VFX 9.5 – What is new?

March 2006

Uwe Habermann, Venelina Jordanova

Table of content

New features for developers

Inheritance architecture

Vfxformbase.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.

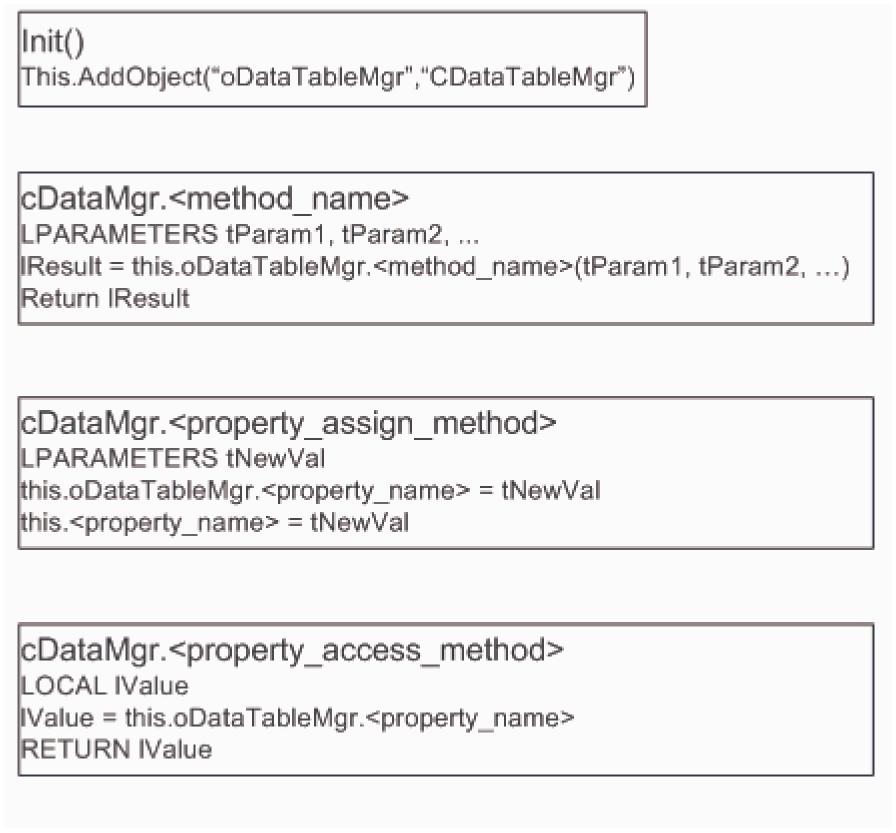
There is an additional inheritance layer in VFX 9.5. The VFX form classes, available in previous VFX versions in *Vfxform.vcx* class library, are now placed in *Vfxformbase.vcx* class library. To the previous class names is appended a suffix *vfxbase*.

For every class from this first-level library there is a 1:1 inherited class in the class library *Vfxform.vcx*. Now the class library *Vfxform.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 forms in the whole application based on this class.

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

Class cDataMgr (vfxappl.vcx) – base class Container

cDataMgr class is an intermediate layer designated to provide a middle tier between cDataTableMgr class and forms in the application.
The class has the same methods and properties as cDataTableMgr (vfxctrl.vcx). The same parameters are passed to the methods as in cDataTableMgr. The properties have assign and access methods.



```
Init()
This.AddObject("oDataTableMgr","CDataTableMgr")
```

```
cDataMgr.<method_name>
LPARAMETERS tParam1, tParam2, ...
lResult = this.oDataTableMgr.<method_name>(tParam1, tParam2, …)
Return lResult
```

```
cDataMgr.<property_assign_method>
LPARAMETERS tNewVal
this.oDataTableMgr.<property_name> = tNewVal
this.<property_name> = tNewVal
```

```
cDataMgr.<property_access_method>
LOCAL lValue
lValue = this.oDataTableMgr.<property_name>
RETURN lValue
```

In cDataMgr.Init() an object of class cDataTableMgr (`oDataTableMgr`) is added to it's properties.
When a method is called it forwards the parameters passed to it to the corresponding method from oDataTableMgr (the method with the same name). The value returned from oDataTableMgr is given back to the calling method.

Example:
cDataMgr.Method1()
```
   LPARAMETERS Param1, Param2, …
   lResult = This.oDataTableMgr.Method1(Param1, Param2, …)
   RETURN lResult
```

When working with properties Access() and Assign() methods are used to pass the property values between cDataMgr and oDataTableMgr. On Assign() the value is assigned to *oDataTableMgr.PropertyName* and *cDataMgr.PropertyName*. On Access() the property value is read from *oDataTableMgr.PropertyName* and returned as required.

Example:
cDataMgr.Property1_assign()
```
   LPARAMETERS vNewVal
   This.oDataTableMgr.Property1 = vNewVal
   This.Property1 = vNewVal
```

