scViewerX

Documentation

Version 9

Software Companions

http://www.softwarecompanions.com

IMPORTANT

Contact Software Companions:

sales@softwarecompanions.com

www.softwarecompanions.com

Trademarks

Acrobat	Registered trademark of Adobe Systems Inc.
AutoCAD	Registered trademark of Autodesk, Inc.
Microsoft Windows	Registered trademark of Microsoft Corp.
Microsoft Office	Registered trademark of Microsoft Corp.
Microsoft Excel	Registered trademark of Microsoft Corp.
Microsoft PowerPoint	Registered trademark of Microsoft Corp.
Microsoft Word	Registered trademark of Microsoft Corp.

All other brand and product names used in this document are trademarks of their respective companies.

Table of Contents

1 General Information

1.1 Welcome to ScViewerX

scViewerX is a powerful ActiveX control that allows you to view, print and convert PLT plotter files, Adobe PDF, Autodesk DWF, CGM, TIFF and several other formats, inside your own application, or on a WEB page. You can implement the scViewerX component in any development environment and framework that can use ActiveX controls, for example Visual Basic, Visual Studio, C#.NET and Delphi.

1.2 Measure and annotate your documents with scViewerX

scViewerX includes an extensive set of annotation tools which includes text, polylines, polygons, rectangles, circles, ellipses, stamps, barcodes, pictures and more. You will find several tools for measurement and counting, which may be useful for e.g.: take-off measurement applications.

1.3 Print your drawings with scViewerX

scViewerX provides full support for printing your documents. You have full control of scaling, printer selection, paper selection and much more. Print the whole document, a displayed portion, or a user selected area only. Print multiple pages on a single paper sheet (n-up printing) or print a large page on several paper sheets (poster or tile printing). You can add custom header and footer to the printed document. You may also add a watermark to all printed pages.

1.4 Convert your drawings with scViewerX

The control includes several methods for converting between different file formats. scViewerX can convert your PDF files back to an editable CAD format, for example DXF. You have full control of the conversion and can change resolution, number of colors, scale and other settings. Pen tables are available for PLT, DWF and CGM files, and may be used to override widths, styles and colors.

1.5 Build your application with ScViewerX

scViewerX contains more than 300 methods, properties and events that may be used by your application. You will find methods for converting, comparing drawings, deskew image files, printing and much more. The control includes methods for merging, splitting, encrypting and signing PDF files.
This document describes each method, property and event available to the programmer.
Included with the SDK you will find complete applications written in C#, VB.Net and C++/MFC with full source code available.

2 The scViewerX Interface

2.1 scViewerX Methods

2.1.1 *AddCompareDocument*

Load a file that will be compared to the currently loaded file.

Syntax	BOOL AddCompareDocument(BSTR FileName)	
Parameters	OutputFileName	Name of file to compare to the currently loaded file.
Returns	BOOL	TRUE if the compare file was successfully loaded.

2.1.2 *AddImagePlaceholder*

Load an image file and add to the currently viewed document. The image will be placed using a text position and extents in millimeters. The added image will become a part of the viewed document (not markup).

Syntax	LONG AddImagePlaceholder(BSTR ReplaceText, BSTR ImageFileName, double Width, double Height)	
Parameters	ReplaceText	Text in file to replace with given image. Text position will be used as image origin.
	ImageFileName	Filename of image to load. Supported file formats : PNG, JPEG, TIFF and BMP.
	Width	Width of if image in millimeters.
	Height	Height of if image in millimeters.
Returns	LONG	Non-zero if the image file was successfully added.

2.1.3 *AddImagePosition*

Load a file and add to the currently viewed file. The image will be placed using a rectangle given in millimeters. All coordinates are relative to top left corner. The added image will become a part of the viewed document (not markup).

