID and not by name (as this has been so far). This allows to keep the dropped folder even if it's renamed later.
The form is entered in Edit mode on DragOver method of DocumentManagement container and not on DragDrop (as it has been). This is done to avoid error when saving after dropping more than one file from Windows Explorer.

cGridMoverDialog.

Incremental search is allowed in source and destination grid.

Properties

oActiveColumn – Keep the last column which has focus.

Methods

onHeaderDblClick() – All headers' dblClick events are bind to this method;
onHeaderMouseDown() – All headers' MouseDown events are bind to this method;
onHeaderMouseEnter() – All headers' MouseEnter events are bind to this method;
onHeaderMouseUp() – All headers' MouseUp events are bind to this method;
onTextKeyPress() – All texts' KeyPress events are bind to this method.

Combobox to open forms - class cXPOpenCombo (vfxappl.vcx)

Instead of using the open dialog a new combobox in the standard toolbar can be used to run forms. User rights, assigned per form and per user group are considered. Items in the combobox list are read from vfxfopen table and the display sequence is controlled by the new field in the table TbrCboSort. If the value in this field is 0, the record is not displayed into combobox.

Properties:

nUserLevel – The instance of class is used when the user level is greater this value. Otherwise, the control is hidden. Set this property to 0, to enable it for all user levels.

Methods:

EnableCombo() – enables the control and fills its list values
DisableCombo() – disables the control and makes it Visible = .F.
ItemExecute() – executes the action associated with currently selected item: runs a form/command

A reference to the control, instantiated in main toolbar is kept into goProgram.oXPOpenCombo.

With this new user interface class it is possible in a VFX application to use a combo box in the main toolbar instead of the XP open dialog. For this purpose is used the class cXPOpenCombo.
An object of this class can be placed in cAppNavBar (appl.vcx). If it exists in the main toolbar on application startup, a reference to it is kept in goProgram.oXPOpenCombo property.
This combo box can be used instead of XP open dialog for user levels above a certain value or all user levels. For this purpose the object's property nUserLevel is used. If this has value of 0, the

combo box is used for all user levels. Otherwise it is used for users who have level greater than its value and for user levels below or equal to it – XP open dialog is used.
What will be used is determined on goProgram.StartOpenForm().
The combo is started by calling its method EnableCombo(). This method reads values from vfxfopen table and fills combobox list. To load necessary data records LoadVFXFopen() is used. In combo box are displayed all records from vfxFopen which have tbrCboSort > 0. The combo box has two columns. Only the first column is visible and it shows the display names of the forms as entered in the *Title* field in vfxfopen table. The list is ordered by the values, saved in field *TbrCboSort*. The second column holds the command to be executed when the item is selected.
The command to be executed is processed by calling the method ItemExecute(). This method executes the command of the currently selected item.
The class is designed to be used by mouse. The method ItemExecute() is called only when an item in the combo is selected with the mouse.

cSearchDialogBasevfxbase class

A new operation “Start with” is added in the Operator combobox. This operation searches for the records in which the selected field starts with the string entered in the Value field of the dialog. The operation “Start with” is available only for fields of character or memo data type. The behaviour of the operator corresponds to precending behaviour of the operator “Equals” for character or memo data type.

The operator “Equals” is changed now so, that for character or memo data types it searches for exact match.

Help

For each form it is possible to show an own help file. If a help topic cannot be found the start page of the help file gets shown automatically.
The file name (without path) should be set up in the form’s property cFormHelpFile. It is opened from the same location as the main help file, set in goProgram.cHelpFile property.

lFixField for comboboxes

A combobox can be used as a fix field. Fot this, its property lFixField should be set to .T..

Pick Dialog

For pick dialog it is now possible to preset the width of columns as well as the sort order. The settings of the developer may override the settings of the end-user in any case.
Parameterized cursoradapters are now supported as the data source for pick dialog. For this purpose must be keyed up following proerties of tha CA object
cIdFieldFullName – Name of field which is used as a parameter value. This value is retrieved in Init method of Pick dialog from the callung form’s data session and is stored nito property vIDValue to be available later, when executing into Pick dialog data session
cPickWhereClause – Where expression to be used in pick dialog. It usually includes This.vIDValue, instead of parameter used in the data form.

