Edraw Office Viewer Component V7.5

User's Guide

Contents

Edraw Office Viewer Component

Edraw Office Viewer Component is the contains a standard ActiveX control that acts as an ActiveX document container for hosting MS Word, Excel, PowerPoint, Visio and Project in a custom form or Web page. The control is lightweight and flexible, and gives developers new possibilities for using Office program 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 Office Viewer Component" to your Visual Basic 6.0 project

1. From the Project Menu select Components…
2. Select control "Edraw Office Viewer 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 Office Viewer Component" 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 "Edraw Office Viewer Component" and click OK Button
5. The control should appear in the toolbox as "Edraw Office Viewer Component"
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. EDOffice.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 Office Viewer Component 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:

1. Edraw Component itself is the 32 bit and so does Office.
2. If you use Edraw Component 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.

Note: New method will be noted with *.

Methods

boolean IsOfficeInstalled(BSTR ProgID);

Returns whether the client installs the Special MS Office program.

ProgID: Word.Application, Excel.Application, PowerPoint.Application, Visio.Application, MSProject.Application

BSTR GetCurrentProgID();

Returns the program id of the current opened document.

Return: The value can be one of the follow values.

Word.Application

Excel.Application

PowerPoint.Application

Visio.Application

MSProject.Application

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.

How to do office automation in C#

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

```
<script language="javascript">
function OfficeAutomation()
{
EDOffice.CreateNew("Excel.Application");
var objExcel = EDOffice.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>
```

IDispatch GetApplication();*

Returns the Automation interface of the application.

Text Replace in the MS Word:

```
'vbscript code
<script language="vbscript">
Sub ReplaceText()
Set objWord = document.OA1.GetApplication
objWord.Selection.Find.ClearFormatting
objWord.Selection.Find.Replacement.ClearFormatting
objWord.Selection.Find.Text = "edraw"
objWord.Selection.Find.Replacement.Text = "cutedraw"
objWord.Selection.Find.Forward = True
objWord.Selection.Find.Wrap = 1
objWord.Selection.Find.Format = False
objWord.Selection.Find.MatchCase = False
objWord.Selection.Find.MatchWholeWord = False
objWord.Selection.Find.MatchWildcards = False
objWord.Selection.Find.MatchSoundsLike = False
objWord.Selection.Find.MatchAllWordForms = False
objWord.Selection.Find.Execute ,,,,,,,,,2
End Sub
</script>


' javascript code:
var appWord = document.OA1.GetApplication;
```

```
appWord.Selection.Find.Execute('somevalue', false, false, false,false, false, 1, false, false, 'somevalue', 2, false, false, false, false);

//Get Office Version
var version = appWord.Version;
```

boolean CreateNew(BSTR ProgID);

Creates a new, empty document.

ProgID: Word.Application, Excel.Application, PowerPoint.Application, Visio.Application, MSProject.Application.

Example
The following vb script shows how to open word.

```
Sub NewDoc_Example()
EDOffice.CreateNew "Word.Application"
End Sub
```

boolean Open(BSTR FileName, [in, optional] VARIANT ProgID);

Opens the specified document.

FileName: The name of the document (needs the full paths or url).
ProgID: **Word.Application, Excel.Application, PowerPoint.Application, Visio.Application, MSProject.Application.**

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

```
Sub LoadFile_Example()
EDOffice.Open  "c:\test.xlsx", "Excel.Application"
End Sub


Sub LoadURL_Example()
EDOffice.Open  "http://www.ocxt.com/demo/samples/sample.xls", "Excel.Application"
End Sub
```

The MS PowerPoint is the single instance program. Only one instance is allowed at the computer. The open method will return false if a PowerPoint instance has existed in the computer.

boolean NewWord([in, optional] VARIANT TemplatePath, [in, optional] VARIANT NewTemplate);

Creates a new, empty word document. Or creates a word document from a template.

TemplatePath: Optional Object. The name of the template to be used for the new document. If this argument is omitted, the Normal template is used.

NewTemplate: Optional Object. True to open the document as a template. The default value is False.

boolean OpenWord([in] BSTR FileName, [in, optional] VARIANT ConfirmConversions, [in, optional] VARIANT ReadOnly, [in, optional] VARIANT AddToRecentFiles, [in, optional] VARIANT PasswordDocument, [in, optional] VARIANT PasswordTemplate, [in, optional] VARIANT Revert,

[in, optional] VARIANT WritePasswordDocument, [in, optional] VARIANT WritePasswordTemplate, [in, optional] VARIANT Format);

FileName: Required Object. The name of the document (paths are accepted).

ConfirmConversions: Optional Object. True to display the Convert File dialog box if the file isn't in Microsoft Word format.

ReadOnly: Optional Object. True to open the document as read-only.

AddToRecentFiles: Optional Object. True to add the file name to the list of recently used files at the bottom of the File menu.

PasswordDocument: Optional Object. The password for opening the document.

PasswordTemplate: Optional Object. The password for opening the template.

Revert: Optional Object. Controls what happens if FileName is the name of an open document. True to discard any unsaved changes to the open document and reopen the file. False to activate the open document.

WritePasswordDocument: Optional Object. The password for saving changes to the document.

WritePasswordTemplate: Optional Object. The password for saving changes to the template.

Format: Optional Object. The file converter to be used to open the document. Can be a WdOpenFormat constant.

To specify an external file format, apply the OpenFormat property to a FileConverter object to determine the value to use with this argument.

boolean NewExcel([in, optional] VARIANT Template);

Creates a new workbook. The new workbook becomes the active workbook.

Template: Optional Object. Determines how the new workbook is created. If this argument is a string specifying the name of an existing Microsoft Excel file, the new workbook is created with the specified file as a template.

boolean OpenExcel(BSTR FileName, [in, optional] VARIANT UpdateLinks, [in, optional] VARIANT ReadOnly, [in, optional] VARIANT Format, [in, optional] VARIANT Password, [in, optional] VARIANT WriteResPassword, [in, optional] VARIANT IgnoreReadOnlyRecommended, [in, optional] VARIANT Origin, [in, optional] VARIANT Delimiter, [in, optional] VARIANT Editable, [in, optional] VARIANT Notify, [in, optional] VARIANT Converter, [in, optional] VARIANT AddToMru);

Opens a excel document.