cDataMgr.Property1_access()
```
   RETURN This.oDataTableMgr.Property1
```

Properties:
aCargo – array
aLinkedtable
cAliasName
cIDfieldname

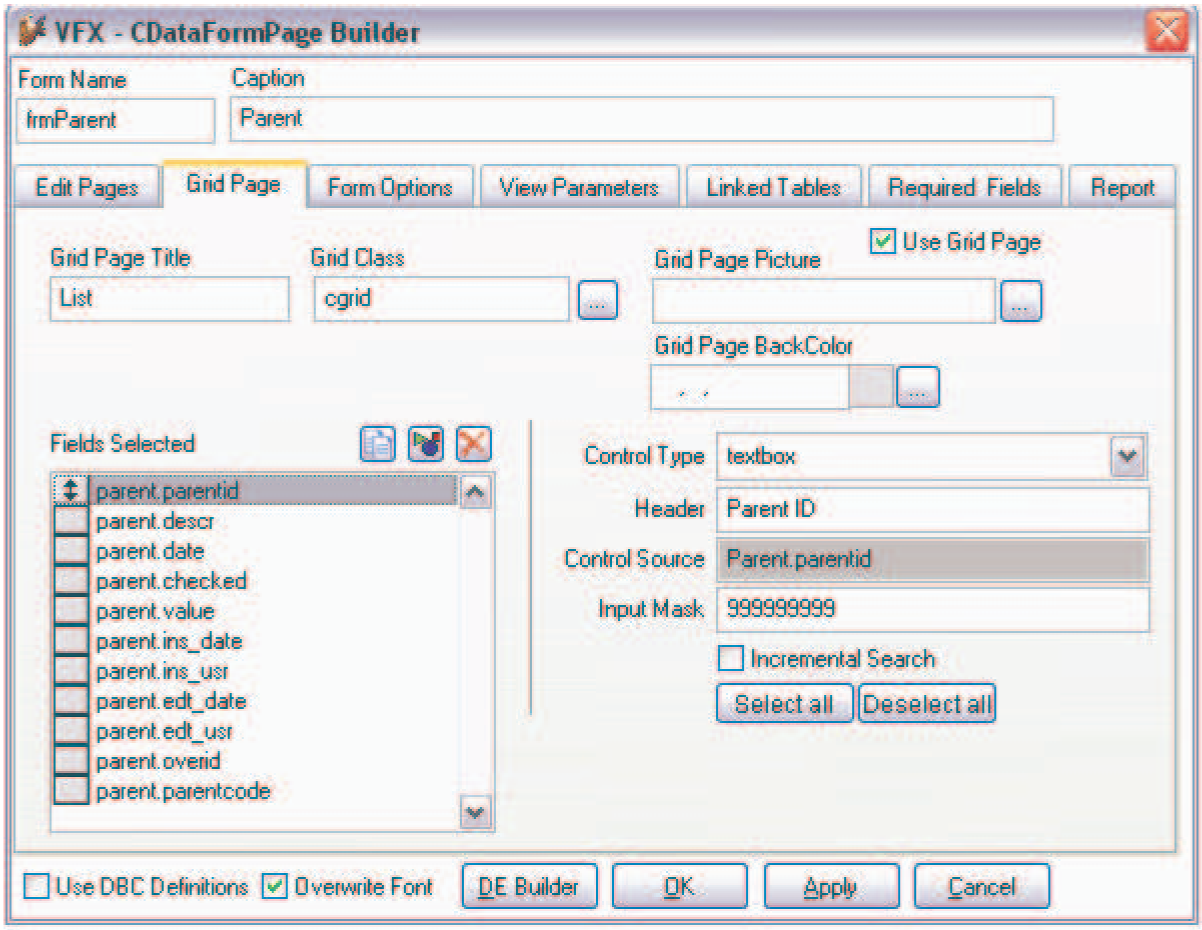
cLinkedTables
cValidchildpositionexpr – Contains an expression evaluated to check for valid child position. Empty if alias is not a part of relation
Extrabuffer – An user buffer
nLinkedtable
Setfilter

Methods:
About()
CheckForValidChildPosition() – Check if current child row relates to current row in cWorkAlias
Delete() – Delete the current record
GoBottom() – Move the record pointer to Bottom Position
GoTop() – Move the record pointer to Top Position
Insert()
LogError() – Saves error information in vfxlog.dbf
MakeValidChildPositionExpr() – Constructs an expression. Evaluated to check for valid child position depending on relation to parent alias
Modified() – Return .T. if record is modified
Next() - Move the record pointer to Next Position
Previous() – Move the record pointer to Previous Position
Release() – Release this object
Requery() – Refresh the current DynaSet
Save() – Save current DynaSet
SynchronizeLinkedTables()
Undo() – Rollback
Update() – Used to make a TABLEUPDATE()
UpdateForeignKey() – Updates ForeignKey Fields

Builders

Form builders

On grid page in all form builders, the developers can fast and easy change incremental search settings for all columns in the grid. This can be done using the two new buttons on grid page in builders *Select all* and *Deselect all* under the *Incremental search* checkbox.



Form Name: frmParent
Caption: Parent

Grid Page Title: List
Grid Class: cgrid
Grid Page Picture:
[✓] Use Grid Page
Grid Page BackColor:

Fields Selected:
- parent.parentid
- parent.descr
- parent.date
- parent.checked
- parent.value
- parent.ins_date
- parent.ins_usr
- parent.edt_date
- parent.edt_usr
- parent.overid
- parent.parentcode

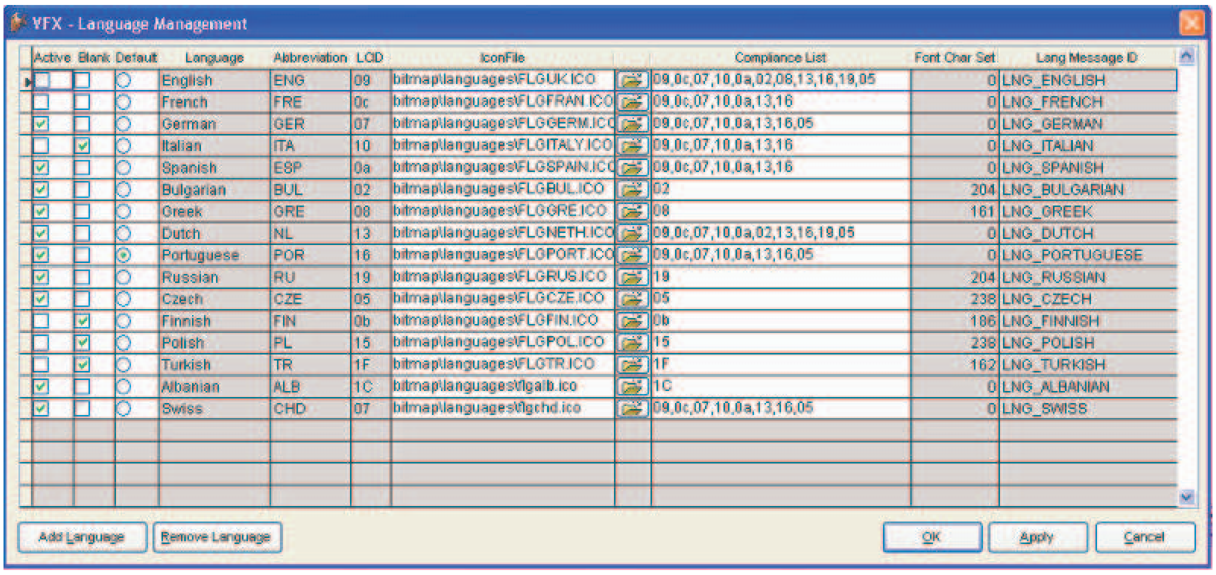
Control Type: textbox
Header: Parent ID
Control Source: Parent.parentid
Input Mask: 999999999
[ ] Incremental Search

