Edraw Viewer Component

for Excel V8

User's Guide

Contents

Edraw Viewer Component for Excel

Edraw Viewer Component for Excel is the enhanced solution of office viewer and focus in the Microsoft Excel. It contains a standard ActiveX control that acts as an ActiveX document container for hosting MS Excel sheets in a custom form or Web page. The control is lightweight and flexible, and gives developers new possibilities for using Excel workbook in a custom solution.

The control is designed to handle specific issues that make using ActiveX documents from a non-top-level host window difficult, and serves as a starting place for constructing your own embedded object file viewer or editor as an ActiveX control.

How to add "Edraw Viewer Component for Excel" to your Visual Basic 6.0 project

1. From the Project Menu select Components…
2. Select control "EDExcel ActiveX Control Module" in the controls table
3. Click the OK Button
4. The control will now appear in your toolbox
5. Drag and drop the control on your form

How to add "Edraw Viewer Component for Excel" to your .NET project

1. Open .NET
2. Right-click on the toolbox and select "Choose Items…"
3. Select the COM Components Tab

4. Check “EDExcel Control” and click OK Button

5. The control should appear in the toolbox as "Edraw Viewer Component for Excel"

6. Double-click on the control to add to your form

7. Resize and move the control on the form as desired

8. Add a button to the form

9. Double-click on the button to access code editor and enter the following code within the Click event: Note: Modify code to point to an existing document file on your web server

10. EDExcel1.Open "http://www.edrawsoft.com/demo/samples/sample.xls"

11. Run the application and click on the button

For ASP.NET or PHP project, you needn't add the component to the toolbar. View the “read me.txt” file in the “install folder\samples\ASP.NET\” folder

More help: http://www.edrawsoft.com/officeviewer.php

About .NET 64 Bit Windows:

Edraw Viewer Component for Excel can work well on 64 bit machine (Windows 7 64 bit or Vista 64 bit). But you need to pay attention to the below issue:

NOTE: To run the Any CPU version, you need change the file path in the \redist\x64\reg.bat. Then run it as the administrator.

1. Edraw Component X86 version (/redist/x86) is the 32 bit. If you use the x86 version in VB.Net, C#.NET, VC, be sure to compile your application to x86 instead of AnyCPU. Because AnyCPU .Net application will run as 64 bit process on 64 bit windows.
2. To use the component in the pure x64 project, you need use the 64 bit redistribution package in the /redist/x64 folder. Run the reg.bat with the Administrator privilege to register the 64 bit ocx file and dll.
4. Make sure the *.csproj has added the <PlatformTarget>AnyCPU</PlatformTarget> item.

more help:

http://www.edrawsoft.com/officeautomationcsharp.php
http://www.edrawsoft.com/64bit-activex-net.php

What's New

1. Added the 64 bit component redistribution package to support the pure 64 bit project.

2. Implemented "exclusive" Excel feature. If some excel workbooks get opened outside of the component, the component window and other Excel window can avoid interfering with each other.

3. Avoided to affect other MS Excel interface outside of the component when the component disables the current Excel window.

4. Added support to hide multiple workbooks on the windows taskbar even when the Windows in Taskbar option is unchecked.

5. Improved the MS Excel document open speed. Now the developers can decide whether close the old document before open a new document.

6. Added support to disable the MS Excel system button: Minimize Box, Maximize Box and Close Button.

7. Fixed the possible screen lock when the users switch to ribbon File menu by shortcuts.

8. Added support to prevent MS Excel window resize.

9. Fixed bug: the component can't hide MS Excel ribbon 2010 64 bit in IE 9.

10. Added ability to hide the Excel formula bar dynamically.

11. Added ability to prevent copy, save, print hotkey in MS Excel.

12. Fixed the print dialog event.

IMPORTANT: *For the component V7.x developers, you needn't change your project code to upgrade the V8 because the two version have the same methods and events interface. Only replace and regsvr32 the redistribution package.*

Methods

boolean IsExcelInstalled();

Returns whether the client installs the MS Excel program.

IDispatch ActiveDocument();*

Returns the Automation interface of the document object.

The method allows you to obtain a reference to the IDispatch interface of the embedded object. From this interface you can automate the object to perform tasks, edit parts of the document, or gather information about what a user has added or removed.

For example, you can create a new worksheet then write the cells:

```
<script language="javascript">
function ExcelAutomation()
{
objExcel.CreateNew();
var objExcel = objExcel.GetApplication();
var worksheet = objExcel.ActiveSheet;
worksheet.cells(1,1).value ="100";
worksheet.cells(1,2).value ="101";
worksheet.cells(1,3).value ="102";
worksheet.cells(2,1).value ="103";
worksheet.cells(2,2).value ="104";
worksheet.cells(2,3).value ="105";
}
</script>
```

How to do office automation in C#

IDispatch GetApplication();*

Returns the Automation interface of the application.

boolean CreateNew();

Creates a new, empty document.

Example
The following vb script shows how to open word template.

```
Sub NewDoc_Example()
edexcel.CreateNew
End Sub
```

boolean Open(BSTR FileName);

Opens the specified document.

FileName: The name of the document (needs the full paths or url).

Example
The following vb script shows how to open word file.

```
Sub LoadFile_Example()
edexcel.Open  "c:\test.xlsx"
End Sub


Sub LoadURL_Example()
edexcel.Open  "http://www.ocxt.com/demo/samples/sample.xls"
End Sub
```

boolean Save();

Saves the specified document. If the document hasn't been saved before, the Save As dialog box prompts the user for a file name.

boolean SaveAs([in] BSTR FilePath, [in, optional] VARIANT FileFormat);

Saves the document to specified location with the specified format.