Filename: The file name of the workbook to be opened.

UpdateLinks: Optional Object. Specifies the way links in the file are updated. If this argument is omitted, the user is prompted to specify how links will be updated. Otherwise, this argument is one of the values listed in the following table.

ReadOnly: Optional Object. True to open the workbook in read-only mode.

Format: Optional Object. If Microsoft Excel is opening a text file, this argument specifies the delimiter character, as shown in the following table. If this argument is omitted, the current delimiter is used.

Password: Optional Object. A string that contains the password required to open a protected workbook. If this argument is omitted and the workbook requires a password, the user is prompted for the password.

WriteResPassword: Optional Object. A string that contains the password required to write to a write-reserved workbook. If this argument is omitted and the workbook requires a password, the user will be prompted for the password.

IgnoreReadOnlyRecommended: Optional Object. True to have Microsoft Excel not display the read-only recommended message (if the workbook was saved with the Read-Only Recommended option).

Origin: Optional Object. If the file is a text file, this argument indicates where it originated (so that code pages and Carriage Return/Line Feed (CR/LF) can be mapped correctly).

Delimiter: Optional Object. If the file is a text file and the Format argument is 6, this argument is a string that specifies the character to be used as the delimiter. For example, use Chr(9) for tabs, use "," for commas, use ";" for semicolons, or use a custom character. Only the first character of the string is used.

Editable: Optional Object. If the file is a Microsoft Excel 4.0 add-in, this argument is True to open the add-in so that it's a visible window. If this argument is False or omitted, the add-in is opened as hidden, and it cannot be unhidden. This option doesn't apply to add-ins created in Microsoft Excel 5.0 or later. If the file is an Excel template, use True to open the specified template for editing or False to open a new workbook based on the specified template. The default value is False.

Notify: Optional Object. If the file cannot be opened in read/write mode, this argument is True to add the file to the file notification list. Microsoft Excel will open the file as read-only, poll the file notification list, and then notify the user when the file becomes available. If this argument is False or omitted, no notification is requested, and any attempts to open an unavailable file will fail.

Converter: Optional Object. The index of the first file converter to try when opening the file. The specified file converter is tried first; if this converter doesn't recognize the file, all other converters are tried. The converter index consists of the row numbers of the converters returned by the FileConverters property.

AddToMru: Optional Object. True to add this workbook to the list of recently used files. The default value is False.

boolean NewPowerPoint([in] long WithWindow);

Creates a presentation.

WithWindow: Whether the presentation appears in a visible window.

boolean OpenPowerPoint([in] BSTR FileName, [in] long ReadOnly, [in] long Untitled, [in] long WithWindow);

Opens the specified presentation

FileName: Required String The name of the file to open.

ReadOnly: Optional, Specifies whether the file is opened with read/write or read-only status.

Untitled: Optional, Specifies whether the file has a title.

WithWindow: Optional, Specifies whether the file is visible.

boolean NewVisio([in] BSTR TemplatePath);

Opens a new Visio document object.

TemplatePath: Required String. The type or file name of the document to add; if you do not include a path, Visio searches the folder or folders designated in the Application object's TemplatePaths property and all published templates, including published third-party templates.

Note: To create a new drawing based on no template, pass a zero-length string ("") to the Add method.

To create a new drawing based on a template, pass "templatename.vst" to the Add method.

Visio opens stencils that are part of the template's workspace and copies styles and other settings associated with the template to the new document. If the template file name is invalid, no document is returned and an error is generated.

To create a new stencil based on no stencil, pass ("vss").

To open a copy of a stencil, pass ("stencilname.vss").

To open a copy of a drawing, pass ("drawingname.vsd").

boolean OpenVisio([in] BSTR FileName, [in] short Flags);

Opens an existing Visio file using extra information passed in an argument.

FileName: Required String. The name of the file to open.

Flags: Required Integer. Flags that indicate how to open the file.

boolean NewProject([in, optional] VARIANT SummaryInfo, [in, optional] VARIANT Template, [in, optional] VARIANT FileNewDialog, [in, optional] VARIANT FileNewWorkpane);

Creates a new, empty project document.

SummaryInfo: Optional Boolean True if the Project Information dialog box is displayed when creating the project. The default is equal to the corresponding setting on the General tab of the Options dialog box.

Template: Optional String . A path and file name for a template to use when creating the project. If Template is omitted, a blank project is created.

FileNewDialog: Optional Boolean . True if the Templates dialog box is displayed when creating the project. If Template is specified, FileNewDialog is ignored.

FileNewWorkpane: Optional Boolean True if Project displays the New Project workpane before creating a new file. The default value is False.

boolean OpenProject([in, optional] VARIANT Name, [in, optional] VARIANT ReadOnly, [in, optional] VARIANT Merge, [in, optional] VARIANT TaskInformation, [in, optional] VARIANT Table, [in, optional] VARIANT Sheet, [in, optional] VARIANT NoAuto, [in, optional] VARIANT UserID, [in, optional] VARIANT DatabasePassWord, [in, optional] VARIANT FormatID, [in, optional] VARIANT Map, [in, optional] VARIANT openPool, [in, optional] VARIANT Password, [in, optional] VARIANT WriteResPassword, [in, optional]

VARIANT IgnoreReadOnlyRecommended, [in, optional] VARIANT XMLName);

Opens a project or imports data.

Name: Optional String. The name of the project file, source file, or data source to open. If Name is not specified, Project displays the Open dialog box.

ReadOnly: Optional Boolean. True if the file is opened read-only. If selectively importing data instead of loading a complete project, ReadOnly is ignored.

Merge: Optional Long Specifies whether to automatically merge the file (MPX and XMLDOM formats only) with the active project. To automatically merge XLS, CSV, or TXT file formats, you can set the merge key in the import map that you are using. The Map argument should be used in place of Merge, which is included primarily for backward compatibility. If Map is specified, Merge is ignored.

TaskInformation: Optional Boolean. True if the file contains information on tasks for a project saved under a non-Project file format. False if the file contains information on resources. The Map argument should be used in place of TaskInformation, which is included primarily for backward compatibility. If Map is specified, TaskInformation is ignored. The default value is True if the active view is a task view; otherwise it is False.