[ ] Use DBC Definitions
[✓] Overwrite Font

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.It is also possible column content for VFX predefined languages to be blanked in case the application does not use these languages or they can just be set to Inactive. After the language is once *blanked*, it can be filled again by unchecking the *Blank* checkbox. This will read the message texts from VFX template Vvfxmsg.dbf and will fill the blank fields. Here, in the dialog, developer has also a chance to change default language for the application.


VFX - Language Management

Active	Blank	Default	Language	Abbreviation	LCID	IconFile		Compliance List	Font Char Set	Lang Message ID
		○	English	ENG	09	bitmap\languages\FLGUK.ICO		09,0c,07,10,0a,02,08,13,16,19,05	0	LNG_ENGLISH
		○	French	FRE	0c	bitmap\languages\FLGFRAN.ICO		09,0c,07,10,0a,13,16	0	LNG_FRENCH
✓		○	German	GER	07	bitmap\languages\FLGGERM.IC		09,0c,07,10,0a,13,16,05	0	LNG_GERMAN
	✓	○	Italian	ITA	10	bitmap\languages\FLGITALY.ICO		09,0c,07,10,0a,13,16	0	LNG_ITALIAN
✓		○	Spanish	ESP	0a	bitmap\languages\FLGSPAIN.IC		09,0c,07,10,0a,13,16	0	LNG_SPANISH
✓		○	Bulgarian	BUL	02	bitmap\languages\FLGBUL.ICO		02	204	LNG_BULGARIAN
✓		○	Greek	GRE	08	bitmap\languages\FLGGRE.ICO		08	161	LNG_GREEK
✓		○	Dutch	NL	13	bitmap\languages\FLGNETH.ICO		09,0c,07,10,0a,02,13,16,19,05	0	LNG_DUTCH
✓		●	Portuguese	POR	16	bitmap\languages\FLGPORT.ICO		09,0c,07,10,0a,13,16,05	0	LNG_PORTUGUESE
✓		○	Russian	RU	19	bitmap\languages\FLGRUS.ICO		19	204	LNG_RUSSIAN
✓		○	Czech	CZE	05	bitmap\languages\FLGCZE.ICO		05	238	LNG_CZECH
	✓	○	Finnish	FIN	0b	bitmap\languages\FLGFIN.ICO		0b	186	LNG_FINNISH
	✓	○	Polish	PL	15	bitmap\languages\FLGPOL.ICO		15	238	LNG_POLISH
	✓	○	Turkish	TR	1F	bitmap\languages\FLGTR.ICO		1F	162	LNG_TURKISH
✓		○	Albanian	ALB	1C	bitmap\languages\flgalb.ico		1C	0	LNG_ALBANIAN
✓		○	Swiss	CHD	07	bitmap\languages\flgchd.ico		09,0c,07,10,0a,13,16,05	0	LNG_SWISS

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.

VFX – Help Wizard

The VFX – Help Wizard automatically fills out unique *HelpContextIDs* in all controls of a project when the button *Set HelpContextIDs* is pressed.
If any doubled HelpContextIDs are found after the wizard finishes its work,they are stored in the file *VFXHelpDoubledEntries.txt* in the current project's folder and this file is open in IDE.Using it programmer as the chance to clear doubles *HelpContextIDs.*

Vfxfopen table

Two new columns are added to the Vfxfopen table:

VFXFOpen-Field	*Description*	*Example*
IconFile	File path to the icon displayed next to the object	Bitmap\customers.ico
NotVisible	If True the object will not be visible in the Open/XPOpen dialog	T

New properties of the Application object

cRequiredFieldInitProps – Default value used if a form's cRequiredFieldInitProps is empty. Stores the required fields' form controls' initial properties. Commonly properties like BackColor or FontColor are used to distinguish these controls on a form.

cRequiredFieldFailureProps – Default value used if a form's cRequiredFieldFailureProps is empty. Stores the required fields' form controls' failure properties. Commonly properties like BackColor or FontColor are used to distinguish these controls on a form when a wrong value is entered.

cSecurityTablesList – Contains a semicolon delimited list of all security tables, which should be updated, when a user is deleted or username is changed

lCloseReportDialogOnExecution – If this is .T. report dialog is closed when report is generated or export is finished

nShowFilterName - Defines whether the name of the current active filter will be shown, if the filter is saved. 0 - Use form settings; 1 - Force to .T.; 2 - Force to .F.. This value is taken into account only if ThisForm.nFilterBehaviour = 2 - VFX95.

lActivateThemes – Controls if themes should be used or not. This is taken into account only when user customization is not allowed, otherwise the property of goUser is used

nSaveWithoutTransaction - Save record without transaction (only for cOneToMany): 0 - Use form's setting, 1 - Save without transaction, 2- Do not save without transaction

nUseMemoForm - Use cMemoForm object to edit memo field content: 0 - Use control setting; 1 - Always use cMEmoForm; 2 - Do not use cMemoForm

nIPaddressesListAllowed – Defines the meaning of the list with IP addresses in vfxipaddresses.dbf : 0 - Not used, 1 - list of IP addresses allowed to login, 2 - list of IP addresses NOT allowed to login

cMainMenu - Main menu file name including extension.

cTableManagerClass - Name of the class providing data access

lAutoHideXPOpen – Controls the autohide function of XP open dialog. .T. – Auto hide ON, .F. – Auto hide OFF

nXPDialogAutoHideInterval - The interval in seconds after which the XP dialog autihides itself (viable if AutoHide is enabled)

Send error report to a WebService

End users have the chance to send error report to you right at the moment, the error has been raised.

