HOW TO DEVELOP ACTIVEX INVISIBLE COMPONENT LIBRARY IN DELPHI

Editor:
Tomáš Mandys, tomas.mandys@2p.cz (2p plus)

Home site:
http://www.2p.cz

Document status:
Version 1.0 First release

Table of Contents

Disclaimer

1. Introduction

Delphi support developing ActiveX libraries using *TActiveXControl* class that defines the core behavior and interfaces required of an *ActiveX* control and connects that behavior to any VCL control derived from *TWinControl*. This allows the control to be embedded in any ActiveX container, including Internet Explorer, Visual Basic, PowerBuilder, Paradox, Borland C++, IntraBuilder and, of course, Delphi. ActiveX has typical OCX extension and must be registered using *regsvr32*.

But custom class must be descendant of a *TWinControl* and it means that implemented component is a visible control in both design and run-time. But not all components should be visible – in Delphi terminology controls - all the time. Do many examples of invisible components exist – timers, communication components, database connectors, loggers, protocol implementation, etc.

Let's demonstrate how to develop invisible ActiveX components.

2. Invisible component

The typical invisible component should be indicated on a design form as an icon or bitmap and totally hidden in run-time.

We have developed *AxCmps.pas* unit that is equivalent of common *AxCtrls.pas* unit.

2.1. AxCmps.pas

The main class *TActiveXComponent* is descendant of *TActiveXControl* and overrides some behavior. First is defined special *TActiveXComponentControl* that server visual indication of a component in design time. In run-time is sleeping behind the scene. The key property of the ActiveXComponentControl is BitmapId that references a component icon that appears on the design form.

TActiveXComponent has assigned one *TActiveXComponentControl* instance and controls it – hides it if run-time mode is recognized and shows it if design time is recognized. Such behavior was not easy to implement since Microsoft IDE (Visual Basic, Visual Basic for Application, etc.) and Delphi IDE behaves partially differently when manipulation with ActiveX library and a thing that works perfectly in Delphi does not work in VB.

2.2. Custom component

Custom component will be descendant of *TActiveXComponent* and will implement a custom interface. Interface is defined in Delphi using Type Library editor TLB.

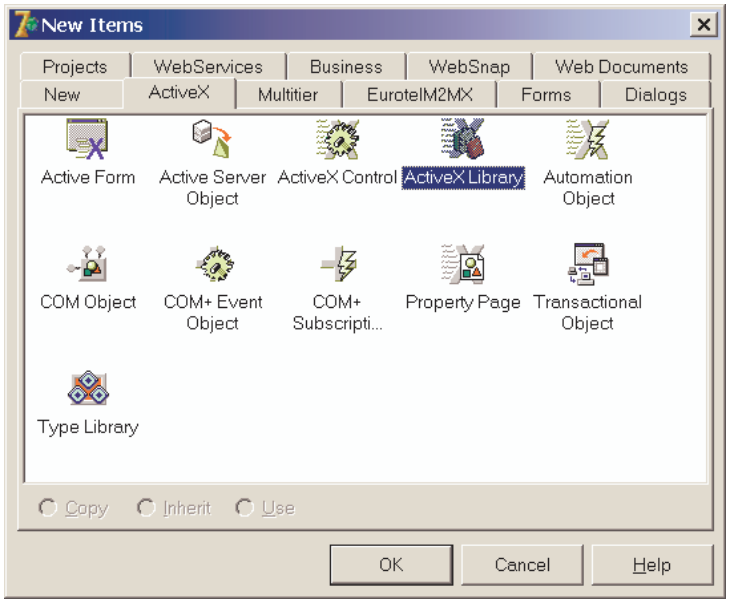
3. Example

3.1. Developing component

Let's say that you want create *TStringList* reusable as *ActiveX* component in *OCX* library.