Table: Optional String. The name of a table in which to place the resource or task information for a project saved under a non-Project file format. Table is required if the value of the Merge argument is pjMerge. The Map argument should be used in place of Table , which is included primarily for backward compatibility. If Map is specified, or Name specifies a database file or format, Table is ignored. The default value for Table is the name of the active table.

Sheet Optional String The sheet to read when opening a workbook created in Microsoft Excel version 5.0 or later. The Map argument should be used in place of Sheet , which is included primarily for backward compatibility. If Map is specified, or if the file specified by Name is not a Microsoft Excel file, Sheet is ignored.

NoAuto: Optional Boolean. True if any Auto_Open macro is prevented from running. The default value is False.

UserID: Optional String. A user ID to use when accessing a database. If Name or FormatID is not a database, UserID is ignored.

DatabasePassWord: Optional String. A password to use when accessing a database. If Name or FormatID is not a database, DatabasePassWord is ignored.

FormatID: Optional String. String for the file or database format. If Project recognizes the format of the file specified with Name, FormatID is ignored. FormatID can be one of the following values.

Format String Description

"MSProject.mpp.9" Project file

"MSProject.mpt.9" Project template

"MSProject.mpd" Project database

"MSProject.mpw" Project workspace

"MSProject.mpx" Project 4.0 MPX file

"MSProject.odbc" ODBC database

"MSProject.mdb.9" Microsoft Access database

"MSProject.xls5.9" Microsoft Excel workbook

"MSProject.csv.9" CSV (comma-delimited) file

"MSProject.txt.9" TXT (tab-delimited) file

"MSProject.XMLDOM" XML file

Map: Optional String. The name of the import/export map to use when importing data.

openPool Optional Long The action to take when opening a resource pool or sharer file. When opening a master project, the value for this argument is also applied to the subprojects.

Password: Optional String. A password to use when opening password-protected project files. If Password is incorrect or omitted and a file requires a password, the user is prompted for the password.

WriteResPassword: Optional String. A password to use when writing to a write-reserved project file. If WriteResPassword is omitted and the file requires a password, the user is prompted for the password.

IgnoreReadOnlyRecommended: Optional Boolean. Variant True to prevent Project from displaying an alert that the project should be opened read-only. If the project was not saved with a read-only recommendation, IgnoreReadOnlyRecommended is ignored.

XMLName: Optional Variant. This is the XML DOM object that is passed to the function when the FormatID is MSProject.XMLDOM. The method should fail if this format is specified and XMLName is not a valid XML DOM Object. If the formatID is anything other than MSProject.XMLDOM, XMLName should be NULL and the method should fail otherwise. Only one of XMLName or Name may be specified, with the other one being NULL.

DoNotLoadFromEnterprise: Optional Variant.

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 WdSaveFormat
{
wdFormatDocument = 0,
wdFormatTemplate = 1,
wdFormatText = 2,
wdFormatTextLineBreaks = 3,
wdFormatDOSText = 4,
wdFormatDOSTextLineBreaks = 5,
wdFormatRTF = 6,
wdFormatUnicodeText = 7,
wdFormatEncodedText = 7,
wdFormatHTML = 8,
wdFormatWebArchive = 9,
wdFormatFilteredHTML = 10,
wdFormatXML = 11
}WdSaveFormat;

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;

enum PpSaveAsFileType
{
ppSaveAsPresentation = 1,
ppSaveAsPowerPoint7 = 2,
ppSaveAsPowerPoint4 = 3,
ppSaveAsPowerPoint3 = 4,
ppSaveAsTemplate = 5,
ppSaveAsRTF = 6,
ppSaveAsShow = 7,
ppSaveAsAddIn = 8,
ppSaveAsPowerPoint4FarEast = 10,
ppSaveAsDefault = 11,
ppSaveAsHTML = 12,
ppSaveAsHTMLv3 = 13,
ppSaveAsHTMLDual = 14,
ppSaveAsMetaFile = 15,
ppSaveAsGIF = 16,
```

```
ppSaveAsJPG = 17,
ppSaveAsPNG = 18,
ppSaveAsBMP = 19,
ppSaveAsOpenXMLPresentation = 24,
ppSaveAsOpenXMLPresentationMacroEnabled = 25,
ppSaveAsOpenXMLShow = 28,
ppSaveAsOpenXMLShowMacroEnabled = 29,
ppSaveAsOpenXMLTemplate = 26,
ppSaveAsOpenXMLTemplateMacroEnabled = 27,
}PpSaveAsFileType;
```

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

```
Sub SaveAs_Example()
EDOffice.SaveAs  "c:\test.doc", 8
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()
EDOffice.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 PrintDialog();

Calls the print dialog.

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

Prints all or part of the specified document with settings.

PrintRange: Optional Object. The page range. Can be any WdPrintOutRange constant.

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.
Pages: Optional Object. The page numbers and page ranges to be printed, separated by commas. For example, "2, 6-10" prints page 2 and pages 6 through 10.
Copies: Optional Object. The number of copies to be printed.

```
enum WdPrintOutRange
{
      wdPrintAllDocument = 0,
      wdPrintSelection = 1,
      wdPrintCurrentPage = 2,
      wdPrintFromTo = 3,
      wdPrintRangeOfPages = 4
};
```

Example
The following vb script shows how to print the 3-6 page in a document.

```
Sub PrintOut_Example()
EDOffice.PrintOut 3, 3, 6
End Sub
```

BSTR GetActivePrinter();

Returns the name of the active printer.

void SetActivePrinter([in] BSTR PrinterName);

Sets the name of the active printer.

boolean PrintPreview();

Starts a print preview.

boolean PrintPreviewExit();

Exits a current print preview.

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

Sets the password, 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()
EDOffice.SetValue  "Password", "1234"
EDOffice.SetValue  "WritePassword", "5678"
'EDOffice.SetValue "Domain", "www.edrawsoft.com"
'EDOffice.SetValue "ProtectModeReminder", "The ActiveX Controls is not available for Internet sites under the enhanced security configuration."
EDOffice.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.

