

*The extensive Application  
Development Framework that  
migrates VFX for Visual FoxPro  
Applications to Silverlight!*

# Silverswitch 1.0



# Developer Manual

## **Copyright**

Visual Extend is a product from dFPUG c/o ISYS GmbH. Any reuse of VFX related material needs the written permission of dFPUG c/o ISYS GmbH; also VFX related publications must have the copyright notice of dFPUG c/o ISYS GmbH.

# Table of Content

Copyright .....	2
Table of Content.....	3
Introduction.....	5
Development Tools .....	6
Visual FoxPro 9 and Visual Extend 13 .....	6
Visual Studio 2010.....	6
Silverlight 4 Tools.....	6
Silverlight 4 Toolkit .....	6
Programming Languages .....	6
Installation.....	7
Start Page .....	8
Creation of a New Project.....	9
VFX – Silverlight Wizard .....	10
What does the wizard do? .....	10
Select Silverlight Solution.....	10
Data access .....	11
Application settings.....	12
Form Selection .....	12
Enter project names for forms.....	12
Select Form Properties .....	13
Report Selection.....	14
Start of the New Silverswitch Application.....	15
Further development with Visual Studio.....	16
Silverswitch Builders and Wizards .....	17
Update Project.....	17
VfxPickField Builder .....	18
DataGrid Builder.....	19
Architecture.....	20
Server projects.....	20
Client projects .....	20
Class Hierarchy .....	20
Application Object .....	21
Properties .....	21
Forms .....	24
Properties .....	24
Limit the Amount of Loaded Data .....	24
Login Behavior and Client Selection .....	25
Login Behavior Values .....	25
Login Behavior – NoLogin .....	26
Login behavior – LoginAtStartup .....	27
Login behavior – OptionalLogin.....	29
Execution of VFP Code .....	30

GUINEU .....	30
Using VfxControl:VfxActionButton .....	31
With VFP developed functions execution .....	32
Server Site .....	32
Client Site .....	33
Parameter Structure and Passing Parameters .....	33
Multilingual Applications .....	35
Runtime Localization .....	35
Localization of forms .....	36
Localization of MessageBoxes .....	37
Required Fields .....	38
Upload and Download files .....	39
Display Pictures .....	39
Properties .....	39
Upload Files .....	39
Properties .....	40
Data Handling .....	42
Error tracking .....	43
Resource Table .....	44
Further Information .....	45
Forum, Newsgroup .....	45
Technical Documentation .....	45
Tips .....	46

## **Introduction**

Silverswitch is an extensive development environment for the development of business applications with Microsoft Silverlight.

In connection with Visual Extend for VFP the VFP developer is presented with the simple task, to migrate existing VFX for VFP projects to Silverswitch. The converted projects have similar behavior in comparison to the original VFX for VFP applications. With Silverswitch the developers discover completely new perspectives.

The migrated Silverswitch applications are Internet applications which work with VFP as well as SQL Server databases even if the original VFX for VFP application doesn't use SQL Server database.

Silverlight applications run on Windows and Mac OS. The number of platforms which support Silverlight is constantly growing.

With Silverswitch it is possible to execute VFP code on both the client-side and the server-side.

## Development Tools

The following development tools are required for Silverswitch.

### Visual FoxPro 9 and Visual Extend 13

The migration of existing VFX for VFP projects is realized with the VFX – Silverlight Wizard. To use this assistant, installations of VFP 9 and VFX 13 are required. The VFX – Silverlight Wizard is included in the full version of VFX 13 but not in the trial version.

The trial version of VFX 13 can be downloaded from the website of Visual Extend:

<http://www.visualextend.com>

### Visual Studio 2010

Silverswitch applications are developed with Visual Studio 2010 and Silverlight 4.

For the development of Silverswitch projects any of the available Visual Studio 2010 versions can be used. It is possible to use the free available version Visual Web Developer 2010 Express. Throughout the handbook it is assumed that Visual Web Developer 2010 Express is being used, designated by the general term „Visual Studio“. Other versions of Visual Studio can have more advanced capabilities, which however are not necessary for the development of Silverswitch applications.

Visual Web Developer 2010 Express can be downloaded from the following website of Microsoft:

<http://www.microsoft.com/express/Downloads>

### Silverlight 4 Tools

After installing Visual Studio, Silverlight 4 Tools also have to be installed. Silverlight 4 Tools can be downloaded from the following website:

<http://www.silverlight.net/getstarted>

### Silverlight 4 Toolkit

Silverlight 4 Toolkit can be downloaded from the Codeplex website:

<http://silverlight.codeplex.com>

Silverlight 4 Toolkit includes controls, based on the existing Silverlight 4 base classes and offers extended functionalities.

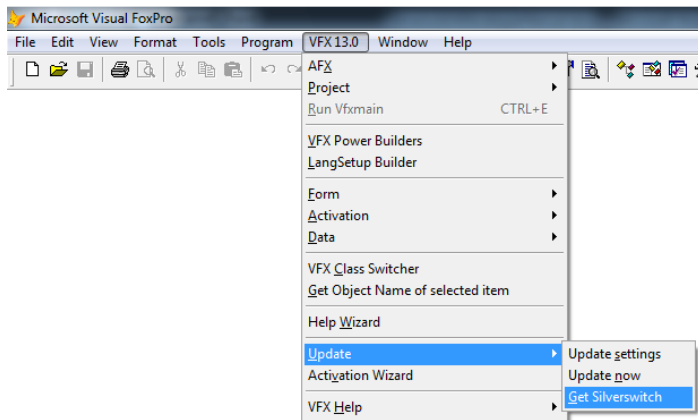
### Programming Languages

The development of Silverswitch applications is realized with the C# programming language. Programming with VFP is possible by using GUINEU. In addition, VFP code can be executed in COM servers.

The user interface is described with the XAML language.

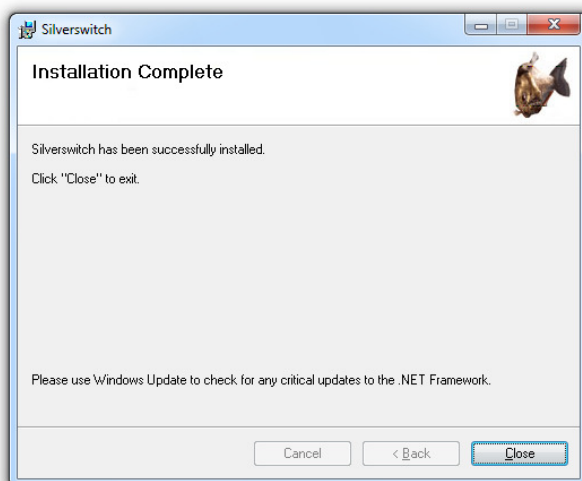
## Installation

Silverswitch is an extension to Visual Studio and is installed with a Windows MSI Installer. The installer has the name Silverswitch.msi. From VFX for VFP the installation package can be downloaded through the “Update” menu entry and a click on “Get Silverswitch”.



All instances of Visual Studio should be closed during installation.

The installer detects all installed versions of Visual Studio.



After successful installation the extension template "Silverswitch" appears in the "New project" dialog under the Visual C# category. Based on this template project new Silverswitch applications are developed.

The de-installation has to be done by the control panel.

## Start Page

Silverswitch contains a start page for Visual Studio. It is set as default start page by the installer. The start page can be changed in the Tools/Options dialog of Visual Studio.



On the left panel of the Start Page there are options for managing projects identical to these on the start page of Visual Studio. New projects can be created and existing ones can be opened. A list of recently used projects is also available.

Several pages with information are on the Start Page of Silverswitch. They are accessible through the tab buttons. The first page appears at startup and shows the version number of Silverswitch as well as logos with links to the developer, sponsor and distributor of Silverswitch. The information on the following pages appears only if there is an Internet connection.

The second and third page displays the last 50 messages of the German speaking and English speaking forum. The following page shows the portal page of dFPUG. Many files can be downloaded from the portal. The next page shows the website of Silverswitch. There is also a page with links about Silverswitch. Finally a very important page will be discussed: the project menu. Its functionality needs some special attention.

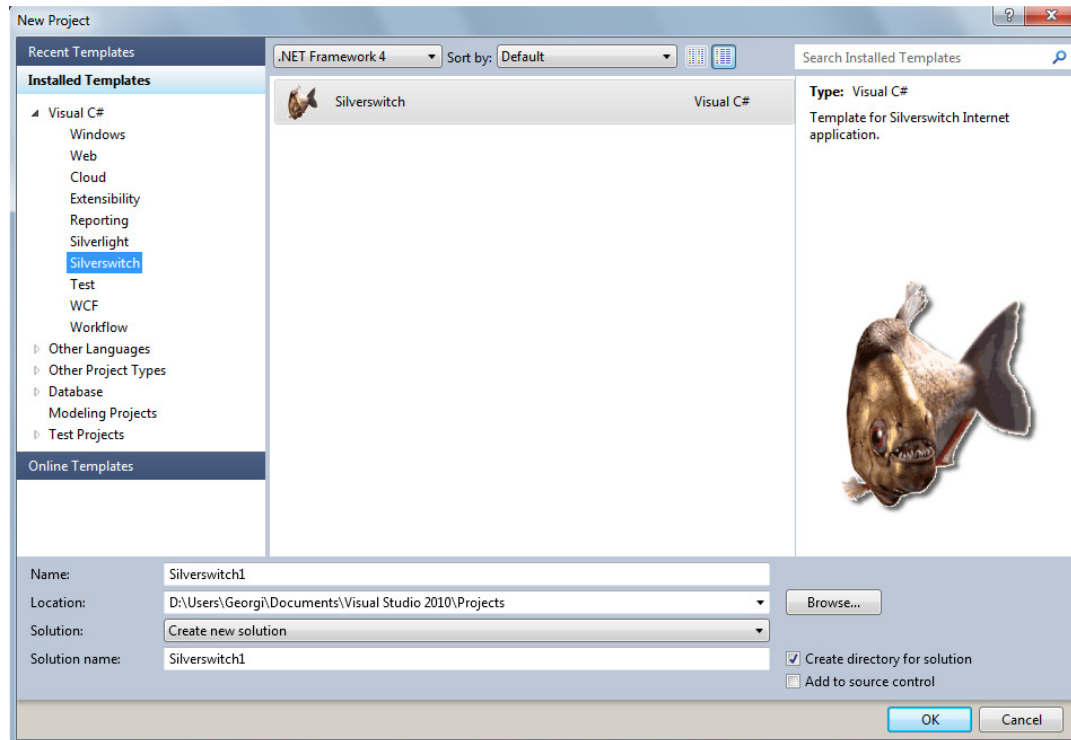


## Creation of a New Project

To create a new project based on Silverswitch in Visual Studio the dialog “New project” must be opened via the start page, by selecting the menu entry “File”, “New project...”, or by clicking the toolbar button “New project”.

In this dialof “Silverswitch” can be found under “Visual C#”, “Silverswitch”. Then, in the middle part of the dialog select “Silverswitch”.