FilePath: The name for the document. If a document with the specified file name already exists, the document is overwritten without the user being prompted first.
FileFormat: The format in which the document is saved. Can be any WdSaveFormat constant.

enum XlFileFormat

```
{
xlAddIn = 18,
xlCSV = 6,
xlCSVMac = 22,
xlCSVMSDOS = 24,
xlCSVWindows = 23,
xlDBF2 = 7,
xlDBF3 = 8,
xlDBF4 = 11,
xlDIF = 9,
xlExcel2 = 16,
xlExcel2FarEast = 27,
xlExcel3 = 29,
xlExcel4 = 33,
xlExcel5 = 39,
xlExcel7 = 39,
xlExcel9795 = 43,
xlExcel4Workbook = 35,
xlIntlAddIn = 26,
xlIntlMacro = 25,
xlWorkbookNormal = -4143,
xlSYLK = 2,
xlTemplate = 17,
xlCurrentPlatformText = -4158,
xlTextMac = 19,
xlTextMSDOS = 21,
xlTextPrinter = 36,
xlTextWindows = 20,
xlWJ2WD1 = 14,
xlWK1 = 5,
xlWK1ALL = 31,
xlWK1FMT = 30,
xlWK3 = 15,
xlWK4 = 38,
xlWK3FM3 = 32,
xlWKS = 4,
xlWorks2FarEast = 28,
xlWQ1 = 34,
xlWJ3 = 40,
xlWJ3FJ3 = 41,
xlUnicodeText = 42,
xlHtml = 44
}XlFileFormat;
```

Example
The following vb script shows how to save as a word document.

```
Sub SaveAs_Example()
edexcel.SaveAs  "c:\test.xls"
End Sub
```

boolean CloseDoc([in, optional] VARIANT SaveChanges);

Closes the specified document or documents.

SaveChanges: Specifies the save action for the document.

boolean IsDirty();

Returns True/False if file has been altered or needs save.

boolean IsOpened();

Returns True/Fase if file has been opened.

boolean OpenFileDialog([in, optional] VARIANT Filter);

Calls the standard file dialog to open the office document.

Filter: The file filter string.

Example
The following vb script shows how to open only the docx file dialog.

```
Sub OpenFileDialog_Example()
edexcel.OpenFileDialog  "Microsoft Excel
Files(*.xl;*.xlsx;*.xlsb;*.xlam;*.xltx;*.xltm;*.xls;*.xlt;*.xla;*.xlm;*.xlw)|*.xl;*.xlsx;*.xlsb;
*.xlam;*.xltx;*.xltm;*.xls;*.xlt;*.xla;*.xlm;*.xlw||”
End Sub
```

boolean SaveFileDialog([in, optional] VARIANT Filter);

Calls the standard file dialog to save the office document.

Filter: The file filter string.

boolean PageSetupDialog();

Calls the page setup dialog.

boolean DocPropertiesDialog();

Call the document properties dialog.

boolean PrintDialog();

Calls the print dialog.

boolean PrintOut([in, optional] VARIANT FromPage, [in, optional] VARIANT ToPage, [in, optional] VARIANT Copies);

Prints all or part of the specified document with settings.

FromPage: Optional Object. The starting page number when Range is set to wdPrintFromTo.

ToPage: Optional Object. The ending page number when Range is set to wdPrintFromTo.
Copies: Optional Object. The number of copies to be printed.

Example
The following vb script shows how to print the 1-6 page in a document.

```
Sub PrintOut_Example()
edexcel.PrintOut 1, 6, 1
End Sub
```

boolean PrintPreview();

Starts a print preview.

boolean PrintPreviewExit();

Exits a current print preview.

boolean SetValue([in] BSTR Name, [in] BSTR Value);

Sets the pasword, writepassword, domain and protectmodereminder for the document.

Name: The name string.
Value: The value string.

Example

The following vb script shows how to open a password-protected document. if the 1.docx file has the password 1234, the writepassword 5678, you can use the follow sample.

```
Sub SetValue_Example()
edexcel.SetValue  "Password", "1234"
edexcel.SetValue  "WritePassword", "5678"
'edexcel.SetValue "Domain", "www.edrawsoft.com"
'edexcel.SetValue "ProtectModeReminder", "The ActiveX Controls is not available for Internet sites under the enhanced security configuration."
edexcel.Open "c:\1.xls"
End Sub
```

boolean ProtectDoc(long ProtectType, [in, optional] VARIANT Password);

Helps to protect the specified document from changes. When a document is protected, users can make only limited changes, such as adding annotations, making revisions, or completing a form.

ProtectType: The protection type for the specified document. WdProtectionType.
Password: Optional Object. The password required to remove protection from the specified document.

```
typedef enum XlProtectType
{
        XlProtectTypeNormal = 0x00000001,
        XlProtectTypeWindow = 0x00000002,
        XlProtectTypeStruct = 0x00000004,
        XlProtectTypeDrawingObjects = 0x00000010,
        XlProtectTypeContents = 0x00000020,
        XlProtectTypeScenarios = 0x00000040,
        XlProtectTypeUserInterfaceOnly = 0x00000080,
}XlProtectType;
```

Example
The following vb script shows how to protect a document for only revisions.

```
Sub ProtectDoc_Example()
edexcel.ProtectDoc  XlProtectTypeNormal|XlProtectTypeWindow|XlProtectTypeStruct
End Sub
```

boolean UnProtectDoc([in, optional] VARIANT Password);

Removes protection from the specified document. If the document isn't protected, this method generates an error.

Password: Optional Object. The password required to remove protection from the specified document.

Example
The following vb script shows how to unprotect a document with password 832-f2322.

```
Sub Unprotect_Example()
edexcel.UnProtectDoc  "832-f2322"
End Sub
```

BSTR GetDocumentName();

Returns the name of the specified object.

BSTR GetDocumentFullName();

Specifies the name of a document, template, or cascading style sheet, including the drive or Web path.

boolean IsReadOnly();

Determines if changes to the document cannot be saved to the original document.

boolean DisplayGridline([in] boolean Show);

Shows/HIdes the grid lines in the office program.

boolean SwitchViewType([in] XlViewType ViewType);

Switches to the different view for office program.

ViewType: The view type for the specified document. XlViewType.

```
typedef enum XlViewType
{
        xlNormalView = 1 ,
        xlPageBreakPreview = 2,
}XlViewType;
```

Example
The following vb script shows how to switch to webview after document opened.

```
Sub DocumentOpenedEvent ()
Edword.SwitchViewType 1
End Sub

<SCRIPT FOR=OA1 EVENT= DocumentOpened ()>
   DocumentOpenedEvent()
</SCRIPT>
```

boolean ViewZoomTo([in] long ZoomFactor);

Zooms to the special zoom factor.

ZoomFactor: The zoom factor.