What settings do you need to set in your application for this:

Application settings:
The following properties of *goProgram* should be set in order for the error report (ER) to be sent to an error registration web service (WS).

cWSDL – WSDL link to connect to
 (Ex.: http://register.something.com/regservice.wsdl)
cServiceName – Name of the WS
 (Default: vfxregservice)
cRemoteMethod – Name of WS method to be called to receive the ER
 (Default: ReceiveErrorInfo)

These properties are keyed up in the *Error Handling* section of VFX Application Builder.
The ER is sent in the form of XML, which has the following fields:

FirstName – User's reginfo
LastName – User's reginfo
Company – User's reginfo
Email – User's reginfo
RegKey – User's reginfo
Version – Application's version taken from the EXE file.
Producer – Name of application producer taken from goProgram.cCompanyName (should not be empty)
AppName – Name of application taken from goProgram.cAppName (should not be empty)
ErrorInfo – Error report

The whole operation of sending the ER to the WS is done by the form's method SendToWS().

If sending the error report with WS fails the error report is sent via e-mail

Server side settings:
All the ERs received for an application are stored in the table *ReceivedErrors* that is part of the application's *RegData* database. For every application that the WS is receiving ERs from there should be a separate line in config.vfx. The content of config.vfx file is same as for standard VFX application. It is very important to fill right value in the field *ClientName*. In *ClientName* field should be eneteerd the name of the application the ER is received from (should be exactly the same as in application's goProgram.cAppName). The application name received with the ER is used to locate the corresponding *RegData* database and connect to it.

The *ReceivedErrors* table has the following structure:

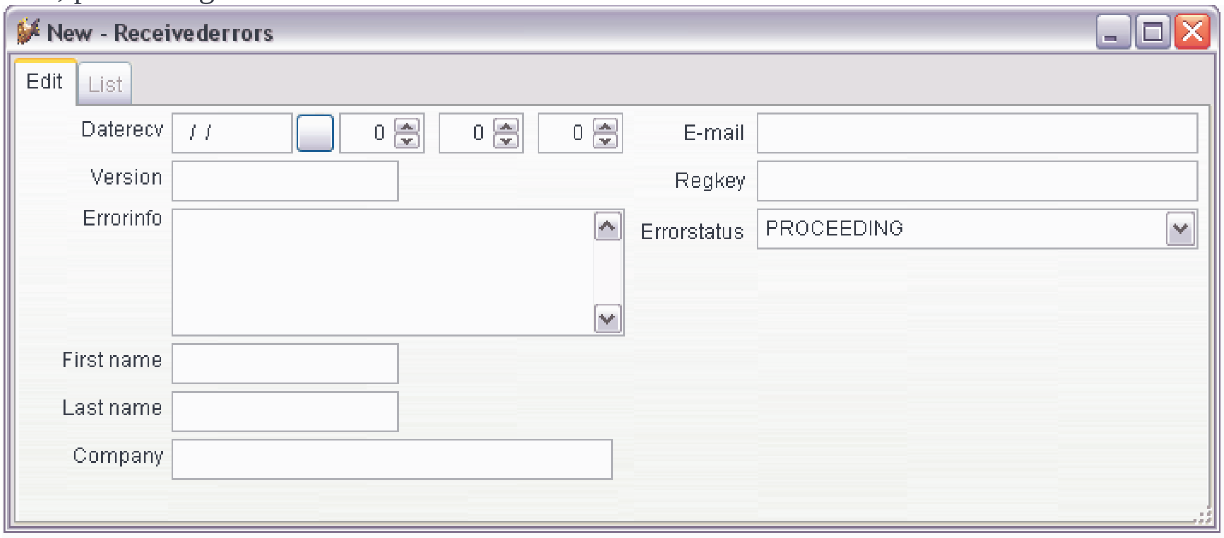
ErrID [I(Autoinc)]
DateRecv[DateTime] – Date and time of receiving of the ER
ErrorInfo[Memo] – Error report
FirstName[C(15)] – User's reginfo
LastName[C(15)] – User's reginfo
Company[C(30)] – User's reginfo
Email[C(30)] – User's reginfo
RegKey[C(30)] – User's reginfo
Version[C(15)] – Application's version taken from EXE file
Producer[C(20)] – Application producer name
Application[C(20)] – Application name

The server response to a received ER consists of a XML with two fields:
nStatus
cText

If the error was logged successfully in the database nStatus is 0. If an error occurred nStatus is greater than 0 and cText contains the error description.

VFX – Customer Management

VFX – Customer management application includes a form used to maintain the errors arose in your application. *Receivederrors* form works with the table where error reports received by the web service are stored. This form allows you to maintain status of the received errors and also if necessary to enter new error reports by hand. The information that your application send to the web service includes date and time when the reported error arrose, application version, information about the error, name, family, organization, where the error is registered, e-mail for reference, registration key of the customer. For every record, the error status can be selected among the list of: received, read, proceeding or fixed.