The project name as well as folder names and form names have to much the naming conventions for classes. So no spaces or special characters are allowed.



The new project should get the name of the VFX application which should be migrated. Additionally, a folder must be specified in which the new project is stored. This folder must be selected in the VFX – Silverlight Wizard later.

The project contains a COM Server which is developed with VFP. The project is intended to be proceeded with the VFX – Silverlight Wizard. Direct execution is not expected but possible if the COM server is registered.

## VFX – Silverlight Wizard

The most development step is the migration of an existing VFX for VFP application to Silverswitch. The VFX project must be opened with VFP.

If the prepared Silverswitch solution should remain open in Visual Studio while the VFX - Silverlight Wizard runs, all files must be saved in Visual Studio before the VFX - Silverlight Wizard starts.

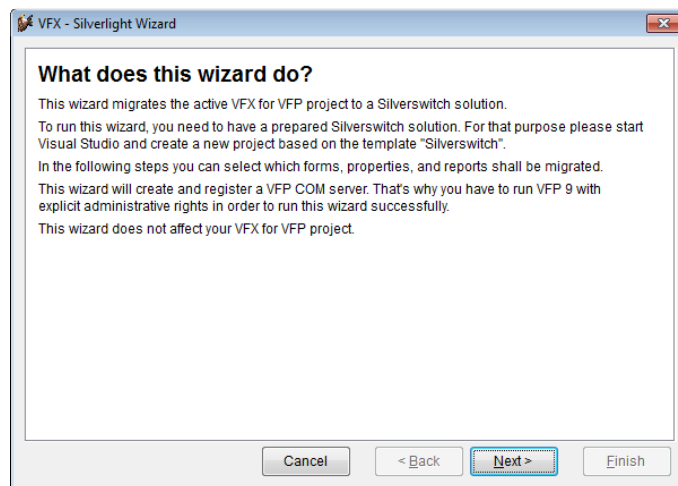
The VFX – Silverlight Wizard is integrated in Visual Extend for VFP and can be started via the menu item "VFX 13.0", "Project", "Silverlight Wizard". The assistant allows the migration of forms and reports to Silverswitch.

The assistant creates and registers a VFP COM server. On Windows Vista and newer Windows versions VFP 9 must run explicitly with administrator rights.

The migrated Silverswitch application accesses the same database as the VFX for VFP application. Each table in the database must have a primary key.

### What does the wizard do?

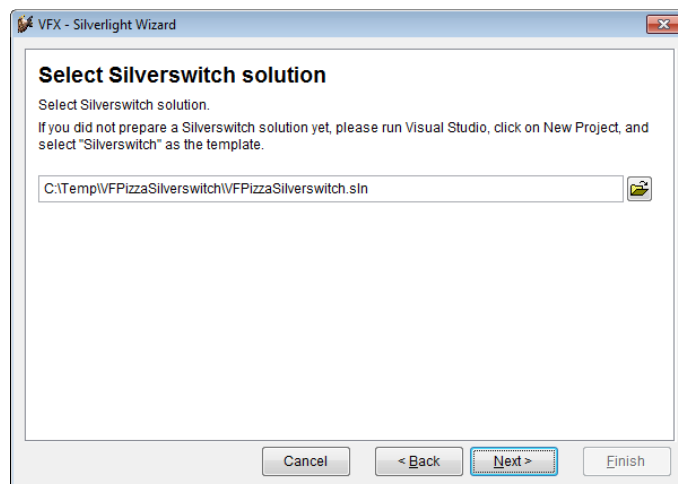
The first step explains the function of the wizard. The wizard helps you to migrate an active VFX project to a Silverswitch solution. There has to be a prepared Silverswitch solution to complete this wizard. Through the next steps of the wizard can be selected which forms, features and reports should be migrated.



The wizard does not affect the VFX for VFP project.

### Select Silverlight Solution

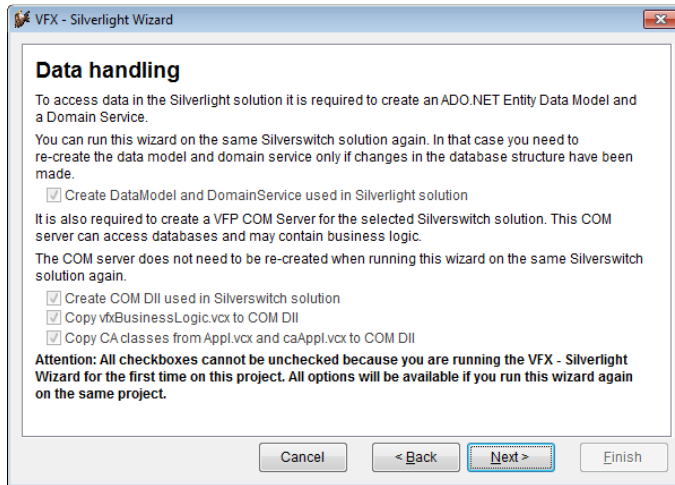
In this step the previously prepared Silverswitch solution is selected.



The forms and reports which the VFX – Silverlight Wizard generated, will be added to the selected solution.

## Data access

This step adds components for data access to the Silverswitch solution.



When working with a SQL Server database the Silverswitch application accesses data through an ADO.NET Entity Data Model and a domain service. Both components are generated by the wizard. If the wizard runs on the same solution again and in the meantime changes to the data structure are done, the wizard will generate new ADO.NET Entity Data Model and Domain Service.

The first time the wizard runs, a VFP COM server is created. Through the COM Server access to VFP databases is realized. When VFP databases should be used, the database can be accessed through the VFP COM Server. Business logic created with VFP can also be executed in the COM Server.

The VFX – Silverlight Wizard looks for the file config.vfx in the VFX project folder. If the file exists, the wizard generates another config.vfx, in which all paths are replaced with full paths. That new file will be copied to the folder VFPComDomainService of the Silverswitch solution. It will be used for data handling via the COM Server Dll. If there is no config.vfx in the VFX project folder, a new config.vfx will be created.

The wizard looks for SQL databases in config.vfx. If one or more are found the client name will be added and connection strings to them will be defined in <application name>.Web\Web.Config file, too.

Connection to SQL databases will be defined in Web.Config only in case the VFX system tables are in the SQL database, too. In case the VFX tables are in a DBC the connections to this row will be defined only in config.vfx but not in Web.config.

**Create DataModel and DomainService used in Silverlight solution** - Based on WCF RIA Services technology a data model and a domain service are generated.

The wizard checks the solution for a data model named <database name>Model.cs in the folder VfxDataLayer.Web\Models and domain service with the name <database name>DomainService.cs in the folder VfxDataLayer.Web\Services. If one of them does not exist, creation of DataModel and DomainServices is selected. By default in that case they will be created.

Model and service files are generated in the Silverswitch template project. In the model every table structure is implemented in a corresponding class with the same name. In the service are implemented Get, Insert, Update, Delete and ClearPersisted methods for each table.

For generating model and service is used a SQL database if there is one in config.vfx. If there is not, the database from the first row in config.vfx is used. If there is no config.vfx file, the database included in the project is used.

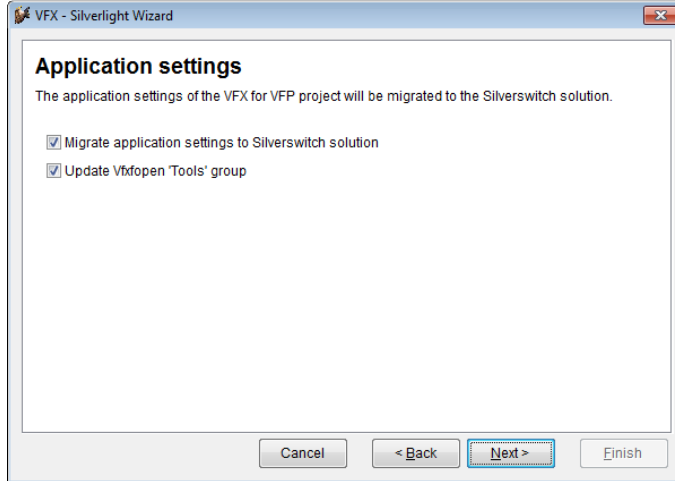
Nevertheless, the generated model and service can handle both SQL and VFP databases if they have the same structure.

It is important the database that is used for model and service generating to have primary keys and relations defined. The primary keys and relations are defined in the generated model and used for database handling.

**Create COM Dll used in Silverlight solution** - If selected, a COM Dll project is created in the folder \VfxDataLayer.Web\VFPComDomainService of the Silverswitch solution. A new Dll will be built and referred as well. The name of the new COM Dll project will be <Project name>COMDomainService. So there will be a specific COM Dll project and specific register Dll for the Silverswitch solution if that option is chosen. COMDomainService.dll came with the Silverswitch template project and will be unregistered and deleted.

## Application settings

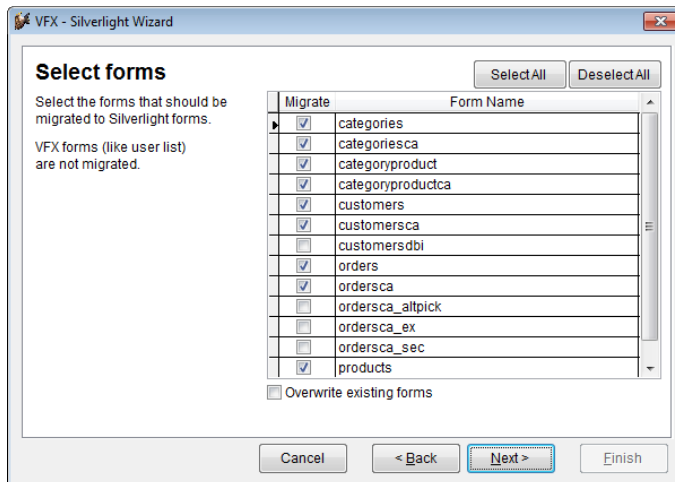
Application settings can be migrated from the application object of the VFX project.



In VFX applications VFX forms can be started by the tools menu. Since there is no menu in Silverswitch applications an additional group can be created in the open dialog to open VFX forms.

## Form Selection

In this step forms of the VFX for VFP project are listed. By default forms which are included into the VFX for VFP project are selected. VFX forms that are included in the template project are not in the list and are not going to be migrated. An example of such a form is the user list.



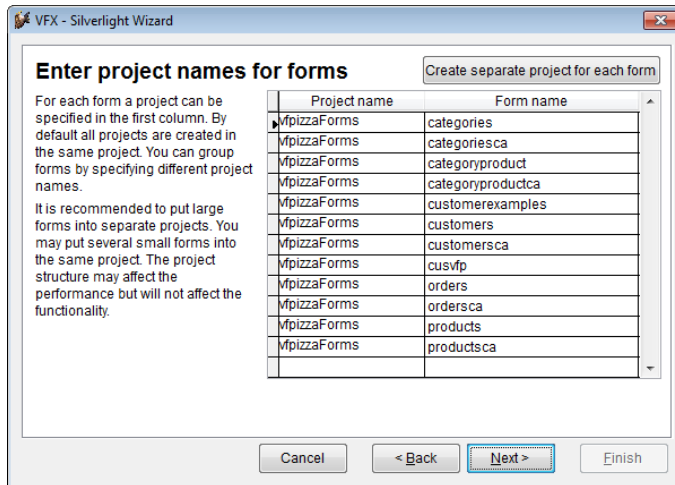
If the data environment of a form uses a data source that does not have a primary key, a warning appears in the log of the wizard. The form can be migrated to Silverswitch and the migrated form can run, but processing of data is not possible as long as no primary key is added to the data source.