Syntax	LONG AddImagePosition(BSTR ImageFileName, double Left, double Top, double Right, double Bottom)	
Parameters	ImageFileName	Filename of image to load. Supported file formats : PNG, JPEG, TIFF and BMP.
	Left	Left position of the added image.
	Top	Top position of the added image.
	Right	Right position of the added image.
	Bottom	Bottom position of the added image.
Returns	LONG	Non-zero if the image file was successfully added

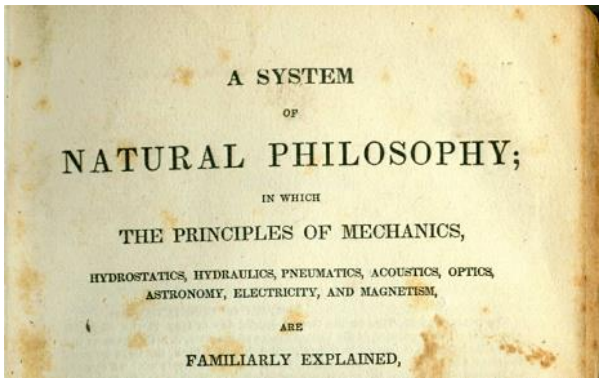
2.1.4 *AddQRImage*

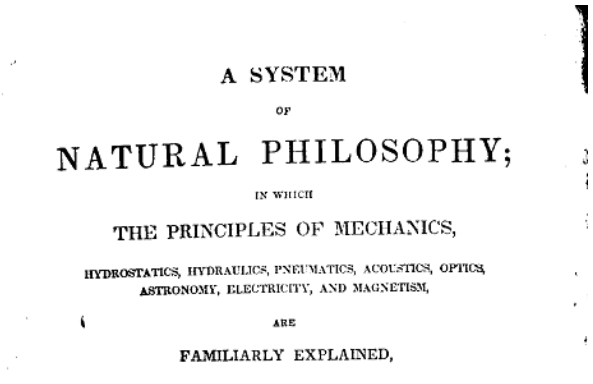
Add a QR barcode image to the currently viewed document. The image will be placed using a rectangle given in millimeters. All coordinates are relative to top left corner. The added image will become a part of the viewed document (not markup).

Syntax	LONG AddQRImage(BSTR Text, long Size, long Margin, long Level, double Left, double Top, double Right, double Bottom)	
Parameters	Text	Text to encode as a QR 2D barcode image.
	Size	Scale of QR image, normally set to 1.
	Margin	White space around QR Image in pixels.
	Level	Error correction level for QR encoding, the supported values are: 1. 7% 2. 15% 3. 25% 4. 30%
	Left	Left position of the added image.
	Top	Top position of the added image.
	Right	Right position of the added image.
	Bottom	Bottom position of the added image.
Returns	LONG	Non-zero if the image file was successfully added.

2.1.5 *BinarizeImage*

The binarize filter can convert a color image to black and white only. Like the defox filter described below, it can also be used to remove stain from old, scanned documents. The pictures below show an image before and after running binarize filter (Threshold value used is 0.5). The sample file used is included in the SDK. You may use the IsRasterFormat property to see if the currently loaded document can be processed using this filter. If you want to binarize scanned PDF files you must use the LoadPDFFile method to load the PDF file as a raster image.





Syntax	LONG BinarizeImage(long Page, double Threshold)	
Parameters	Page	Page number to binarize.
	Threshold	Threshold value to use for the binarization. Threshold values must be between 0 to 1.
Returns	LONG	Non-zero if the file was successfully binarized.

2.1.6 *CheckFile*

Check if the given file can be loaded by the control.

Syntax	BOOL CheckFile(BSTR FileName);	
Parameters	FileName	Name of file to check.
Returns	BOOL	TRUE if the file can be loaded.

2.1.7 *CheckFileFormat*

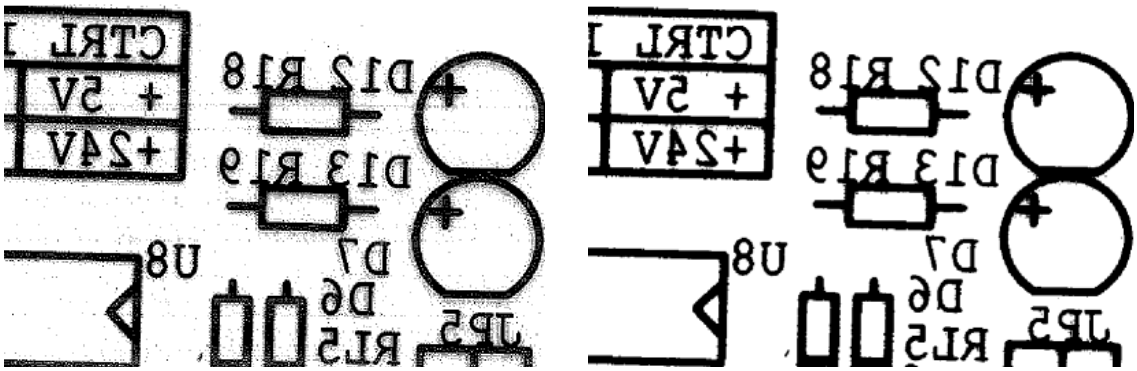
Check if the given file can be loaded and return the detected file format.