Example
The following vb script shows how to zoom the document to 200%.

```
Sub DocumentOpenedEvent ()
Edword.ViewZoomTo  200
End Sub

<SCRIPT FOR=OA1 EVENT= DocumentOpened ()>
   DocumentOpenedEvent()
</SCRIPT>
```

boolean DisableViewRightClickMenu(boolean Disable);

Disables the right click menu in the MS Excel.

boolean DisableFileCommand([in] WdUIType UIType, [in] boolean Disable);

UIType: The enum type need to disable in the UI. WdUIType.

Disable: True to disable the command button or menu item.

```
enum WdUIType
{
        wdUIDisalbeOfficeButton   =    0x00000001,
        wdUIDisalbeNew          =    0x00000002,
        wdUIDisalbeOpen          =    0x00000004,
        wdUIDisalbeUpgradeDocument     =    0x00000008,
        wdUIDisalbeSave         =    0x00000010,
        wdUIDisalbeSaveAs=    0x00000020,
        wdUIDisalbeSendAsAttachment   =    0x00000040,
        wdUIDisalbeClose =    0x00000100,
        wdUIDisalbePrint =    0x00000200,
```

```
        wdUIDisalbePrintQuick   =    0x00000400,
        wdUIDisalbePrintPreview   =    0x00000800,
        wdUIDisalbeSaveAsMenu   =    0x00001000,
        wdUIDisalbePrepareMenu   =    0x00002000,
        wdUIDisalbePermissionRestrictMenu   =    0x00004000,
        wdUIDisalbeSendMenu   =    0x00008000,
        wdUIDisalbePublishMenu          =    0x00010000,
        wdUIDisalbeServerTasksMenu   =    0x00020000,
        wdUIDisalbeCopyButton   =    0x00040000,
        wdUIDisalbeCutButton   =    0x00080000,
        wdUIHideMenuHome   =    0x01000000,
        wdUIHideMenuInsert   =    0x02000000,
        wdUIHideMenuPageLayout   =    0x04000000,
        wdUIHideMenuReferences   =    0x08000000,
        wdUIHideMenuMailings   =    0x10000000,
        wdUIHideMenuReview   =    0x20000000,
        wdUIHideMenuView   =    0x40000000,
        wdUIHideMenuDeveloper   =    0x80000000,
        wdUIHideMenuAddIns   =    0x00100000,
        wdUIHideMenuFormat   =    0x00200000,
        wdUIHideMenuEdit   =    0x00400000,
        wdUIHideMenuTool   =    0x00800000,
}WdUIType;
```

Note: The component disabled the Office menu, New button and Open button in default.

DWORD dwDisableCommand = wdUIDisalbeOfficeButton | wdUIDisalbeNew| wdUIDisalbeOpen;

The function need be set in the BeforeDocumentOpened event.

If you want to enable the three button, follow the samples.

Example
The following vb script shows how to enable the office main menu, new button and open button.

```
Sub DocumentOpenedEvent ()
Edword.DisableFileCommand 1 , false    'wdUIDisalbeOfficeButton
Edword.DisableFileCommand 2 , false    'wdUIDisalbeNew
Edword.DisableFileCommand 4 , false   'wdUIDisalbeOpen
End Sub

<SCRIPT FOR=OA1 EVENT= DocumentOpened ()>
   DocumentOpenedEvent()
</SCRIPT>
```

The following vb script shows how to diable the saveas and save button.

```
Sub DocumentOpenedEvent ()
Edword.DisableFileCommand 16 , true    'wdUIDisalbeSave
Edword.DisableFileCommand 32 , true    'wdUIDisalbeSaveAs
End Sub

<SCRIPT FOR=OA1 EVENT= DocumentOpened ()>
```

```
   DocumentOpenedEvent()
</SCRIPT>
```

boolean UpdateOffice2003ToolbarButton([in] long OfficeID, [in] boolean Visible, [in] boolean Enabled);

Disables or hides the office 2000, 2003 toolbar button.

OfficeID: The office MSO id for every button.

Visible: True to set the command button visible.

Enabled: True to enable the command button.

The function works only in the Office 2000/2003 version.

The following vb script shows how to hide the saveas and save button.

```
Sub DocumentOpenedEvent ()
Edword.UpdateOffice2003ToolbarButton 3 , false, false    `save button
Edword.UpdateOffice2003ToolbarButton 748 , false, false    `saveas button
End Sub

<SCRIPT FOR=OA1 EVENT= DocumentOpened ()>
   DocumentOpenedEvent()
</SCRIPT>
```

boolean InvisibleOffice2003Toolbar([in] BSTR ToolbarName);

Disables or hides the office 2000, 2003 toolbar.

ToolbarName: The office 2000, 2003 commandbar name.

The following vb script shows how to hide the standard toolbar, web toolbar and formatting toolbar.

```
Sub DocumentOpenedEvent ()
Edword.InvisibleOffice2003Toolbar "Standard"
Edword.InvisibleOffice2003Toolbar "Web"
Edword.InvisibleOffice2003Toolbar "Formatting"
End Sub

<SCRIPT FOR=OA1 EVENT= DocumentOpened ()>
   DocumentOpenedEvent()
</SCRIPT>
```

boolean HttpInit();

Initializes the HTTP connection.

boolean HttpAddPostString(BSTR Name, BSTR Value);

Adds the post parameter.

Name: The parameter name.
Value: The parameter value.

boolean HttpAddPostFile(BSTR LocalFilePath, [in, optional] VARIANT NewFileName);

Adds the post file.

LocalFilePath: The full file path needs to upload to the server. It the parameter is NULL, the function will add the file which is opening in the component.
NewFileName: The new file name for the upload file.

boolean HttpAddPostOpenedFile([in, optional] VARIANT NewFileName);

Adds the opened office file to post queue.

NewFileName: The new file name for the upload file.

boolean HttpPost(BSTR WebUrl, [in, optional] VARIANT WebUsername, [in, optional] VARIANT WebPassword);

Sends the specified request to the HTTP server.

WebUrl: A string containing the web url from which uploads data via HTTP.
WebUsername: A string containing the user name.
WebPassword: A string containing the access password.

NOTE: If you have not set the WebUsername and WebPassword, you need enable the window anonymous authentication.