## Enter project names for forms

In this step forms can be grouped in projects. By default all forms are generated in a project with a name <Projectname>Forms. It is possible to choose custom project names. It is also possible to include any number of forms into one project. When the Silverlight client application is built, a XAP file is generated from each project and that file is transferred to the client through the Internet at runtime. By dividing the application into separate projects the size of the XAP files is being decreased and consequently the load time is shortened.

Forms can be grouped selectively into projects. It makes sense to put extensive forms into their own project while smaller forms are grouped together in collective projects. A click on the button „Create separate project for each form“ generates a project name for each form.

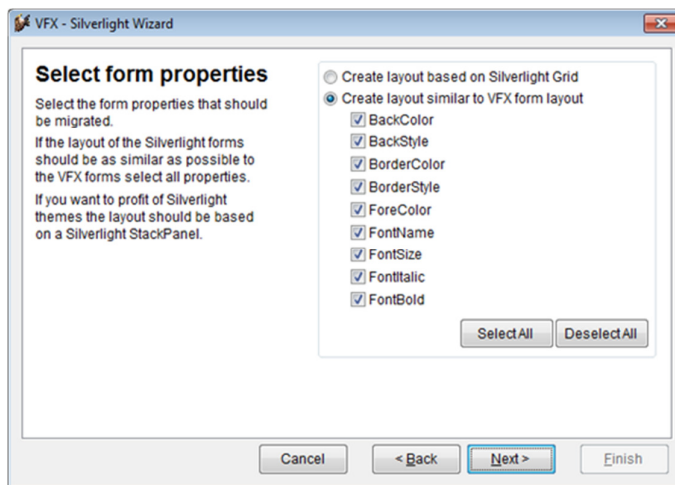
New client projects of the Silverswitch solution will be created for every selected VFP form. The name of the new project will be View<form name>. XAML files will be saved in the folder Views of the corresponding Silverswitch project.



## Select Form Properties

By default the Silverlight forms are generated with a layout that is close to the layout of the corresponding VFP form. To achieve this all layout properties are migrated to Silverlight.

If the end-user should be able to use styles, not all layout properties should be migrated to Silverlight. All properties which are hard-coded in Silverlight forms cannot be changed by styles at runtime.



“Create layout based on Silverlight StackPanel” – The position of controls in the VFP form is not migrated. Rather the layout of the generated Silverlight form will be positioned in a Silverlight Grid.

A Silverlight grid is similar to a HTML table. In this table typically a column for labels and a column for text boxes and other controls for data entry are placed. The advantage of this layout is that forms are better scalable. Such forms can be displayed properly on devices with small screens.

In general for each field in a VFX form there is a label followed by another control like a textbox, or an editbox. Both controls belong together. The VFX form builders set the tag property of the label to the same value as the other control has as controlsourc. If the VFX – Silverlight Wizard generates the layout based on a Silverlight Grid the connection between the controls is recognized and both controls are put in a row of a Grid. Each control is generated in its own column. So the first column of the Grid will contain the label and the second column will contain the other control like textbox or editbox.

### Requirements:

Labels: class – *clabel*

Tag – corresponding with data source property of data control

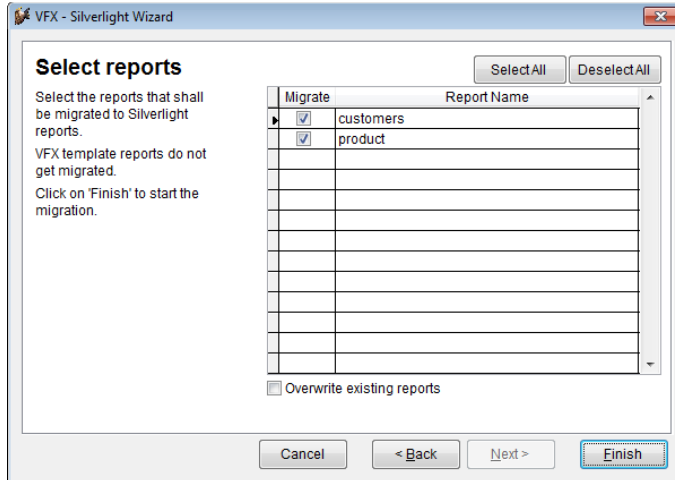
Data controls: class – *ctextbox*, *cpickfield*, *cpickalternate*, *cpickdate*, and others having *ControlSource* property.

**Create layout similar to VFX form layout** – The layout of the generated Silverlight form will be as similar as possible to the original VFX form. The generated Silverlight form will have a Grid as the outmost container.

This container will have just one column and one row. All controls are positioned in this Grid cell using the Silverlight Margin property. This allows absolute positioning of the controls within this cell, similar the use of top and left coordinates in VFP.

## Report Selection

In this step all report files included in the VFX for VFP project are listed. By default, all reports are migrated except VFX template reports, like the template files for grid reports.



Reports must be based on the data sources provided by the calling form.

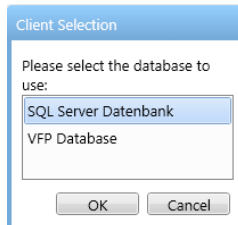
If the Silverswitch solution is open in Visual Studio during the migration process of the VFX – Silverlight Wizard, Visual Studio will detect the ongoing changes in the solution and will open a dialog asking if it should reload the solution. In this case the project has to be reloaded.

## Start of the New Silverswitch Application

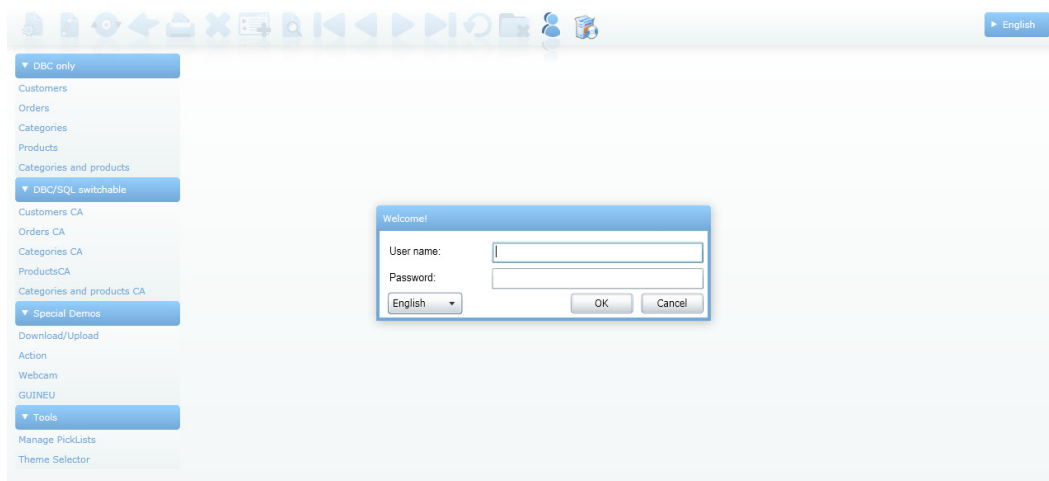
The new created solution can be tested immediately. For this purpose the solution can be started by pressing the green arrow button in the toolbar or by pressing F5.

### Welcome to Silverswitch

The application starts in the default Internet browser displaying a splash-screen and a progress bar. If the application is configured to use more than one data source, a client selection dialog appears.



A login window appears. After successful login the application runs. A toolbar appears at the top and left the open dialog appears.



By default each user of a Silverswitch application must be logged in with user name and password. If authentication is not activated, the login dialog is not shown.

## **Further development with Visual Studio**

Further development of the Silverswitch application can be done in Visual Studio. It is also possible to modify the forms of the application in VFX for VFP, and then to migrate the modified forms with the VFX – Silverlight Wizard again.

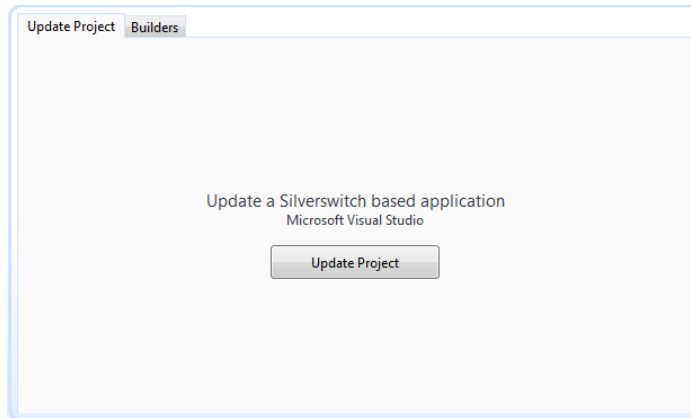


## Silverswitch Builders and Wizards

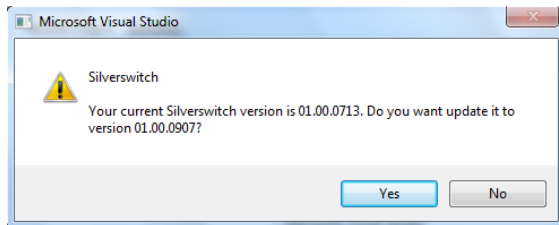
Choosing the Project Menu Tab will lead you to the page below.

### Update Project

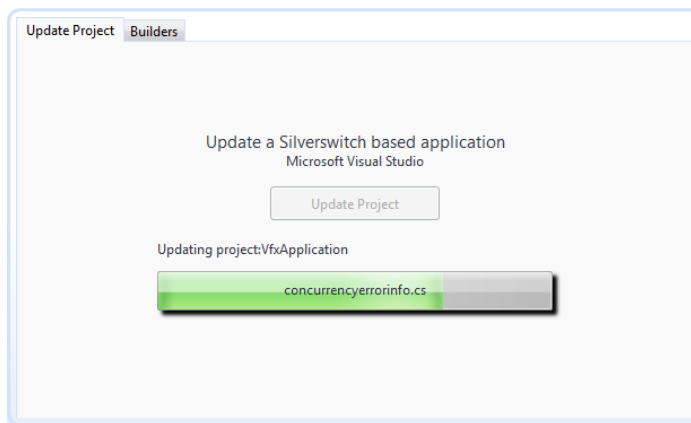
A click on the Update Project button will check the current version of your Silverswitch project.