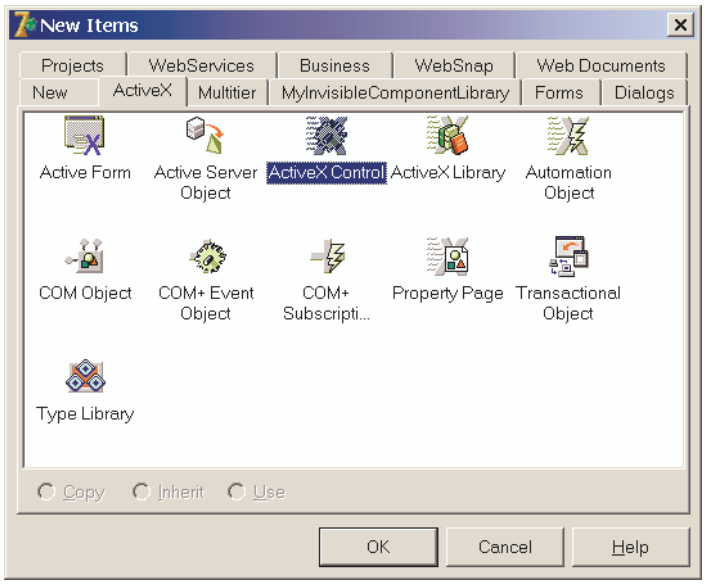
1. run Delphi IDE

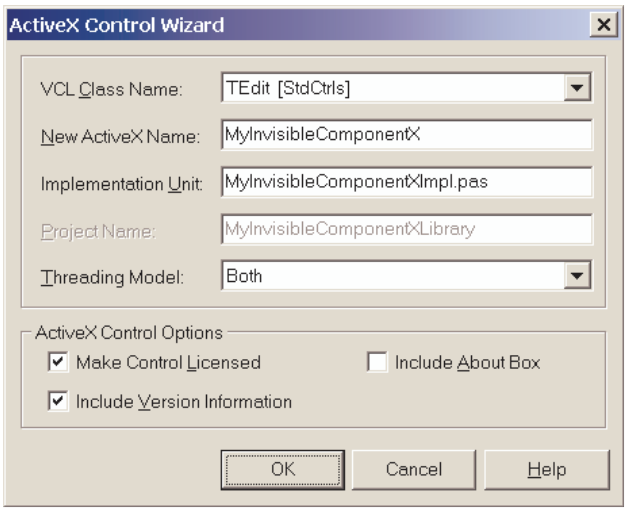
2. change language of type library to *"Pascal"* if you prefer it to *"IDL"* (*Tools/Environment options/Type library/Language*)
3. create new project – ActiveX library (*File/New/ActiveX/ActiveX library*)



Save project for example as *"MyInvisibleComponentXLibrary"*

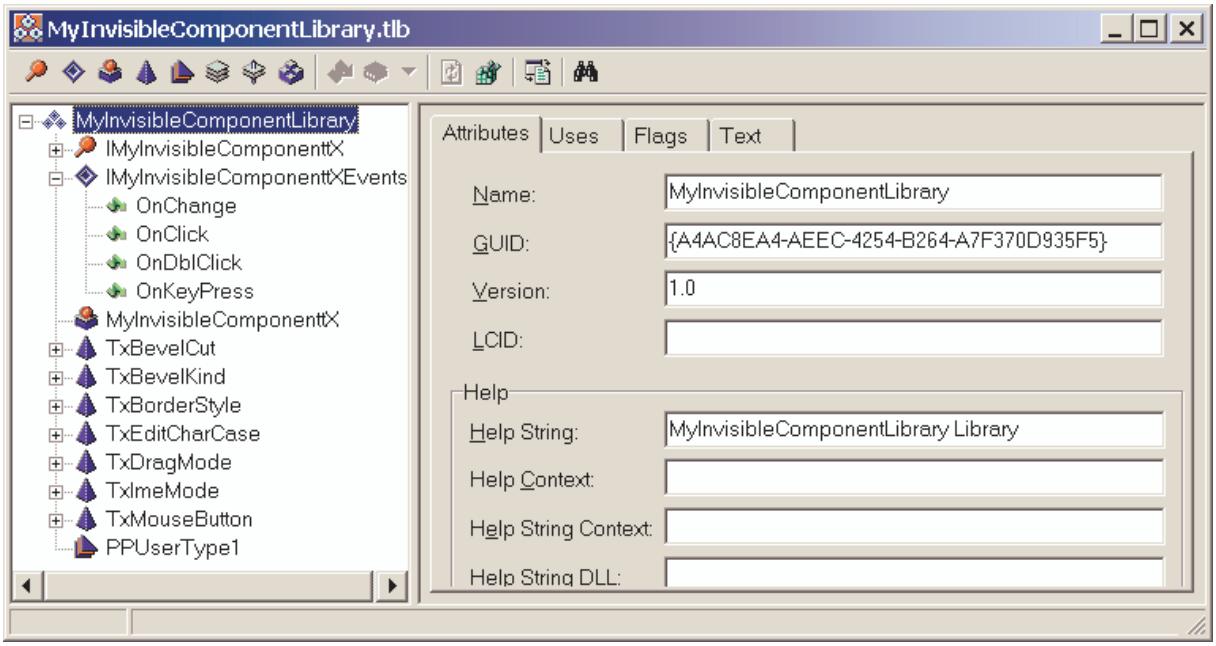
4. add new ActiveX control (*File/New/ActiveX/ActiveX control*)