Syntax	LONG CheckFileFormat(BSTR pFileName)	
Parameters	FileName	Name of file to check.
Returns	LONG	One of the following values is returned: 0. Unknown file format – cannot be loaded 1. HPGL/2 2. Gerber RS-274D 3. Gerber RS-274X 4. Calcomp plotter format 5. CALS 6. TIFF 7. PNG 8. Windows BMP 9. JPEG 10. CGM (Computer Graphics Metafile) 11. Autodesk DWF or DWFx 12. Adobe PDF 13. JEDMICS C4 (tiled group 4) 14. WebP Image Format 15. Microsoft Word 16. Microsoft Excel 17. Microsoft Powerpoint 18. GIF 19. JPEG 2000 20. Text 21. LibreOffice Draw Formats 22. Intergraph Raster Format 23. Autodesk DXF 24. Excellon Drill Format 25. Gerber format loaded using external reader. 26. HEIC

Please note that Microsoft Office formats can only be loaded if you have Microsoft Office 2007 or newer installed on your system. You can also load Office formats if you have LibreOffice installed.

2.1.8 *CleanupImage*

Remove noise from an image page. This method is only supported for raster formats. You may use the IsRasterFormat property to see if the currently loaded document can be cleaned. If you want to cleanup scanned PDF files you must use the LoadPDFFile method to load the PDF file as a raster image.
Below are two pictures that show an image before and after CleanupImage has been applied:



Syntax	LONG CleanupImage(long Page)	
Parameters	Page	Page number to cleaunp. Page numbers start at index 0.
Returns	LONG	Non-zero if the page was successfully cleaned.

2.1.9 *CloseCompareDocument*

Close the added compare file and return to normal view mode.

Syntax	LONG CloseCompareDocument()	
Parameters	None	
Returns	LONG	Non-zero if the compare file was successfully closed.

2.1.10 *CloseFile*

Close the currently loaded file.

Syntax	LONG CloseFile()	
Parameters	None	
Returns	LONG	Non-zero if the file was successfully closed.

2.1.11 *CopyToClipBoard*

Copy a region clipboard defined by four coordinates to clipboard.
If all four coordinates are set to zero, the whole drawing will be copied to the clipboard.

Syntax	BOOL CopyToClipBoard(long x1, long y1, long x2, long y2)	
Parameters	x1	Left coordinate of part to copy.
	y1	Top coordinate of part to copy.
	x1	Right coordinate of part to copy.
	y2	Bottom coordinate of part to copy.
Returns	BOOL	TRUE if the area was successfully copied.

2.1.12 *CreateSearchablePDF*

Create a searchable PDF file from a raster file, for example a scanned PDF file. You can create a searchable PDF file from all supported raster formats which includes PDF, TIFF, PNG, JPEG and more. This method is only available if the Tesseract OCR engine is installed on the system. See appendix K for more information about tesseract. You can use the OCRAvailable property to check if Tesseract is installed on the system.

Syntax	LONG CreateSearchablePDF(BSTR OutputFileName)	
Parameters	OutputFileName	Name of destination PDF file with searchable text.
Returns	LONG	Non-zero if the process was successfully completed.

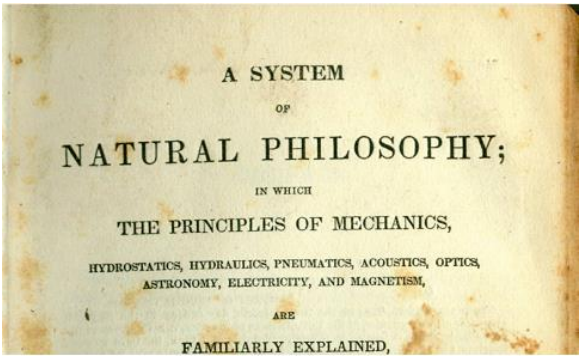
2.1.13 *Crop*

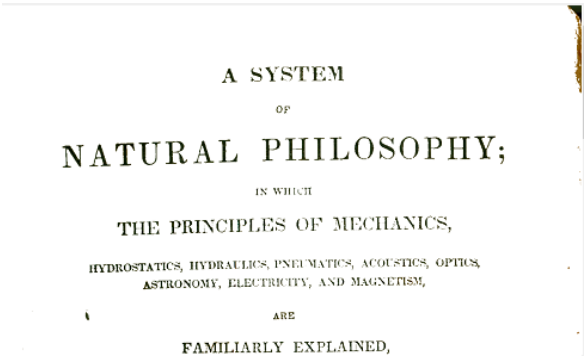
Crop a document page to fit the provided rectangle. Please note that this function doesn't work for PDF files, unless you load them as raster or vector, using the LoadPDFFile method. All coordinates are world coordinates.

