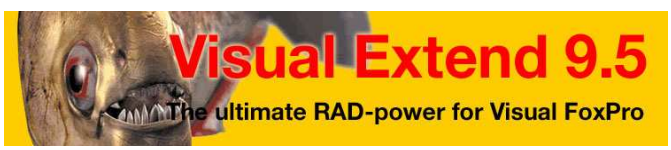

# VFX 9.5 – What is new?
# 3. Quarter 2006

October 2006




*Uwe Habermann, Venelina Jordanova*

# Table of content

# New features for developers

## *Search dialog*

Filter definitions are now able to be saved in *VfxFilter.dbf* or in *VfxRes.dbf*. This feature is available simultaneously for User and for System filter definitions. The feature is controlled by the property *lUseFilterTable* of *cFoxApp* class (*goProgram* object). When the value of this property is set to .T., *VfxFilter.dbf* table is used for keeping User and System filter definitions.

Additionally, search dialog uses *cDateTextBox* class in Value column for fields of Date and DateTime types.

## *Download and extract Zip file into EXE folder*

This feature allows downloading additional file along with application update and extracting its content into EXE folder. For this purpose, has been added a new property of cFoxApp class (goProgram object) – *cAddFilesDownloadURL*. The property keeping download URL for this additional file to be downloaded and to be extracted into EXE folder. Also a new method AddFilesUpdate has been created in cFoxApp class, which process download, extracts the ZIP content and deletes the downloaded file.

## *Unique Fields*

To cTextBoxBase class is added a new property *lUniqueField*. If it's set to .T. – VFX checks for uniqueness of the Field, bound to this textbox (its ControlSource).
This property can be keyed up in all form builders through the checkbox *Unique field*. When the value in the field must be unique, mark this checkbox. This functionality is available only when the selected control is of class cTextBox and only for data fields from Master table.
The property can also be keyed up in VFX – TextBox Builder.

### VFX – Data Environment Builder

In Data Environment Builder, on Aliases page is added a new *NoData* checkbox which controls the property *NoData* of CursorAdapter instance.
If a parameter is specified in *cWhereclause* the value of property *NoData* is automatically set to .T. This avoids duplicate execution of CursorFill when form loads.
If later the parameter in *cWhereclause* is removed, the checkbox still keeps the value of .T. and the developer must manually set the value on .F. if desired.

On Indexes page can also be selected the Collation for the created index from a drop down combobox. By default, new indexes are created with Machine collation.

## *Product activation*

### Application protection

In VFX 9.5 activation key, used to activate developed application, can be generated in two different formats: the format that developers know by far and a new Microsoft compatible format.
The developer can choose whether the application will use Long activation key (VFX 9.0 behavior) or a Short activation key in Microsoft compatible format. Microsoft compatible activation key format is a new feature, which allows the developer to use 25 characters long activation keys. Additionally, when using this key format, is that it can be time limited. These features are controlled

by properties of the class *cVFXActivate*. They can be keyed up in the Application Builder. The properties can be alternatively set in the class *cVFXActivation* (*Appl.vcx*).

*nProductActivationBehavior* – 1 – *(Default)* VFX 9.0 activation key format; 2 – Microsoft compatible activation key format;

*lUseTimeLimitedActivationKey* – .T. when the used activation key will be time limited. Allowed only if nProductApplicationBehavior is set to 2. Default value is .F.;

*dStartActivationDate* – keep the initial date for calculating active days, when using time limited activation key. Default value: 01.Jan.2000. Last date of validity period for the time limited key will be calculated by adding days of validity, kept in activation key string, to this date.

*nDefaultValidityDays* – keep the default number of days used when issuing a time limited the activation key. This value is used when activation key is automatically generated by the registration web-service.

Additionally, developer can choose whether the Activation Key will be Hardware tolerant. It is possible to define the number of hardware parameters tolerance. When hardware tolerant feature is used, the activation key is still valid when a number hardware parameters changed. This feature can be keyed up the Application Builder. The properties which control it are in the class *cVFXActivation* (*Appl.vcx*).

*nHardwareParametersTolerance* – number of tolerant hardware parameters. When the value is greater than 0, the product activation is hardware tolerant. Default value 0.

*cStoreHardwareParameters* – name of text file the information about initial values of hardware parameters is saved.

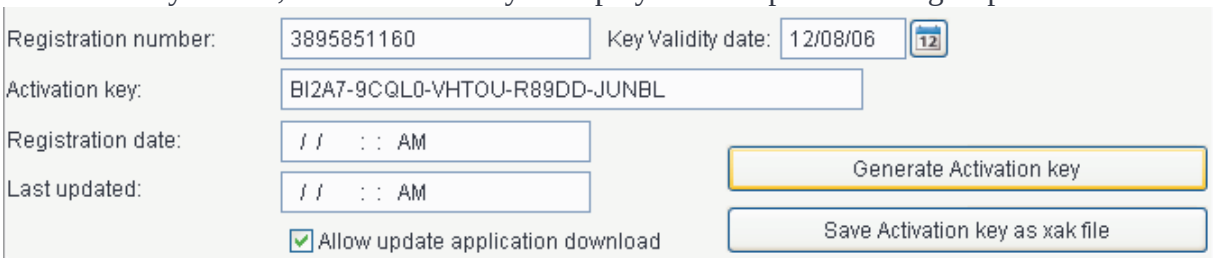
*cEncryptPassword* – a password used to encrypt and decrypt the content of hardware parameters file.

If Hardware tolerant product activation feature is used, at the end-user side the values of hardware parameters, active at the time of activation, are saved in an encrypted text file. This file is encrypted with a password set in the property *cEncryptPassword* of *cVFXActivate* class. The name of this text file is keyed up in *cStoreHardwareParameters* property of *cVFXActivate* class. The file is created along with the INI file, conforming Windows compatibility rules.

When the application is checked for valid activation, hardware parameters' values are also checked. The application checks if the evaluation of activation pattern matches the initial one. If the expressions do not match, the text file keeping original parameters will be used to compare current hardware parameter values and initial hardware parameter values. If the number of non-matching values is less or equal to the setting *nHardwareParametersTolerance* it will be considered that the activation key is still valid.
If there is found an activation key, but no hardware parameters file is found, the validation of activation key will be done without tolerance.