Errorstatus: PROCEEDING

New properties for end-users

Users administration

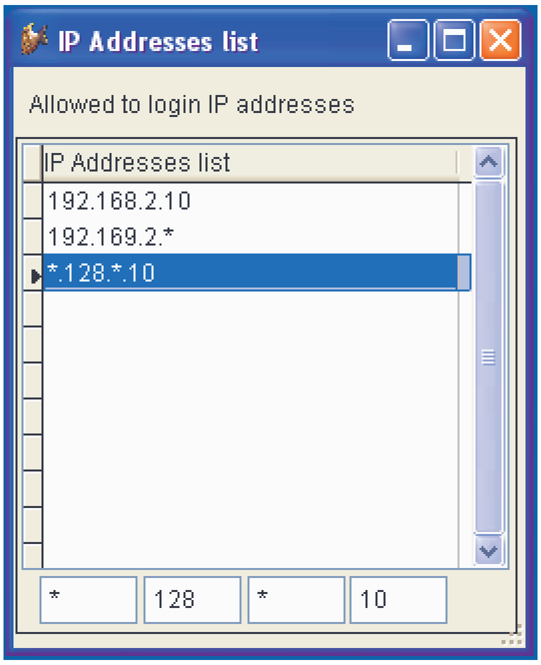
List of IP addresses allowed/not allowed to login

The administrator users have now the chance to define a list of IP addresses which the application is allowed/not allowed to be started from. This feature is controlled by the property goProgram.nIPaddressesListAllowed. The list is stored in the file DATA\vfxipaddresses.dbf . The meaning of the IP addresses in it is defined by the value of **nIPaddressesListAllowed.**

goProgram.nIPaddressesListAllowed

0 – Feature is not used
1 – Used. List of IP addresses which the application is ALLOWED to start from
2 – Used. List of IP addresses which the application is NOT ALLOWED to start from

This restriction is not applied in cases when the application is started by an administrator user.
This list of IP addresses should be defined by an administrator, using the **IP Addresses list** form (Tools -> Login IP addresses)



IP Addresses list
192.168.2.10
192.169.2.*
.128..10

*	128	*	10

The IP address entered can be either a full IP address like 192.168.2.10, or a mask containing metachar (*) for any part of the IP address like *.128.*.10 or 192.168.*.*. If a mask is entered, when checking the IP address(es) of the current machine the parts of the IP address provided are compared against the corresponding parts in every entry in the list.

If a user logging in has administrative rights access is granted otherwise the IP address(es) of the current machine is(are) examined. If the list contains allowed IPs and any of the IP addresses assigned to the current computer are in this list or matches the conditions set by the IP mask, the user is granted access. If the list contains IPs not allowed and any of the IPs of the current machine are in the list or matches the conditions set by the IP mask, the user is denied access. If access is denied a message box is displayed.

If the feature is enabled and the list is empty no check of IPs is made and all users are granted access

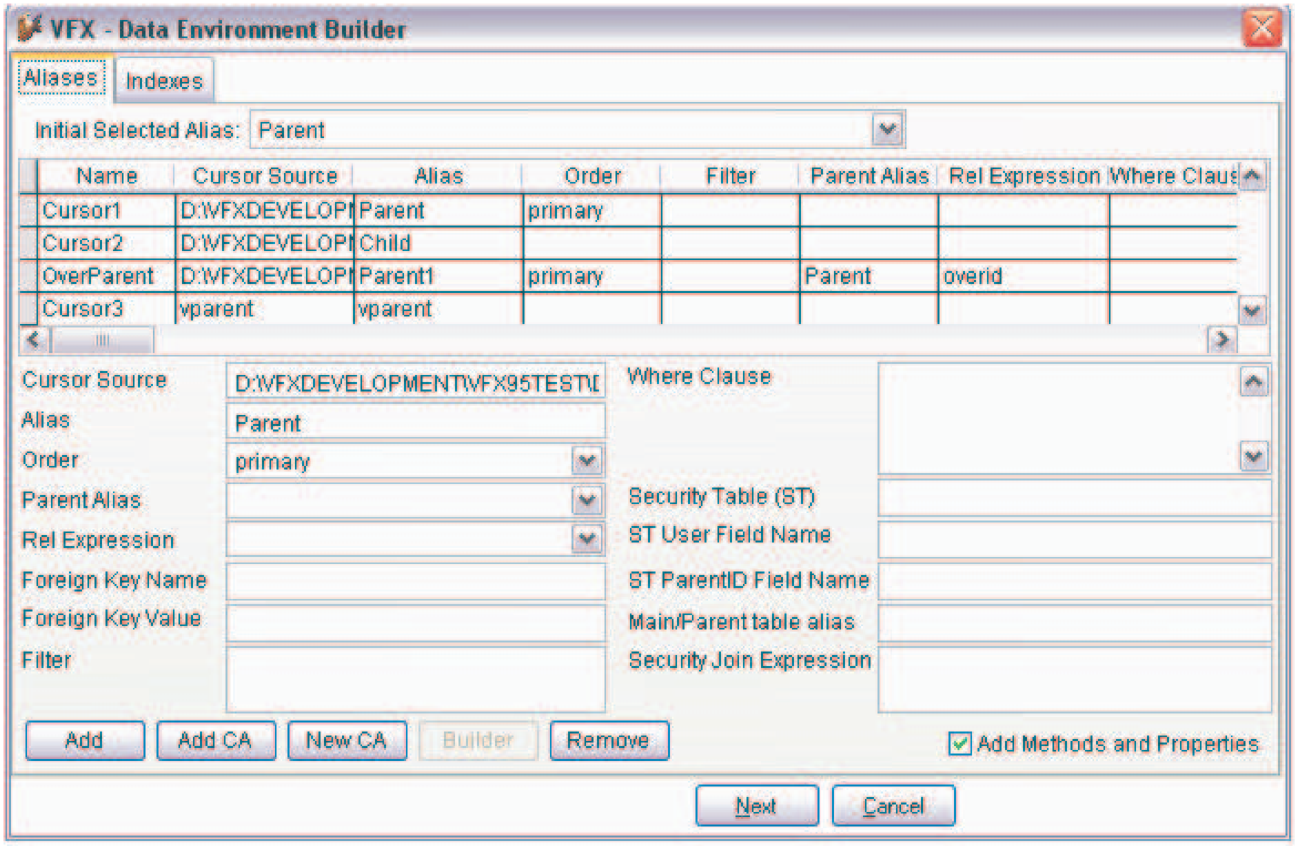
Security rights management

This new security feature allows applying view restrictions for users on “per record” base. For every record in the table can be defined a list of users allowed to view and edit it.