Syntax	long Crop(long Page, DOUBLE x1, DOUBLE y1, DOUBLE x2, DOUBLE y2)	
Parameters	Page	Page number to crop. Pages start at index 0.
	x1	Left coordinate of the crop rectangle.
	y1	Top coordinate of the crop rectangle.
	x1	Right coordinate of the crop rectangle.
	y2	Bottom coordinate of the crop rectangle.
Returns	LONG	Non-zero if the file was successfully cropped.

2.1.14 *DefoxImage*

The defox image filter can be used to remove stain from old, scanned documents. A stained image is displayed to the left below and an image that has been defoxed to the right. For this example, a threshold value of 0.45 was used. The sample image file is included in the SDK. This method is only supported for raster formats, and you may use the IsRasterFormat property to check if the currently loaded document can be processed. If you want to remove stained from scanned PDF files you must use the LoadPDFFile method to load the PDF file as a raster image.





Syntax	LONG DefoxImage(long Page, double Threshold)	
Parameters	Page	Page number to remove stain from.
	Threshold	Threshold value to use for the destaining. Threshold

		values must be between 0 to 1.
Returns	LONG	Non-zero if the file was successfully defoxed.

2.1.15 *DeskewImage*

Deskew a skewed image by the given angle. Deskewing will straighten an image. scViewerX can calculate the recommended skew angle by using the GetDeskewAngle method. This method is only supported for raster image formats, check the IsRasterFormat propery to see if the current file can be deskewed. . If you want to deskew scanned PDF files you must use the LoadPDFFile method to load the PDF file as a raster image.

Syntax	LONG DeskewImage(long Page, double Angle)	
Parameters	Page	Page number to deskew. Page numbers start at index 0.
	Angle	Deskew angle in radians.
Returns	LONG	Non-zero if the file was successfully deskewed.

2.1.16 *DetectQR*

Search for QR codes on the given page. If QR codes are found the number of detected QR codes will be returned. Use the GetQRCode method to obtain the text for each QR code.

Syntax	BOOL DetectQR(LONG Page, LONG* NumberOfQR)	
Parameters	Page	Page number to search. Page numbers start at index 0. Set Page to -1 to search all pages in the open document.
	NumberOfQR	Number of QR codes found on the given page.
Returns	BOOL	True if the page was sucessfully searched.

2.1.17 *DrawHotspots*

Highlight all hotspots that have the given ID by drawing a rectangle. You can set draw color and line width in pixels for the drawn rectangle. Hotspots may be present in CGM files.

Syntax	LONG DrawHotspots(BSTR ID, OLE_COLOR DrawColor, LONG Width)	
Parameters	ID	All hotspots sharing this ID will be highlighted
	DrawColor	Draw color for rectangle outline (RGB)
	Width	Width of rectangle outline in pixels.
Returns	LONG	Non-zero if success.

2.1.18 *DrawToDC*

Draw the whole drawing, or a part of it, to a user provided device context.
The device context may be any type of device, for example metafile, bitmap, printer and dibsection. Offsets and clip rectangle are using screen coordinates.

Syntax	long DrawToDC(OLE_HANDLE DC, double Scale, long Page, double Rotation, long OffsetX, long OffsetY, long ClipX1, long ClipY1, long ClipX2, long ClipY2, OLE_COLOR BackColor, VARIANT_BOOL Mirror, VARIANT_BOOL Flip)

Parameters	DC	Device context handle.
	Scale	Scale factor .
	Page	Page number of document to draw (first page is 0).
	Rotation	Rotation factor in degrees.
	OffsetX	Horizontal offset to add for drawing.
	OffsetY	Vertical offset to add for drawing.
	ClipX1	left corner of clip rectangle.
	ClipY1	Lower corner of clip rectangle.
	ClipX2	Right corner of clip rectangle.
	ClipY2	Upper corner of clip rectangle.
	BackColor	Background color to use.
	Mirror	Set to true to mirror the drawing horizontally.
	Flip	Set to true to flip the drawing vertically.
Returns	LONG	Non-zero if success.