## Generating activation key in Customer Management application

When the application, for which Customer Management is started, is set to use Microsoft compatible activation key format, the activation key is displayed in 5 alphanumeric groups.




The Key Validity date control is visible only when the application uses time limited activation key. The default value of the validity date is calculated by adding *Activation Key Validity Days (nDefaultValidityDays)* to current date.

## VFX – Registration Web Service

The Web Service is added the ability to generate and return a temporary valid Activation Key to the customer. This feature is available only for applications that are set to use time limited activation keys. The generated activation key is returned to the customer in the moment he registers online. And this key is valid for a number of days, which value is preset for the application in the property *nDefaultValidityDays* of the class *cVFXActivation*. The customers register information, the given activation key and the key validity date are stored in the customers database.
The application can be set to use time limited activation keys in the Application Builder. This setting is valid only for applications which use Microsoft compatible activation key format. The default value of key validity days is set also in the Application Builder.

## *Application Impersonation*

The *impersonation* is a new VFX functionality that raises data security for VFX applications
This feature allows users to have access to Data through the Application even when in OS they have been denied access to Data folder. This feature can be used in very simple way. It is only needed to fill correctly the three new fields in config.vfx file.

*Win User Name* – Windows User Name to be used for impersonation of the application.
*Win Password* – Windows User Password
*Win Domain Name* – Windows domain name (if applicable), in which is the user.

When running the application and selecting a Client in client selection dialog, the application automatically uses the user rights of the specified Window User, instead of rights of the user currently logged in. These security user rights are applied when accessing the data folder and all folders that application accesses.

When Impersonation is used, it is required to fill username and password for all client data sources in Config.vfx.

## *Installation Wizard*

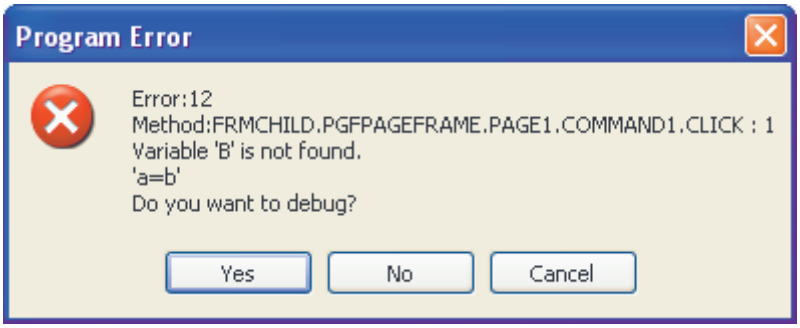
The VFX – Installation Wizard prepares a folder, ready to be distributed to the end-users' computers. This folder contains all the files necessary to run the application at the end-user computer, without need developer to create and deploy installation file.


The VFX – Installation Wizard can be run from VFX 9.5 Menu using *Project, Installation Wizard.*

Using the directory selector the developer can choose the folder in which the Install Wizard will place application files. If the folder does not exist it will be created automatically. By default, installation folder is proposed to be INSTALL folder under active project folder.
Below the directory selector is placed a checkbox *Build EXE file*. If it is marked while preparing application files, the wizard will automatically build an EXE file of the active project. If there is no EXE file in the project folder of the active project, the *Build EXE file* checkbox will be checked and disabled, so the EXE file will always be built.
There are 9 checkboxes for the different languages. By default all are languages except Chinese simplified and Chinese traditional are selected. If some of the language DLLs are not available on the developer's computer the corresponding checkbox in the wizard is also disabled.
Clicking the OK button starts preparing of install folder and files copying.
This process copies VFP runtime libraries, all additional OCX and DLL files used by VFP. During this process will also be created a DATA folder. In this Data folder will be created an empty DBC and all VFX tables, which are not included in EXE, will be copied there.

## Runtime Errors handling

When the application is started in IDE and if an error occurred, instead of displaying standard end-user error message, the developer is given the chance to debug the error, to ignore it or to abort the program execution.
The message box with the description of the error is displayed:




There are three possible answers:
*Yes* – the program execution will be debugged;
*No* – the error will be ignored;
*Cancel* – the program execution will be aborted.

In all cases the error details will be logged in vfxLog.dbf.

## Database repair

When for some reason, the reindexing is not successful for all tables, the database repair makes a list of problematic tables and automatically repairs all of them after the reindex operation.

## Run database maintenance on application start

VFXMain.prg accepts a set of parameters used to perform automatic database maintenance:

*tcDatabaseMaintenance* – the values of *tcDatabaseMaintenance* must be one of the listed: "$REPAIR$", "$PACK$", "$PACKMEMO$", "$REINDEX$" or "$TABLE$"<table name>.
Only one value may be passed at a time. When parameter starts with $TABLE$, the specified table is repaired.

*tcPath* – a $ delimited list of client names to be located in vfxpath (this option can be used with vfxpath only - not with config.vfx). If empty all records in vfxpath are processed. If it does not start with $, it is considered to be a database path.

*tcDBC* – database name. This parameter must be passed when in *tcPath* is passed a folder path. When the parameter *tcPath* contains a list of client names searched in VFXPath table, database name

## Class cDatabaseTools

### Properties

*cCurrentTable* – the name of the currently selected table

*cDatabaseName* – the name of the database

*cDataDir* – the path where the database is located

*cOldActiveDBC* – the name of the previously used database

*lSilentMode* – When set to .T. no messages are displayed. Default is .F. When the form is run with parameters, it is automatically set to .T.

### Methods

*Init(tcarg, tcAction, tcPath, tcDBC)*

*tcAction* – the values of *tcAction* could be "$REPAIR$", "$PACK$", "$PACKMEMO$", "$REINDEX$" or <table name>. If does not start with $, it is considered a tablename is passed for repairing. A combination of multiple values is invalid.

*tcPath* – a $ separated list of client names to be found in vfxpath (this option can be used with vfxpath only - not with config.vfx). If empty all records in vfxpath are processed. If it does not start with $, it used as a database path.

*tcDBC* - database name

*ExcludeFiles(tcFileList, taFiles)* – excludes a list of files from an array. Both the list and the array are passed as parameters.