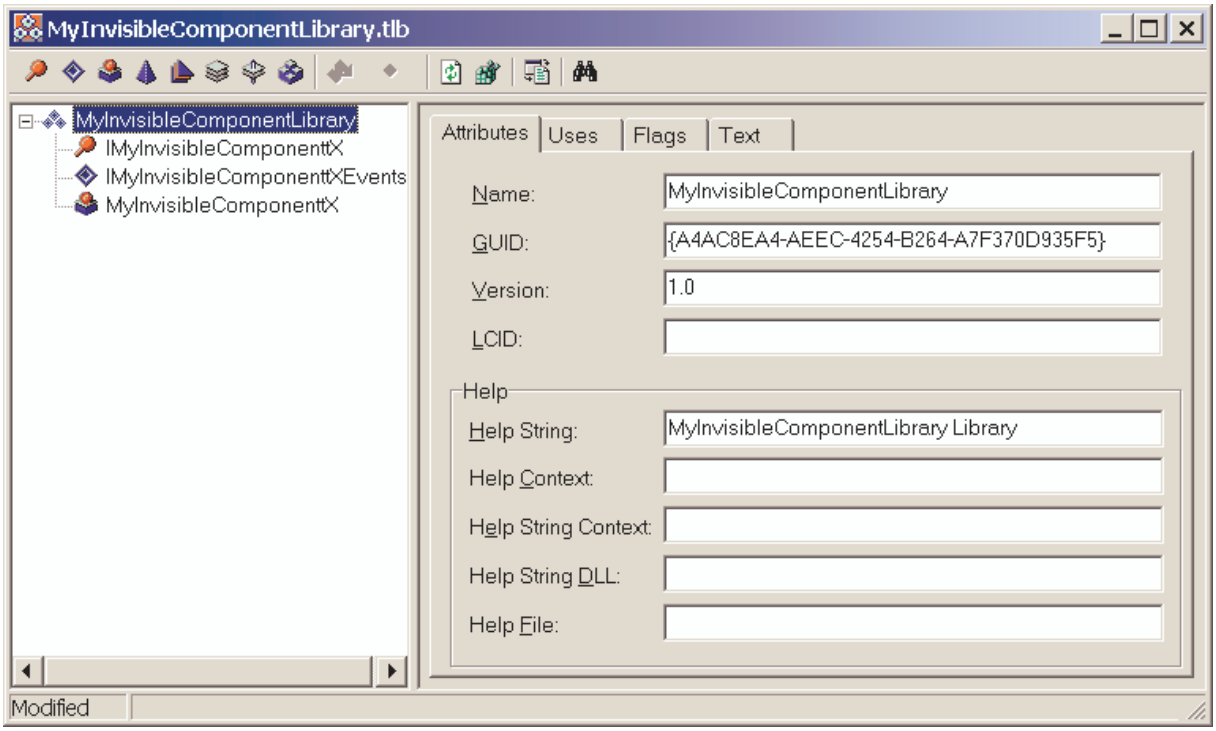


- select any *"VCL Class Name"*, you will change it directly in IDE later
- enter your new component name in *"New ActiveX Name"*
- enter your *"Implementation unit"* name
- if you want license your library via *.LIC* file check *"Make control licensed"*
- *"Include version information"* enable to insert *"Project/Options/Version info"* to library

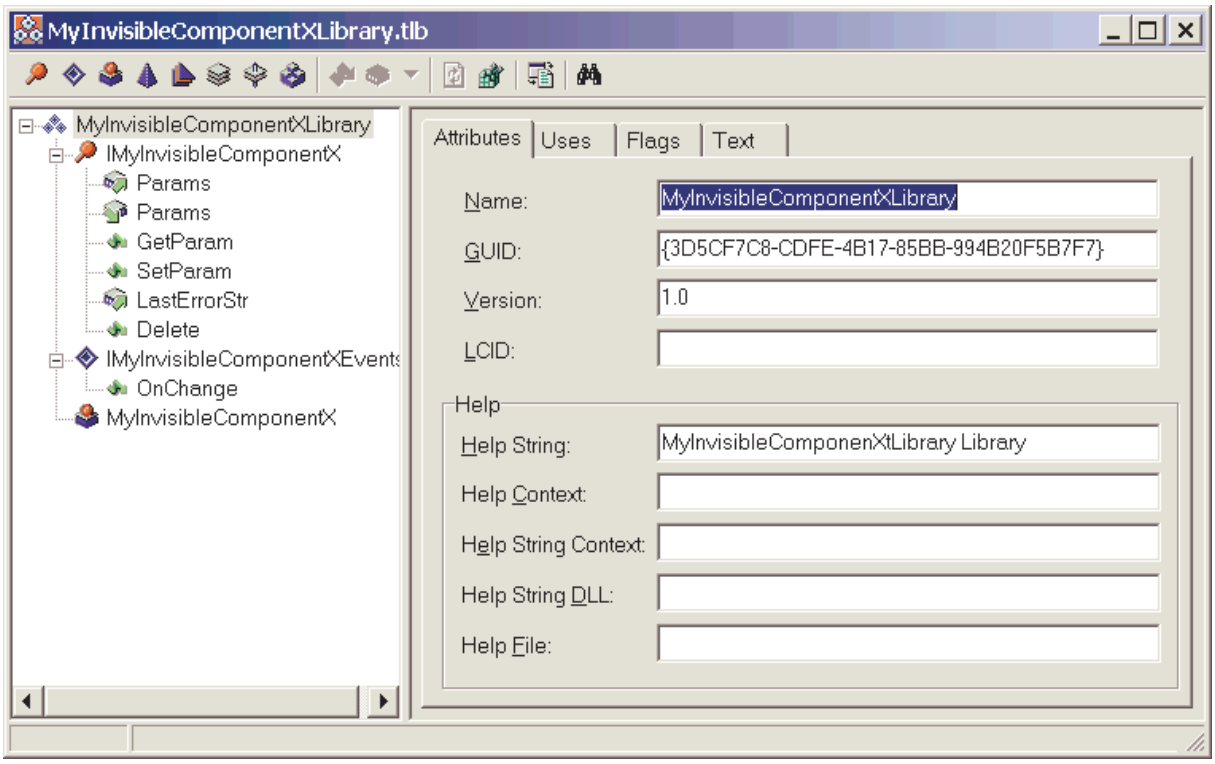
Delphi wizard prepares type library and skeleton of new class implementation.