New properties added to cBaseDataAccess class:

cSecurityTable – Name of Security table
cSecurityUserFieldName – Name of field in Security Table for user name
cSecurityParentIDFieldName - Name of ID field in Security Table and main table
cParentAlias – Alias of main table or parent(when cSecurityJoinExpression is used). See the example below.
cSecurityJoinExpression – Additional join expression to the security table. Used when tables is child table. See the example below.

These properties can be keyed up for *Cursoradapter* object of the Dataenvironment using
VFX - Data Environment Builder



Initial Selected Alias: Parent

Name	Cursor Source	Alias	Order	Filter	Parent Alias	Rel Expression	Where Claus
Cursor1	D:\VFXDEVELOP	Parent	primary				
Cursor2	D:\VFXDEVELOP	Child					
OverParent	D:\VFXDEVELOP	Parent1	primary		Parent	overid	
Cursor3	vparent	vparent					

Cursor Source	D:\VFXDEVELOPMENT\VFX95TEST\
Alias	Parent
Order	primary
Parent Alias	
Rel Expression	
Foreign Key Name	
Foreign Key Value	
Filter	
Where Clause	
Security Table (ST)	
ST User Field Name	
ST ParentID Field Name	
Main/Parent table alias	
Security Join Expression	

For every table used in an application a security relations table containing two fields – Name of ID field and User is created that contains the records “allowed” for viewing by a user. Name of ID field in the security table should be the same as the name of the ID field in the table for which the security table is created.

When the table content is queried, the CursorAdapter class builds additional JOIN between the table and its security table on ID field and considering User so that only the records allowed for the current user are retrieved. The Security Join Expression property is supplied for additional security considerations or specific conditions when joining a child table through its parent table toward security table.

When a record is inserted in a main (parent) table, corresponding security record is inserted into the security table. When a record is deleted, all corresponding records in security table are deleted too. For child tables it is not necessary to manage security records, as far as child tables do not have their own security table. How to know if the alias that a record is deleted from is parent or child? For this purpose is used again the property *cSecurityJoinExpression* – it is supposed that is this property is empty, current alias is parent table. If there is a join expression in the property *cSecurityJoinExpression*, then the alias is considered as child table and the security relations table is not managed.

Example
It is required to provide security management for OrderDetails table which is child table for Orders table. The relation Orders-OrderDetails is on orderid field. The SelectCmd of CA, generated with CA Wizard is "Select * from OrderDetails"

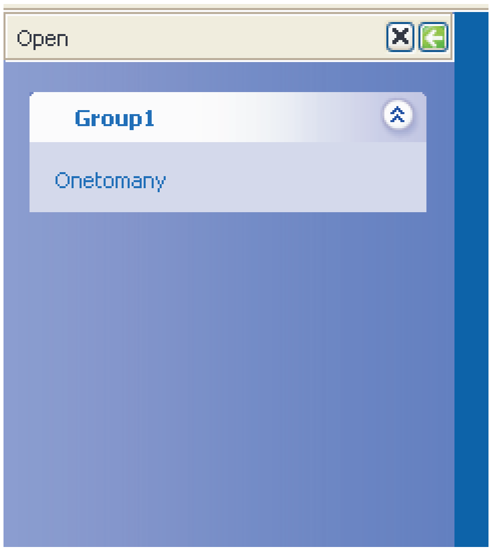
Properties of CA object (same colors are used in the command shown below):
cSecurityTable – "**OrderSecurity**"
cSecurityUserFieldName – "**UserName**"
cSecurityParentIDFieldName – "**OrderId**"
cSecurityJoinExpression – "**JOIN Orders ord on OrderDetails.OrderID= ord.OrderID**"
cParentAlias – **ord**

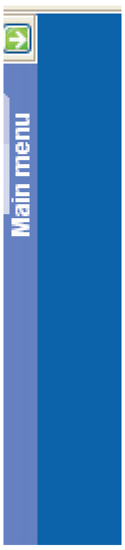
The resultant SelectCmd will be
```
   Select * from OrderDetails JOIN Orders ord on OrderDetails.OrderID=
   ord.OrderID
   JOIN OrderSecurity SecurityTable on SecurityTable.OrderId = ord.OrderId AND
   UserName = ?goUser.User
```
The colored syntax is used to make view of how the properties' values are used at run time in the application.

End-users interface enhancements

Auto Hide XP Open Dialog

With this feature the XP Open Dialog can be set to auto hide itself after an interval of inactivity over it by the user. When the feature is enabled, a green button with a white arrow is displayed in the XP Open Dialog, which indicates that the dialog will and can autohide.





The XP Open Dialog can be hidden manualy by pressing the button with the white arrow next to the close button in the top right corner. To show the dialog again the mouse pointer should be moved over the visible part of the dialog.
The Autohide behavior is controlled by a Timer object placed on the XP Open Dialog. It controlls when the dialog will slided out of view. The interval of inactivity, after which the dialog hides automatically, is controlled by the property *goProgram.nXPDialogAutohideInterval*. By default its value is 15 sec and can be keyed up in *VFX – Application builder*.

