1. Description

TCaptureX library permits the text extraction from the various applications. It contains a COM object called TextCaptureX, that exposes some methods that allow text extraction.

1.1. Creating a TextCaptureX object

First of all, the TCaptureX library must be registered on the system. This can be done using the following command:

regsvr32 <path>\TCaptureX.dll

where **<path>** is the actual path to the TCaptureX library. The registrations process is automatically done from the setup.

Following are some examples on how to create a TextCaptureX object in your application.

1.1.1. Visual C++:

In order to use the COM object in your C++ application, add the following import statement to your C++ source file, like this:

```
#import "<path>\TCaptureX.dll"
```

Optionally, you may add also:

```
using namespace TCaptureXLib;
```

in order to reference the objects without specifying the namespace.

Then create the object in one of the following two ways:

```
ITextCaptureXPtr obj(__uuidof(TextCaptureX));
```

Or

```
ITextCaptureXPtr obj = NULL;
HRESULT hr = obj.CreateInstance(__uuidof(TextCaptureX));
```

In order to destroy the object, call:

```
obj.Release();
```

Remark: before creating the object, **CoInitialize** API function must be called.

1.1.2. C#

- add TCaptureX library to the list of the project references
  Optionally, you may add at the beginning of the file:

```
using TCaptureXLib;
```

in order to reference the objects without specifying the namespace.

The actual creation is:

```
TextCaptureXClass obj = new TextCaptureXClass();
```

1.1.3. Visual Basic

- add TCaptureX library to the list of the project references
  The actual creation is:

```
Dim obj As New TextCaptureX
```

1.2. Using a TextCaptureX object

TextCaptureX object exposes the following methods:

- GetActiveWindowHwnd
- CaptureInteractive
- CaptureActiveWindow
- GetTextFromPoint
- GetTextFromRect

1.2.1. GetActiveWindowHwnd method

```
HRESULT GetActiveWindowHwnd([out,retval] LONG* hwnd);
```

It returns the handle of the active window. If the active window is a MDI container, it returns the handle of the active MDI child.

Examples:

Visual C++:

```
LONG hWnd = obj->GetActiveWindowHwnd();
```

C#:

```
long hWnd = obj.GetActiveWindowHwnd();
```

Visual Basic:

```
Dim hWnd As Long
hWnd = obj.GetActiveWindowHwnd()
```

1.2.2. CaptureInteractive method

```
HRESULT CaptureInteractive([out] LONG* hwnd, [out] LONG* left,
[out] LONG* top, [out] LONG* width, [out] LONG* height,
[out,retval] LONG* selection);
```

This method permits an automatic selection of the capture window and coordinates. When calling this method, the user is able to select with the mouse a rectangle on the screen, no matter what window. First left click specifies the point where the rectangle starts, the second left click specifies where the selection ends. At any time, pressing ESC key, aborts the selection.

Parameters:
LONG* hwnd : will contain the handle of the window
LONG* left: the X coordinate of the selected rectangle
LONG* top: the Y coordinate of the selected rectangle
LONG* width: the width of the selected rectangle
LONG* height: the width of the selected rectangle
LONG* selection: a custom value specifying the capture result:

- **0**, if the selection was ok
- **1**, if the user cancelled the capture
- **2**, if an error occurred

Examples:
Visual C++:

```
long hWnd = 0;
long nLeft = 0;
long nTop = 0;
long nWidth = 0;
long nHeight = 0;
LONG result = obj->CaptureInteractive(&hWndTarget, &nLeft,
&nTop, &nWidth, &nHeight);
if(0 == result) //selection ok
{
      //do something
}
```

C#:

```
long hWnd = 0;
long nLeft = 0;
long nTop = 0;
long nWidth = 0;
long nHeight = 0;
long result = obj.CaptureInteractive(out hWndTarget, out
nLeft, out nTop, out nWidth, out nHeight);
if(0 == result) //selection ok
{
        //do something
}
```

Visual Basic:

```
Dim result As Long
Dim hWnd As Long
Dim left As Long
Dim top As Long
Dim width As Long
Dim height As Long
result = obj.CaptureInteractive(hWnd, left, top, width,
height)
If(0 == result) Then 'selection ok
{
        'do something
}
End If
```

1.2.3. CaptureActiveWindow method

```
HRESULT CaptureActiveWindow([out,retval] BSTR* result);
```

This method captures the text from the window that is currently active, that is the window returned by the GetActiveWindowHwnd method. The return value is the entire text from the active window.
Remark: The application where this method is called is responsible of NOT containing the active window of the system.

Parameters:
BSTR* result : the text from the active window

Examples:
Visual C++:

```
_bstr_t result = obj->CaptureActiveWindow();
```

C#:

```
string result = obj.CaptureActiveWindow();
```

Visual Basic:

```
Dim result As String
result = obj.CaptureActiveWindow()
```

1.2.4. GetTextFromPoint method

```
HRESULT GetTextFromPoint([in] LONG hwnd, [in] LONG x, [in] LONG y,
[out,retval] BSTR* result);
```

This method captures the word specified by the point (x, y) in screen coordinates, in the window specified by hwnd.
Remark: One may use CaptureIntercative method to grab these parameters, that is window handle and coordinates.

Parameters:
LONG hwnd : handle of the window to capture from
LONG X : X coordinate of the point
LONG Y : Y coordinate of the point
BSTR* result : the word (if any) located at the specified point

Examples:
Visual C++:

```
LONG hwnd = 0;
LONG X = 0;
LONG Y = 0;
//set the values
.......
bstr_t bstrResult = obj->GetTextFromPoint(hwnd, X, Y);
```

C#:

```
long hwnd = 0;
long X = 0;
long Y = 0;
//set the values
.......
string result = obj.GetTextFromPoint(hwnd, X, Y);
```

Visual Basic:

```
Dim result As String
Dim hwnd As Long
Dim X As Long
Dim Y As Long
‘set the values
.......
result = obj.GetTextFromPoint(hwnd, X, Y)
```

1.2.5. GetTextFromRect method

```
HRESULT GetTextFromRect([in] LONG hwnd, [in] LONG left, [in] LONG top,
[in] LONG width, [in] LONG height, [out,retval] BSTR* result);
```

This method captures the text from the rectangle specified (left, top, width, height) in screen coordinates, in the window specified by hwnd.

Remark: One may use CaptureIntercative method to grab these parameters, that is window handle and coordinates.

Parameters:
LONG hwnd : handle of the window to capture from
LONG left : X coordinate of the upper left point of the rectangle
LONG top : **Y** coordinate of the upper left point of the rectangle
LONG width : width of the rectangle
LONG height : height of the rectangle
BSTR* result : the text located in the area specified by the rectangle

Examples:

Visual C++:

```
            LONG hwnd = 0;
            LONG left = 0;
            LONG top = 0;
            LONG width = 0;
            LONG height = 0;
            //set the values
            .......
            bstr_t bstrResult = obj->GetTextFromRect(hwnd, left, top,
      width, height);
```

C#:

```
            long hwnd = 0;
            long left = 0;
            long top = 0;
            long width = 0;
            long height = 0;
            //set the values
            .......
            string result = obj.GetTextFromRect(hwnd, left, top, width,
height);
```

Visual Basic:

```
            Dim result As String
            Dim hwnd As Long
            Dim left As Long
            Dim top As Long
            Dim width As Long
            Dim height As Long
            ‘set the values
            .......
            result   =   obj.GetTextFromRect(hwnd,   left,   top,   width,
height)
```