5. Open type library visual editor and remove all obsolete methods, properties, enumerations introduces for *TEdit* class.



Add properties, methods and events that you need publish in *ActiveX* component. Note that for *TString* use *IStrings* and for string use *WideString*.



6. Delete obsolete methods from skeleton *"MyInvisibleComponentXImpl"*, probably all methods except *FEvents*, *DefinePropertyPages*, *EventSinkChanged*, *InitializeControl*.

7. Add *AxCmps* and *MyInvisibleComponent* to interface clause.

8. Change *TMyInvisibleComponentX* ancestor from *TActiveXControl* to *TActiveXComponent* that implements all visible features, i.e. in design time appears as icon, in run-time is hidden.

```
TMyInvisibleComponentX = class(TActiveXComponent, IMyInvisibleComponentX)
```

9. Add *FMyInvisibleComponent* of *TMyInvisibleComponent* to private section of *TMyInvisibleComponentX* declaration. Declare all methods to implement *IMyInvisibleComponentX* interface and events for *IMyInvisibleComponentXEvents*.

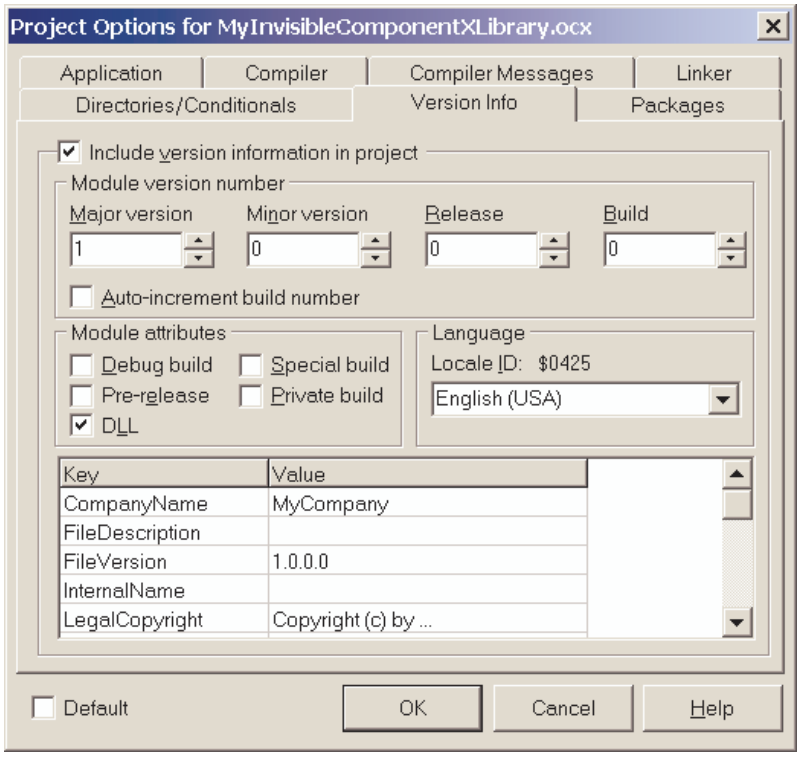
   Note that for *IStrings* use *GetOleStrings* and *SetOleStrings* procedures. It seems that *ActiveX* library compiled using Delphi have problem if an exception is raised in library. I observed that it is better safe all code to try expect statement, return result code and make *LastErrorStr* property to publish reason of exception (see *Delete* function implementation).

10. Override *TMyInvisibleComponentX.InitializeControl* to initialize properties.

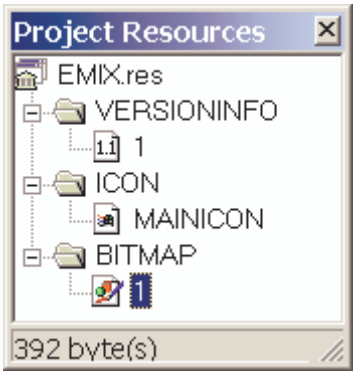
11. Override *TMyInvisibleComponentX.DefinePropertyPages* to assign special purpose property editor (for *IStrings* type).