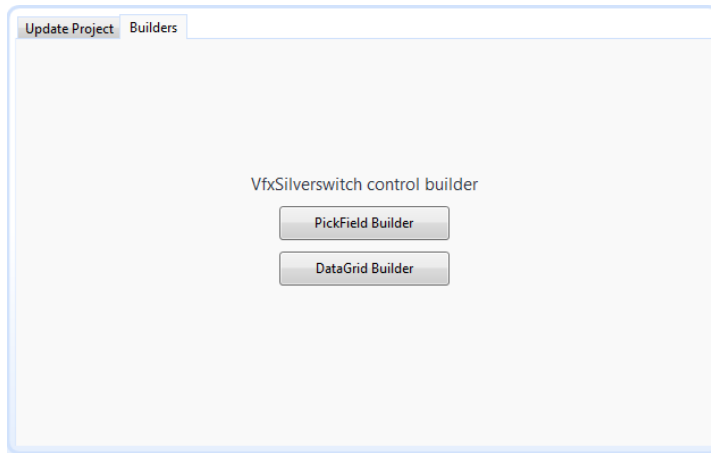
If the installed version of Silverswitch is newer than the version of your project a message box appears.



Your approval will start the update process.

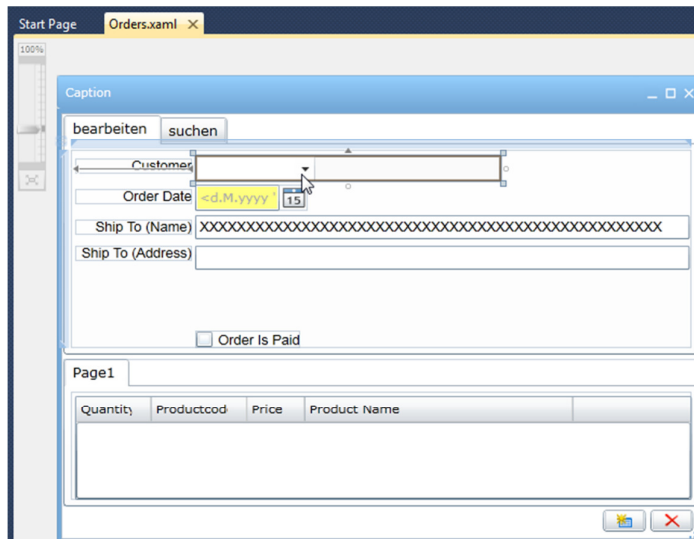


On the second tab of the Project Menu page the user can benefit from builders for different controls.



## VfxPickField Builder

In order to use the builder for a specific control a form should already be opened in designer. This form should contain a control of type corresponding to the builder. Keep the start page open and go to the form which is being designed. Select the control which you wish to use with the builder.



In this case the selected control is VfxPickField. Go back to the start page and Project Menu section. Click the Builder button. The builder starts. You can change properties of the control and save the changes to the designed form by clicking the OK button.

Silverswitch - PickField Builder

**Table Name**  
customers

Select	Field Name	Header Name
<input checked="" type="checkbox"/>	customerid	customerid
<input checked="" type="checkbox"/>	customername	customername
<input type="checkbox"/>	address	address
<input type="checkbox"/>	contactperson	contactperson
<input type="checkbox"/>	phone	phone
<input type="checkbox"/>	phonetype	phonetype
<input type="checkbox"/>	email	email
<input type="checkbox"/>	skypename	skypename
<input type="checkbox"/>	url	url

**Return Expression**  
customername

**Return Expression Description**  
customername

OK Cancel

## DataGrid Builder

Silverswitch - DataGrid Builder

Read Only  Selected

Row Details Visibility Mode  
VisibleWhenSelected

**Table Name**  
orders

**Table columns**

Select	Field Name
<input checked="" type="checkbox"/>	orderdate
<input checked="" type="checkbox"/>	shiptoname
<input type="checkbox"/>	totalsum
<input type="checkbox"/>	paid
<input type="checkbox"/>	shiptoaddress
<input type="checkbox"/>	customerid
<input type="checkbox"/>	orderid

Auto Fit  
Width 80

Control Type  
Header Order Date

Header Picture  
Control Source orderdate

Input Mask  
 Read Only  
 Incremental Search

Select All Deselect All

OK Apply Cancel

## Architecture

A Silverlight Internet-Application consists of at least two projects: a server project and a client project. The server project is running on the server side and contains the data access and business logic.

The client project is the actual Silverlight project and is running on the client computer. The client project is split into many small subprojects for optimization reasons.

### Server projects

Silverswitch applications have two server projects.

The startup project of the solution has the name `Silverswitch.Web`. It is running on the Internet Information Server. This project contains the main page `default.htm`.

The user enters an Internet address in his browser to start the application. `Default.htm` loads and detects if the Silverlight plug-in installed and executes the application on the client-side.

The second server project is named `VfxDataLayer.Web`. This project contains the data model, that allows access to databases, and domain services which allow the exchange of data between client projects and server projects.

### Client projects

When compiling a Silverlight project a file with the name extension `.XAP` is created. In fact the file is a ZIP archive of the DLLs that contain the logic and resources of the project.

The user types the address of the application in his browser. The main page contains the name of the initial XAP file and the Silverlight Plug-In version required. Then the browser downloads the initial XAP file over the Internet and executes it with the Silverlight Plug-in. If the required Plug-in version is not installed the user is directed to the Silverlight download page.

Silverswitch is built by the module concept. The application is executed with loading the file `VfxLoader.xap`. Subsequently `VfxMainScreen.xap` and `VfxForm.xap` are loaded.

### Class Hierarchy

The class hierarchy implemented in Silverswitch is similar to the hierarchy that was implemented in VFX for VFP. The functionality of Silverswitch is contained in class files with the prefix "Base". From these class files one to one inheritance exist in class files without the prefix "Base". Custom extensions can be integrated in the latter mentioned class files.

## Application Object

The application object is defined in the VfxApplicationBase project in the program file Controls\VfxAppObjectBase.cs. A one-to-one derivation for individual adjustment exists in the VfxApplication project in the class file Controls\VfxAppObject.cs.

The application object provides global properties and methods. Settings can be applied to the properties of the application object that will affect the behavior of the entire application.

### Properties

**AddUserToCaption** – Determines whether to add the user name to the application caption.

**AllowRelogon** – Allows the user to login again without leaving the application.

**AllowUserCustomization** – If the value of this property is set to true, logged-in users can apply individual settings to the application. If no user is logged, no individual settings can be made. When the value of this property is set to false, no user can make individual settings.

**ApplicationName** – This property specifies the name of the application. The name appears in the toolbar next to the icon of the application.

**AskToSave** – Determines if there will be a MessageBox asking for saving prior to some operations.

**AutoEdit** – Determines if a form will enter edit mode on interactive change of controls. Has the following settings: AlwaysEnabled, AlwaysDisabled, UseObjectSettings

**Century** – This property sets the format of the century in date fields. If this property is false, the century is not shown, the year appears in two digit format (yy). If this property is set to true, the year is displayed in four digit format (yyyy).

**ChildGridCopy** – Allows to copy child grid records. Default value is GlobalSettings.AlwaysDisabled.

**CurrentConnectionInfo** – Global property for the current connection. Holds information about database type client name and availability.

**CurrentLanguage** – Global property for current selected language.

**DisableFormResize** – When the value of this property is set to true, users cannot change the size of forms at runtime. If false, forms are resizable at runtime. During the closing of a form the user selected size is saved in the table VfxResources and restored the next time the form is opened.

**EditDateFieldName** – The date of the last edit of a record.

**EditTimeFieldName** – The name of the field that stores the time of last modification of a record. If a field with the specified name is in a table, the value is automatically populated every time a record is saved. The type of the field must be C(8).

**EditUserFieldName** – The name of the field in a table that stores the name of the user who last edited the record. If a field with the specified name is in a table, the value is automatically populated each time a record is saved. The field must be of type char length 32.

**ErrorDetailLevel** – Determines how detailed the log of an encountered exception will be. The exception log is stored in the table VfxLog. It has three levels:MessageOnly,CallStackInformation,FullDetailedInformation.

**FilterFieldsSource** – Global behavior for FilterDialog used for all forms.

**HideOpenDialog** – Determine whether OpenDialog will be visible in the application.

**HideWhenEmpty** – This property can determine whether controls are hidden when no records are stored in a table. In this case, a note appears on the form. When the user clicks on this note, the form is switched into new mode.

**InsertDateFieldName** – Contains the name of the field which stores the date of the creation of a record. If a field with the specified name is in a table, the field is filled automatically during the first save of a record. The type of the field should be date or DateTime. If the field is of type DateTime a timestamp is inserted when saving.

**InsertTimeFieldName** – The name of a field that stores the time of the creation of a record. If a field with the specified name is in a table, the value is filled automatically during the first saving. The type of the field must be C(8).

**InsertUserFieldName** – The name of a field in a table that stores the name of the user that has created the record. If a field with the specified name is in a table, the value is filled automatically during saving. The field must be of type char with length 32.

**KeepUserLog** – Keep login/logout history for users.

**LangID** – If runtime localization is not used, the abbreviation of the default language is specified in this property. The default is English.

**LoginBehavior** – This property specifies whether a user login is required at startup, optional or not required. It determines the startup behavior of the application.

Optional login (default behavior)

The application starts without login dialog. It uses the default database. This is the first database that is found in the file Web.config. If no database is available in the file Web.config, the first database in the file Config.vfx is used. While the application is running, a login by clicking on the “Login” in the toolbar button is possible. This setting is suitable for applications where some functionality will be accessible to public, while other parts of the application are protected by user login. This way Internet-applications are made possible, which are read-only without login while a write access requires a login.

LoginAtStartup

The application starts with user login. This behavior is the same of VFX for VFP applications. If the Web.config and Config.vfx files found more than one database, the client selection dialog appears before the user login dialog. This is recommended for Internet and Intranet-applications, which should not be public accessible.

NoLogin

The application starts without login. It uses the default database. This is the first database that is found in the file Web.config. If no database is stored in Web.config, the first database in the file Config.vfx is used. A user login is not possible. This setting is recommended when an Internet application should be available to the public or non-public intranet applications where user login should not be used.

**MainForm** – The form specified in this property opens automatically when the application starts. If user login is required, the form will open after login.

**MaintenanceServerMethodName** – Server method responsible for maintenance requests.

**MaintenanceTimerFileName** – Name of the file which appearance in maintenance folder will stop the application.

**MaintenanceTimerInterval** – Interval in seconds after which the application will check for appearance of maintenance file in the maintenance folder.

**MultiInstance** – Determines globally if forms can be opened more than once.

**PasswordHistoryCount** – Determines the number of unique new passwords that have to be associated with a user account before an old password can be reused.

**PasswordLength** – This property specifies the minimum length of passwords. If this property is set to 0, a password is not required.

**PasswordStrengthLevel** – Indicates the strength level of required password. Four levels of security are available.

Weak – No requirements.

Medium – The password must contain characters from minimum 2 of the following 4 categories:

Uppercase characters (A through Z)

Lowercase characters (a through z)

Base 10 digits (0 through 9)

Non-alphabetic characters (for example,!, \$, #, %)

Strong – The password must contain characters from minimum 3 of the above mentioned 4 categories.