For Excel:

```
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()
EDOffice.ProtectDoc  XlProtectTypeNormal|XlProtectTypeWindow|XlProtectTypeStruct
End Sub
```

For Word:

```
enum WdProtectType
{
        wdAllowOnlyRevisions = 0,
        wdAllowOnlyComments = 1,
        wdAllowOnlyFormFields = 2,
        wdAllowOnlyReading = 3,
        wdNoProtection = -1,
}WdProtectType;
```

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

```
Sub ProtectDoc_Example()
EDOffice.ProtectDoc  0
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()
EDOffice.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.

For Excel:

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

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

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

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

For Word:

```
enum WdViewType
{
        wdMasterView = 5,
        wdNormalView = 1 ,
        wdOutlineView = 2,
        wdPrintPreview = 4 ,
        wdPrintView = 3,
        wdReadingView = 7,
        wdWebView = 6,
}WdViewType;
```

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

```
Sub DocumentOpenedEvent ()
EDOffice.SwitchViewType 6
```

boolean ViewZoomTo([in] WdPageFit FitType, [in, optional] VARIANT ZoomFactor);

Zooms to the special zoom factor.

FitType: The zoom mode for the specified document. WdPageFit.
ZoomFactor: The zoom factor.

For Word:

```
enum WdPageFit
{
        wdPageFitTextNone = 0 ,
        wdPageFitFullPage = 1,
```

```
        wdPageFitBestFit = 2,
        wdPageFitTextFit = 3 ,
}WdPageFit;
```

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

```
Sub DocumentOpenedEvent ()
EDOffice.ViewZoomTo -1, 200
' fit best fit
'EDOffice.ViewZoomTo 2
End Sub

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

For Excel and PowerPoint:

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

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

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

boolean DisableViewRightClickMenu(boolean Disable);

Disables the right click menu in the MS Word and 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 ()
EDOffice.DisableFileCommand 1 , false    'wdUIDisalbeOfficeButton
EDOffice.DisableFileCommand 2 , false    'wdUIDisalbeNew
EDOffice.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 ()
EDOffice.DisableFileCommand 16 , true    'wdUIDisalbeSave
EDOffice.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 ()
EDOffice.UpdateOffice2003ToolbarButton 3 , false, false    'save button
EDOffice.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 ()
EDOffice.InvisibleOffice2003Toolbar "Standard"
EDOffice.InvisibleOffice2003Toolbar "Web"
EDOffice.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, ([in, optional] VARIANT FileFormat);*

Adds the opened office file to post queue.

NewFileName: The new file name for the upload file.

FileFormat: The format in which the document is saved before upload. Can be any constant. (See SaveAs method)

The method allows the developer to save the open file at the special format then upload. For example, you can save the open ms word files as the *.docx format then upload.

```
if(document.OA1.IsOpened)

{

document.OA1.SetAppFocus();

document.OA1.HttpInit();

var sFileName = "test_docx1.docx";

document.OA1.HttpAddPostOpenedFile(sFileName, 12 );

document.OA1.HttpPost("http://localhost/officeviewer/upload_weboffice.php");

}
```

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.

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()
EDOffice.HttpInit
EDOffice.HttpAddpostString "author", "anyname"
EDOffice.HttpAddpostString "Data", "2010-5-15"
EDOffice.HttpAddPostOpenedFile
EDOffice.HttpPost "http://localhost:1320/Samples/UploadAction.aspx"
End Sub
</script>
```

Save as docx file then upload.

```
<script language="vbscript">
Sub UploadFile()
EDOffice.HttpInit
EDOffice.HttpAddpostString "author", "anyname"
EDOffice.HttpAddpostString "Data", "2010-5-15"
EDOffice.HttpAddPostOpenedFile "newname.docx", 12
EDOffice.HttpPost "http://localhost:1320/Samples/UploadAction.aspx"
End Sub
</script>
```

Save as doc file then upload.

```
<script language="vbscript">
Sub UploadFile()
EDOffice.HttpInit
EDOffice.HttpAddpostString "author", "anyname"
EDOffice.HttpAddpostString "Data", "2010-5-15"
EDOffice.HttpAddPostOpenedFile "newname.doc", 0
EDOffice.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.doc"
document.OA1.HttpPost("http://localhost:1833/test/UploadAction.aspx");
```

Or:

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

Or:

```
OA1.SaveAs("d:\1.docx", 12)
OA1.SaveAs("d:\2.docx", 12)
OA1.HttpInit()
OA1.HttpAddPostFile "d:\1.docx"
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()
EDOffice.HttpInit
EDOffice.HttpAddpostString "author", "anyname"
EDOffice.HttpAddpostString "Data", "2010-5-15"
EDOffice.HttpAddPostOpenedFile newfilename
EDOffice.HttpPost "http://localhost:1320/Samples/UploadAction.aspx"
'EDOffice.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, BSTR ProgID, [in, optional] VARIANT WebUsername, [in, optional] VARIANT WebPassword);

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

A full asp.net sample can be downloaded from

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

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.
ProgID: Word.Application, Excel.Application or PowerPoint.Application.
WebUsername: A string containing the user name.
WebPassword: A string containing the access password.

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()
	EDOffice.HttpInit();
	EDOffice.HttpAddpostString("DocumentID", "Tester.xls");
	EDOffice.HttpOpenFileFromStream(strDownloadPath, ”Excel.Application”);
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.
EDOffice.AddPostArgument(L"EDA_GETSTREAMDATA", L"EDA_YES"); This couple of parameters tell the destinated webpage EDOffice will received file as stream.

At the web side, webpage will decide to read which file or database record according 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 = EDOffice.HttpDownloadFileToTempDir "http://www.edrawsoft.com/demo/1.doc"
      EDOffice.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()
EDOffice.FtpConnect “ftp.edrawsoft.com”, “username”, “password”
EDOffice.FtpDownloadFile “ftp.edrawsoft.com/archieve/001.doc”, “c:\temp.doc”
EDOffice.FtpDisConnect
End Sub
</script>
```

long GetErrorCode();

Returns the error code.

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 SetComponentSize(long Width, long Height);

Sets the width and hieght of the component.

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.
More Detail: http://www.edrawsoft.com/show-hide-office-menus.php

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

Resizes the width and height of component.

boolean SetAppFocus ();

Sets the office program focus.

boolean ExitOfficeApp ();

Exits the office program.

**boolean GetOfficeVersion ();*

Returns the current office version number.

**boolean GetOfficeHwnd ();*

Returns the HWND of the current office main window.

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>
```

boolean WordDisableSaveHotKey(boolean Disable);

Disables the Save keycodes in the MS Word.