There are two ways of enabling/disabling the autohide feature.

1. ***User customization is ON***
   The feature can be turned on through the Customize dialog (Tools->Customize…, available if customization is enabled for this user). Every user can change his/her setting at any time. For the setting to take effect the program must be restarted.
   Controlling property:
   goUser.AutoHideXPOpen
   0 – ON
   1 – OFF
2. ***User customization is OFF***
   The feature is controlled by a property of goProgram, which is set at design time and cannot be changed by the user at runtime.
   Controlling property:
   goProgram.lAutoHideXPOpen
   .T. – ON
   .F. – OFF

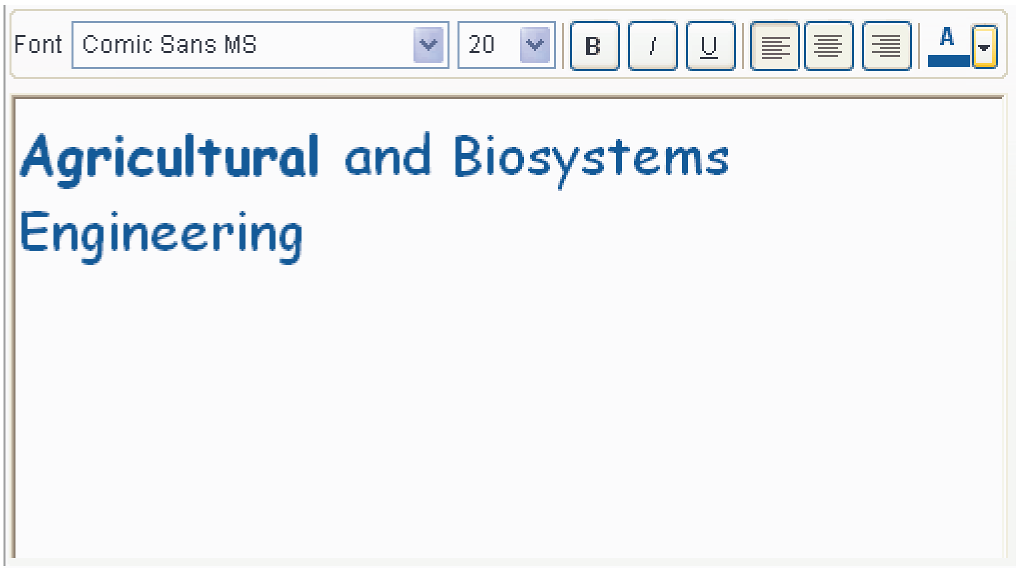
Enhanced color depth support

If the color depth of the display is less than or equal to 256 colors, the XP Style open dialog will automatically switch to using the bitmaps provided with VFX more suitable for such colordepths. The system color depth is retrieved during application startup and stored in the property *goProgram.nColorDepth* in the form of a positive integer (Ex.: nColorDepth = 32 for 32-bit color). If this value is less than or equal to 8 the XP Style open dialog will automatically use the files *blue16.bmp*, *tabclose16.bmp* and *tabopen16.bmp* instead of *blue.bmp*, *tabclose.bmp* and *tabopen.bmp* from the project's bitmap folder.

RTF Support

***cRTFControl* (vfxCtrl.vcx)**

This control can be added from any form builder for fields of *Memo* or *General* data type.



Agricultural and Biosystems Engineering

This new vfx control enables edition of RTF texts. It has its own toolbar for easy text formatting. Toolbar contains controls for selecting font, size and type of chanracters (bold, italic, underline), text alignment and text color.
This control is designated to edit data contained in fields of *Memo* and *General* data type.
RTF data can be print in reports or save as BMP and RTF files. It also can be sent with an e-mail as BMP and RTF attachment.

It’s recommended in *onPrint* method of the form to have similar code:

```
PARAMETERS tlpreview

LOCAL lcMessageText
IF Type("tlpreview") == "N" AND INLIST(tlpreview, 1,2,6,7,11,12)  &&1 - email
with PDF,
&&11 - email with HTML,
&&12 - email with XML,
&&2 - save as PDF,
&&6 - save as HTML,
&&7 - save as XML
MESSAGEBOX("RTF is not supported in this format!", 0+48, ThisForm.Caption)
   RETURN .F.
ELSE
   DODEFAULT(tlpreview)
ENDIF
```