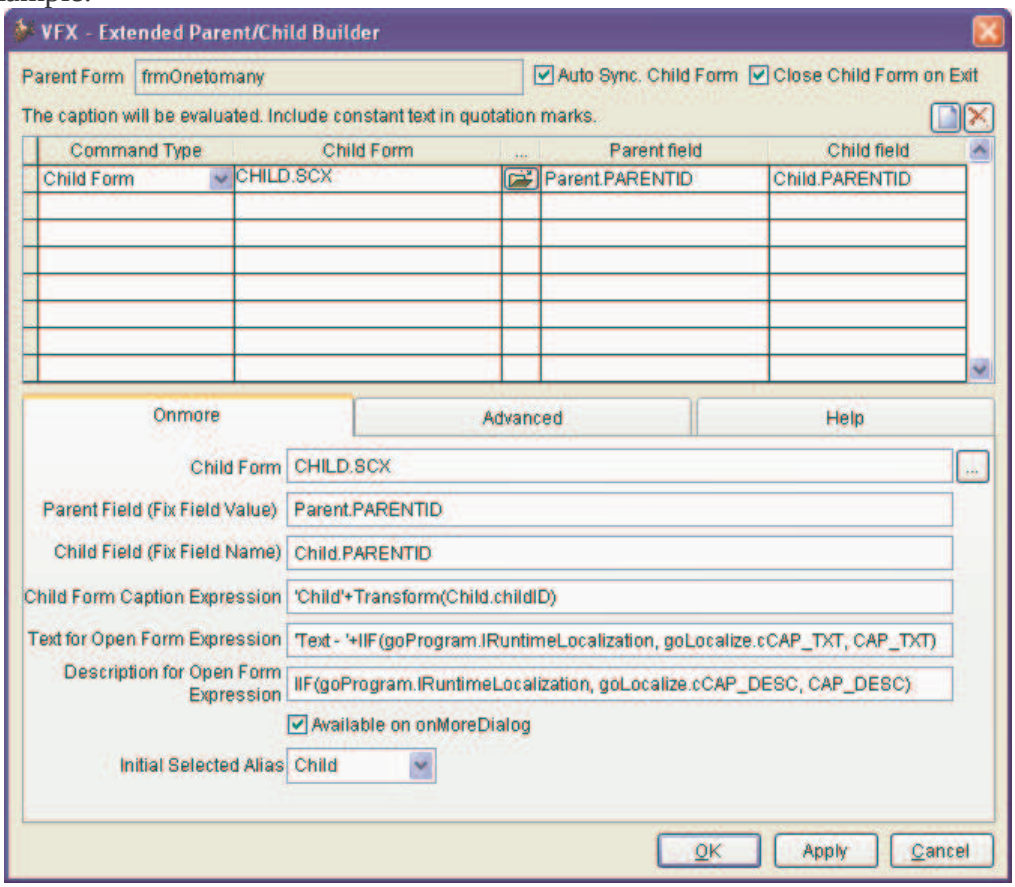
Run child form from child grid

Child forms can now be started out of childgrids. The child form gets synchronized when moving the record pointer in the child grid of the parent form.
Parent/Child Builder