The follow code is demo how to upload the opened file to server with the HTTP mode. It can also post multiple files in a post request.

```
<script language="vbscript">
Sub UploadFile()
edexcel.HttpInit
edexcel.HttpAddpostString "author", "anyname"
edexcel.HttpAddpostString "Data", "2010-5-15"
edexcel.HttpAddPostOpenedFile newfilename.xlsx
edexcel.HttpPost "http://localhost:1320/Samples/UploadAction.aspx"
End Sub
</script>
```

Or you can save your file to server with the HttpAddPostFile method.

```
OA1.Save()
OA1.CloseDoc()
OA1.HttpInit()
OA1.HttpAddPostFile "d:\1. xls "
document.OA1.HttpPost("http://localhost:1833/test/UploadAction.aspx");
```

Or:

```
OA1.SaveAs("d:\1.xls", 0)
OA1.SaveAs("d:\2. xls ", 0)
OA1.HttpInit()
OA1.HttpAddPostFile "d:\1. xls "
document.OA1.HttpPost("http://localhost:1833/test/UploadAction.aspx");
```

Note: If you try to save the opened file to remote server with the "HTTP" methods, you need write a receipt page in your web server. Because the component uploads the file by HTTP mode. The follow is some sample code.

```
ASP.NET:
//Default.aspx
<script language="vbscript">
Sub SavetoServer()
edexcel.HttpInit
edexcel.HttpAddpostString "author", "anyname"
edexcel.HttpAddpostString "Data", "2010-5-15"
edexcel.HttpAddPostOpenedFile
edexcel.HttpPost "http://localhost:1320/Samples/UploadAction.aspx"
'edexcel.Save "http://localhost:1320/UploadAction.aspx?author=name&Data=2"
End Sub
</script>

//UploadAction.aspx.cs file
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.Drawing.Imaging;
using System.Text.RegularExpressions;

public partial class UploadAction : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      if (Request. QueryString["author"] == "name" && Request. QueryString["Data"] == "2")
      {
         Response.Write("0\n");
         Response.Write("We have receipted the right param from Edraw Diagram ActiveX
Control.");
      }
```

```
      if (Request.Files.Count == 0)
      {
         Response.Write("0\n");
         Response.Write("There isn't file to upload.");
         Response.End();
      }
      if (Request.Files[0].ContentLength == 0)
      {
         Response.Write("0\n");
         Response.Write("Failed to receipt the data.\n\n");
         Response.End();
      }
      string fullFileName = Server.MapPath(Request.Files[0].FileName);
      Request.Files[0].SaveAs(fullFileName);
      Response.Write("Upload Successfully.");
      Response.End();
   }
}

PHP:
<?php
header("http/1.1 200 OK");
$user = iconv("UTF-8", "UNICODE", $_POST['user']);
$passwd = iconv("UTF-8", "UNICODE", $_POST['passwd']);
$sql = sprintf("username=%s  passwd=%s", $user,$passwd);
echo $sql;
$sql = sprintf("file=%s  size=%s error=%s tmp=%s",
$_FILES['trackdata']['name'],$_FILES['trackdata']['size'],$_FILES['trackdata']['error'],$_FILES
['trackdata']['tmp_name']);
echo $sql;
$handle = fopen($_FILES['trackdata']['name'],"w+");
if($handle == FALSE)
{
     exit("Create file error!");
}
$handle2 = fopen($_FILES['trackdata']['tmp_name'],"r");
$data = fread($handle2,$_FILES['trackdata']['size']);
echo $data;
fwrite($handle,$data);
fclose($handle2);
fclose($handle);
exit(0);
?>

ASP:  (review the full code in the install folder\samples\asp\)
<%@Language=VBScript %>
<!-- #include file="./include/upload.inc" -->
<!--#include file="./include/conn.asp"-->
<%
Set Uploader = New UpFile_Class
Uploader.NoAllowExt="cs;vb;js;exe"
Uploader.GetData (Request.TotalBytes)
Request.TotalBytes
if Uploader.isErr then
  select case Uploader.isErr
       case 1
       Response.Write "Fail to receipt the data."
       case 2
       Response.Write "The file is too big to upload"
```

```
	End select
	'Response.End
End if
Dim id
If "" <> Request.QueryString("id") then
	id = Request.QueryString("id")
End if
if id<>0 then
   Sql="SELECT * from doc where doc.id = "&id
else
   Sql="SELECT * from doc"
End if
rs.Open Sql,conn,1, 3
for each formName in Uploader.file
	set file=Uploader.file(formName)
	If id<>0 then
		If("" = Request.QueryString("isAip")) Then
			rs("DocContent") = Uploader.FileData(formname
			rs("DocID") = Uploader.Form("DocID")
			rs("DocTitle") = Uploader.Form("DocTitle")
			rs("DocType") = Uploader.Form("DocType")
		Else rs("AipContent")  = Uploader.FileData(formname)
			rs("state") = 2
		End if
	Else
		rs.AddNew
		rs("DocID") = Uploader.Form("DocID")
		rs("DocTitle") = Uploader.Form("DocTitle")
		rs("DocContent") = Uploader.FileData(formname)
		rs("Docdate") = Now()
		rs("DocType") = Uploader.Form("DocType")
		rs("state") = 1
	End If
set file=nothing
rs.Update
exit for
next
rs.Close
Set rs=Nothing
conn.close
set conn = Nothing
Set Uploader = Nothing
%>

JSP:
jsp:
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page language="java" import="com.jspsmart.upload.File>
<jsp:useBean id="mySmartUpload" scope="page" class="com.jspsmart.upload.SmartUpload"
/>
<%
String sPath="C:\\"
mySmartUpload.initialize(pageContext);
mySmartUpload.upload();
String TempName=mySmartUpload.getRequest().getParameter("TempName");
mySmartUpload.save(sPath);
File myFile =mySmartUpload.getFiles().getFile(0);
%>
```

boolean HttpOpenFileFromStream(BSTR WebUrl, [in, optional] VARIANT WebUsername, [in, optional] VARIANT WebPassword);

Opens a file from a stream with the HTTP/HTTPS.

WebUrl: A string containing the web url from which downloads data via HTTP. The WebUrl must include the file extend name so that the component know the file type. For example: http://www.edrawsoft.com/Getfile.aspx?ID=1002&FileName=guid.xls, or http://www.edrawsoft.com/sample.xlsx.
WebUsername: A string containing the user name.
WebPassword: A string containing the access password.