Adding the RTF Control to a Report

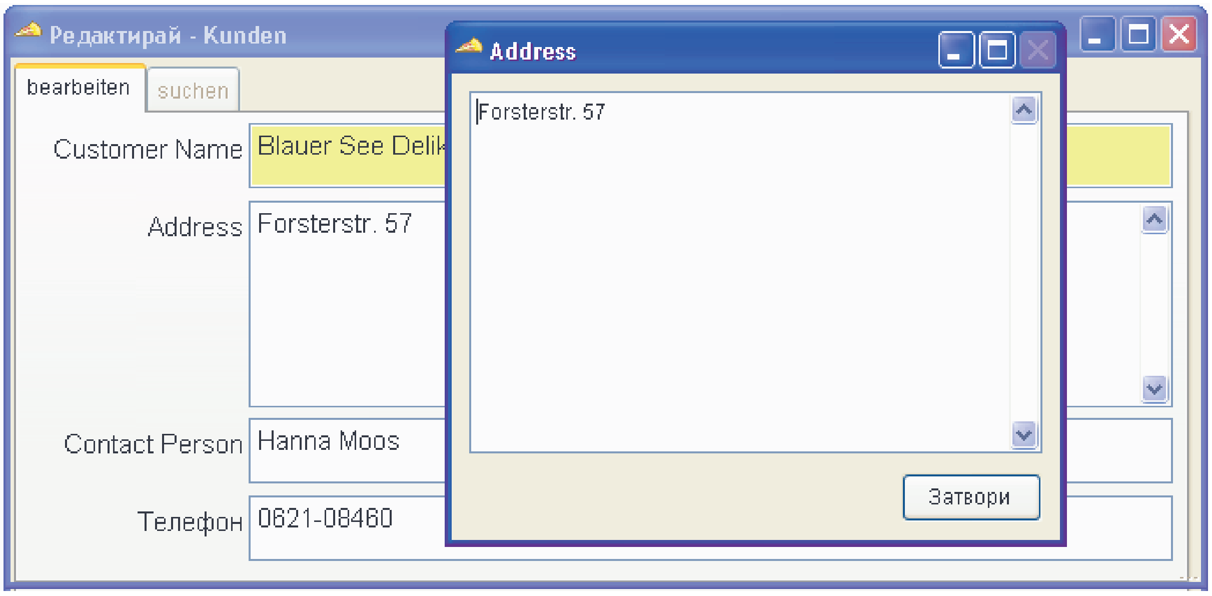
Add a rectangle control to a report wherever you want the RTF text to appear. Remember that the height is not important. The ReportListener will stretch the rectangle to fit the text.
Next, set the MemberData XML for the rectangle. To do this, open the rectangle properties and select the “Other” tab. Click on the “Edit Settings…” button in the “Run-Time Extensions” section. Then click on the “Edit XML…” button in the “Run-time extensions” screen.
You will see there is already some XML code in the editbox. You need to add an “rptctrl” node to tell the custom ReportListener that this control has a custom handler. Add the following text just before the </VFPData> closing tag, where “MyTable.rtfmemo” is a valid expression that evaluates to RTF text.

<rptctrl class="RTFHandler" expr="MyTable.rtfmemo">

Select OK on all the open dialogs and save the report.

cMmemoform for cEditbox controls

To the cEditbox control is added functionality to call a form for easy editing the memo data. When the user DblClicks on an editbox the memo form opens with larger text area where the text content could be seen and edited conveniently.



Customer Name: Blauer See Delik

Address: Forsterstr. 57

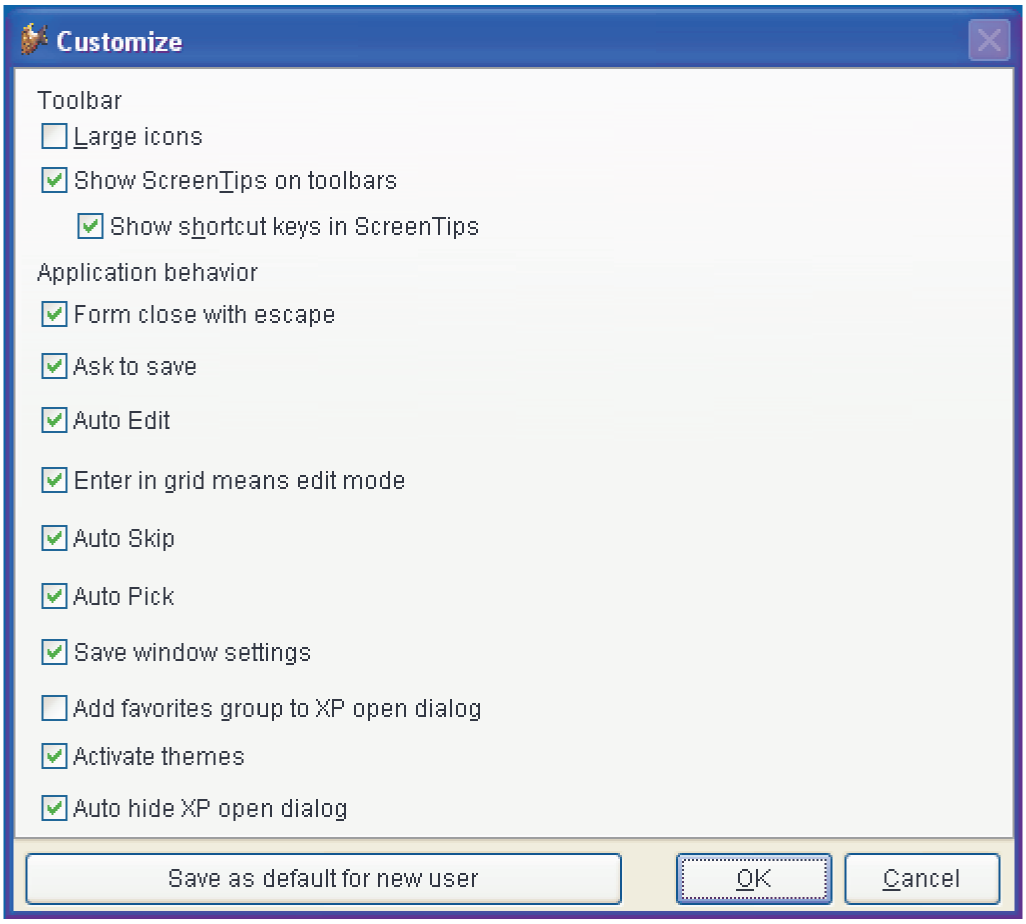
Contact Person: Hanna Moos

Телефон: 0621-08460

Forsterstr. 57

Customize dialog

The new *Activate themes* and *Autohide XP Open* dialog features can be set per user in *Customize* dioalog.


Customize

Toolbar

- [ ] Large icons
- [✓] Show ScreenTips on toolbars
  - [✓] Show shortcut keys in ScreenTips

Application behavior

- [✓] Form close with escape
- [✓] Ask to save
- [✓] Auto Edit
- [✓] Enter in grid means edit mode
- [✓] Auto Skip
- [✓] Auto Pick
- [✓] Save window settings
- [ ] Add favorites group to XP open dialog
- [✓] Activate themes
- [✓] Auto hide XP open dialog

Search dialog

Additionally it's added one more operator. This is the operator NOT CONTAIN. This operator can be applied to character and memo data types.