Disalbe: Disables the CTRL+S, SHIFT+F12, Alt+SHIFT+F2, F12

boolean WordDisablePrintHotKey(boolean Disable);

Disables the Print keycodes in the MS Word.

Disalbe: Disables the CTRL+P, CTRL+SHIFT+F12, CTRL+F2, ALT+CTRL+I

boolean WordDisableCopyHotKey(boolean Disable);

Disables the Copy keycodes in the MS Word.

Disalbe: Disables the CTRL+C, CTRL+V, CTRL+X, SHIFT+DEL, SHIFT+INSERT, ALT+CTRL+V, CTRL+SHIFT+C, CTRL+INSERT

boolean WordShowRevisions([in] boolean Show);

Shows/Hides the revisions for the office program.

boolean WordAcceptAllRevisions([in] boolean Receipt);

Accepts or rejects all the tracked changes in a document or range.

long WordGetBookmarkCount();

Returns the number of bookmarks.

BSTR WordReadBookmarkInfo([in] long Pos, [in] boolean NameOrValue);

Returns the name or value of the special bookmark.

Pos: The index of bookmarks in the document. From 1 to …
NameOrValue: True returns the bookmark name, False returns the bookmark value.

Example
The following java script shows how to read the bookmark information.

```
function Readbookmark_Example ()
{
Var count = edword.GetBookmarkCount();
For(int i=1; i<=count; i++)
{
Var name = edword.GetBookmarkInfo(i, true);
Var value = edword.GetBookmarkInfo(i, false);
}
```

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

Writes the value for the special bookmark.

Name: The bookmark name.
Value: The bookmark value.

boolean WordInsertFile([in] BSTR FilePath, [in, optional] VARIANT InDocPos);

Inserts a file to opened Word file.

FilePath: The file path need to be inserted.
InDocPos: The insert position. WdInPocPos.

```
enum WdInDocPos
{
        wdInDocumentPosCursor = 1,
        wdInDocumentPosStart = 2,
        wdInDocumentPosEnd = 3 ,
}WdInDocPos;
```

boolean WordInsertText([in] BSTR Text, [in, optional] VARIANT InDocPos);

Inserts text content to opened Word file.

Text: The text string need to be inserted.
InDocPos: The insert position. WdInPocPos.

```
enum WdInDocPos
{
        wdInDocumentPosCursor = 1,
        wdInDocumentPosStart = 2,
        wdInDocumentPosEnd = 3 ,
}WdInDocPos;
```

boolean WordInsertPicture([in] BSTR FilePath, [in] boolean InlineObject, [in, optional] VARIANT InDocPos);

Inserts a picture to opened Word file.

FilePath: The image path need to be inserted.
InDocPos: The insert position. WdInPocPos.

```
enum WdInDocPos
{
        wdInDocumentPosCursor = 1,
        wdInDocumentPosStart = 2,
        wdInDocumentPosEnd = 3 ,
}WdInDocPos;
```

boolean WordInsertBreak([in] WdBreakType BreakType);

Inserts a break to opened Word file.

BreakType: The break type. WdBreakType.

```
typedef enum WdBreakType
{
        wdPageBreak = 7,
        wdColumnBreak = 8,
        wdSectionBreakNextPage = 2,
        wdSectionBreakContinuous = 3,
        wdSectionBreakEvenPage = 4,
        wdSectionBreakOddPage = 5,
        wdLineBreak = 6,
```

```
        wdLineBreakClearLeft = 9,
        wdLineBreakClearRight = 10,
        wdTextWrappingBreak = 11,
}WdBreakType;
```

Example
The following java script shows how to insert a line break.

```
function InsertBreak_Example ()
{
Edword.WordInsertBreak( 6 );
}
```

boolean WordGotoItem([in] WdGoToItem What, [in] WdGoToDirection Which, [in, optional] VARIANT Count, [in, optional] VARIANT Name);

Goes to the specified item in the Word document.

What: Optional Object. The kind of item to which the range or selection is moved. Can be one of the WdGoToItem constants.

Which: Optional Object. The item to which the range or selection is moved. Can be one of the WdGoToDirection constants.

Count: Optional Object. The number of the item in the document. The default value is 1. Only positive values are valid. To specify an item that precedes the range or selection, use wdGoToPrevious as the Which argument and specify a Count value.

Name: Optional Object. If the What argument is wdGoToBookmark, wdGoToComment, wdGoToField, or wdGoToObject, this argument specifies a name.

```
enum WdGoToItem
{
        wdGoToStart = 101,
        wdGoToEnd = 102,
        wdGoToBookmark = -1 ,
        wdGoToComment = 6 ,
        wdGoToEndnote = 5 ,
        wdGoToEquation = 10 ,
        wdGoToField = 7 ,
        wdGoToFootnote = 4 ,
        wdGoToGrammaticalError=  14 ,
        wdGoToGraphic = 8 ,
        wdGoToHeading=  11 ,
        wdGoToLine = 3 ,
        wdGoToObject = 9 ,
        wdGoToPage = 1 ,
        wdGoToPercent = 12 ,
        wdGoToProofreadingError = 15 ,
        wdGoToSection = 0 ,
```

```
        wdGoToSpellingError = 13 ,
        wdGoToTable = 2 ,
}WdGoToItem;

enum WdGoToDirection
{
        wdGoToAbsolute = 1,
        wdGoToFirst = 1,
        wdGoToLast = -1 ,
        wdGoToNext = 2 ,
        wdGoToPrevious = 3 ,
        wdGoToRelative = 2 ,
}WdGoToDirection;
```

Example
The following java script shows how to go to the file end.

```
function GoToItem_Example ()
{
Edword.WordGoToItem( 102, 1 );
}
```

boolean WordReplaceText([in] BSTR Text, [in] BSTR ReplaceText, [in] boolean MatchWholeWord, [in] boolean MatchCase);

Replaces all the specified string value with another string value.