A full asp.net sample can be downloaded from

http://www.edrawsoft.com/download/openfromstreamtest.zip

Example:
Either you want to open an appointed file or open a file from database, for client side, all what you need do is the same, like following:

```
<script language="vbscript">
Sub DownloadFile()
      edexcel.HttpInit();
      edexcel.HttpAddpostString("DocumentID", "Tester.xls");
      edexcel.HttpOpenFileFromStream(strDownloadPath,"", "");
End Sub
</script>
```

Before you call function HttpOpenFileFromStream, you should do two things, one is to initialize http for clearing all parameters and cookies in http, another thing is to appoint the file or database record. And then use HttpOpenFileFromStream to send the request to the destinated webpage. Before HttpOpenFileFromStream send request, it will add a couple of parameters automatically.
edexcel.AddPostArgument(L"EDA_GETSTREAMDATA", L"EDA_YES"); This couple of parameters tell the destinated webpage edexcel will received file as stream.

At the web side, webpage will decide to read which file or database reacord accordding to the post parameters. And you should add boundary flag 'EDA_STREAMBOUNDARY' to file data, following is the asp.net demo.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.Drawing.Imaging;
```

```
using System.Text.RegularExpressions;

public partial class UploadAction : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      if (Request.Params["EDA_GETSTREAMDATA"] == "EDA_YES")
      {
         //string fullFileName = Server.MapPath("testnewone.txt");
         //String fs = File.ReadAllText(fullFileName);

         String fullFileName = Server.MapPath(Request.Params["DocumentID"]);
         Byte[] fs = File.ReadAllBytes(fullFileName);

         Response.Write("Get Stream Successfully!");
         Response.Write("EDA_STREAMBOUNDARY");
         Response.BinaryWrite(fs);
         Response.Write("EDA_STREAMBOUNDARY");
      }
      else
      {
         if (Request.Params["author"] == "anyname" && Request.Params["Data"] == "2007-
5-15")
         {
            Response.Write("0\n");
            Response.Write("We have receipted the right param from Office ActiveX Control.");
         }
         if (Request.Files.Count == 0)
         {
            Response.Write("0\n");
            Response.Write("There isn't file to upload.");
            Response.End();
         }
         if (Request.Files[0].ContentLength == 0)
         {
            Response.Write("0\n");
            Response.Write("Failed to receipt the data.\n\n");
            Response.End();
         }
         for (int i = 0; i < Request.Files.Count; i++)
         {
            string fullFileName = Server.MapPath(Request.Files[i].FileName);
            Request.Files[i].SaveAs(fullFileName);
         }
         Response.Write("Upload Successfully.");
         Response.End();
      }
   }
}
```

asp:

```
If l_bWrSreamBoundary
Then Response.Write "EDA_STREAMBOUNDARY"
While Not l_stream.EOS And Response.IsClientConnected
Response.BinaryWrite(l_stream.Read(l_nChunkSize))
Wend l_stream.Close
If l_bWrSreamBoundary
Then Response.Write EDA_STREAMBOUNDARY"
```

BSTR HttpDownloadFileToTempDir(BSTR WebUrl, [in, optional] VARIANT WebUsername, [in, optional] VARIANT WebPassword);

Downloads a file from a remote server then save it to OA temporary directory with HTTP.

WebUrl: A string containing the web url from which downloads data via HTTP. The WebUrl must include the file extend name so that the component know the file type. For example: http://www.edrawsoft.com/Getfile.aspx?ID=1002&FileName=guid.edxz, or http://www.edrawsoft.com/sample.edxz.
WebUsername: A string containing the user name.
WebPassword: A string containing the access password.
Return Value: Local temporary file path to save the download file.

The component provides the method to download file from a server then save to a local disk file. Support Http and HTTPS.

```
<script language="vbscript">
Sub DownloadFile()
	Dim sPath;
	sPath = edexcel.HttpDownloadFileToTempDir "http://www.edrawsoft.com/demo/1.xls"
	edexcel.Open sPath
End Sub
</script>
```

Note: You should make sure you the file exists in the web site firstly. A simple method is to enter the WebUrl in the Internet Explore. If IE can download the file, the component can do it too.

FAQ: If you are using Windows 2003 as your server, you maybe need to add the mime types to Internet Information Server.

boolean FtpConnect(BSTR WebUrl, [in, optional] VARIANT WebUsername, [in, optional] VARIANT WebPassword);

Creates a FTP connect.

boolean FtpDownloadFile(BSTR RemoteFile, BSTR LocalFile);

Downloads a file from remote server then save it to local file with FTP.

RemoteFile: The remote file path which saves to FTP site.
LocalFile: The full file path which downloads to the local disk.

boolean FtpUploadFile(BSTR LocalFile, BSTR RemoteFile, boolean OverWrite);

Uploads a local disk file to remote server with FTP.

LocalFilePath: The full file path needs to upload to the server. It the parameter is NULL, the function will add the file which is opening in the component.
RemoteFile: The remote file path which saves to FTP site.
OverWrite: Overwrites the exsited file if the same file exists at the FTP server.

boolean FtpDisConnect();

Closes the FTP Connect.

The follow code is demo how to download a file to local disk with the FTP mode.