2.1.19 *DrawToDCEx*

Draw the whole drawing, or a part of it, to a user provided device context.
The device context may be any device (metafile, bitmap, printer, dibsection etc.).
This function may include markup if available.

Syntax	long DrawToDCEx(OLE_HANDLE hDC, long Page, long WorldX1, long WorldY1, long WorldX2, long WorldY2, long ClipX1, long ClipY1, long ClipX2, long ClipY2, OLE_COLOR BackColor, VARIANT_BOOL IncludeMarkup)	
Parameters	DC	Device context handle.
	Scale	Scale factor .
	Page	Page number of document to draw (first page is 0).
	WorldX1	Left coordinate of the drawing rectangle to draw.
	WorldY1	Top coordinate of the drawing rectangle to draw.
	WorldX2	Right coordinate of the drawing rectangle to draw..
	WorldY2	Bottom coordinate of the drawing rectangle to draw.
	ClipX1	left corner of clip rectangle.
	ClipY1	Lower corner of clip rectangle.
	ClipX2	Right corner of clip rectangle.
	ClipY2	Upper corner of clip rectangle.
	BackColor	Background color to use.
	IncludeMarkup	Set to true to include markup (if available).
Returns	LONG	Non-zero if success.

2.1.20 *DrillSetFormat*

Change the settings needed to load Excellon drill files correctly. Excellon files usually do not include any information about the internal format used. To be sure that such files are loaded correctly you may need to change the format settings before loading the file.

Syntax	long DrillSetFormat(VARIANT_BOOL Incremental, long Preceding, long Succeeding, long Units, VARIANT_BOOL Leading,

	VARIANT_BOOL Trailing)	
Parameters	Incremental	If true the drill files contains incremental coordinates. Set to false for absolute coordinates.
	Preceding	Number of digits before decimal point.
	Succeeding	Number of digits after decimal point.
	Units	Units used in drill file. Set to 0 for Inch and 1 for Milimetres.
	Leading	Leading zeros suppressed if set to true.
	Trailing	Trailing zeros suppressed if set to true.
Returns	LONG	Non-zero if success.

2.1.21 *EnumeratePen*

Return the real pen number for the given pen index.

Syntax	long EnumeratePen(long Index)	
Parameters	Index	Pen index (0..GetNumPens())
Returns	Long	Real pen number.

2.1.22 *Export*

Export the loaded document to another format.

Syntax	BOOL Export(BSTR FileName, BSTR Format, double Scale, long BitsPixel, long DPI)	
Parameters	Filename	Name of the exported file.
	Format	File format identifier to use for the exported file. See below of a list of supported file format identifiers.
	Scale	Scale factor to apply to the conversion.
	BitsPerPixel	Bits per Pixel – for raster formats output only. See note 1 below.
	DPI	Dots per Inch – for raster format output only. See note 1 below.
Returns	BOOL	TRUE if the file was successfully exported.

Available output formats:

Format ID	**Description**
BMP	Windows Bitmap Format
CALS	CALS Type 1 CCITT-G4 Raster Format
CGM	Computer Graphics Metafile (Binary)
DWF	Drawing Web Format
DXF	Autodesk Drawing Exchange Format. Version 2000.
DXFR12	Autodesk Drawing Exchange Format. Version 12
EMF	Windows Enhanced Metafile
GBR	Gerber RS-274X
HEIC	HEIC Image Format
HPRTL	HP-RTL
JPEG	JFIF Compliant JPEG Format
JPEG2K	JPEG 2000 Format

PCX	Paintbrush Format
PDF	Acrobat PDF
LEGACYPDF	Acrobat PDF writer from version 5.
PDFRASTER	Acrobat PDF raster only (Flattened)
PLT	HPGL/2 Plotter Format
PNG	Portable Network Graphics
PS	Adobe Postscript
SVG	Scalable Vector Graphics
TIFF	Tagged Image File Format
WEBP	Google WebP Image Format
WMF	Windows Metafile

Notes:

1. Raster formats includes BMP, CALS, HPRTL, JPEG, JPEG2K, PCX, PDFRASTER, PNG, TIFF, GIF, HEIC and WEBP.
2. LEGACYPDF may be faster for some files, but it does not support full transparency, so markup may appear different in the converted PDF compared to how it looks on the screen. If you for example just want to convert TIFF files with huge extents (width or height more than 2 meters), the legacy exporter may be faster and give better results.