*tcFileList* – a ";" separated list of the files to be excluded

*taFiles* – an array containing a list of files some of which are to be excluded

*FillSourceList(taFiles,taTable, tlShowProgress)* – fills an array with details about .dbf files

*taFiles* – an array containing a list of .dbf files

*taTable* – an array containing details about the .dbf files in *taFiles*

*tlShowProgress* – if set to .T. a progress bar will be displayed

*RunMaintenance(tcTable)* – sets the data in the mover and calls the *Doit*() method

*tcTable* – the name of the table which *tcAction* parameter contains. This table is selected in the mover control. If *tcAction* does not contain the name of a table *tcTable* is blank.

## *New properties in goProgram*

*cAddFilesDownloadURL* - keeps download URL for additional ZIP file to be downloaded and extracted into EXE folder

*lCheckforDBUpdate* –Controls if client database update first checks if update is needed without need of exclusive access to tables. The default value is .F. This property is keyable in Application Builder. The database update operation opens tables in exclusive mode. Using this property, developers can activate the feature to check for structural changes without requiring exclusive access to the database. When this property is set to .T., before starting the database update all tables are checked for structural differences. If it is not necessary to run database update, the operation is not started.

*lUseVfxFilterTable* – when .T. VFX uses *vfxFilter* table to save filter information instead of vfxRes table.

*nXPdialogTotalTimeToSlideout* – Total time in milliseconds for the XP open dialog to slide out (if XP open dialog autohide is enabled)

*nWindowState* – used to control the window state when application starts

*0* – Run normal

*1* – Run minimized

*2* – Run maximized (Default)

# New fewtures for end-users

## *The class cComboPickList*

### cComboPicklist class

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

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

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

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

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

### Properties:

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

*cPickList* – Contains code from *Vfxpdef.Code*

*cProcCode* – Contains code from *vfxpdef.ProcCode*

*lAutoAdjustColumnWidths* – If .T., the column widths will be adjusted to the width of the widest length of the content.

*lAutoGCode* – If .T., the code field will be generated automatically when new items are added and only Description column is visible.

*lShowOnlyDescr* – If .T. only Description column is visible.

*nUserLevel* – User Level allowing the user to insert new items.

*nFieldLen* – Internally used. Keeps Field Len value for current pick definition.

### Methods:

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

*FillItems* – fills the Combobox with pick items defined in *Vfxplist.dbf* table.

*SetComboboxSettings* – executes when the *cComboPickList* is initializing. Here are set up the properties of *cComboPickList* control according to *Vfxpdef.dbf* table and user rights.

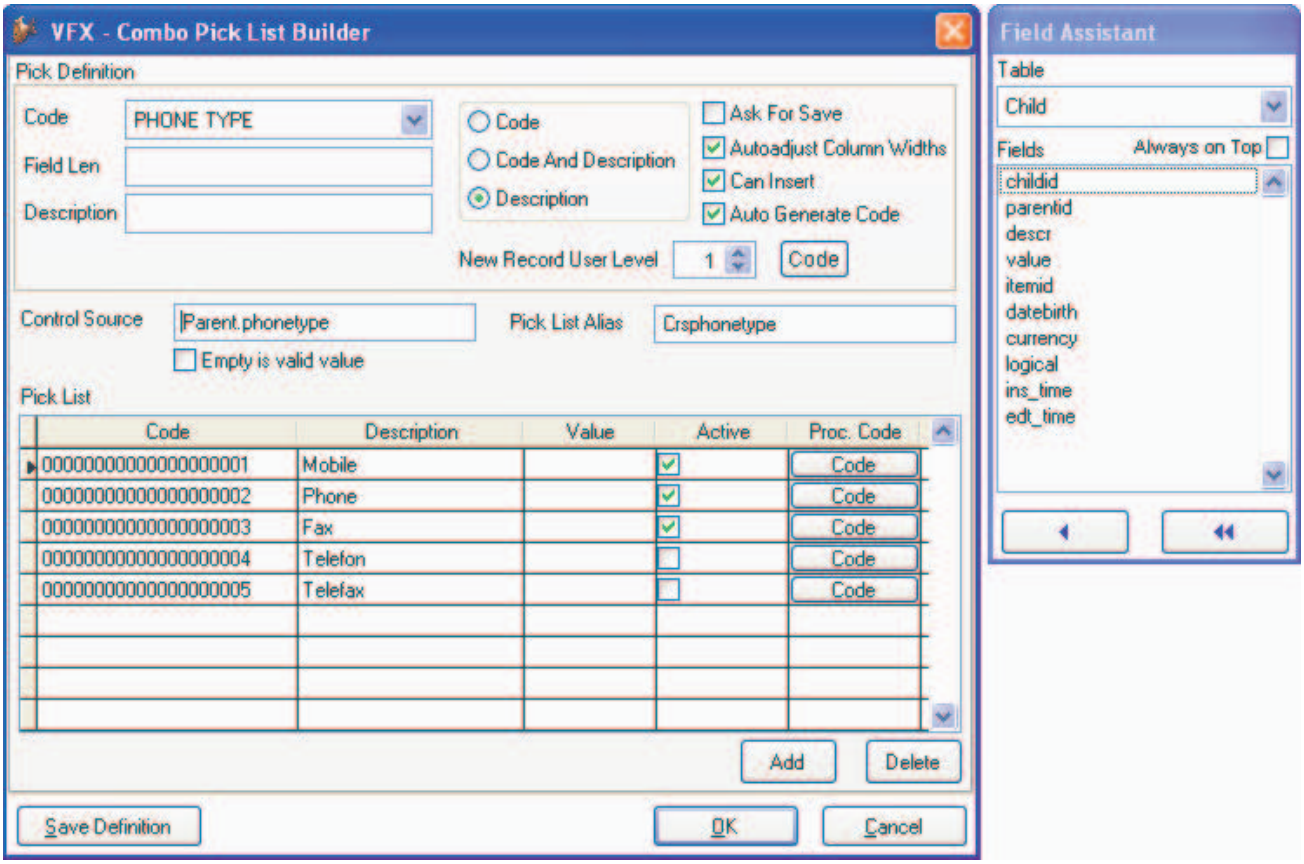
For *cComboPicklist* class it is possible to define two different code snippets. In *Vfxpdef* table in *ProcCode* memo field and in *Vfxplist* table in *ProcCode* memo field.