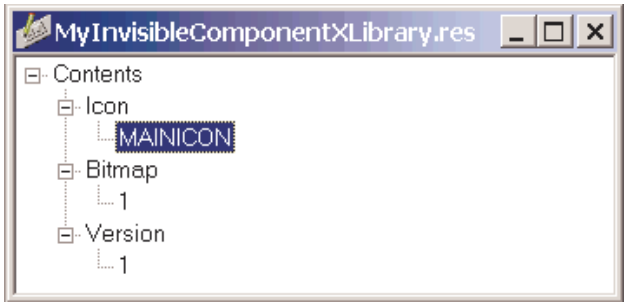
12. Change initialization section *TActiveXControlFactory* to *TActiveXComponentFactory* and remove *"TEdit"* parameter.

13. Edit project properties, advertise version, developer, copyright etc.



14. Edit *"MyInvisibleComponentXLibrary.res"* resource (using *ImageEditor* for example) containing bitmap identified by ID number or modify existing one. Add line `Control.BitmapId:=ID` to *InitializeControl* method where *ID* is bitmap identifier in resource, default value is *"1"*. Size of bitmap 26x26 pixels is expected. Note that I have in my Delphi 5 *Project/Resources* menu item but similar item is missing in Delphi7 – maybe it's a hack. Using this command you can add bitmap directly in IDE. External resource file must linked to library using `{$R file}` directive.





15. If you need use *ActiveX* library also in Delphi IDE you should change also **_TLB.PAS*.

Add *OleCmps* to interface uses section and change ancestor of *TMyInvisibleComponentX* from *TOleControl* to *TOleComponent* – it's hack that hides obsolete methods of *TControl* class in property editor.

Warning: *_*TLB.PAS* is generated by IDE when saving type library so changes made manually are overwritten.

```
TMyInvisibleComponentX = class(TOleComponent)
```

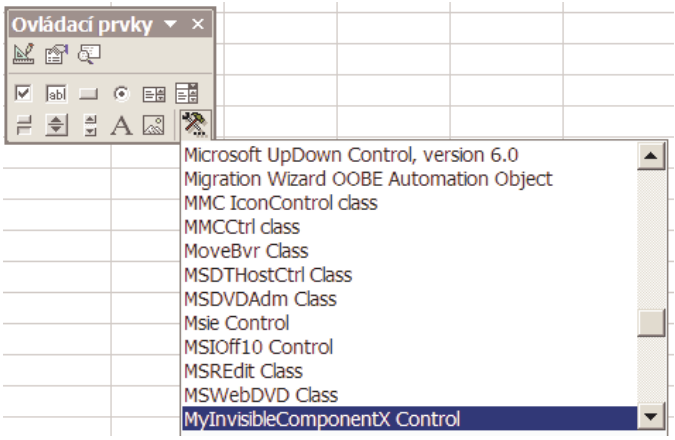
16. Ensure that license key in *.LIC* file is the same as license string passed in initialization section of *TMyInvisibleComponentXImpl* to *TActiveXComponentFactory.Create*. If not the same then the component is unable successfully place on a form.

17. Register library using *regsvr32* utility

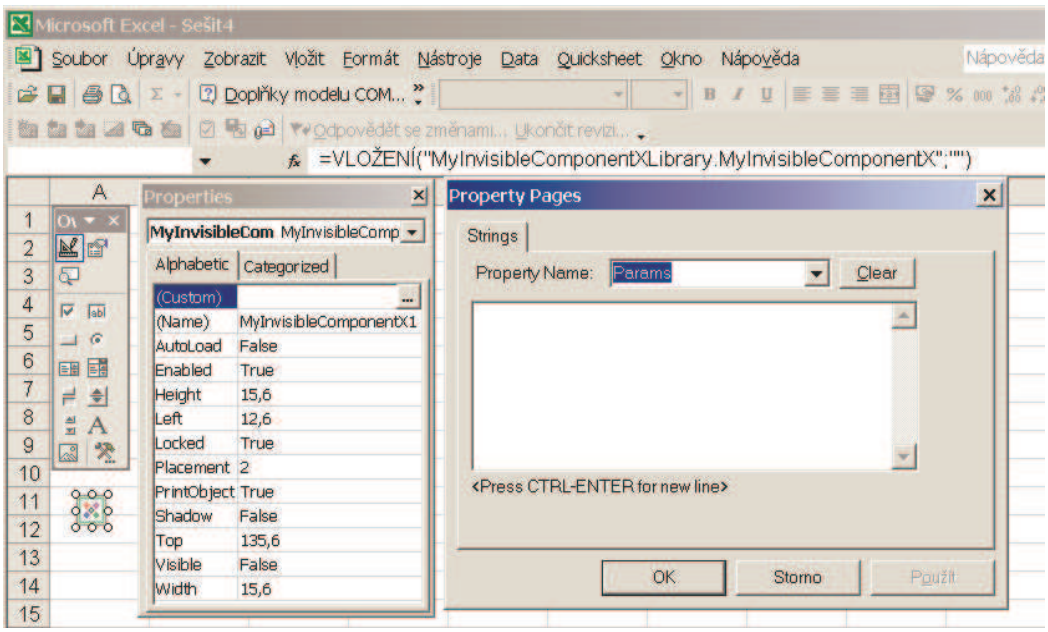
```
Regsvr32 MyInvisibleComponentXLibrary.ocx
```