```
<script language="vbscript">
Sub UploadFileviaFTP()
edexcel.FtpConnect “ftp.edrawsoft.com”, “username”, “password”
edexcel.FtpDownloadFile “ftp.edrawsoft.com/archieve/001.xls”, “c:\temp.xls”
edexcel.FtpDisConnect
End Sub
</script>
```

long GetErrorCode();

Returns the error code. You can referent the codes here:

http://msdn.microsoft.com/en-us/library/aa385465(v=vs.85).aspx

http://msdn.microsoft.com/en-us/library/aa384325(v=vs.85).aspx

BSTR GetTempFilePath([in, optional] VARIANT RefFileName);

Gets a temporary file path.

RefFileName: The referent file name.

boolean ClearTempFiles();

Clears these temporary files created by the component. The function won't delete any other files out of temporary ED directory.

boolean ExcelAddWorkSheet([in] long Index);

Adds a new worksheet to an opened Excel file.

boolean ExcelDeleteWorkSheet ([in] long Index);

Deletes a new worksheet from an opened Excel file.

boolean ExcelActivateWorkSheet ([in] long Index);

Activates the worksheet by the index.

long ExcelGetWorkSheetCount ();

Returns the counts of worksheets in the opened workbook.

boolean ExcelSetCellValue([in] long Column, [in] long Row, [in] BSTR Value);

Populates a specified cell with a string value.

BSTR ExcelGetCellValue([in] long Column, [in] long Row);

Returns the contents of the specified cell. ActiveCell (Column = -1, Row = -1)

boolean ExcelSetRowHeight([in] long Row, [in] double Height);

Sets the height of the specified rows.

boolean ExcelSetColumnWidth([in] long Column, [in] double Width);

Sets the width of the specified columns.

boolean ExcelDeleteRow([in] long Row);

Deletes the specified row.

boolean ExcelDeleteColumn([in] long Column);

Deletes the specified column.

boolean ExcelInsertRow([in] long Row);

Inserts a new row after the specified row.

boolean ExcelInsertColumn([in] long Column);

Inserts a new column after the specified column.

boolean ExcelInsertPageBreakInRow([in] long Row);

Inserts a page break after the specified row.

boolean ExcelInsertPageBreakInColumn([in] long Column);

Inserts a page break after the specified column.

boolean ExcelCopyToClipboard();

Copies the whole sheet to clipboard.

boolean ExcelPasteStringToWorksheet([in] BSTR bstText);

Inserts data to the worksheet like clipboard.

void ShowRibbonTitlebar([in] VARIANT_BOOL Show);

Shows/Hides the title bar in the office ribbon interface.

For Office 2007 and Office 2010, you can hide the Ribbon title bar with the method in the BeforeDocumentOpened event.

void ShowMenubar([in] VARIANT_BOOL Show);

Shows/Hides the menu bar in the office interface.

For Office 2000, Xp, 2003, you can hide the menu bar with the method in the BeforeDocumentOpened event. For office 2007 and higher version, you can use the DisableFileCommand method.

boolean SetComponentSize([in] long Width, [in] long Height);

Resizes the width and height of component.

void DisableStandardCommand([in] CommandType CmdType, [in] BOOL Disable);

Disables the standard file, save, print commands.

You can disable the standard commands in the office. Then write your own process in the event.

```
enum CommandType{
        cmdTypeSave = 0x00000001,
        cmdTypeClose = 0x00000002,
        cmdTypePrint = 0x00000004,
        cmdTypeRightClick = 0x00000008,
        cmdTypeDoubleClick = 0x00000010,
        cmdTypeIESecurityReminder = 0x00000020,
}CommandType;
```

The follow code is demo how to disable the print process.

```
function OA_DocumentBeforePrint()
{
document.OA1.DisableStandardCommand(4, true);//cmdTypePrint = 0x00000004,
}
<script language="javascript" for="OA1" event="DocumentBeforePrint()">
```

```
  OA_DocumentBeforePrint();
</script>
```

Event

[id(1), helpstring("Occurs after the control has completed the initialization.")]

void NotifyCtrlReady();

Example
The following script to hide the toolbar and grid when the component was initialized.

```
function edexcel_NotifyCtrlReady()
{
        document.all.edexcel.LicenseName = "";
        document.all. edexcel.LicenseCode = "";
        document.all.edexcel.BorderStyle = 1;
        document.all.edexcel.Toolbars = false;
}
<SCRIPT LANGUAGE=javascript FOR=edexcel EVENT=NotifyCtrlReady>
<!--
        edexcel_NotifyCtrlReady();
//-->
</SCRIPT>
```

[id(2), helpstring("Called when the new document is created.")]

*void **NewDocument**();*

[id(3), helpstring("Called before document is opened or new document added.")]

*void **BeforeDocumentOpened**();*

Example
The following script to add office UI setting before a document opened.

```
function edexcel_BeforeDocumentOpened()
{
        EDExcel.DisableFileCommand 1 , false    'wdUIDisalbeOfficeButton
        EDExcel.DisableFileCommand 2 , false    'wdUIDisalbeNew
        EDExcel.DisableFileCommand 4 , false  'wdUIDisalbeOpen
        EDExcel.DisableFileCommand 16 , true    'wdUIDisalbeSave
```

```
        EDExcel.DisableFileCommand 32 , true    'wdUIDisalbeSaveAs
}
<SCRIPT LANGUAGE=javascript FOR=edexcel EVENT= BeforeDocumentOpened >
<!--
        edexcel_ BeforeDocumentOpened ();
//-->
</SCRIPT>
```

[id(4), helpstring("Called when document is opened or new document added.")]

*void **DocumentOpened**();*

Example
The following script to add office automation after a document was opened.

```
function edexcel_DocumentOpened()
{
        var objExcel = document.OA1.GetApplication();
        var worksheet = objExcel.ActiveSheet;
        worksheet.cells(1,1).value ="100";
        worksheet.cells(1,2).value ="101";
        worksheet.cells(1,3).value ="102";
        worksheet.cells(2,1).value ="103";
        worksheet.cells(2,2).value ="104";
        worksheet.cells(2,3).value ="105";
}
<SCRIPT LANGUAGE=javascript FOR=edexcel EVENT= DocumentOpened >
<!--
        edexcel_ DocumentOpened ();
//-->
</SCRIPT>
```

[id(5), helpstring("Called before document is closed (may be canceled).")]

*void **BeforeDocumentClosed**();*

[id(6), helpstring("Called before document is saved (may be canceled).")]

*void **BeforeDocumentSaved**();*

[id(7), helpstring("Called before right click the component.")]

void ***WindowBeforeRightClick****();*

You can add your own right click menu here.

[id(8), helpstring("Called before double click the component.")]

void ***WindowBeforeDoubleClick****();*

[id(9), helpstring("Called before double click the component.")]

void ***WindowSelectionChange****();*

[id(10), helpstring("Called before double click the component.")]

void ***DocumentBeforePrint****();*

[id(11), helpstring("Called when the file was downloaded completely.")]

void ***WindowActivate****();*

[id(12), helpstring("Called when the file was downloaded completely.")]

void ***WindowDeactivate****();*

[id(13), helpstring("Called before downloading the file.")]

void ***BeforeDownloadFile****();*

[id(14), helpstring("Called when the file was downloaded completely.")]

void ***DownloadFileComplete****();*

[id(15), helpstring("Called when the file was uploaded completely.")]

void ***UploadComplete****();*

[id(16), helpstring("Called when the IE is in the protection mode.")]

void ***IESecurityReminder****([in,out] VARIANT* Cancel);*

Example
The following script to reminder the user added your site in trusted site list.