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

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

## VFX – Combo Pick List Builder

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




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

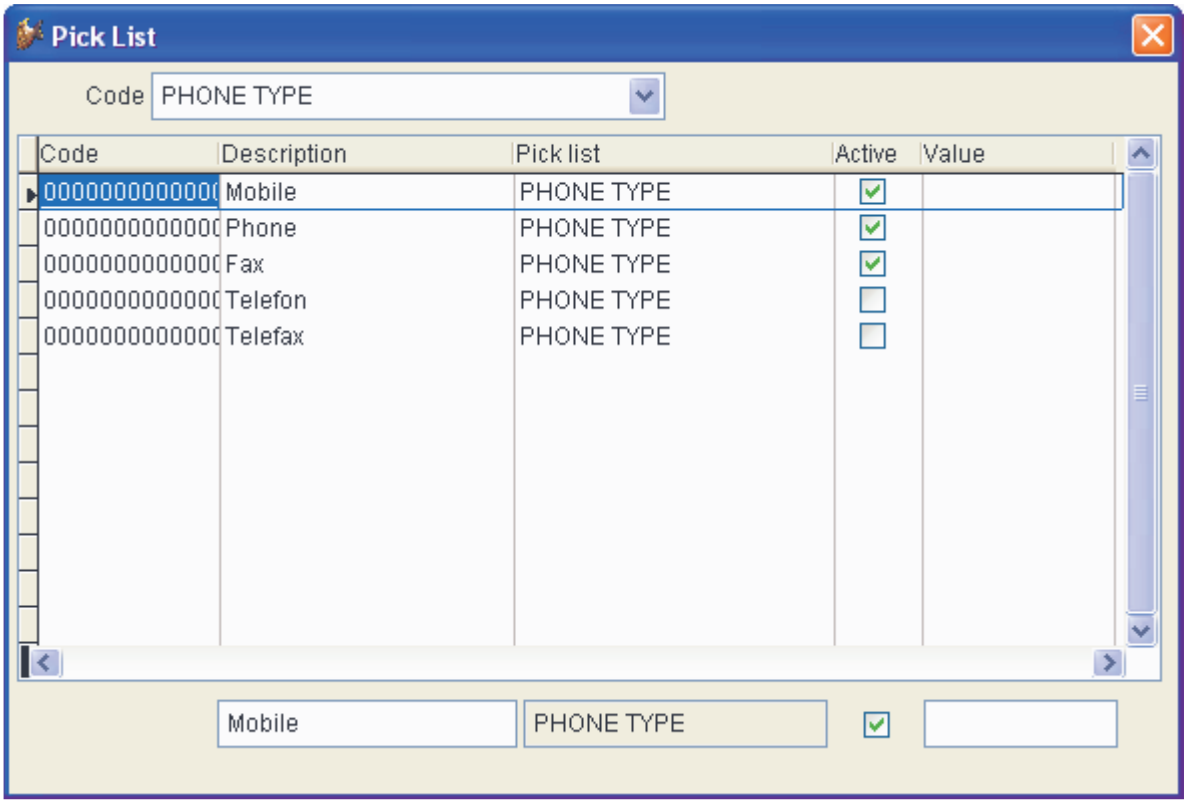
It is also possible the class *cComboPicklist* to show only the Description text. In this case if *Can Insert* option is set to .T., the option *Auto Generate Code* is set to .T. automatically. When *Auto Generate Code* is set to .T., when an item is added to Pick List, the code is generated as next

consecutive value. In this case in VFX – Combo Pick List Builder and in Pick list maintenance form, the column *Code* is Read Only. The length of generated code is defined by the value in the *Field Len* text box. If this value is empty, the length is defined according to the size of the field specified as *Control Source* field.
The option *Auto Generate Code* is available only if the *cComboPickList* is set to show the *Description field only*. In other cases the code is entered manually.

A piece of Code which will be common for the *cComboPicklist* and will be executed every time when the selected item of the *cComboPickList* control is changed, can be written in an edit box, which will appear when the button *Code* from *Pick Definition* area is pressed.

A piece of code which will be specific for the particular list item and will be executed only when this item is selected from the list, can be written in an edit box, which will appear when the button *Code* from *Pick List* grid column is clicked.

### Pick list maintenance form




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

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

## *cDateTextBox class*

The class *cDateTextBox* is a text box for entering a date or datetime. Whit double clicking or pressing F9 a calendar control is called. This control is designed to be used in *Search Dialogs* and in *Child Grids* for fields of Date or DateTime type.

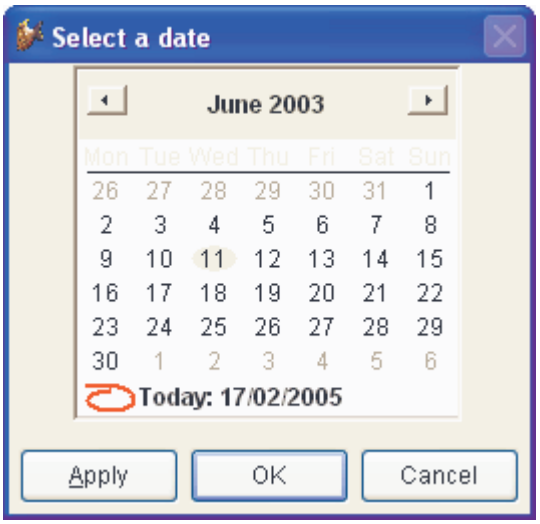
For the Textbox for date selection are available the following hotkeys:

| | |
|---|---|
| +, = | Next day |
| -, _ | Previous day |
| T, t | Today |
| F, f | The first day (beginning) of the current month |
| L, l | The last day of the current month |
| Y, y | New Year 's day |
| E, e | End of the year |
| P, p | Previous month |
| N, n | Next month |

These hotkeys are defined with the constant MSG_CPICKDATEHOTKEYS for different languages.

When the control is assigned to field of DateTime type, the upper hotkeys keep the current time settings. Only when calling *New Year's day* or *End of the Year*, time settings are restored.

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




## *The class cDocumentManagement*

### Drag and drop to cDocumentManagement

The functionality of *cDocumentManagement* class is extended with drag-drop compatibility. New records can be created by simply drag and drop the desired file from Windows explorer. Users can also drag and drop items from Outlook. A correspondent record is created in Document Management for email, task, contact or a folder.