3.2. Using new component in Excel

1. Open Microsoft Excel
2. Show *Control tools* toolbar (right click at any toolbar and check "*Control tools*")
3. Click *"Add control"*
4. Select *"MyInvisibleComponentXLibrary.ocx"*



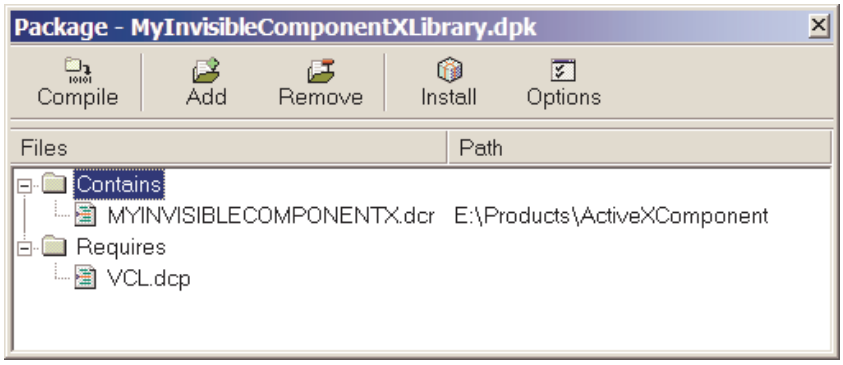
5. Place *MyInvisibleComponentX* to sheet



3.3. *Using new component in Delphi*

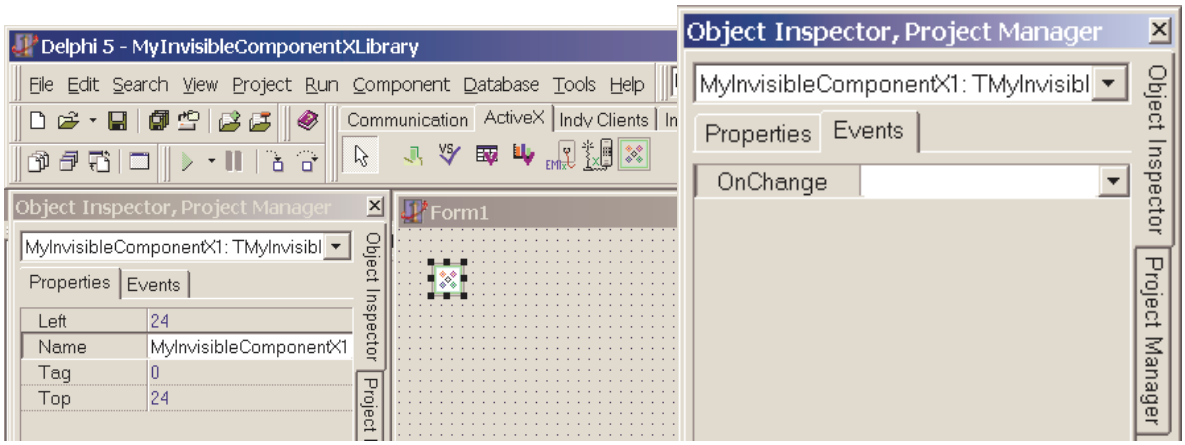
Create new package *MyInvisibleComponentXLibrary.dpk* containing *MyInvisibleComponentXLibrary_TLB* and component resource *MyInvisibleComponentXLibrary.dcr* that contains 26x26 pixels bitmap named *"MYINVISIBLECOMPONENTX".*

Link resource using `{$R 'MYINVISIBLECOMPONENTX.dcr'}` directive, compile and install.



Now should appear your component at ActiveX palette tab. Assure that *OleCmps.TOleComponent* is ancestor of class in *TLB*. If is not then change it manually and recompile package. If you plan use *OleCmps* in more packages create new package containing *OleCmps* and introduce name of this package to *"requires"* section.

```
requires
  OleCmps;
```