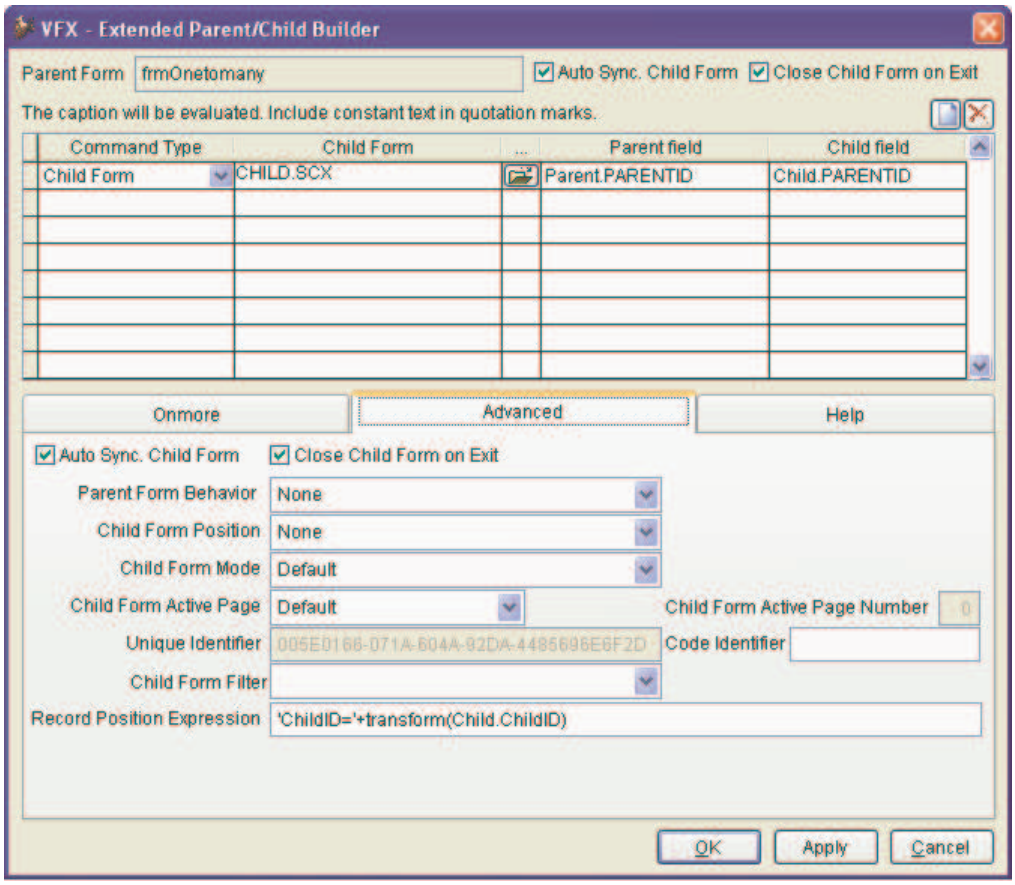
A new combobox “Initial Selected Alias” is available in Parent/Child Builder. This combo is displayed only if the form inherits onetomany class or contains child grids placed in it. The combo lists the aliases of the childgrids on the form. The alias selected in the combo will be used as parent alias when opening the child form.
A child form can be invoked in two ways – by clicking on “More” button of the toolbar or by doubleclicking in the textboxes of the grid (the code in the DblClick Event of the textboxes is added by cChildgrid builder). It is possible to add more than one child form to one childgrid. In this case “On more” dialog is displayed, so the end used should select which child form to be opened. In the invoked “On more” dialog are listed only the child forms of the current childgrid. If