# Besa UI Automation Guide

http://www.besasoftware.com

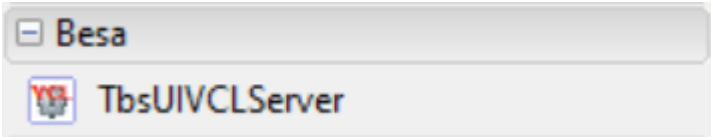
**Version 1.X**
**2021-05-15**

## Contents

Contents ..... 2
Introduction ..... 3
Adding Custom Components ..... 3
Sample code for ***TEdit*** control: ..... 3
Sample code for ***TLabel*** control: ..... 4
Using Table, Rows and Cells ..... 4

## Introduction

A light-weight and fast UI Automation Server component. Simple usage. Only Drag and Drop and Set *Active* property to *True*. It is ready. Supports custom component definition.




## Adding Custom Components

This is the documentation of the the settings of the **TbsUIVCLServer** component. We will then give some examples of custom component definitions. For more information about Microsoft UI Automation: UI Automation Overview

Delphi has two type visual control type:

1. Windows controls *(TWinControl based like: TButton,TPanel...)*
2. Native Delphi controls *(TGraphicControl based like: TLabel,TSpeedButton...)*

If you want to add your custom component to Besa UI Automation system. You must create a proxy class. If your class is TWinControl based your proxy class must be derivated from **TbsWinControl** else must be derivated from **TbsControl**. This proxy class must be registered with **AddVCLProviderClass** method.

### Sample code for ***TEdit*** control:

```
uses
  bsUICommon, bsUIVCL, BSUIAutomationCore, ...;

type
  // For windows controls used TbsWinControl
  TbsUiEdit=class(TbsWinControl)
  public
    function DoGetValue:String; override;
    function DoSetValue(AValue:String):Boolean; override;
    function DoGetControlType:Integer; override;
  end;


implementation


{ TbsUiEdit }


function TbsUiEdit.DoGetControlType: Integer;
begin
  Result:= UIA_TextControlTypeId;
end;

function TbsUiEdit.DoGetValue: String;
begin
  Result := TCustomEdit(FControl).Text;
end;
```

```
function TbsUiEdit.DoSetValue(AValue: String):boolean;
begin
  Result:=True;
  TCustomEdit(FControl).Text:=AValue;
end;

initialization
  //Register your provider class
  AddVCLProviderClass(TCustomEdit, TbsUiEdit);
end.
```

## Sample code for ***TLabel*** control:

```
Uses
  bsUICommon, bsUIVCL, BSUIAutomationCore, ...;


type
  // For delphi controls used TbsControl
  TbsUiLabel=class(TbsControl)
    function DoGetValue:String; override;
    function DoGetControlType:Integer; override;
  end;

implementation

{ TbsUiLabel }

function TbsUiLabel.DoGetControlType: Integer;
begin
  Result:=UIA_TextControlTypeId;
end;

function TbsUiLabel.DoGetValue: String;
begin
  Result:=TLabel(FControl).Caption;
end;

initialization
  AddVCLProviderClass(TCustomLabel, TbsUiLabel);

end.
```

## Using Table, Rows and Cells

Besa UI Automation components support Grids. If you have a grid component (must be TWinControl based) you must create a proxy class from **TbsTable** class. A table has Rows and Cells. And you must create proxy class for rows and cells. Proxy class required for rows and cells because UI Automation need bounds rectangle and set/get value for cells. Your row class must be derivated from **TbsTableRow** and cell class derivated from **TbsTableCell**.

if your component based:

| **Delphi Class** | **UI Class** |
|---|---|
| TCustomGrid | TbsUiCustomGrid |
| TStringGrid | TbsUiStringGrid |

| | |
|---|---|
| TDBGrid | TbsUiCustomDBGrid |
| *else use* | TbsTable |

Sample code for TStringGrid's base class ***TCustomGrid*** control:

```
uses
  bsUICommon, bsUIVCL, BSUIAutomationCore, ...;

type
  TbsUiCustomGridCell=class(TbsTableCell)
  protected
    function DoSetValue(AValue:String):Boolean; override;
    function DoGetValue:String; override;
    function GetCellRect(ARow,AColumn:LongWord): TRect; override;
  end;

  TbsUiCustomGridRow=class(TbsTableRow)
  public
    function GetTableCellClass:TbsTableCellClass; override;
    function GetRowRect(ARow:LongWord): TRect; override;
  end;

  TbsUiCustomGrid=class(TbsTable)
  protected
    function DoGetRowCount:Cardinal; override;
    function DoGetColumnCount:Cardinal; override;
    function DoGetColumnHeaders: TArrayOfString; override;
    function GetTableRowClass:TbsTableRowClass; override;
  end;

implementation

{ TbsUiCustomGrid }

function TbsUiCustomGrid.DoGetColumnCount: Cardinal;
begin
  Result:=// Get Col Count;
end;

function TbsUiCustomGrid.DoGetRowCount: Cardinal;
begin
  Result:= // Get Row Count;
end;

function TbsUiCustomGrid.DoGetColumnHeaders: TArrayOfString; override;
begin
  Result:=// Return Column Headers as array of string
end;

function TbsUiCustomGrid.GetTableRowClass: TbsTableRowClass;
begin
  Result:= TbsUiCustomGridRow;
end;

{ TbsUiCustomGridRow }
```

```
function TbsUiCustomGridRow.GetRowRect(ARow: LongWord): TRect;
begin
  // Calculate Row Rect.
  Result:=CalculatedRect;
end;

function TbsUiCustomGridRow.GetTableCellClass: TbsTableCellClass;
begin
  Result:= TbsUiCustomGridCell;
end;

{ TbsUiCustomGridCell }

function DoSetValue(AValue:String):Boolean; override;
begin
   //Set Row,Col cell's value
   Result:=True; // if succesfully
end;

function DoGetValue:String; override;
begin
  Result:=//Get Row, Col cell's value
end;

function TbsUiCustomGridCell.GetCellRect(ARow, AColumn: LongWord): TRect;
begin
  // Calculate Cell Rect
  Result:=CalculatedRect;
end;

initialization
  AddVCLProviderClass(TCustomGrid, TbsUiCustomGrid);

end.
```