Text: Optional Object. The text to be searched for.

ReplaceText: The replacement text.

MatchCase: Optional Object. True to specify that the find text be case-sensitive. Corresponds to the Match case check box in the Find and Replace dialog box (Edit menu).

MatchWholeWord: Optional Object. True to have the find operation locate only entire words, not text that's part of a larger word. Corresponds to the Find whole words only check box in the Find and Replace dialog box.

Boolean WordMergeAndCompare([in] BSTR TargetFilePath);

Compares and merges documents then displays it in the current windows.

TargetFilepath: Required String.

boolean WordDisableDragAndDrop(boolean Disable);

Disables drag and drop.

long WordGetRevisionCount();

Returns the number of Revisions.

BSTR WordReadRevisionInfo([in] long Pos, [in] WdRevisionType RevType);

Returns the name or value of the special bookmark.

Pos: The index of bookmarks in the document. From 1 to …
RevType: The revision type. WdRevision.

```
enum WdRevisionType
{
        wdRevisionAuthor = 0,
        wdRevisionDate = 1,
        wdRevisionType = 2,
        wdRevisionText = 3,
}WdRevisionType;
```

Example
The following java script shows how to read the bookmark information.

```
function ReadRevision_Example ()
{
Var count = edword.WordGetRevisionCount();
For(int i=1; i<=count; i++)
{
Var author = edword.WordReadRevisionInfo(i, 0);
Var text = edword. WordReadRevisionInfo (i, 3);
}
```

boolean WordAcceptRevision([in] long Pos, [in] boolean Accept);

Accepts or rejects the specified tracked change.

Pos: The index of bookmarks in the document. From 1 to …
Accept: Accepts or rejects the revision.

boolean WordDisableViewRightClickMenu(boolean Disable);

Disables the right click menu in the MS Word.

boolean WordCopyToClipboard();

Copies the whole content to the clipboard.

boolean WordCopyToClipboardAsPicture();

Copies the whole content to the clipboard as picture.

boolean WordPasteFromClipboard([in, optional] VARIANT InDocPos);

Pastes to the opened Word file from the clipboard data.

InDocPos: The position. WdInDocPos.

```
enum WdInDocPos
{
        wdInDocumentPosCursor = 1,
        wdInDocumentPosStart = 2,
        wdInDocumentPosEnd = 3 ,
}WdInDocPos;
```

boolean WordPasteSpecialFromClipboard([in] WdPasteDataType lFormatType, [in] boolean vFloatOverText, [in, optional] VARIANT InDocPos);

Pastes to the opened Word file from the clipboard data with special format.

lFormatType: Paste format type. WdPasteDateType.

vFloatOverText: True = float the object over text.

InDocPos: The position. WdInDocPos.

```
enum WdPasteDataType
{
        wdPasteBitmap = 4,
        wdPasteDeviceIndependentBitmap = 5,
        wdPasteEnhancedMetafile = 9,
        wdPasteHTML = 10,
        wdPasteHyperlink = 7,
        wdPasteMetafilePicture = 3,
        wdPasteOLEObject = 0,
        wdPasteRTF = 1,
        wdPasteShape = 8,
        wdPasteText = 2,
}WdPasteDataType;
```

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.

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.

boolean SlideShowPlay([in] VARIANT_BOOL bLoopUntilStopped);

Plays the slide show file.

Example
The following java script shows how to play the slide show from slide 2.

```
function EDOffice1_DocumentOpened()
{
EDOffice1.SlideSetStartingSlide(2);
EDOffice1.SlideShowPlay(true);
}
<script language="javascript" for=" EDOffice1" event="DocumentOpened">
  EDOffice1_DocumentOpened();
</script>
```

boolean SlideShowExit();

Exits the slide show play window.

boolean SlideGotoFirst();

Goes to the first slide in the opened PowerPoint file.

boolean SlideGotoPrevious();

Goes to the previous slide in the opened PowerPoint file.

boolean SlideGotoNext();

Goes to the next slide in the opened PowerPoint file.

boolean SlideGotoLast();

Goes to the last slide in the opened PowerPoint file.

boolean SlideGotoPage([in] long nPage);

Goes to the specified slide in the opened PowerPoint file based on the index.

long SlideGetCount();

Returns the count of slides.

long SlideGetCurrentShowPosition();

Returns the current show position of slides.

boolean SlideSetStartingSlide([in] long Start);

Sets the starting slide.

boolean SlideSetEndingSlide([in] long End);

Sets the ending slide.

boolean SlideExportSlideToImage([in] long Index, [in] BSTR FilePath, [in] BSTR FilterName, [in, optional] VARIANT Width, [in, optional] VARIANT Height);

Exports the slide to image.
Index: The index of the slide.
FilePath: The name of the file to be exported and saved to disk. You can include a full path; if you don't, Microsoft PowerPoint creates a file in the current folder.
FilterName: The graphics format in which you want to export slides. The specified graphics format must have an export filter registered in the Windows registry. You can specify either the registered extension or the registered filter name. Microsoft Office PowerPoint will first search for a matching extension in the registry. If no extension that matches the specified string is found, PowerPoint will look for a filter name that matches.
Width: The width in pixels of an exported slide.
Height: The height in pixels of an exported slide.