To control the functionality to drag and drop from Outlook, to the class *cDocumentManagement* were added number of new properties:

*cDocumentTypeFieldName* – Name of the field in Document table where is stored the document type. Document type field is internally used. There are four values for document type, stored in this field: 0 – *(Default when adding records with new button)* File or RTF document, 1 –

Mail from Outlook, 2 – Task from Outlook and 3 – Contact from Outlook, 4 – Folder from Outlook. The default field name is *DocType* (*vfxDocuments.dbf*),

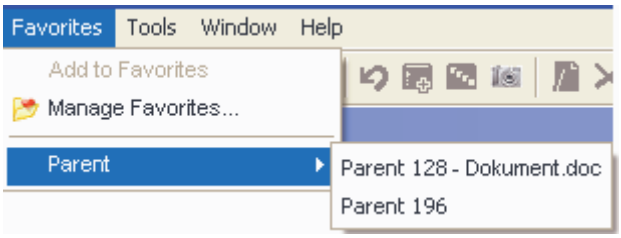
*cEntryIDFieldName* – Name of the field in Document table where to store the EntryID of an Outlook Item. Used only for documents dropped into document management object from Outlook. The default value is *EntryID* (*vfxDocuments.dbf*).

If one of the properties *cDocumentTypeFieldName* or *cEntryIDFieldName* is empty, the feature to Drag and Drop elements(mails, tasks, contacts) from Outlook will be disabled.

The class *cDocumentManagement* supports also the functionality to add ChildGrid records to favorites menu. Users can do this by *Add To Favorites* menu, when a document record is selected. When opening from Favorites, the record pointer will be positioned on the corresponding document row in cDocumentManagement grid. To support this functionality are used the two cChildGrid properties *cFavoriteID* and *cFavoriteDescr*.

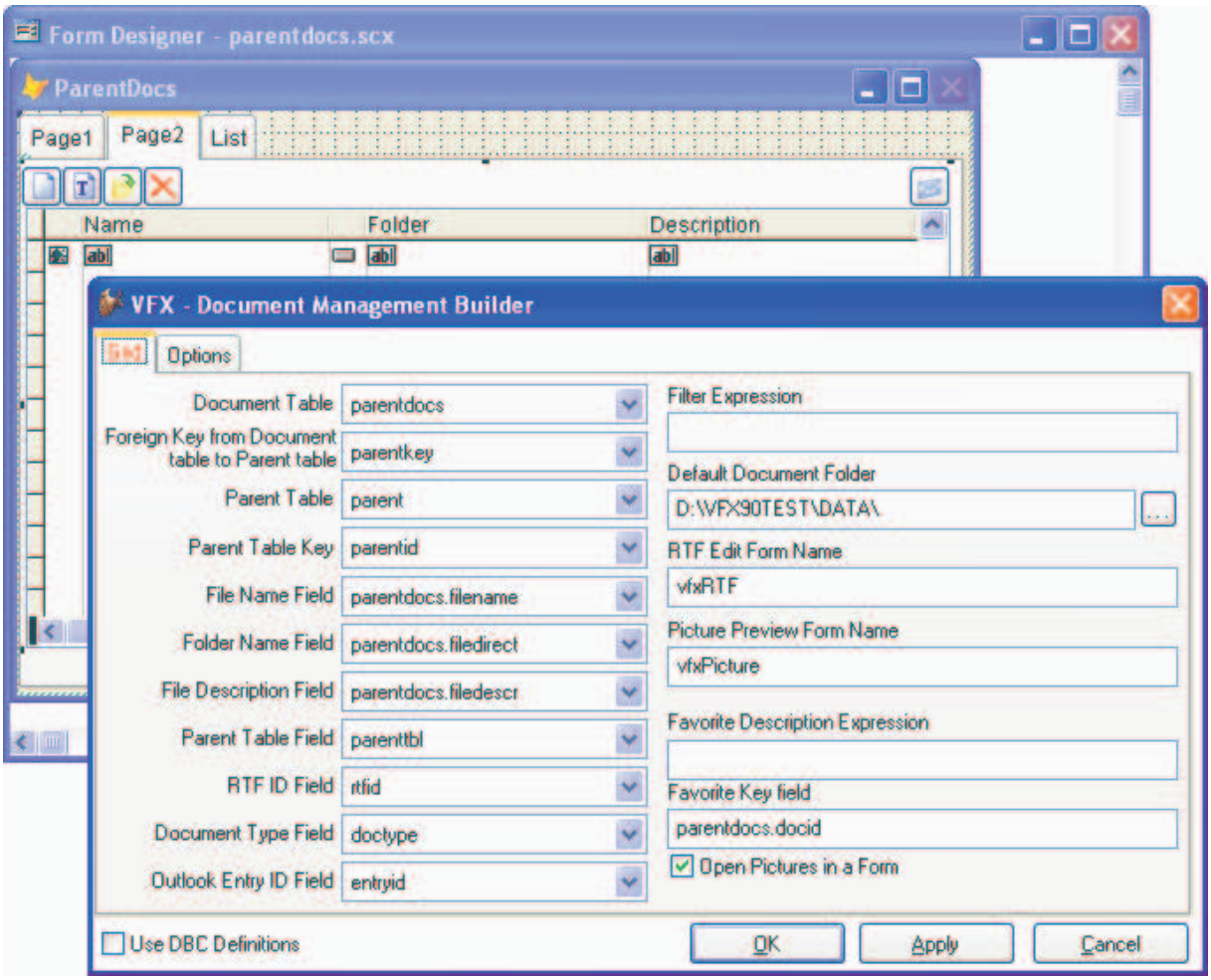
The string result from evaluating expression specified in *cFavoriteDescr* property, will be added to the form's description in menu file. If this property is empty, the favorite records from Document Management will be added with a description created in following way - to the expression used for description of form's favorites, will be appended:

- first 10 characters from file description if it is not empty;
- or in other case, the file name (As shown in the picture below).



### VFX – Document Management Builder

All the properties of the class can be keyed up in VFX – Document Management Builder.