4. Links

http://support.microsoft.com/?kbid=238228

HOWTO: Build an Office 2000 COM Add-In in Visual Basic

http://www.geocities.com/nafmis/FAQ/COMFAQ199907.html

The most valuable info concerning invisible ActiveX library written in Delphi

http://www.2p.cz/download/

Complete source code of demo application

5. Appendix

5.1. AxCmps.pas

```
unit AxCmps;

interface
uses
  SysUtils, Classes, Controls, ActiveX, AxCtrls, {$IFDEF REGISTRATION}AxCtrlsReg,
{$ENDIF}Windows, ComServ, ComObj, Graphics;

type
  TActiveXComponentControl = class;

  TActiveXComponent = class(TActiveXControl, IOleControl)
  private
    fUserMode: WordBool;
  protected
    procedure ModeChanged; virtual;
    procedure InitializeControl; override;
    procedure ActiveXLoaded; virtual;
    { IOleControl }
    function GetControlInfo(var ci: TControlInfo): HResult; stdcall;
    function OnMnemonic(msg: PMsg): HResult; stdcall;
    function OnAmbientPropertyChange(dispid: TDispID): HResult; stdcall;
    function FreezeEvents(bFreeze: BOOL): HResult; stdcall;
  public
    property UserMode: WordBool read fUserMode;
    procedure Initialize; override;
  end;

  TActiveXComponentControl = class(TCustomControl)
  private
    fBitmapId: Integer;
    procedure SetRunTime(const Value: Boolean);
    function GetRunTime: Boolean;
  protected
    procedure Paint; override;
    procedure DesignPaint; virtual;
  public
    property BitmapId: Integer read fBitmapId write fBitmapId;
    constructor Create(AOwner: TComponent); override;
    property RunTime: Boolean read GetRunTime write SetRunTime;
  end;

  TActiveXComponentFactory = class({$IFDEF
REGISTRATION}TRegActiveXControlFactory{$ELSE}TActiveXControlFactory{$ENDIF})
  public
    constructor Create(ComServer: TComServerObject;
      ActiveXControlClass: TActiveXControlClass;
      const ClassID: TGUID;
      ToolboxBitmapID: Integer; const LicStr: string; MiscStatus: Integer;
      ThreadingModel: TThreadingModel = tmSingle);
  end;

implementation
```

```
{ TActiveXComponent }

procedure TActiveXComponent.Initialize;
begin
  fUserMode:= True;  // VB does not call OnAmbientPropertyChange, so it must be
runtime default option
  inherited;
  ModeChanged;
end;

procedure TActiveXComponent.InitializeControl;
begin
  inherited;
{  if (Control <> nil) and (Control is TActiveXComponentControl) then
    TActiveXComponentControl(Control).BitmapId:= ; // default is 1
//UpperCase(Self.ClassName); }
end;

procedure TActiveXComponent.ModeChanged;
begin
  if (Control <> nil) and (Control is TActiveXComponentControl) then
    begin
      TActiveXComponentControl(Control).RunTime := fUserMode;
    end;
end;

function TActiveXComponent.GetControlInfo(var ci: TControlInfo): HResult;
begin
  Result := inherited GetControlInfo(ci);
end;

function TActiveXComponent.OnMnemonic(msg: PMsg): HResult;
begin
  Result := inherited OnMnemonic(msg);
end;

function TActiveXComponent.OnAmbientPropertyChange(
  dispid: TDispID): HResult;
var
  AD: IAmbientDispatch;
begin
  Result := inherited OnAmbientPropertyChange(dispid);
  if (dispid = DISPID_UNKNOWN) or (dispid = DISPID_AMBIENT_USERMODE) then  // called
only by Delphi
    if (ClientSite <> nil) and (ClientSite.QueryInterface(IAmbientDispatch, AD) =
S_OK) then
    begin
      fUserMode:= AD.UserMode;
      ModeChanged;
    end;
end;

function TActiveXComponent.FreezeEvents(bFreeze: BOOL): HResult;
begin
  Result := inherited FreezeEvents(bFreeze);
  if not bFreeze then
    ActiveXLoaded;   // for VB - according
http://www.geocities.com/nafmis/FAQ/COMFAQ199907.html
end;
```

```
procedure TActiveXComponent.ActiveXLoaded;
begin
end;

{ TActiveXComponentControl }

constructor TActiveXComponentControl.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  if AOwner is TWinControl then
    Parent := (AOwner as TWinControl);
  Width := 26;
  Height := 26;
  ControlStyle:= [csOpaque, csFixedWidth, csFixedHeight, csNoStdEvents {,
csReplicatable, {csNoDesignVisible}];
  Constraints.MaxHeight:= Width;
  Constraints.MinHeight:= Width;
  Constraints.MaxHeight:= Height;
  Constraints.MinHeight:= Height;
  Visible:= False;
  fBitmapId:= 1;
end;

procedure TActiveXComponentControl.Paint;
begin
  inherited Paint;
//  if not RunTime then   // visibility is controled by ActiveX,
//  if tested here - MS Visual *, Excel does not show icon in designed if project
being reloaded (even placing works OK)
    DesignPaint;
end;

procedure TActiveXComponentControl.DesignPaint;
var
  PaintRect: TRect;
  Bmp: TBitmap;
begin
  PaintRect := Rect(1, 1, Width, Height);
  Bmp:= TBitmap.Create;
  try
    Bmp.LoadFromResourceId(hInstance, fBitmapId);
    Canvas.StretchDraw(PaintRect, Bmp);
  finally
    Bmp.Free;
  end;
{  Canvas.Pen.Style := psDot;
  Canvas.Brush.Style := bsClear;
  Canvas.Rectangle( 0, 0, Width, Height); }
end;

procedure TActiveXComponentControl.SetRunTime(const Value: Boolean);
begin
  Visible:= not Value;
  SetDesigning(not Value, True);
  Repaint;
end;

function TActiveXComponentControl.GetRunTime: Boolean;
begin
  Result:= not (csDesigning in ComponentState);
end;
```

```
{ TActiveXComponentFactory }

constructor TActiveXComponentFactory.Create(ComServer: TComServerObject;
  ActiveXControlClass: TActiveXControlClass; const ClassID: TGUID;
  ToolboxBitmapID: Integer; const LicStr: string; MiscStatus: Integer;
  ThreadingModel: TThreadingModel);
begin
  inherited Create(ComServer, ActiveXControlClass, TActiveXComponentControl, ClassID,
ToolboxBitmapID, LicStr,
  MiscStatus or OLEMISC_SIMPLEFRAME or OLEMISC_ACTSLIKELABEL or
OLEMISC_INVISIBLEATRUNTIME,
  ThreadingModel);
end;

end.
```