there is only one child form, defined for the current childgrid, it is invoked without displaying the “On more” dialog. If the edit or list page of the form is active all the child forms for the form are displayed.
Sample – OneToManyPageFrame form in VFX100test.

VFX – Extended Parent/Child Builder

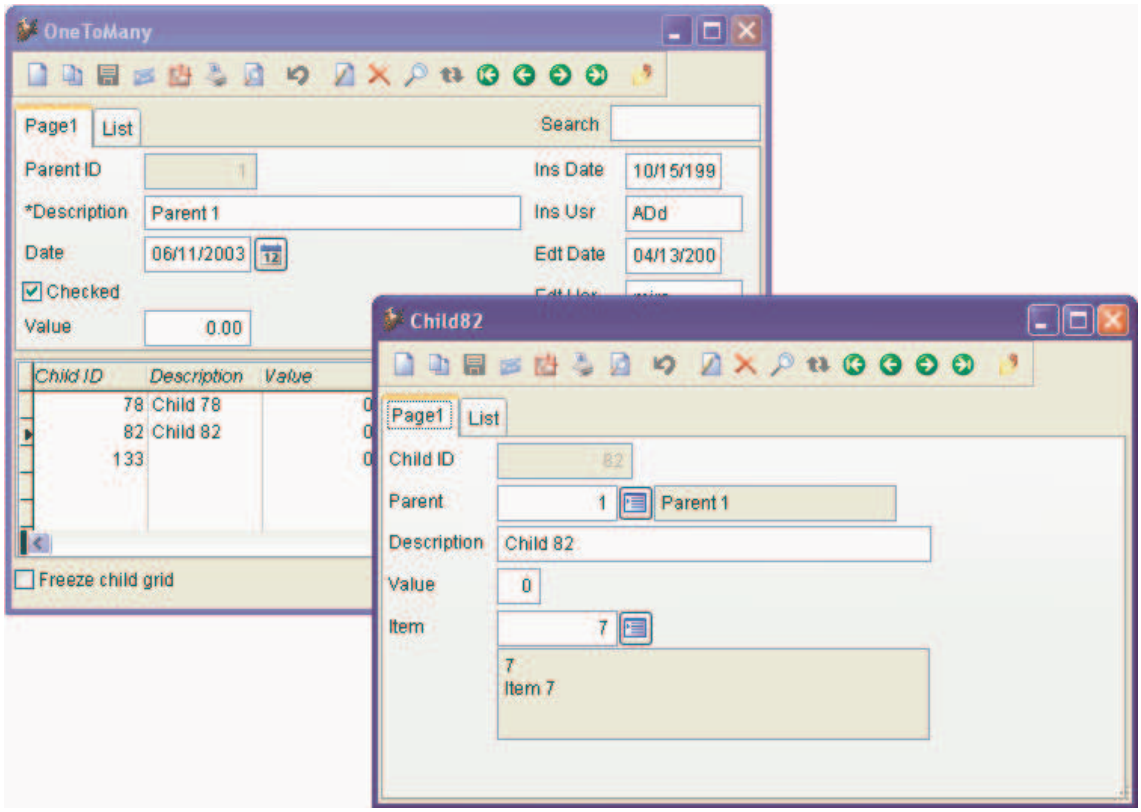
The evaluation of the expression in the textbox Record Position Expression is executed at the parent side, same as Child Form Caption Expression. So in these expressions can be used aliases and variables from Parent form. The expression must be in format that can be used in ECALUATE() function and character constants must be written in quotation marks.
The Text and Description for Open Form are also made as Expressions, evaluated at the parent side. This can be used for Localization.
Here, an example:



Expressions in Text for Open Form Expression and Description for Open Form Expression contain code to get localized strings and to use them.



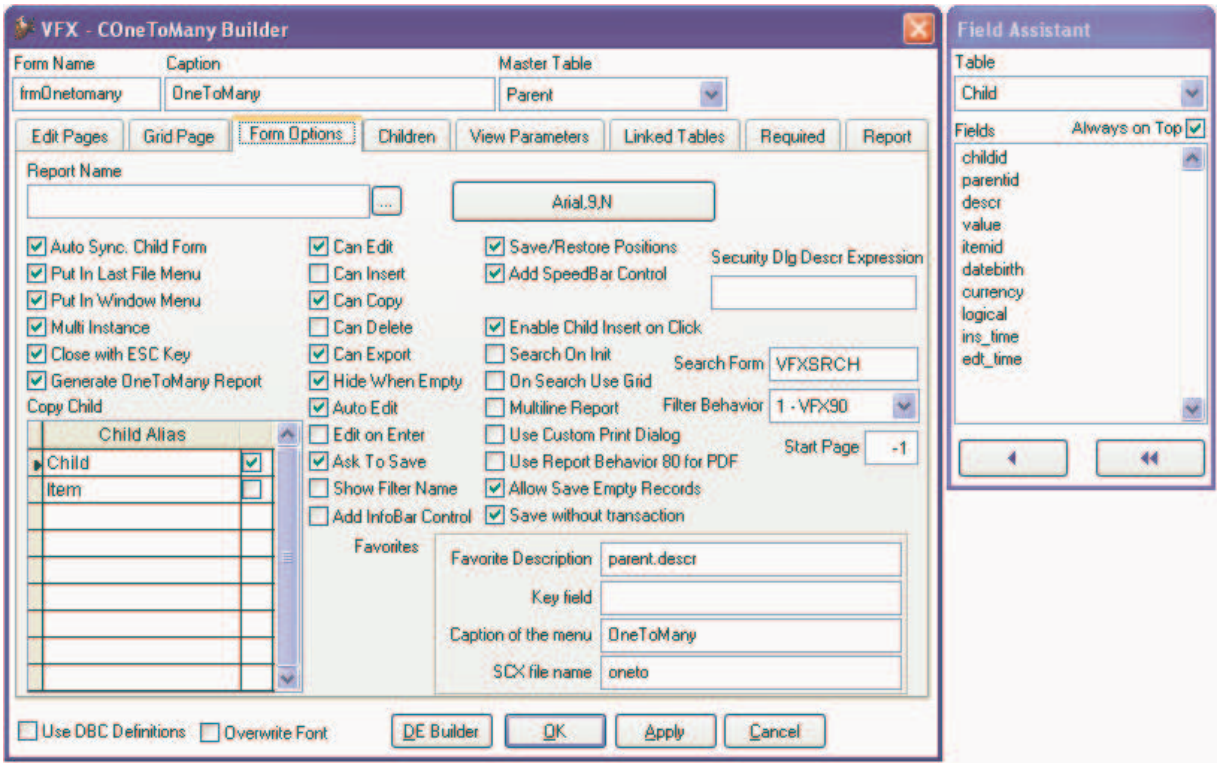
This example, you can find in VFX100Test. The parent form is OneTo.scx. The Child form will be opened in the record which is active in the child grid:



New features in cOneToMany form classes.

One-to-many reports

In forms based on the class cOnetomany there is a new property *lGenerateOneToManyReport*. This controls whether to generate onetomany reports. By default the property value is set to .F. (for backward compatiability with current behavior).
This is additionally controlled byt the Property *nGenerateOneToManyReport* of goProgram object This property controls the usage of OneToManyReport FRX for all OneToMany forms in the application. (0 – Use form setting, 1 – Force to .T., 2 – Force to .F.). The value of property *nGenerateOneToManyReport* of cFoxApp can be set from the Application Builder.
The value of property *lGenerateOneToManyReport* of OneToMany forms can be set in Form Builders (*VFX - COneToMany Builder*, *VFX - COneToManyPageFrame Builder* and *VFX - CTreeOneToMany Builder*), page Form Options, check box "*Generate OneToMany Report*".

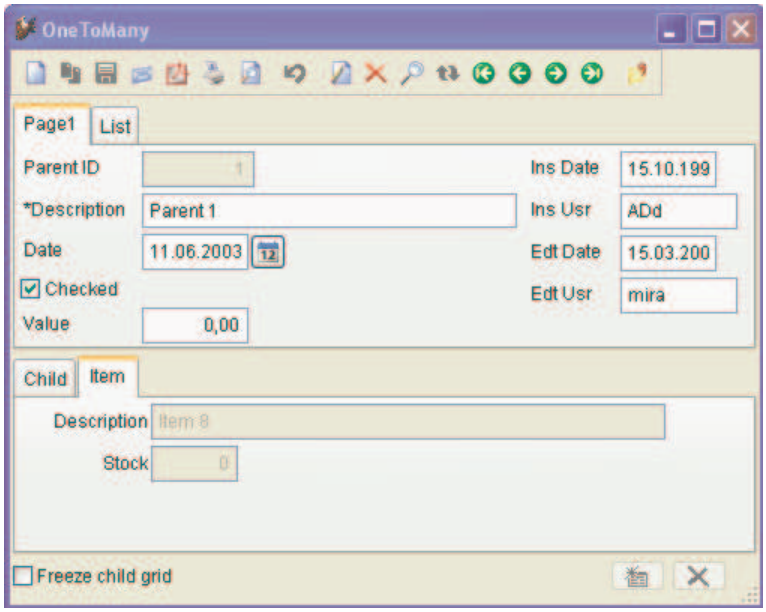


Child Edit Pages

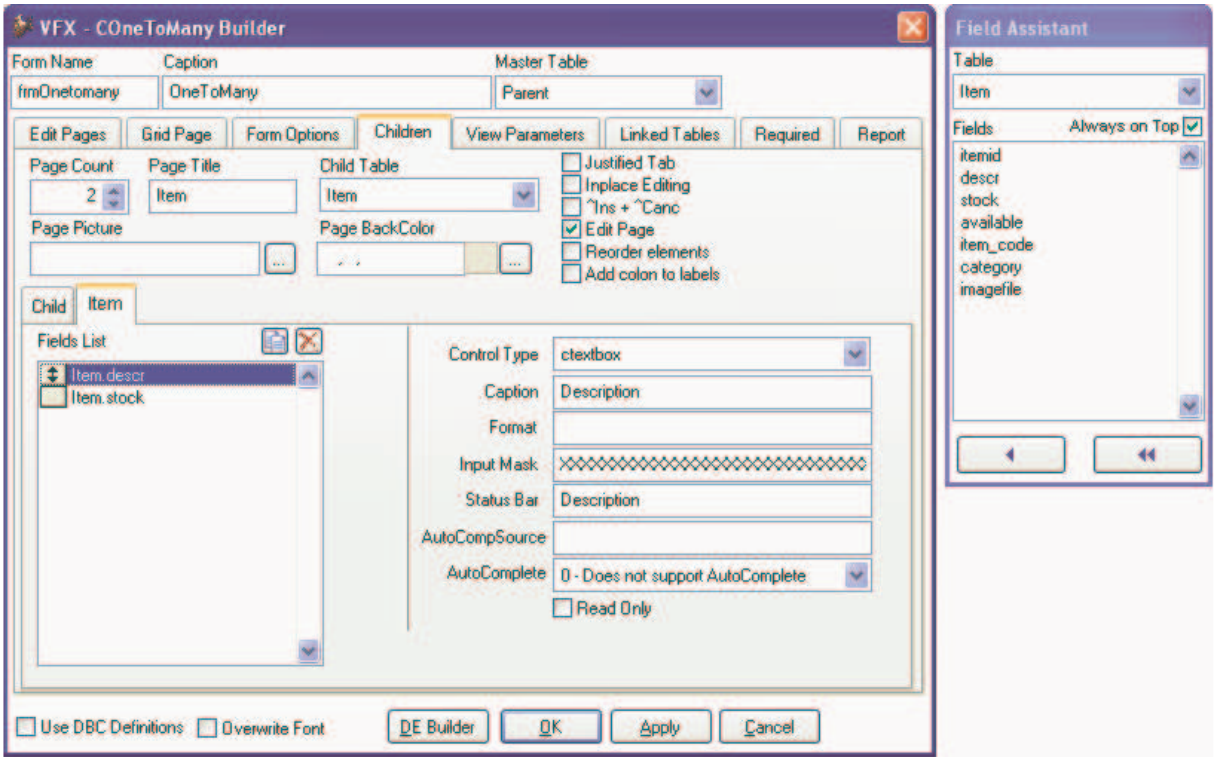
In RunTime for every edit page in the page frame - *pgfChildGrid* (for OneToMany form) and *pgfPageFrame* (for OneToManyPageFrame form) are added the properties *lCanUpdate* and *lEditing*.

lEditing – controls whether all controls on Edit page are editable. If this property is set to .F., all controls on the page are disabled.

lCanUpdate – controls whether on this page, child records can be added and deleted. If this property is set to .F., buttons New and Delete are disabled.



These properties can be keyed up in Form Builders (*VFX - COneToMany Builder*, *VFX - COneToManyPageFrame Builder* and *VFX - CTreeOneToMany Builder*), page *Children*, check boxes "*Inplace Editing*" and "*^Ins + ^Canc*".



The values set for these properties are saved and read from *cPagesSettings* property of the page frame.

Format of cPagesSettings property.

For each page in the page frame is written a separate row in *cPagesSettings* property of the PageFrame. This row has the following format:
<nPageType>_ <cAlias>;<lEditing>;<lCanUpdate>
where:

nPageType – is the type of the page: 0 – used for parent page; 1 – used for child edit page; 2 – used for child grid page.

cAlias – keep the working alias for the child pages. For parent pages it is empty.

lEditing – keep the value of lEditing property of child edit page. For child grid pages and parent pages it is empty.

lCanUpdate - keep the value of lCanUpdate property of child edit page. For child grid pages and parent pages it is empty.

Example:

2_ child;;

1_ item;.F.;.F.