In the VFX – Document Management Builder, all fieldname selection comboboxes are filled automatically with the default values, if vfxDocuments is selected as the document table. The fields are also automatically filled, if fields with same names exist in the selected document table (if it is not vfxDocuments).

In the text box *Favorite Key Field* is expected to write an expression which will be used for *cFavoriteID* property of cChildGrid. Usually here should be written the ID field from Document table. This value is filled by default from the Builder.
In the text box *Favorite Description Expression* is expected to write the expression, which will be used for *cFavoriteDescr* property of cChildGrid.

## *The class cChildGrid*

### Favorites support

Child records can be added to Favorites. The end-user can do that by positioning on child record which will be added, add selecting from menu *Favorites*/*Add To Favorite*. The favorite record, created for Child Grid will be displayed under the favorites group, created for the form (will use form's properties cFavoriteScx and cFavoriteMenu).

## Properties

*cFavoriteID* – similar to Form's *cFavoriteID* property, but keeps the ID field name of the child table.

*cFavoriteDescr* – similar to the Form's property, but keeps the Description Expression for child table. This expression will be evaluated and added to Form's description in Favorite menu.

## Method

*ViaFavorite* – similar to Form's method, but additionally to the action of opening the form and positioning to the parent record, it will position the record pointer of the child grid at the proper record.

### VFX – CChildGrid Builder

In CChildGrid builder, as well as in the builders for forms that contain child grids (cOneToMany Builder, cOneToManyPageFrame Builder and cTreeViewOneToMany Builder), developers have the chance to specify for every column if it will allow sorting without having incremental search on that column. This is very useful, when childgrid alias is based on a parameterized cursoradatpter object or on a parameterized view. In this case, you may allow users to edit data, specifying that grid columns will not support incremental search and in same time, you may allow sorting on different grid columns.

**Incremental Search**. Mark this option, if you want to have incremental search in the current column. Selecting this option automatically checks the options Read Only and Allow Sort for the current column.

**Allow Sort**. Check this option, if you want to allow the end user sort the grid by this column. Unselecting this option will also unselect Incremental search option.

### *Record positioning in one-to-many forms*

Until now in all one-to-many forms after save or undo the record pointer was always positioned on the first child record. To get rid of this inconvenience, a new array property was created in data form classes– *aChildAliasRecno[1,1]*. This array is filled up in the beginning of OnSave method and OnUndo method with child aliases and current record numbers. After saving or reverting the data, using this array property the record pointers are restoring to the position before save or undo operation.

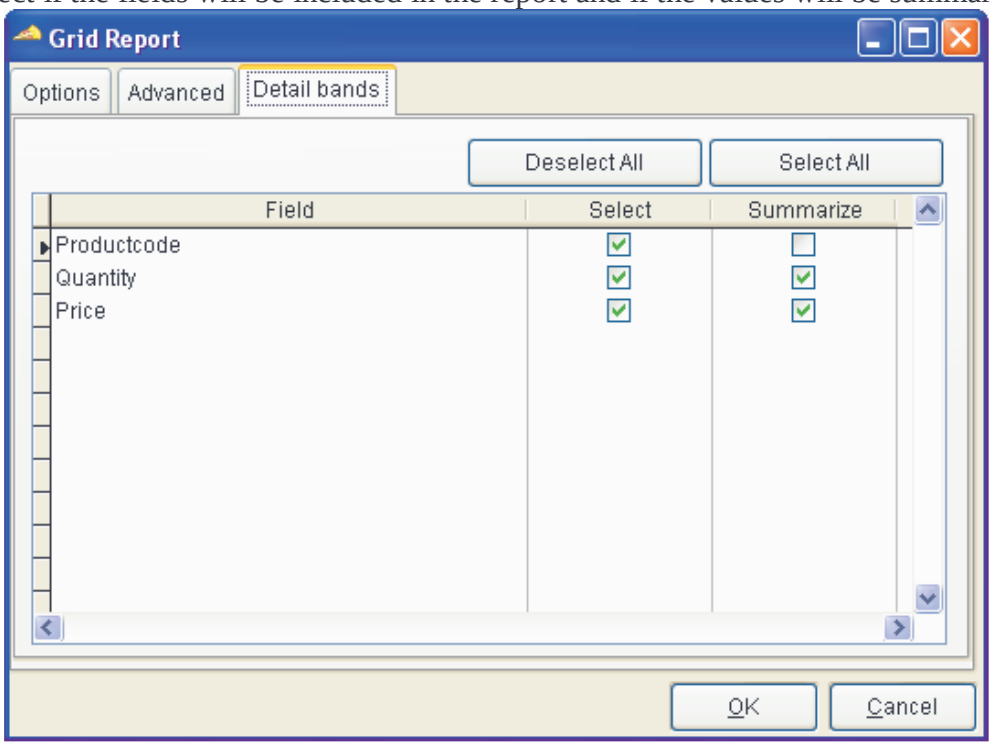
### *Grid report*

For OneToMany or OneToManyPageFrame it is also possible to generated reports to contain child detail band, consisting of fields from a child alias.

In *VFX – cOneToMany Builder* and in *VFX – cOneToManyPageFrame Builder* forms, on page Report is added a combobox *Band*, which allows the developer to select the position of each field (in Parent band or in Child band).

If *Use Grid Fields For Report* is selected, it is not possible for developer to specify report fields. In that case all fields from *cWorkAlias* will be printed in Parent band. If the form is *OneToMany* in Child band will be printed all fields from the child grid, placed on the first child page. If the form is *OneToManyPageFrame*, in Child band will be printed all fields from the grid on the first met Grid Child Page (if one exists).

In *Grid Report dialog* is added a page *Detail Bands* where are listed the fields, to be positioned in Child band. The fields positioned in Parent band are listed on *Advanced* page. This additive page *Detail Bands* is visible only for *OneToMany* and *OneToManyPageFrame* forms. In this page the user can select if the fields will be included in the report and if the values will be summarized.