Best – The password must contain characters from all of the above mentioned 4 categories.

**PasswordValidityDays** – This property sets the period of validity of passwords. If a password has expired, the user is prompted for registering a new password during login. Login is not possible without entering a new password. If the value of this property is 0, passwords remain valid indefinitely.

**PathToApplicationLogo** – Path to application logo.

**RelogonQuit** – Determines if the application should be closed if the user clicks cancel in the login dialog.

**RequiredFieldFailureStyle** – This property sets the style of the mandatory fields if they are empty when saving is initiated.

**RequiredFieldInitStyle** – This property sets the style of the mandatory fields. The default is “Required”.

**RuntimeLocalization** – If the value of this property is set to true, the application localization is executed at runtime. All texts to be displayed are read at runtime from the table VfxMsg and displayed in the selected language. If the value of this property is set to false, no texts from the table VfxMsg are read at runtime.

**SaveFormLayoutResolutionDependent** – Defines if a form’s layout should be saved depending on the user resolution.

**ShowFilterActivatedInFormCaption** – Indicates if a message is shown in form title when a filter is applied.

**ShowIntroForm** – This property determines whether to display the splash screen.

**ShowNTLogonFieldInUserManagement** – When true Ntlogonfield is displayed in user list.

**WorkAliasRecordsLimit** – Specifies the quantity of loaded records.

## Forms

Forms have properties that control the behavior of the form at runtime.

### Properties

Properties of forms are specified in the file with the extension .xaml.cs of a form.

**AskToSave** – If the value of this property is set to true, the user is asked to save changes when attempting to close the form and any unsaved change exists.

**AutoEdit** – If the value of this property is set to true, the controls on a form are enabled. The manipulation of data can be started immediately. If the value of this property is set to false, the controls on a form are disabled. If the user wants to edit the data, he must click the “Edit” button first to toggle the form into edit mode and the controls will switch their mode to “enabled”.

**AutoResizeControl** – Specifies if the controls are resized automatically.

### Limit the Amount of Loaded Data

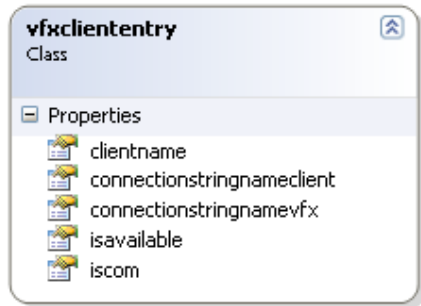
For quantity restriction of loaded records the property `WorkAliasRecordsLimit` can be used. The restriction can be applied to the application object as well as to the controller of each particular form. Values from 1 to the desired number of loaded records can be set to this property. Any limitation set to the controller of a form overrides the limitation of the application object. There are also some special values. If -1 is set to the property then there is no requirement for quantity restriction. In other words all records will be loaded. When 0 is set to the property of the controller the limitation of the application objects is considered as default. 0 set to the property of the application object has the same effect as -1 (all records loaded). The following table summarizes all possible states:

<code>VfxAppObject.WorkAliasRecordsLimit=</code> <code>Controller.WorkAliasRecordsLimit=</code>	-1	0	Any number $b > 0$
-1	All records loaded	All records loaded	All records loaded
0	All records loaded	All records loaded	b records loaded
Any number $a > 0$	a records loaded	a records loaded	a records loaded



## Login Behavior and Client Selection

The application can run with different client databases. “Client” is a database with specific structure and logic. There are two ways to set clients that an application can use. One way is through the Config.vfx file and COM Server that the application uses. Another way is to include a client into the file web.config. Clients included in one of the mentioned files are acknowledged through the query method GetAvailableClients in VfxSystemDomainService. This method returns all clients that an application can use. The order of extraction of the clients is important. First are taken into account all clients from Web.config after that all clients from Config.vfx. GetAvailableClients returns the type IQueryable<vfxcliententry>. The vfxcliententry class contains information about the name of clients, and connection names.



If no client database is available the following message is shown:



## Login Behavior Values

Silverswitch supports 3 different login behaviors:

**No login** – Application doesn't enforce user authentication.

**Optional login** – It is possible to start the application without user authentication but it is also possible to login later through VfxLoginDialog.

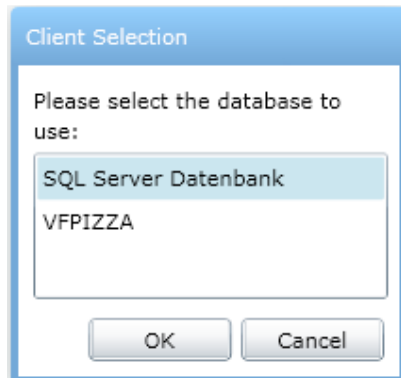
**Login at start up** – The user must authenticate at the beginning when the application starts.

To set specific login behavior change the value of property LoginBehavior in VfxAppObject. This property is of type enum with name LoginBehaviors. Available values are:

- LoginBehaviors.LoginAtStartUp
- LoginBehaviors.NoLogin
- LoginBehaviors.OptionalLogin (Default value)

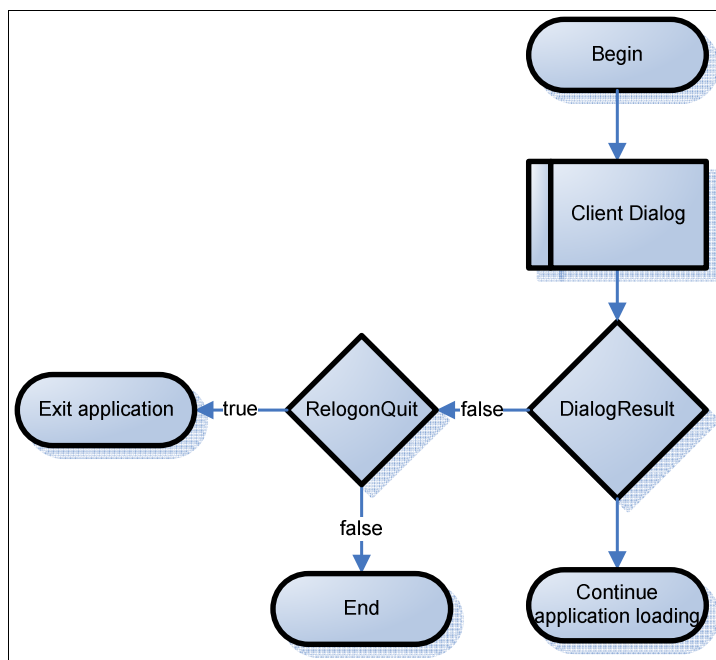
## Login Behavior – NoLogin

This value means that the application does not support user authentication. When the application starts the client selection dialog appears, if more than one client database is available. After the user selects a database the application starts.



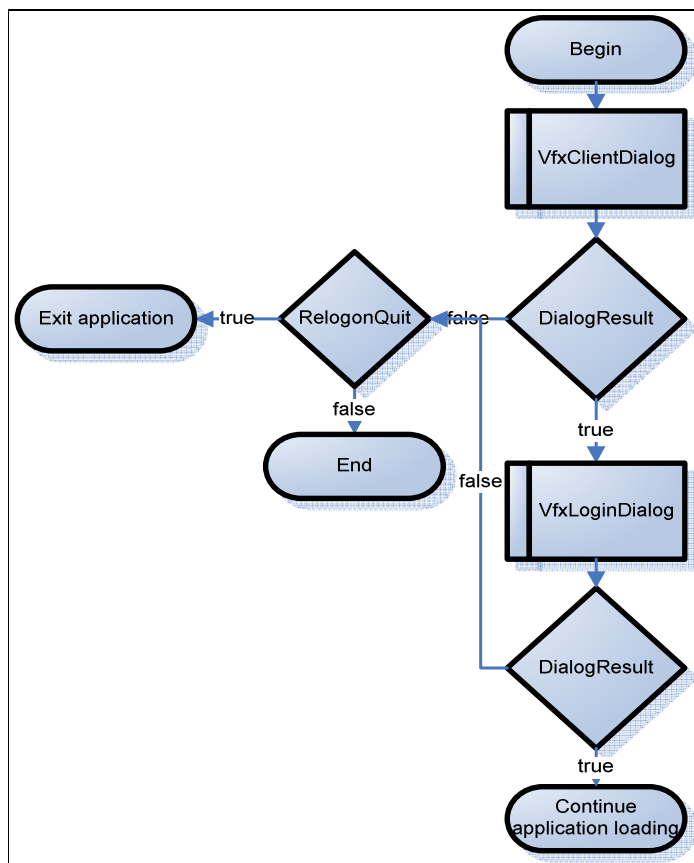
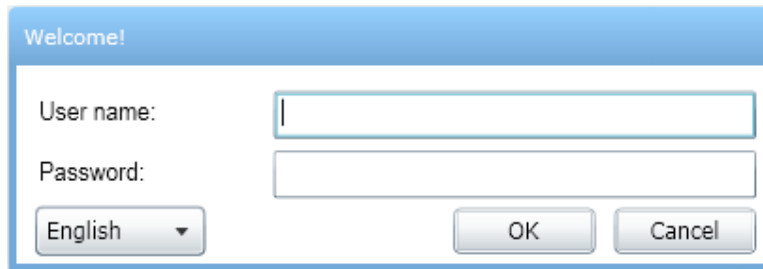
The Relogin button is not available. The user cannot select other clients at runtime.

The next figure shows the schema by which the application works when LoginBehavior is set to NoLogin.




## Login behavior – LoginAtStartup

The user must authenticate at startup of the application. The client selection dialog appears before the login dialog. It is possible that the login dialog appears immediately if the application has only one available client database.

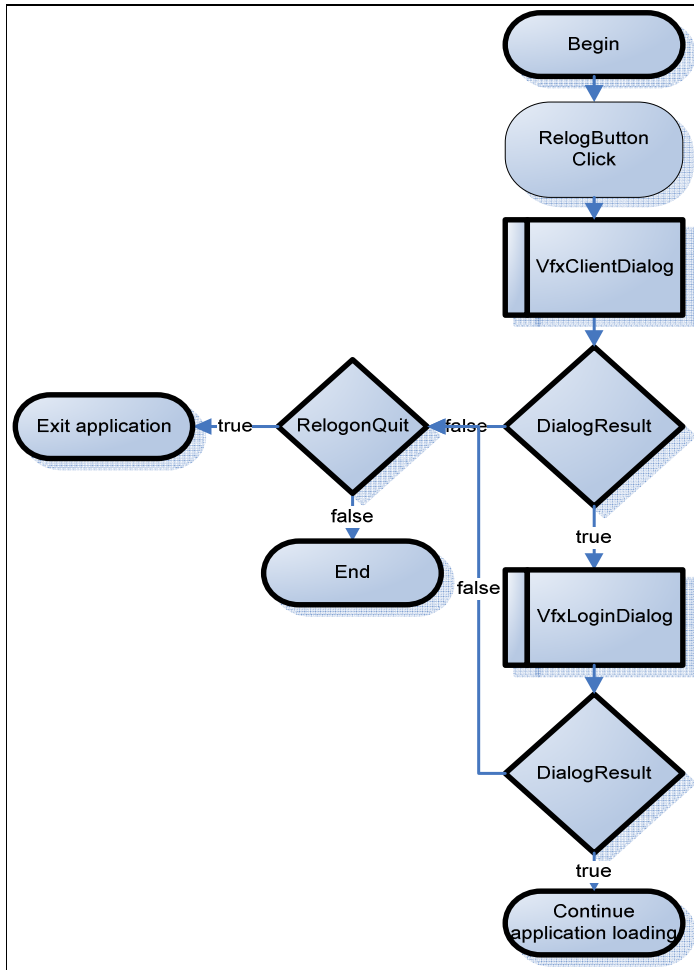


**The figure shows the scheme when LoginBehavior is set to LoginAtStartup.**

With this setting it is possible for the user to login again a different database with different user name. To do so the user must click on the Login button  in the toolbar.