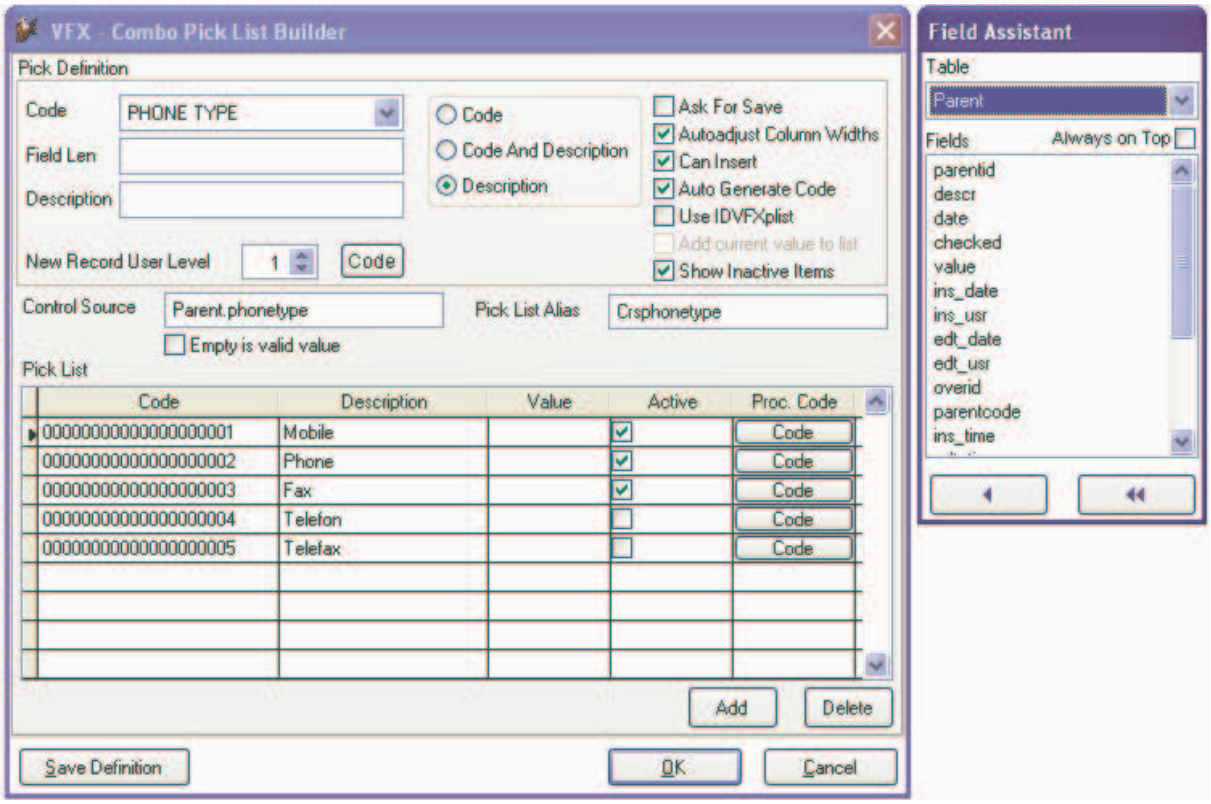
Class cComboPicklist

When the ControlSource of the combobox is not a required field of the form and if its Style is Dropdown List, a new option is available in context menu – Reset. When selecting this menu bar, the value of the control is set to be empty.

If the *cComboPicklist* is keyed up to show only Code, it is possible to set the class in runtime automatically to add current data value of the control source to the drop-down list value. This is done if current value is not found among pick list items. The processing is made in *FillItems* method, which is called by Requery method of *cComboPicklist* class. This can be managed by *'Add current value to list'* checkbox in the builder.

In the class cComboPicklist is added a new property *lAddCurrentValueToList* which controls whether the data value gets added to the pick list automatically.
When the class is set up in this way, in its Init event, the *cComboPicklist* is adding a reference to itself to a new array of *cDataFormVfxBase, aAdditionalControlsToRequery*. The *cDataFormVfxBase* form's mthod calls *Requery* method of *cComboPicklist* controls, refered by the array *aAdditionalControlsToRequery*.

The other new feature of *cComboPicklist* is the possibility to control displaying the items that are marked as inactive. This can be managed by '*Show Inactive Items'* checkbox. If it is not marked, the inactive items are not displayed in the drop-doen list of the combobox. When it is marked, The inactive items are diaplyed in drop-down list as disabled.
The new property of the class *lShowInactiveItems* controls whether inactive values should be shown.



New properties of the class:

lAddCurrentValueToList – If .T., the current value will be added to combobox if not found. Allowed only when code is shown.

lShowInactiveItems – If .T. inactive items are listed in combobox drop-down list.

nColumnCnt – Internally used. Contains value of ColumnCnt field from vfxPDef table.

lIDVfxPlist – Internally used. If set to .T., the table vfxplist contains the Autoinc field idvfxplist and this field is used as BoundColumn for the combobox.

Methods:

RightClick – creates a context popup menu with the options – Cut, Copy, Paste, and Reset. Reset bar is awailable, if the ControlSource of the combobox is not a required field of the form and if its Style is Dropdown List.
ResetValue – replaces the ControlSource of the class with an empty value. Called by the context menu bar Reset.