A sample part of OneToMany report is shown below:

**Order**

*Page* 1 *Date* 08/25/06
*Time* 18:16:44

| *Order Date* | *Ship To (Name)* | | |
|---|---|---|---|
| 07/04/96 | Vins et alcools Chevalier | | |
| *Productcode* | *Quantity* | *Price* | |
| PCardi | 12 | 6.40 | |
| Apfel | 10 | 1.50 | |
| Tomate | 5 | 2.10 | |
| | 27 | 10.00 | |
| 07/05/96 | Toms Spezialitдten | | |
| *Productcode* | *Quantity* | *Price* | |
| Pveget | 9 | 6.80 | |
| Jдgerm | 40 | 1.80 | |
| | 49 | 8.60 | |
| 07/08/96 | Victuailles en stock | | |
| *Productcode* | *Quantity* | *Price* | |
| PDiavo | 6 | 6.50 | |
| Schoko | 15 | 1.60 | |
| Tirami | 20 | 2.80 | |
| | 41 | 10.90 | |

If no fields from Detail bands are selected in *Grid Report dialog*, a Standard report without child detail band will be printed.

## *Image scanning*

VFX includes features for easy and user-friendly image scanning. There are two VFX classes developed for this and a function in VFX .FLL
The class *cScannerControl* is a custom class, providing an object-oriented shell for VFX.FLL scanning functions. The class *cScanForm* if a form class, giving the end-users interface to preview and scan images.

### The class cScannerControl

Class *cScannerControl* is a part of *Vfxctrl.vcx* library. It gives developer the change to acquire images from scanner, simply by filling a few properties.
You can see an example how to use this class in VFX95Test application, in form Item.

### Properties

*cFileName* – full path and name of the file where to store the scanned image

*cFileFormat* – file format string is based on known file extensions. The value can be one among: *BMP*, *JPG*, *GIF*, *PNG*, *TIFF*. The default value is *TIFF*.

*cRunOnFailure* – string containing a method name to call in case scanning failed. The method is user-defined method of a form and is used to display error messages. Default value *"This.ScanFailed()"*

*cRunOnSuccess* – string containing a method name to call when scanning is completed succesfully. The method is user-defined and is used to display a message informing that the scanning was successful. Default value "*This.ScanSuccesful()*"

*lUseDefaultTWAINSource* – If set to .T. the default TWAIN source will be used Its default value is .T.

*nScanResult* – Keeps the return value of *GetTWAINImage()* function. This property can be used in a user-defined method to determine what error occurred and process it accordingly.

*cTWAINSource* – string containing TWAIN Source to be used. The default value is empty string. If lUseDefaultTWAINSource property is .F. and cTWAINSource property is also empty and in case there are more than one TWAIN sources on the computer, a dialog for TWAIN Source selection is displayed

*lDisplayAdvancedSettingsDialog* – depending on TWAIN source, this feature may not be always available. The default value is .F.

*lDisplayProgressDialog* – If set to .T., a progress dialog is displayed. The default value is .T.

*lDisplayMessages* – If set to .T., a message is displayed when the scanning was completed successfully. The default value is .T.

### Methods

*GetTWAINSources* – calls *GetTWAINSources* function in *VFX.FLL*
*Return value*: the return value of *GetTWAINSources()* function:
*0 – if operation is completed successfully*
*nErrorNumber – when an error occurred*

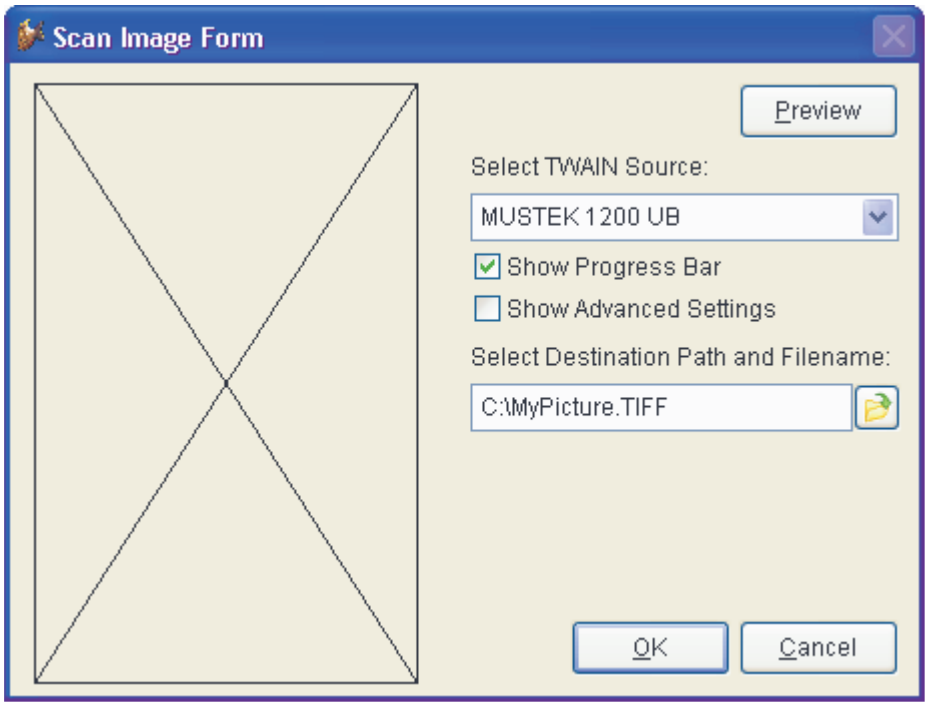
*GetTWAINImage* – calls *GetTWAINImage* function in *VFX.FLL*
*Return value*: the return value of *GetTWAINImage()* function:
*0 – if operation is completed successfully*
*nErrorNumber – when an error occurred*

*ScanFailed* – displays a default error message if an error occurs during scanning

*ScanSuccessful* – displays a default message that scanning was completed successfully

## The class cScanForm

Class *cScanForm* is a form class, designated to be displayed to end-users. The class is a part of *vfxformbase.vcx* library. It is suitable either to be instantiated and controlled by code or to be called as a modal dialog.




When used as modal dialog, the form returns full pathname of scanned file or empty string if file is not scanned.
You can see an example how to use this class in VFX95Test application, in form Item2.

## Properties

*cImageFile* –the full path and filename where the image is saved (blank if Cancel button is clicked)

*cTempImageFile* –Internally used to keep the path to the temporary file where the image used for preview is saved.

*lDoNotReleaseForm* –set this property to .T. when the form is instatiated by code and after clickng ok or cancel the form is not released bur hidden.

*lScanned* –Internally used.