When the user clicks on the Login button the VfxClientDialog appears, after that VfxLoginDialog appears.

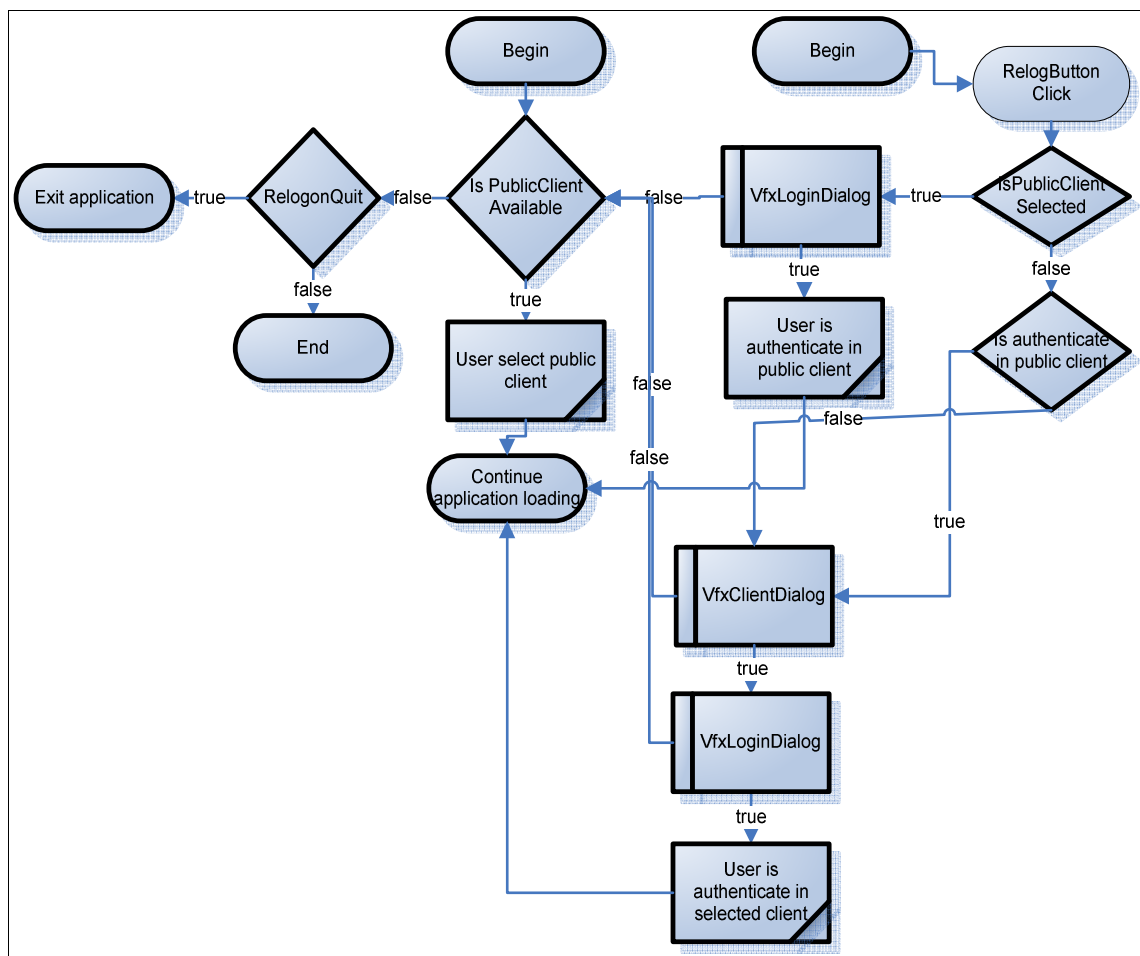


The figure illustrates how the application works when the user clicks on the Login button in the toolbar.

## Login behavior – OptionalLogin

At the beginning the application starts working with the first client database available. With this option the user starts without authentication. Click on the login button  results in different behavior.

1. At first click on the login button the login dialog will show up and the user can authenticate with the first client database. After successful authentication the user continues to work with this database.
2. Click on the login button after the user is authenticated the client selection dialog will show up when there is more than one client database available. After client selection the login dialog is shown and the user must authenticate in the selected client database.



The figure shows functionality schema of the application and RelogButton for LoginBehavior with value OptionalLogin.

## Execution of VFP Code

### GUINEU

VFP code can be executed in all Silverswitch client projects with the help of GUINEU. GUINEU is a VFP runtime environment which executes code in FXP files compiled with VFP.

The GUINEU runtime environment is hosted in the file vfx.guineu.runtime.dll. This DLL is located in the project VfxGuineuRuntime. In the same project is placed also the class VfxGuineu.cs which enables the functionality of GUINEU.

The code has to be created with the development environment of VFP. The compiled FXP file must be included in the respective Silverswitch project.

It is recommended that the PRG file is also included in the Silverswitch project. By double-click on the PRG file in Solution Explorer VFP will start and the file can be edited.

In order to use GUINEU in a C# class a field of type VfxGuineu must be added. The instance is given the name of the compiled VFP file as a parameter.

```
private readonly VfxGuineu _fox = new VfxGuineu("Orders.FXP");
```

The call of functions in the VFP executable file is possible with a single line of C# code.

```
<Output> = _fox.Do("<function name>", <optional parameter>, ...);
```

The evaluation of the return value is optional. As many parameters can be passed as the VFP function can accept. The parameters can be of any type. It is also possible to pass objects of the Silverswitch user interface as parameters. The properties of objects are available in the VFP function and can be accessed.

```
private void xpgfPageFramePage1txtShiptoname_GotFocus(object sender,
System.Windows.RoutedEventArgs e)
{
    _fox.Do("ShipToName_GotFocus", txtTooltip);
}

private void xpgfPageFramePage1txtShiptoname_TextChanged(object sender,
System.Windows.Controls.TextChangedEventArgs e)
{
    OnEdit();
    _fox.Do("Validate", xpgfPageFramePage1txtShiptoname,
xpgfPageFramePage1edtShiptoaddress, chkSpeichern);
}

private void xpgfPageFramePage1txtOrderdate_LostFocus(object sender,
System.Windows.RoutedEventArgs e)
{
    OnEdit();
    var orderDate =
((orders)ViewModel.WorkAliasCollection.CurrentItem).orderdate;
    txtDelivery.Text = _fox.Do("Lieferdatum", orderDate);
}
```

There is a method called ExecuteFunction in the VFP COM server. Through this method one parameter VFP functions can be executed. The result is returned as a string. This COM server method can be used in two ways in the Silverswitch solution – directly using the service method ExecuteCommandService or using the specially developed VfxControl:VfxActionButton control.

In the following example the COM server executes the VFP proper() function and returns the result. This result is get via a callback function Callback()and is set to the variable “propername”.

To execute the COM server procedure is used the VFXDomainContext service method ExecuteCommandService. It is called via InvokeOperation. The callback function gets the result.

So the procedure name "proper" and the procedure Parameter "jack jackson" are passed and get the result "Jack Jackson" in the variable propername.

```
using VfxDataLayer.Web.Services;
using VfxApplicationBase;

namespace Test
{
    publicpartialclassTest : VfxObject
    {
        string propername = "";

        privatevoid btnProper_Click(object sender,
            System.Windows.RoutedEventArgs e)
        {
            VFXDomainContext _VFXDomainContext = newVFXDomainContext();

            IDictionary<string,object> parameters = newDictionary<string,
            object>();
            parameters["procedureName"] = "proper";
            parameters["procedureParmaters"] = "jack jackson";
            parameters["clientName"] =
            VfxAppObjectBase.CurrentConnectionInfo.ClientName;
            _VFXDomainContext.InvokeOperation("ExecuteCommandService",
            typeof(string), parameters, false, Callback, null);

        }
        publicvoid
            Callback(System.ServiceModel.DomainServices.Client.InvokeOperation op)

        {
            propername = op.Value.ToString();
        }
    }
}
```

## Using VfxControl:VfxActionButton

Using the VfxActionButton class can be done similar to the previous example. There is used bound VfxActionButton property that is used internally as function property on VfxActionButton.click.

First,VfxControl:VfxActionButton control is dropped from the Toolbox to the Silverswitch from.

```
<VfxControl:VfxActionButton
    Content="Action"
    Visibility="Visible"
    Height="23"
    HorizontalAlignment="Left"
    Margin="370,209,0,0"
    Name="btnAction"
    VerticalAlignment="Top"
```

```

        ToolTipService.ToolTip="Use VFP function PROPER() to format Customer
Name"
        Width="75"/>

```

Then the btnAction properties are set in the form constructor and the callback function is defined:

```

public partial class Customers : VfxForm.VfxDataForm
{
    public Customers()
    {
        btnAction.ProcedureName = "proper";
        btnAction.SetBinding(VfxActionButton.ProcedureParmatersProperty,
newBinding { Source = customersViewModel, Path =
newPropertyPath("Customers.CurrentItem.customername"), Mode =
BindingMode.OneWay });
        btnAction.Callback = Callback;
    }

    public void
Callback(System.ServiceModel.DomainServices.Client.InvokeOperation op)
    {
        ((customers)((CustomersViewModel)ViewModel).
Customers.CurrentItem).customername = op.Value.ToString();
        this.FormStatus = VfxApplicationBase.FormStatus.EditMode;
    }
}

```

Instead of using the function proper() there can be used any function developed in VFP.

## With VFP developed functions execution

There are two ways to use VFP code.

1. Use ExecuteFunction method of the main vfpCOMDomainService class.

This function accepts exactly one parameter and returns a string. If the function execution is not successful, the function returns an error message in the following format: "Error: " + message()

The code must be written as a method of the class cvfpdomainservice. There is an example in cvfpdomainservice class getdatetime method:

```

LPARAMETERS tcArg
RETURN TRANSFORM(DATETIME())

```

2. Create method of main COM server class vfpCOMDomainService.

In this case changes in the SilverswitchVfxDataLayer.Web project are needed. It is necessary to add new corresponding methods in VFXDomainService.cs and DataHandler.cs.