Class cFooterBar

cFooterBar is a new class in vfxCtrl.vcx. It's similar to cInfoBar container.
The class cFooterBar is designated to show a set of important information to end-users at bottom part of the data form. The developers can place this container to the forms and place there controls which show iformation to users.
In one form it's possible to have more than one FooterBar containers. While resize this containers rest always on bottom of the form and only there width is resized.
For resize functionality a new property in cDataFormVfxbase class is added:
oFooterControl – Keeps a reference to topmost FooterBar container in the form.

It's recommended the controls in this containers to be set as no resizable (Comment = "<NORESIZE>").

As well, this container is added in aAdditionalControls array of the form and is refreshed on Form's Refresh().

Class cMapPoint

The class is designated to encapsulate using of Microsoft Map Point application.

Properties

nUnits – An integer representing distance units. 0 - miles; 1 – kilometers.
oMapPoint – Used internally. Keeps a reference to a MapPoint object.
oMap – Used internally. Keeps a reference to a Map object. Used for finding addresses.
oRoute – Used internally. Keeps a reference to a Route object used for calculating route distances between addresses.

Methods

OpenMapPoint(tlForceNewInstance, tnUnits) – Opens MapPoint application. Creates a new Map object and stores reference to it in oMap property.

tlForceNewInstance – .T. – always create a new instance of the OLE class, .F. – Use running application, if one exists;
tnUnits - distance units. 0 - miles; 1 – kilometers.

FindAddress(tcStreet, tcCity, tcOtherCity, tcRegion, tcPostalCode, tvCountry) – Searches for an address by specified parameters. Finds a collection of locations, that are possible matches to an address. It is required at least one parameter to be passed. If a parameter is not supported for an address—for example, Region for an address in Germany - the parameter is ignored. Returns a FindResults collection.

tcStreet – The street name or street address to find
tcCity – The name of the city

tcOtherCity – The name of the second city to find. Valid only for addresses in the United Kingdom.
tcRegion - The name of the region to find
tcPostalCode - The postcode, postal code, or ZIP Code to find.
tvCountry - The country to find. List country codes can be finded at: http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/mappoint2004/BIZOMVGeoCountry.asp

Usage Example:

```
loMapPoint = NEWOBJECT("cMapPoint")
loMapPoint.OpenMapPoint(.T., 1)
loResults = loMapPoint.FindAddress(lcStreet,lcTown,,,lcZIP, GeoCountryCode)
lnGeoQuality = loResults.ResultsQuality
loLocation = loResults.Item(1)
lnGeoLong = loLocation.Longitude
lnGeoLat = loLocation.Latitude
```

GetLocation(tnLatitude, tnLongitude) – Returns a Location object for given latitude and longitude coordinates, passed as parameters. The Location object can be a place, an address, or latitude and longitude coordinates.

RouteAddWayPoint(tnLatitude, tnLongitude, toWayPoint) – Adds a Way point (WayPoint object) to the current route by:

1. Latitude (tnLatitude) and longitude (tnLongitude) coordinate as parameters.
2. a Location object (toWayPoint) as parameter;
   The location object can be retrieved by:
   - Calling GetLocation method
   - Calling Findaddress method. It is a member (Item) of FindResults collection.

Usage Example:

```
loMapPoint = NEWOBJECT("cMapPoint")
loMapPoint.OpenMapPoint(.T., 1)
loResults = loMapPoint.FindAddress(lcStreet,lcTown,,,lcZIP, GeoCountryCode)
loLocation = loResults.Item(1)
loMapPoint.RouteAddWayPoint(loLocation)
```

RouteClear() – Clears way-points from current route.

RouteMoveWayPoint(tnItemNumber, toAnchor) – Moves a way point.
tnItemNumber – way point item number
toAnchor – a location object (new location)

RouteCalculate() – Calculates a route based on two or more waypoints. Minimum two way points are needed to be added first. Returns reference to the Route object.
After executing RouteCalculate method, properties Distance and DrivingTime of route object are available.

Usage Example:

```
loRoute = This.oMapPoint.RouteCalculate()
lnWayDistance = loRoute.Distance     && in distance units (km. or mile)
lnWayTime = loRoute.DrivingTime * 24   && in hours
```

CloseMapPoint() – Releases MapPoint application instance.