## Methods

*Init(tlShowProgress, tlAdvancedSettings, tcFileName, tcTwainSource, tlDoNotReleaseForm)*

The parameters of the init method are used when the form is invoked with do form, to initialize form's properties.

*tlShowProgress* – Used to set the property *lDisplayProgressDialog* of *cScannerControl* object in the form.

*tlAdvancedSettings* - Used to set the property *lDisplayAdvancedSettingsDialog* of *cScannerControl* object in the form

*tcFileName* – Used to set the value of the edit field in *cfileselector* object in the form.

*tcTwainSource* - Used to set the value of the twain sources *combobox* object in the form.

*tlDoNotReleaseForm* – Used to set the property *lDoNotReleaseForm* property of the form.

*Scan(tcFileName, tcFileFormat*) – scans an image and saves it in a file. Expects the full path to the file and the image format as parameters.

*tcFileName* – the full path and filename where the scanned image is saved.

*tcFileFormat* – the file format in which the scanned image is saved. It is used to set the *cFileFormat* property of *cScannerControl* object in the form.

*Return value: .T. – If scan operation completed successfully*

*SaveLastFolder(tcExportFolderType*) – saves path to the folder which was chosen for the image file in a resource file.

*tcExportFolderType* – internally used.

## Image scanning related functions in VFX.FLL

*GetTWAINSources(@tcTwainSources)* – returns a list of available TWAIN sources, installed on the computer.

*tcTwainSources* – a character string, passed by reference. In this string is stored the resultant ';' separated list of available TWAIN sources.

*Return value: 0 – if operation is completed successfully*
*nErrorNumber – when an error occurred, TWAIN error code*

*GetTWAINImage(tcFileName , tcFileFormat , tlUseDefaultTWAINSource, tcTWAINSource, tlDisplayAdvancedSettingsDialog, tlDisplayprogressDialog)* – scans an image and saves the scanned immage in the file *tcFileName*.

*tcFileName* – full path and name of the file in which to store the scanned image

*tcFileFormat* – file format is based on known format extensions - BMP, JPG, GIF, PNG, TIFF. The default value, if this parameter is omitted, is TIFF.

*tlUseDefaultTWAINSource* – If its value is .T., the default TWAIN source will be used. The default value, if this parameter is omitted, is .T.

*tcTWAINSource* – string containing TWAIN Source to be used. The default value, if this parameter is omitted, is empty string. If tlUseDefaultTWAINSource is .F. and tcTWAINSource is empty and there are more than one TWAIN sources on the computer, a dialog for TWAIN Source selection is displayed.

*tlDisplayAdvancedSettingsDialog* – If its value is .T., an advanced settings dialog is displayed. Depending on TWAIN source, this feature may not be available always. The default value, if this parameter is omitted, is .F.

*tlDisplayprogressDialog* – If its value is .T., a progress dialog is displayed. The default value, if this parameter is omitted, is .T.

*Return value: 0 – if operation is completed successfully*
*nErrorNumber – when an error occurred, TWAIN error code*

## List of TWAIN Error codes

0x00000000 - 0x0000FFFF - General errors
---
0x00000000 - Success (image saved successfully)
0x00000001 - Canceled by user
0x00000002 - TWAIN library load failed
0x00000003 - GDI+ initialization failed
0x00000004 - Invalid image format specified
0x00000005 - Not enough memory
0x00000006 - Invalid source specified
0x00000007 - GDI+ error
0x00000008 - Invalid characters in filename (Unicode convertion failed)
0x00000009 - File save error

0x00010000 - 0x0001FFFF - TWAIN specific errors (MSG_OPENDSM)
---
0x00010000 - Cannot open Data source manager
0x00010001 - Failure due to unknown causes
0x00010002 - Not enough memory to perform operation
0x00010005 - DSM error

0x00020000 - 0x0002FFFF - TWAIN specific errors (MSG_USERSELECT)
---
0x00020000 - Cannot select datasource
0x00020001 - Failure due to unknown causes
0x00020002 - Not enough memory to perform operation
0x00020005 - DSM error

0x00030000 - 0x0003FFFF - TWAIN specific errors (MSG_GETFIRST)
---
0x00030000 - Cannot start datasource enumeration
0x00030001 - Failure due to unknown causes
0x00030002 - Not enough memory to perform operation

0x00030005 - DSM error

0x00040000 - 0x0004FFFF - TWAIN specific errors (MSG_GETNEXT)
---
0x00040000 - Cannot continue data sources enumeration
0x00040001 - Failure due to unknown causes
0x00040002 - Not enough memory to perform operation
0x00040005 - DSM error

0x00050000 - 0x0005FFFF - TWAIN specific errors (MSG_OPENDS)
---
0x00050000 - Cannot open data source
0x00050001 - Failure due to unknown causes
0x00050002 - Not enough memory to perform operation
0x00050003 - No data source found
0x00050004 - Data source is connected to max possible applications
0x00050005 - DSM or DS error

0x00060000 - 0x0006FFFF - TWAIN specific errors (CAP_XFERCOUNT)
---
0x00060000 - Cannot setup number of images to transfer (capability value 1)
0x00060001 - Failure due to unknown causes
0x00060002 - Not enough memory to perform operation
0x00060005 - DSM or DS error
0x00060006 - Unknown capability
0x0006000A - Data parameter out of range
0x0006000D - Capability not supported by source
0x0006000E - Operation not supported by capability
0x0006000F - Capability has dependancy on other capability

0x00070000 - 0x0007FFFF - TWAIN specific errors (MSG_ENABLEDS)
---
0x00070000 - Cannot enable data source
0x00070001 - Failure due to unknown causes
0x00070002 - Not enough memory to perform operation
0x00070005 - DSM or DS error

0x00080000 - 0x0008FFFF - TWAIN specific errors (DAT_IMAGENATIVEXFER)
---
0x00080000 - Cannot perform image transfer
0x00080001 - Failure due to unknown causes
0x00080002 - Not enough memory to perform operation
0x00080005 - DSM or DS error

0x00090000 - 0x0009FFFF - TWAIN specific errors (MSG_ENDXFER)
---
0x00090000 - Cannot complete/cancel image transfer
0x00090001 - Failure due to unknown causes
0x00090002 - Not enough memory to perform operation
0x00090005 - DSM or DS error