## Server Site

The COM server at the server side can handle data access to a VFP database. Actually, also other databases, especially SQL Server databases, can be supported in this way, too. Vfpcomdomainservice.dll uses generated cursoradapter classes for performing Silverswitch service operations.

Data access configuration specified in Config.vfx is used for data source specification.

Data exchange with the Silverswitch project is done in XML format.



## Client Site

COM servers are supported at the client site as well. These COM servers can be used only in Silverlight clients running out-of-browser (OOB) with elevated trust. Such COM servers are suitable for any business logic which may run at the client site and for data access.

This technique is especially useful for Silverlight desktop applications.

## Parameter Structure and Passing Parameters

If a form is going to access data through COM (VFP databases) or SQL Server databases it has to pass parameters to the service methods. The parameters are instances of the `VfxQueryParameter` class stacked together in a list. The `VfxQueryParameter` class has constructors with 3, 4, and 5 parameters which set values to the following properties:

ParameterName

Value

LogicalOperator

TypeName

Operator

Usually the last property is not used.

Parameters are wrapped in a `VfxParameter Manager` instance. `VfxParameterManager` is responsible for serialization, parameter selection, addition, modification, and other functions concerning parameters.

```
public class OrderscaViewModelBase :ViewModel<orders>
{
    private VfxPagedCollectionView _products;
    public OrderscaViewModelBase(IDataFormProperties form)
    : base(new VFPizzaSilverlightDomainContext())
    {
        var parameters = new List<VfxQueryParameter>
        {
            new VfxQueryParameter("@order", orderid, "", "orders"),
            new VfxQueryParameter("@order", "orderid", "", "orderdetails"),
            new VfxQueryParameter("@order", "productid", "", "products"),
            new VfxQueryParameter("customerid", "", "or", "orders"),
            new VfxQueryParameter("orderid", "", "or", "orderdetails")
        };
        ClientParameters = new VfxParametersManager(parameters);
        _products = new
VfxPagedCollectionView(Context.EntityContainer.GetEntitySet<products>())
;
        _form = form;

        LoadProducts();
    }

    public void LoadProducts()
    {
        Context.EntityContainer.GetEntitySet<products>().Clear();
        string pars = ClientParameters.GetSerializedParameters<products>();
        Context.Load<products>(
            ((VFPizzaSilverlightDomainContext)Context).GetProductsQuery(pars),
            loadOperation =>
            {
                }, null);
    }
}
```

Some special parameter names are @TOP, @ORDER, and @WHERE.

@TOP selects the first N entries. Here is an example:

```
new VfxQueryParameter("@TOP", recordsLimit.ToString(), "", tableName)
```

@ORDER sets criteria by chis the selected entries will be ordered. Example:

```
new VfxQueryParameter("@ORDER", "orderid", "", "orders")
```

@WHERE sets a where clause in the second (value) field:

```
new VfxQueryParameter("@WHERE", "orderid > 100", "", "orders")
```

The following operators can be used: ("=";"<">"<">"<=";">=";IN;BETWEEN;LIKE)

The mentioned special parameters can be used together to form a more descriptive query.

## Multilingual Applications

Silverswitch is well prepared for creation of multilingual applications. Furthermore, the VFX - Silverlight Wizard inherits the localization settings of the migrated VFX for VFP application.

### Runtime Localization

With Silverswitch it is possible not only to develop localized applications, but also to allow end-users to change the application language at runtime.



Runtime localization is implemented dynamically and in a memory efficient way in the class LocalizationManager.

The localization settings are realized using properties of the application object in the class VfxApplication/Controls/VfxAppObject.cs.

**RunTimeLocalization** – This property determines whether the application will be localized at runtime. If the value of this property is set to true, the language of the application can be chosen in the login dialog. Additionally, at runtime the language can be changed using the language selection combobox in the main toolbar. The comboboxes will display only languages which are set as active. To mark a language as active it is necessary to set the *IsActive* field in the *VfxLanguage* table to true. If RuntimeLocalization is set to false the language selection comboboxes are not visible.

**LangID** – This property determines the initial language with which the application starts. Localizable controls are all Silverswitch controls that have values for *MessageID* or *TooltipMessageID* properties in the XAML of their form. These properties store string values corresponding with the field *message\_id* in the table *VfxMsg*.

During the localization process:

- These values are added to an array <string> object Messages used as QueryParameter of a Domain Context.
- A dictionary object is created and filled. It stores the *message\_id* and the current language text for each value of *MessageID* or *TooltipMessageID* of the XAML controls. This object uses caption properties or tooltip properties of localized controls.

The last used language is kept on a per user basis in the table VfxUsr. Next time when the same user logs in, the last chosen language is automatically selected.

Language	Abbreviation
English	ENG
French	FRE
German	GER
Italian	ITA
Spanish	ESP
Bulgarian	BUL
Greek	GRE
Dutch	NL
Portuguese	POR
Russian	RU
Czech	CZE
Finnish	FIN
Polish	PL
Turkish	TR
Albanian	ALB
Swiss	CHD
Romanian	RO
Slovak	SVK
Estonian	EST
ChineseSimp	CHS
Japanese	JPN
ChineseTrad	CHT

**Table 1 – List of available languages**

For each available language there is a column in the table VfxMsg. Some texts are not available in every language.

When changing the language, the regional settings are determined. This includes decimal point, date format and others.

## Localization of forms

All forms, which support runtime localization, implement the *IVfxLocalization* interface. In the first occurrence of the *Loaded* event the method *InitializeLocalizationManager(string language)* is called. In this method an instance of *VfxLocalizationManager* object is created. *VfxLocalizationManager* finds all controls which support runtime localization (these are controls that have properties *ToolTipMessageID* or *MessageID*). The *MessageID* property of the control corresponds to the *message\_id* field of the table VfxMsg. For each available language there is a column in the table VfxMsg. Once the message is found it is applied to the control's Caption, Text, Header, or Content properties depending on the type of control. Likewise the value from *ToolTipMessageID* property is used to set the tooltip of the control. Consider the following example of a simplified form containing three controls.

```
<VfxObject:VfxForm x:Class="VfxClass"
  x:Name="frmClass"
  MessageID="CAP_FRMUSERLIST"
  Caption="User List" >
  <VfxObject:VfxGrid x:Name="LayoutRoot">
    <VfxObject:VfxTabControl
      x:Name="xpgfPageFrame">
      <VfxObject:VfxTabPage
        x:Name="xpgfPageFramePage1"
        Header="Edit"
        MessageID="CAP_EDIT"
        ToolTipMessageID="CAP_EDIT">
      <VfxObject:VfxTextBlock
        x:Name="xpgfPageFramePage1lblUserAccess"
        Text="User Access"
```

```

        MessageID="CAP_LBLUSERACCESS"/>
    <VfxObject:VfxButton
        Content="OK"
        Name="button1"
        Click="button1_Click"
        MessageID="CAP_CMDOK"/>
    </VfxObject:VfxTabItem>
</VfxObject:VfxTabControl>
</VfxObject:VfxGrid>
</VfxForm:VfxDataForm>

```

## Localization of MessageBoxes

To show a localized MessageBox the developer has to call the `VfxStartUpFormBase.ShowVfxMessageBox` method. It implements the `VfxLocalizationManager.GetMessageValue` method as well as `VfxMessageBox.Show` method in order to show a localized MessageBox. Here the declaration of `ShowVfxMessageBox` is shown and each of its parameters shortly explained.

```

public void ShowVfxMessageBox(string message, string messageID, string
caption,
string captionID, VfxMessageBoxButtons buttonType,
VfxMessageBoxTypes messageType, int timeout,
Action<VfxMessageBoxResults> callback)

```

### Parameters

**Message** – The text to display in the messagebox if value for MessageID parameter is not found in the table VfxMsg.

**MessageID** – Search constant with this ID in table VfxMsg and if found use it as message text in the messagebox.

**Caption** – The text to display in the title bar of the messagebox if the value for CaptionID parameter is not found in the table VfxMsg.

**CaptionID** – Search constant with this ID in table VfxMsg and if found use it as caption for VfxMessageBox.

**ButtonType** – One of the VfxMessageBoxButtons values that specifies which buttons to display in the messagebox.

**MessageBoxType** – One of the VfxMessageBoxTypes values that specify which icon to display in the messagebox.

**Timeout** – Specifies the number of milliseconds after which the messagebox closes automatically.

**Callback** – Occurs when the messagebox is closed. You can receive the result from the messagebox with this action.

Example:

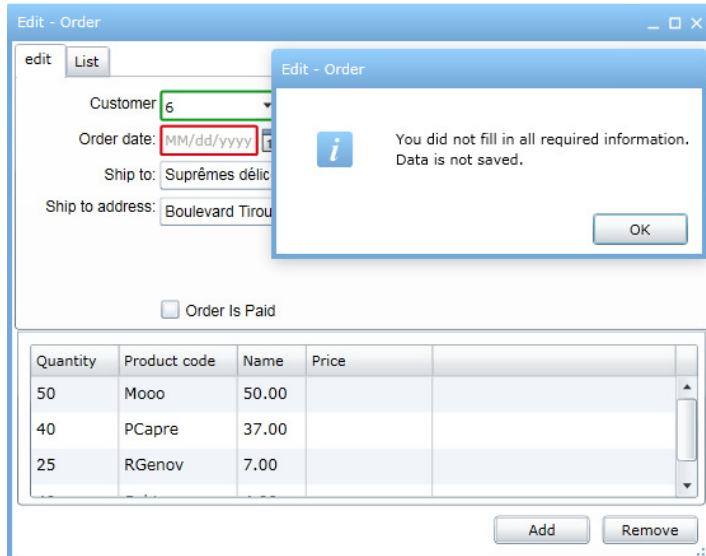
```

ShowVfxMessageBox("Are you sure you want to delete this file?",
"MSG_DELETEFILE", "Delete", "CAP_CMDDELETE",
VfxApplicationBase.VfxMessageBoxButtons.YesNo,
VfxApplicationBase.VfxMessageBoxTypes.ExclamationPoint, 0,
DeleteMessageResult);

```

## Required Fields

Required fields enable the developer to build forms with controls which the user is obligated to fill. The user's request to save an entry with empty required fields will be denied. In this case the controls which have to be filled by the user will change border color from green to red and a messagebox will appear.