```
function edexcel_IESecurityReminder()
{
edexcel.SetValue "Domain", "www.edrawsoft.com"
'or you can customize the whole sentense by set the ProtectModeReminder value.
'edexcel.SetValue "ProtectModeReminder", "The ActiveX Controls is not available for Internet sites under the protection mode. You can add www.yoursite.com to trusted site list."
}
<SCRIPT LANGUAGE=javascript FOR=edexcel EVENT= IESecurityReminder>
<!--
        edexcel_ IESecurityReminder ();
//-->
</SCRIPT>
```

Property

[id(1)] boolean ShowToolbars;
Shows or hides the toolbars.

[id(2)] BSTR LicenseName;
Sets the license name.

[id(3)] BSTR LicenseCode;
Sets the license code.

[id(4)] OLE_COLOR BorderColor;
Sets the border color the control.

[id(5)] long BorderStyle;
Sets the border style of the control.

Constants

```
typedef enum BorderStyle
{
BorderNone = 0,
BorderFlat,
Border3D,
Border3DThin
} BorderStyle;

typedef enum FileCommandType
{
FileNew = 0,
FileOpen,
FileClose,
FileSave,
FileSaveAs,
FilePrint,
FilePageSetup,
FileProperties,
FilePrintPreview
} FileCommandType;

typedef enum XlFileFormat
{
xlAddIn = 18,
xlCSV = 6,
xlCSVMac = 22,
xlCSVMSDOS = 24,
xlCSVWindows = 23,
xlDBF2 = 7,
xlDBF3 = 8,
xlDBF4 = 11,
xlDIF = 9,
xlExcel2 = 16,
xlExcel2FarEast = 27,
xlExcel3 = 29,
xlExcel4 = 33,
xlExcel5 = 39,
xlExcel7 = 39,
xlExcel9795 = 43,
xlExcel4Workbook = 35,
xlIntlAddIn = 26,
xlIntlMacro = 25,
xlWorkbookNormal = -4143,
xlSYLK = 2,
xlTemplate = 17,
xlCurrentPlatformText = -4158,
xlTextMac = 19,
xlTextMSDOS = 21,
xlTextPrinter = 36,
xlTextWindows = 20,
xlWJ2WD1 = 14,
xlWK1 = 5,
```

```
xlWK1ALL = 31,
xlWK1FMT = 30,
xlWK3 = 15,
xlWK4 = 38,
xlWK3FM3 = 32,
xlWKS = 4,
xlWorks2FarEast = 28,
xlWQ1 = 34,
xlWJ3 = 40,
xlWJ3FJ3 = 41,
xlUnicodeText = 42,
xlHtml = 44,
xlWorkbookDefault = 51,
xlOpenXMLAddIn = 55,
xlOpenXMLTemplate = 54,
xlOpenXMLTemplateMacroEnabled = 53,
xlOpenXMLWorkbook  = 51,
xlOpenXMLWorkbookMacroEnabled = 52,
}XlFileFormat;

typedef enum XlViewType
{
xlNormalView = 1 ,
xlPageBreakPreview = 2,
}XlViewType;

typedef enum XlProtectType
{
XlProtectTypeNormal = 0x00000001,
XlProtectTypeWindow = 0x00000002,
XlProtectTypeStruct = 0x00000004,
XlProtectTypeDrawingObjects = 0x00000010,
XlProtectTypeContents = 0x00000020,
XlProtectTypeScenarios = 0x00000040,
XlProtectTypeUserInterfaceOnly = 0x00000080,
}XlProtectType;

typedef enum CommandType{
cmdTypeSave = 0x00000001,
cmdTypeClose = 0x00000002,
cmdTypePrint = 0x00000004,
cmdTypeRightClick = 0x00000008,
cmdTypeDoubleClick = 0x00000010,
cmdTypeIESecurityReminder = 0x00000020,
}CommandType;

typedef enum WdUIType
{
wdUIDisalbeOfficeButton   =    0x00000001,
wdUIDisalbeNew=    0x00000002,
wdUIDisalbeOpen =    0x00000004,
wdUIDisalbeUpgradeDocument    =    0x00000008,
wdUIDisalbeSave=    0x00000010,
wdUIDisalbeSaveAs=    0x00000020,
wdUIDisalbeSendAsAttachment   =    0x00000040,
wdUIDisalbeClose =    0x00000100,
wdUIDisalbePrint =    0x00000200,
wdUIDisalbePrintQuick   =    0x00000400,
wdUIDisalbePrintPreview   =    0x00000800,
```

```
wdUIDisalbeSaveAsMenu = 0x00001000,
wdUIDisalbePrepareMenu = 0x00002000,
wdUIDisalbePermissionRestrictMenu = 0x00004000,
wdUIDisalbeSendMenu = 0x00008000,
wdUIDisalbePublishMenu = 0x00010000,
wdUIDisalbeServerTasksMenu = 0x00020000,
wdUIDisalbeCopyButton = 0x00040000,
wdUIDisalbeCutButton = 0x00080000,
wdUIHideMenuHome = 0x01000000,
wdUIHideMenuInsert = 0x02000000,
wdUIHideMenuPageLayout = 0x04000000,
wdUIHideMenuReferences = 0x08000000,
wdUIHideMenuMailings = 0x10000000,
wdUIHideMenuReview = 0x20000000,
wdUIHideMenuView = 0x40000000,
wdUIHideMenuDeveloper = 0x80000000,
wdUIHideMenuAddIns = 0x00100000,
}WdUIType;