Example

The following java script shows how to export slide into jpg file.

```
function ExportSlide_Example ()
{
edoffice.SlideExportSlideToImage(1, “d:\1.jpg”, “JPG”);
}
```

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 EDOffice_NotifyCtrlReady()
{
        document.all.EDOffice.LicenseName = "";
        document.all. EDOffice.LicenseCode = "";
        document.all.EDOffice.BorderStyle = 1;
        document.all.EDOffice.Toolbars = false;
}
<SCRIPT LANGUAGE=javascript FOR=EDOffice EVENT=NotifyCtrlReady>
<!--
        EDOffice_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 EDOffice_BeforeDocumentOpened()
{
        EDOffice.DisableFileCommand 1 , false    'wdUIDisalbeOfficeButton
        EDOffice.DisableFileCommand 2 , false    'wdUIDisalbeNew
```

```
        EDOffice.DisableFileCommand 4 , false          'wdUIDisalbeOpen
        EDOffice.DisableFileCommand 16 , true    'wdUIDisalbeSave
        EDOffice.DisableFileCommand 32 , true    'wdUIDisalbeSaveAs
}
<SCRIPT LANGUAGE=javascript FOR=EDOffice EVENT= BeforeDocumentOpened >
<!--
        EDOffice_ 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 EDOffice_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=EDOffice EVENT= DocumentOpened >
<!--
        EDOffice_ 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 EDOffice_IESecurityReminder()
{
EDOffice.SetValue "Domain", "www.edrawsoft.com"
'or you can customize the whole sentense by set the ProtectModeReminder value.
'EDOffice.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=EDOffice EVENT= IESecurityReminder>
<!--
        EDOffice_ IESecurityReminder ();
//-->
</SCRIPT>
```

[id(17), helpstring("Occurs when the sheet activates.")]

void SheetActivate();

[id(18), helpstring("Occurs when the sheet deactivate.")]

void ***SheetDeactivate****();*

[id(19), helpstring("Occurs when the sheet calculates.")]

void ***SheetCalculate****();*

[id(20), helpstring("Occurs when the active sheet changes.")]

void ***SheetChange****();*

[id(21), helpstring("Occurs when the new sheet is created.")]

void ***WorkbookNewSheet****();*

[id(22), helpstring("Occurs after a new slide is created.")]

void ***PresentationNewSlide****();*

[id(23), helpstring("Occurs when a slide show starts. Is called for each slide show that starts.")]

void ***SlideShowBegin****();*

[id(24), helpstring("Occurs when a slide show ends. Is called for each slide show that ends.")]

void ***SlideShowEnd****();*

```
[id(25), helpstring("Occurs after the next build starts.")]

void SlideShowNextBuild();
```

```
[id(26), helpstring("Occurs after showing the new slide.")]

void SlideShowNextSlide();
```

```
[id(27), helpstring("Occurs after a slide or slide selection
changes in any view except the Outline view.")]

void SlideSelectionChanged();
```

```
[id(28), helpstring("Occurs after a color scheme is
changed.")]

void ColorSchemeChanged();
```

```
[id(29), helpstring("Occurs after the slide show window is
clicked.")]