5.2. MyInvisibleComponentXImpl.pas

```
unit MyInvisibleComponentXImpl;

{.$WARN SYMBOL_PLATFORM OFF}

interface

uses
  Windows, ActiveX, Classes, StdCtrls, SysUtils,
  ComServ, StdVCL, AXCtrls, AxCmps, MyInvisibleComponentXLibrary_TLB;

type
  TMyInvisibleComponentX = class(TActiveXComponent, IMyInvisibleComponentX)
  private
    { Private declarations }
    FMyInvisibleComponent: TStringList;
    FEvents: IMyInvisibleComponentXEvents;
    fLastErrorStr: string;
    procedure ChangeEvent(Sender: TObject);
  protected
    { Protected declarations }
    procedure DefinePropertyPages(DefinePropertyPage: TDefinePropertyPage); override;
    procedure EventSinkChanged(const EventSink: IUnknown); override;
    procedure InitializeControl; override;

    function Get_Params: IStrings; safecall;
    procedure Set_Params(const Value: IStrings); safecall;
    function GetParam(const Name: WideString): WideString; safecall;
    procedure SetParam(const Name, Value: WideString); safecall;
    function Get_LastErrorStr: WideString; safecall;
    function Delete(aIndex: Integer): WordBool; safecall;
  end;

implementation

uses ComObj;

{ TMyInvisibleComponentX }

procedure TMyInvisibleComponentX.ChangeEvent(Sender: TObject);
begin
  if FEvents <> nil then
```

```
  begin
    try
      FEvents.OnChange();
    except
    end;
  end;
end;

procedure TMyInvisibleComponentX.DefinePropertyPages(DefinePropertyPage:
TDefinePropertyPage);
begin
  DefinePropertyPage( Class_DStringPropPage );
  {TODO: Define property pages here.  Property pages are defined by calling
    DefinePropertyPage with the class id of the page.  For example,
      DefinePropertyPage(Class_MyInvisibleComponentXPage); }
end;

procedure TMyInvisibleComponentX.EventSinkChanged(const EventSink: IUnknown);
begin
  FEvents := EventSink as IMyInvisibleComponentXEvents;
end;

function TMyInvisibleComponentX.Get_LastErrorStr: WideString;
begin
  Result:= fLastErrorStr;
end;

function TMyInvisibleComponentX.Get_Params: IStrings;
begin
  GetOleStrings(FMyInvisibleComponent, Result);
end;

function TMyInvisibleComponentX.GetParam(
  const Name: WideString): WideString;
begin
  Result:=FMyInvisibleComponent.Values[Name];
end;

procedure TMyInvisibleComponentX.InitializeControl;
begin
  inherited;
  FMyInvisibleComponent:= TStringList.Create({if required an Owner use Control});
  FMyInvisibleComponent.OnChange := ChangeEvent;
end;

procedure TMyInvisibleComponentX.Set_Params(const Value: IStrings);
begin
  SetOLEStrings(FMyInvisibleComponent, Value);
end;

procedure TMyInvisibleComponentX.SetParam(const Name, Value: WideString);
begin
  FMyInvisibleComponent.Values[Name]:= Value;
end;

function TMyInvisibleComponentX.Delete(aIndex: Integer): WordBool;
begin
  try
    FMyInvisibleComponent.Delete(0);  // raises exception if index out of item range
    Result:= True;
  except
```

```
    on E: Exception do
    begin
      Result:= False;
      fLastErrorStr:= E.Message;
    end;
  end;
end;

initialization
  TActiveXComponentFactory.Create(
    ComServer,
    TMyInvisibleComponentX,
    Class_MyInvisibleComponentX,
    1,
    '{8B69C9B8-A742-4F5C-98BF-44F7C3D41A03}',
    0,
    tmBoth);
end.
```

5.3. OleCmps.pas

```
(* OleCmps - ActiveX non-visual component development support in Delphi
 * Copyright (c) 2003 by Mandys Tomas-MandySoft
 *)

{ URL: http://www.2p.cz }

unit OleCmps;

interface
uses
  OleCtrls;

type
  TOleComponent = class(TOleControl)  // hide obsolete properties
  private
    fDummy: string;
  published
    property Height: string read fDummy;
    property Width: string read fDummy;
    property Hint: string read fDummy;
    property HelpContext: string read fDummy;
    property Cursor: string read fDummy;
  end;

implementation

end.
```