To implement this behavior the developer has to add the following row to the InitializeValues method of the cs file of the form.

```
private void InitializeDefaultValues()  
{  
    ...  
    RequiredFields="";Orders.CurrentItem.orderdate;Orders.CurrentItem.custome  
    rid;Orders.CurrentItem.shiptoaddress;"  
    ...  
    WorkAlias = "Orders";  
}
```

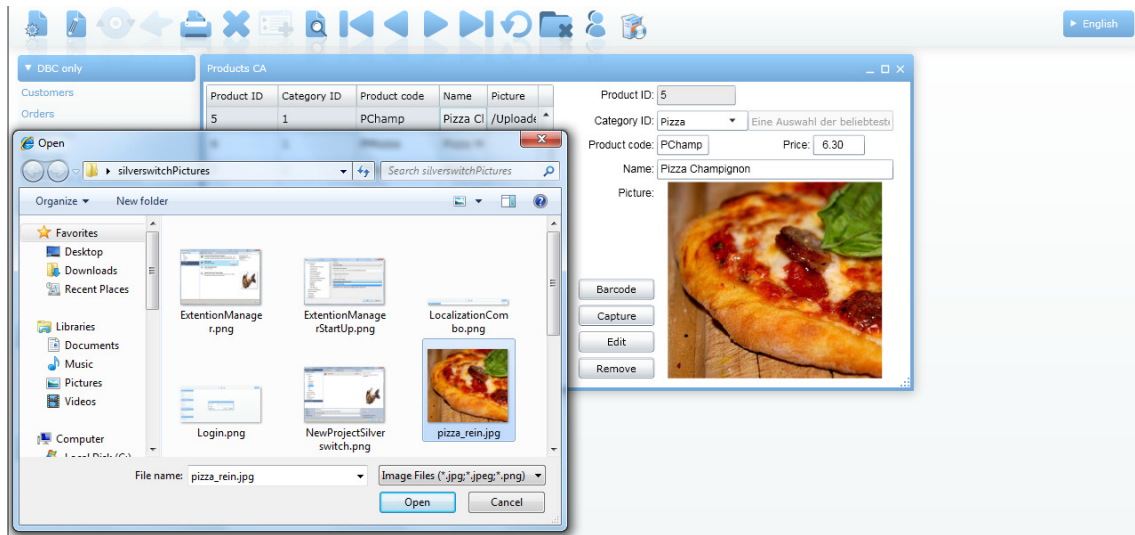
RequiredFields property can contain arbitrary number of field names separated by semicolon. The controls which are bound to these fields will automatically acquire the required fields behavior.

Here is how the controls from the discussed form are implemented in the xaml of the form:

```
<VfxControl:VfxPickField  
    x:Name="xpgfPageFramePage1cntCustomerid"  
    TableName="{Binding Path=Customers, Mode=TwoWay}"  
    FieldList="customerid;customername"  
    FieldTitle="Kundennummer;Kundenname"  
    ReturnExpr="customerid"  
    ReturnExprDesc="customername"  
    SelectedItem = "{Binding Path=Orders.CurrentItem.customers,  
    Mode=TwoWay}">  
  
<VfxObject:VfxDatePicker  
    x:Name="xpgfPageFramePage1cntOrderdate"  
    SelectedDate="{Binding Path=Orders.CurrentItem.orderdate,  
    Mode=TwoWay}">  
  
<VfxObject:VfxTextBox  
    x:Name="xpgfPageFramePage1edtShiptoaddress"  
    Text="{Binding Path=Orders.CurrentItem.shiptoaddress, Mode=TwoWay}"/>
```

## Upload and Download files

There are classes supporting the upload and download of files, and displaying pictures in Silverswitch. They can be used in easy way in Silverswitch forms.



## Display Pictures

The class `VfxImage` can be used to display pictures in forms.

### Properties

**VfxSourcePath** – Source path of the picture at the server side. The path does not include file name. The path must be specified relative from the ClientBin folder where the XAP files are stored.

**VfxSourceFileName** – File name of the picture file.

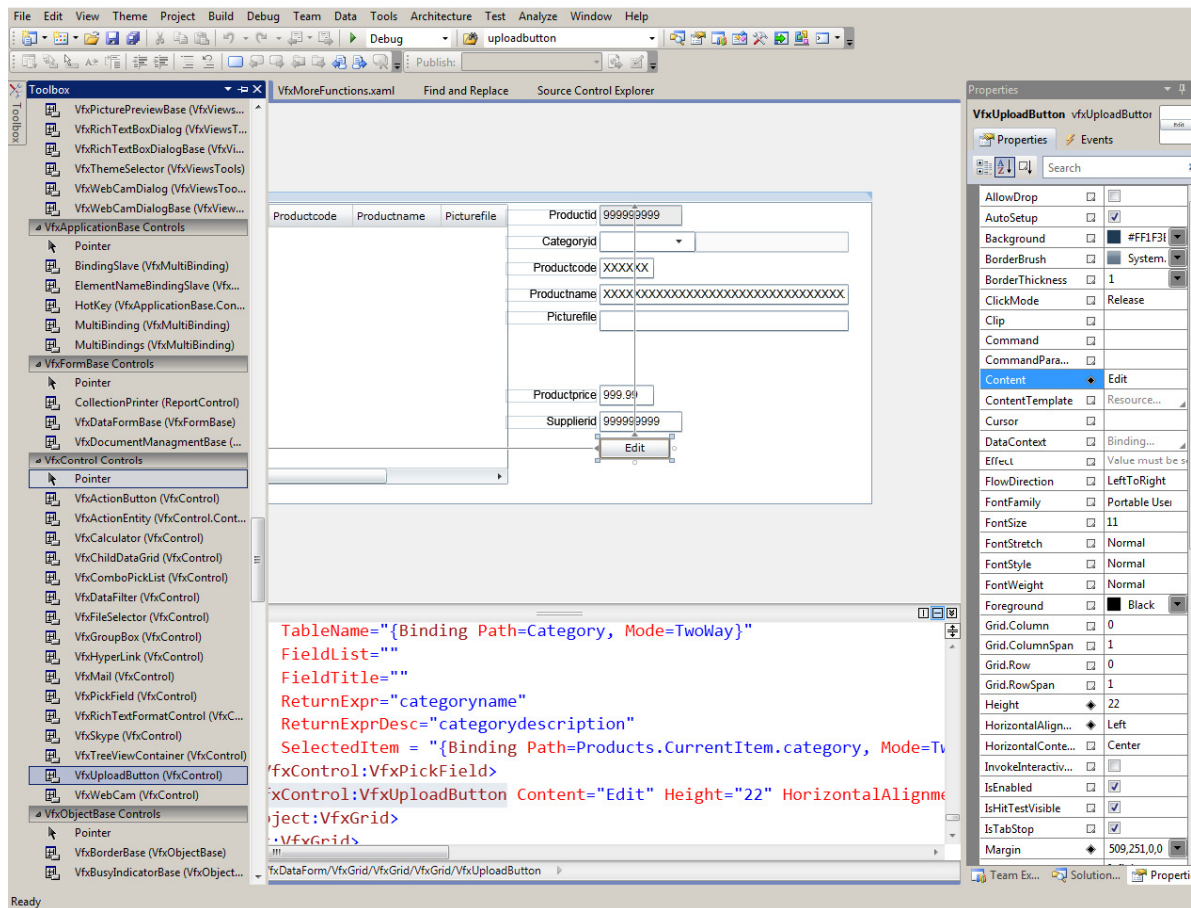
Example declaration of a `VfxImage` control in XAML:

```
<VfxObject:VfxImage Name="vfxImage1"
  VfxSourcePath=" ../UploadedFiles/"
  VfxSourceFileName="margarita.jpg"
  MessageID="MSG_PLEASEWAIT" />
```

Note that the `VfxImage` control also has a `MessageID` property. A localized message along with busy indicator can be displayed during the loading of the image from the server to the client.

## Upload Files

Open XAML file in design mode. Find control `VfxControl:VfxUploadButton` in toolbox and drop in design window.



The control declaration is generated in the XAML window:

```
<VfxControl:VfxUploadButton
    Content="Button"
    Height="23"
    HorizontalAlignment="Left"
    Margin="292,298,0,0"
    Name="vfxUploadButton1"
    VerticalAlignment="Top"
    Width="75" />
```

VfxUploadButton have functionality to upload selected at client site picture, upload it in server folder. If the control is bind after form saving file path and file name will be saved in the database.

## Properties.

**UploadFileType** – This property defines the file type filter for open file dialog. It accepts two values “All” and “Image”. Default UploadFileType value is “All”. If Images will be uploaded it is recommended to set this property to “Image”.

```
UploadFileType="Image"
```

**OpenFileDialogFilter** – Another way to filter displayed files in open file dialog is using the property OpenFileDialogFilter. Example:

```
OpenFileDialogFilter="Documents | *.doc; *.txt | Images | *.jpeg; *.jpg; *.png"
```

**VfxSourcePath** – This property defines the source path. In the example below it is “picturepath” property of “Products” entity. It corresponds with field picturepath of products table in the database.



**VfxSourceFileName** – The property defines file name. In the example below it is “picturefile” property of the Products table.

```
<VfxControl:VfxUploadButton
  Content="Upload"
  Name="vfxUploadButton1"
  UploadFileType="Image"
  VfxSourcePath="{Binding Path=Products.CurrentItem.picturepath,
Mode=TwoWay}"
  VfxSourceFileName="{Binding Path=Products.CurrentItem.picturefile,
Mode=TwoWay}" />
```

Additionally, the property MessageID can be set to localize the button content. For more details about localization see the chapter “Run-time Localization”.

## **Data Handling**

All base functionality needed for data handling is included in the Silverswitch template project. All application specific functionality is generated by the VFX – Silverlight Wizard.

## Error tracking

When an error is raised, there is an exception thrown in Silverlight. It will be handled and will be tracked automatically in a table. The name of the current user, date, time, error message, as well as the stack trace, is saved. Further application behavior after an exception was thrown can be set through properties of the application object. The exception information is logged in the table VfxLog.

The content of the error information depends on the value of the application object property ErrorDetailLevel. It accepts the following values:

**MessageOnly** – Only message property of exception object is logged.

**NoCallStackInformation** – All information except call stack information will be logged.

**FullDetailedInformation** – All information will be logged:

The name of the current user, date and time will be logged in all cases.

If the application is running in release mode the exception is not shown to the end-user. First, the exception type is checked. If it is one of types defined in the application property CriticalErrorTypes, a user friendly dialog is shown. If the error type is not defined in the application object property CriticalErrorTypes, the error will be ignored and execution will continue.

A user friendly dialog is shown if the exception cannot be stored in the error log table.

## **Resource Table**

Silverswitch applications use a resource table to store information about forms on a per user base. The information is used to set the size and position of the form, DataGrid layouts, as well as the current sort order.

Position and size of the form are saved on form close. The user sees the forms always appearing exactly in the same way he closed it.

You can reset the resource file by clicking the Clear Resource button in the user list form. This will delete all entries from in the resource file for the current user.

## **Further Information**

Most recent information about Silverswitch can be found at the website <http://www.Silverswitch.de>.

## **Forum, Newsgroup**

Support about Silverswitch is available in the newsgroup <news://news.dfpug.net>. Access to the newsgroup is also available on the Silverswitch start page.

## **Technical Documentation**

Silverswitch contains a technical reference. The technical reference shows the inheritance between classes. All classes, methods, and properties are documented.

## Tips

1. If an application does not start in the browser, check StartUp project and start page. <application name>.Web project must be set as StartUp project. default.htm must be set as the start page.
2. To debug in Silverlight .cs files, the debugger has to be enabled. If it is not, find main project <application name>.web, open context menu and choose "Properties". At "Properties" window choose "Web" tab. At the bottom of the tab check "Silverlight".