void SlideShowNextClick();
```

Property

[id(1)] boolean ShowToolbars;
Shows or hides the toolbars of MS Word and MS Excel. The property does not work for the other office programs.

[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 WdSaveFormat
{
wdFormatDocument = 0,
wdFormatTemplate = 1,
wdFormatText = 2,
wdFormatTextLineBreaks = 3,
wdFormatDOSText = 4,
wdFormatDOSTextLineBreaks = 5,
wdFormatRTF = 6,
wdFormatUnicodeText = 7,
wdFormatEncodedText = 7,
wdFormatHTML = 8,
wdFormatWebArchive = 9,
wdFormatFilteredHTML = 10,
wdFormatXML = 11,
wdFormatDocumentDefault 16,
wdFormatPDF = 17,
wdFormatXMLDocument = 12,
wdFormatXMLDocumentMacroEnabled = 13,
wdFormatXMLTemplate = 14,
wdFormatXMLTemplateMacroEnabled = 15,
wdFormatXPS = 18,
}WdSaveFormat;

typedef enum PpSaveAsFileType
{
ppSaveAsPresentation = 1,
ppSaveAsPowerPoint7 = 2,
ppSaveAsPowerPoint4 = 3,
ppSaveAsPowerPoint3 = 4,
ppSaveAsTemplate = 5,
ppSaveAsRTF = 6,
ppSaveAsShow = 7,
ppSaveAsAddIn = 8,
```

```
ppSaveAsPowerPoint4FarEast = 10,
ppSaveAsDefault = 11,
ppSaveAsHTML = 12,
ppSaveAsHTMLv3 = 13,
ppSaveAsHTMLDual = 14,
ppSaveAsMetaFile = 15,
ppSaveAsGIF = 16,
ppSaveAsJPG = 17,
ppSaveAsPNG = 18,
ppSaveAsBMP = 19,
ppSaveAsOpenXMLPresentation = 24,
ppSaveAsOpenXMLPresentationMacroEnabled = 25,
ppSaveAsOpenXMLShow = 28,
ppSaveAsOpenXMLShowMacroEnabled = 29,
ppSaveAsOpenXMLTemplate = 26,
ppSaveAsOpenXMLTemplateMacroEnabled = 27,
ppSaveAsOpenXMLAddin = 30,
ppSaveAsOpenXMLTheme = 31,
ppSaveAsPDF = 32,
ppSaveAsXPS = 33,
}PpSaveAsFileType;

typedef enum PjSaveFiletype
{
pjSaveMPP = 0,
pjSaveMPT = 11,
pjSaveTXT = 3,
pjSaveCSV = 4,
pjSaveXLS = 5,
}PjSaveFiletype;

typedef enum WdPrintOutRange
{
wdPrintAllDocument = 0,
wdPrintSelection = 1,
wdPrintCurrentPage = 2,
wdPrintFromTo = 3,
wdPrintRangeOfPages = 4
}WdPrintOutRange;

typedef enum WdProtectType
{
wdAllowOnlyRevisions = 0,
wdAllowOnlyComments = 1,
wdAllowOnlyFormFields = 2,
wdAllowOnlyReading = 3,
wdNoProtection = -1,
}WdProtectType;

typedef enum XlProtectType
{
XlProtectTypeNormal = 0x00000001,
XlProtectTypeWindow = 0x00000002,
XlProtectTypeStruct = 0x00000004,
XlProtectTypeDrawingObjects = 0x00000010,
XlProtectTypeContents = 0x00000020,
```

```
XlProtectTypeScenarios = 0x00000040,
XlProtectTypeUserInterfaceOnly = 0x00000080,
}XlProtectType;

typedef enum PpViewType
{
ppViewHandoutMaster = 4,
ppViewMasterThumbnails = 12,
ppViewNormal = 9,
ppViewNotesMaster = 5,
ppViewNotesPage = 3,
ppViewOutline = 6,
ppViewPrintPreview = 10,
ppViewSlide = 1,
ppViewSlideMaster = 2,
ppViewSlideSorter = 7,
ppViewThumbnails = 11,
ppViewTitleMaster =8,
}PpViewType;

typedef enum WdViewType
{
wdMasterView = 5,
wdNormalView = 1 ,
wdOutlineView = 2,
wdPrintPreview = 4 ,
wdPrintView = 3,
wdReadingView = 7,
wdWebView = 6,
}WdViewType;

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

typedef enum WdPageFit
{
wdPageFitTextNone = 0 ,
wdPageFitFullPage = 1,
wdPageFitBestFit = 2,
wdPageFitTextFit = 3 ,
}WdPageFit;

typedef enum WdRevision
{
wdRevisionAuthor = 0,
wdRevisionDate = 1,
wdRevisionType = 2,
wdRevisionText = 3,
}WdRevision;

typedef enum WdPasteDataType
{
wdPasteBitmap = 4,
```

```
wdPasteDeviceIndependentBitmap = 5,
wdPasteEnhancedMetafile = 9,
wdPasteHTML = 10,
wdPasteHyperlink = 7,
wdPasteMetafilePicture = 3,
wdPasteOLEObject = 0,
wdPasteRTF = 1,
wdPasteShape = 8,
wdPasteText = 2,
}WdPasteDataType;

typedef enum WdBreakType{
wdPageBreak = 7,
wdColumnBreak = 8,
wdSectionBreakNextPage = 2,
wdSectionBreakContinuous = 3,
wdSectionBreakEvenPage = 4,
wdSectionBreakOddPage = 5,
wdLineBreak = 6,
wdLineBreakClearLeft = 9,
wdLineBreakClearRight = 10,
wdTextWrappingBreak = 11,
}WdBreakType;

typedef enum WdGoToItem{
wdGoToStart = 101,
wdGoToEnd = 102,
wdGoToBookmark = -1 ,
wdGoToComment = 6 ,
wdGoToEndnote = 5 ,
wdGoToEquation = 10 ,
wdGoToField = 7 ,
wdGoToFootnote = 4 ,
wdGoToGrammaticalError=  14 ,
wdGoToGraphic = 8 ,
wdGoToHeading=  11 ,
wdGoToLine = 3 ,
wdGoToObject = 9 ,
wdGoToPage = 1 ,
wdGoToPercent = 12 ,
wdGoToProofreadingError = 15 ,
wdGoToSection = 0 ,
wdGoToSpellingError = 13 ,
wdGoToTable = 2 ,
}WdGoToItem;

typedef enum WdGoToDirection{
wdGoToAbsolute = 1,
wdGoToFirst = 1,
wdGoToLast = -1 ,
wdGoToNext = 2 ,
wdGoToPrevious = 3 ,
wdGoToRelative = 2 ,
}WdGoToDirection;

typedef enum WdInDocPos{
```

```
wdInDocumentPosCursor = 1,
wdInDocumentPosStart = 2,
wdInDocumentPosEnd = 3 ,
}WdInDocPos;

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 Office Viewer Component.

DLL and OCX

officeviewer.ocx
EDOfficeViewerX.dll

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

officeviewer.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 "Edraw Office Viewer Component 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 Form_Load event.
   Private Sub Form_Load()
   EDOffice.LicenseName = ""
   EDOffice.LicenseCode = ""
   End Sub
   Add a button and fire the button click event.
   Private Sub OpenDoc()
   EDOffice.Open "d:\1.docx", "Word.Application"
   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 Office Viewer Component 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 EDOffice 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_EDOffice.m_hWnd)) return;

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

See VCEDOfficeDemo c++ example

Upgrade a control in VC++

You can easily upgrade a VC++ project witch is using a EDOffice 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 Office Viewer Component and click OK to all next dialogs.
5. Rebuild the project.

Web Application Issue

Use the officeviewer.cab file

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

```
<object classid="clsid: 7677E74E-5831-4C9E-A2DD-9B1EF9DF2DB4"
id="EDOffice" width="100%" height="100%"
codebase="http://www.yoursite.com/download/officeviewer.cab#7,5,0,356"
>
</object>
```

Note: You should put the officeviewer.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 EDOffice.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 Office Viewer Component 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 or IE8

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 Office Viewer Component.

Firefox Support

Microsoft created Active X for the Internet Explorer browser to properly play various types of media. Although the tool was designed specifically for Internet Explorer, when using Firefox, you will sometimes come across a website instructing you to install Active X to view media properly. For that reason, Mozilla has created a plug-in that will allow Active X to be enabled in Firefox.
Instructions

1. Determine what version of Firefox you are using by clicking on "Help" from the Firefox menu bar. Click "About Firefox" to see the version number listed underneath the Firefox logo.
2. Visit the Firefox Mozilla website to locate the Esker Active X Plug-In (see Resources).

3. Read to see which version of the plug-in is compatible with the version of Firefox you are running. If there is more than one version of Active X for your Firefox version, always download the latest Active X version since it will reflect updates.

4. Ensure that the box is checked to install the "Experimental Add-On." Click "Add to Firefox."

5. Accept the license agreement by clicking on the "Accept and Install" button. Once the automatic installation completes, Active X will be activated automatically for Firefox. If for some reason it does not auto-activate, click on "Tools" and "Add-Ons" from the Firefox menu. Click the "Plug-Ins" tab and click "Enable."

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 officeviewer.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 EDOffice_NotifyCtrlReady()
{
document.EDOffice.LicenseName = "your license name";
document.EDOffice.LicenseCode = "your license code";
}
</script>
<SCRIPT LANGUAGE=javascript FOR=EDOffice EVENT=NotifyCtrlReady>
<!--
 EDOffice_NotifyCtrlReady();
//-->
</SCRIPT>
<object classid="clsid: 7677E74E-5831-4C9E-A2DD-9B1EF9DF2DB4" id="EDOffice"
width="100%" height="100%"
codebase="http://www.yoursite.com/download/officeviewer.cab#7,5,0,356">
</object>
```

5. At last you can call the EDOffice.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/officeviewer.php

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

Support Email: support@ocxt.com