typedef enum OAErrorCode{
eSC_Ok = 0,
eSC_GenericError,
eSC_InvalidFileType,
eSC_InvalidSite,
eSC_WrongDNS,
eSC_CreateTempFileFailed,
eSC_OpenUploadFileFailed,
eSC_SaveOpenedFileFailed,
eSC_NotHttpURL,
eSC_ConnectFailed,
eSC_RequestFailed,
eSC_RequestHeaderFailed,
eSC_EmptyArgument,
eSC_OpenFileArgumentFailed,
eSC_SendRequestFailed,
eSC_WriteDataFailed,
eSC_ReadDataFailed,
eSC_EndRequestFailed,
eSC_OpenFileFailed,
eSC_InvalidReturnData,
}OAErrorCode;
```

System Requirements

Hardware :

Minimum:

Pentium Based MMX processor 500 MHZ
256 MB of ram
SVGA Graphics card
64 MB HardDisk
Recommended:

Intel Celeron or AMD Duron 1 GHz and above
1 GB ram
SVGA Graphics card with some acceleration (1024x768x16bitsColor)

Software :

Minimum:

Windows 2000 with IE6 and above.

Recommended:
IE 6/7/8
Windows 2000/XP/2003/Vista/Windows 7
Office 2000/XP/2003/2007/2010

For web application: The component need IE Protection Mode was Turn Off.

You can added your site at the IE trusted site list to turn off the IE protection mode automatically when the users visit your site with the component.

Please consider also your application's requirements.

Redistribute Files

Here are the files needed for installing Edraw Viewer Component for Excel.

DLL and OCX

EDExcel.ocx
EDOfficeViewerX.dll

Or you can use the cab file in the install folder for web application.

edexcel.cab

Visual Basic Issue

How to add Office ActiveX Control to your Visual Basic 6.0 project

1. From the Project Menu select Components…
2. Select control "EDExcel ActiveX Control Module" in the controls table.
3. Click the OK Button.
4. The control will now appear in your toolbox.
5. Drag and drop the control on your form.
6. Right click the control then choose the View Code… item.
7. Add the NotifyCtrlReady event and set the library file to load.
   ```
   Private Sub EDExcel1_NotifyCtrlReady()
   EDExcel1.LicenseName = ""
   EDExcel1.LicenseCode = ""
   End Sub
   ```
8. Run the project.

Visual C++ Issues

Adding the control to a simple dialog base Application

1. Begin a new MFC AppWizard(exe) dialog base Application.
2. Open The Resources Dialogs.
3. Select the Main Dialog and right click on it.
4. From pop-up menu select "Insert ActiveX Control…".
5. Select Edraw Viewer Component for Excel from the list.
6. Add a member variable to newly created control in the dialog class.
7. This will automatically generate new cpp and .h files which including the information of edexcel control.
8. Call the MFC ClassWizard to add the control event message

To resize the control according the form add a new Window Message Handler WM_SIZE and add the following code in OnSize virtual function

```
if(!::IsWindow(m_edexcel.m_hWnd)) return;

m_edexcel.MoveWindow(0,0,cx,cy);
```

See VCEDExcelDemo c++ example

Upgrade a control in VC++
You can easily upgrade a VC++ project witch is using a EDExcel control of a previous version:

1. Register the new version of control using the regsvr32 utility that is located in windows system directory.
2. From Project menu select Add to Project -> Components and Controls
3. From Components and Control Gallery dialog select the folder Registered ActiveX Controls
4. Find and select Edraw Viewer Component for Excel and click OK to all next dialogs.
5. Rebuild the project.

Web Application Issue

Use the edexcel.cab file

The component can be used for web application. You can embed it at the html Object tag.

<object classid="clsid: **367AD645-0381-4750-9B15-741067C2A118**" id="edexcel" width="100%" height="100%" codebase="http://www.yoursite.com/download/edexcel.cab#8,0,0,520"> </object>

Note: You should put the edexcel.cab file in your own site and change the codebase url when you distribute the component.

You can view the html samples in the install folder\samples\html folder.

The edexcel.cab file is available in the intall folder. You can create your own cab file.

Why do I fail to download the ActiveX control on the client machine

The failure of loading ActiveX control has the following possible causes:

1. The security settings of IE on the client machine are incorrect.

Please verify the following security settings of IE to "Prompt" or "Enabled":
a) Download signed ActiveX controls
b) Run ActiveX Controls and plug-ins
c) Script ActiveX controls marked safe for scripting

The dialog box of the security setting can be launched from menu Tools>Internet Options. Then select the security tab.

How to add Edraw Component to your ASP.NET project

1. Open Visual Studio.
2. Create a new ASP.NET project.
3. Don not attempt to add the Edraw Viewer Component for Excel to the Toolbox. It is a client component. You can add it as the HTML Object.

4. Copy all files at the ASP_c#\ to the new project folder.
(UploadAction.aspx UploadAction.aspx.cs Default.aspx.cs Default.aspx Tester.doc)
5. Then add exist items...
6. Modify the Server Port in the Default.aspx.
6. Run.

About the IE Protection Mode

The component can't run at the IE protection mode. So the client needs add the site in IE trusted site list. The component will pop up the reminder dialog if it run at the IE protection mode.

How to Delete the IE CAB Download

For IE7, IE8, IE9

1. Open a new Internet Explore.
2. Go to Internet Explorer, and click the "Tools" button in the left of browser, and then click Manage Add-ons.
3. Click Toolbars and Extensions.
4. Double Click the add-on you want to delete. In the pop up message box, you can click the Delete button in the bottom.

For IE6

IE Toolbar > Options > General > Setting > View Object...
Then delete the Edraw Viewer Component for Excel.

Convert to Full Version

The trial version has 60 day limiation.

After you put the order, you will get the full version download link and the license key.

Step:

1. Firstly you need uninstall the trial version from your computer. Make sure the edexcel.ocx file has been removed from your computer completed.
2. Install the full version. In the install folder, you can find all files to redistribute.
3. For VC, VB, C# destop application, you can set LicenseName and LicenseCode property directly in the Property panel.
4. For Web Application, you need set the two properies in the NotifyCtrlReady event.

```
<script language="javascript">
function edexcel_NotifyCtrlReady()
{
document.edexcel.LicenseName = "your license name";
document.edexcel.LicenseCode = "your license code";
}
</script>
<SCRIPT LANGUAGE=javascript FOR=edexcel EVENT=NotifyCtrlReady>
<!--
 edexcel_NotifyCtrlReady();
//-->
</SCRIPT>
<object classid="clsid: 367AD645-0381-4750-9B15-741067C2A118" id="edexcel"
width="100%" height="100%"
codebase="http://www.yoursite.com/download/edexcel.cab#8,0,0,520">
</object>
```

5. At last you can call the edexcel.AboutBox method to verify the license. The trial version will show "30 day trial license for evaluation". But the full license version will show your license name.

Online Store: http://www.edrawsoft.com/edexcel.php

License Agreement: http://www.edrawsoft.com/componentagreement.php

Support Email: support@ocxt.com

Other Edraw Components: http://www.edrawsoft.com/officeviewer.php