2.1.23 *ExportExtents*

Basically, doing the same as the Export method, but have two additional parameters to force the extents of the exported file.
The two properties named ExportAlignmentH and ExportAlignmentV, can be used to control the placement of the drawing, if it's smaller than the exported page size.

Syntax	BOOL ExportExtents(BSTR FileName, BSTR Format, double Width, double Height, double Scale, long BitsPixel, long DPI)	
Parameters	Filename	Name of exported file.
	Format	File format to use for the exported file. Please see the Export method for a list of all supported formats.
	Width	Width of exported file in millimetres.
	Height	Height of exported file in millimetres.
	Scale	Scale factor to apply to the conversion. If you set the scale factor to 0.0, the exported document will be scaled to fit the given width and height.
	BitsPerPixel	Bits per Pixel – for raster formats output only.
	DPI	Dots per Inch – for raster format output only.
Returns	BOOL	TRUE if the file was successfully exported.

2.1.24 *ExportPage*

Export a single page of a multipage document to a new file of selected format.

Syntax	LONG ExportPage(BSTR FileName, BSTR Format, LONG Page, DOUBLE Width, Height, DOUBLE Scale, LONG BitsPerPixel, LONG DPI);	
Parameters	Filename	Name of exported file.
	Format	File format to use for the exported file.

		Please see the Export method above for a list of supported formats.
	Page	Page number to export. Page numbers start with index 0.
	Width	Width of exported file in millimetres. Set to 0 to use original page width.
	Height	Height of exported file in millimetres. Set to 0 to use original page height.
	Scale	Scale factor to apply to the conversion. If you set the scale factor to 0.0, the exported document will be scaled to fit the given width and height.
	BitsPerPixel	Bits per Pixel – for raster formats output only.
	DPI	Dots per Inch – for raster format output only.
Returns	LONG	Non-zero if the page was successfully exported.

2.1.25 *ForceRedraw*

Force the control to refresh drawing window. Use this function after toggling vector layer states or other operations that may cause current display to be invalid, for example the DeskewImage method.

Syntax	LONG ForceRedraw()	
Parameters	None	
Returns	LONG	Non-zero if success

2.1.26 *GerberDefineAperture*

Define shape and size for an aperture in the aperture table.
This method will only affect Gerber files using the older RS-274D format.

Syntax	long GerberDefineAperture(long ApertureNo, long Shape, long SizeX, long SizeY, long HoleSize)	
Parameters	ApertureNo	Aperture number to define (values from 10 to 9999 valid).
	Shape	Aperture shape, following values are available: 1. Round 2. Square 3. Oval 4. Polygon
	SizeX	Width of aperture in 1/10000 inch.
	SizeY	Height of aperture in 1/10000 inch.
	HoleSize	Size of drill hole in 1/10000 inch (optional)
Returns	LONG	Non-zero if successfully added.

2.1.27 *GerberLoadAperture*

Load aperture definitions from an existing file.
This will only affect Gerber files using the older RS-274D format

Syntax	long GerberLoadAperture(BSTR FileName)	
Parameters	FileName	Name of file with predefined apertures. Several different formats are supported, including:

		• Gerbview Apertures (*.gba, *.app) • VeriBest Aperture Map (*.map) • Pads Aperture Report (*.rep) • PCB386 GAP Files (*.gap) • Seetrax Apertures (*.txt) • and much more. If you need support for an aperture format that is currently not available, you can request support by sending an e-mail to sales@softwarecompanions.com.
Returns	LONG	Non-zero if successfully loaded.

2.1.28 *GerberSetColors*

Since Gerber files do not contain any color information at all, they are loaded only in black and white mode. You may use this method change the displayed track and flash colors.

Syntax	long GerberSetColors(OLE_COLOR FlashColor, OLE_COLOR TrackColor)	
Parameters	FlashColor	RGB color to use for flash elements, also known as pads.
	TrackColor	RGB color to use for tracks.
Returns	LONG	Non-zero if successfully loaded.

2.1.29 *GerberSetFormat*

Change the settings needed to load RS-274D Gerber files correctly. RS-274 formatted Gerber files do not include any information about the internal format used, To be sure that such files are loaded correctly you may need to change format settings before loading the file.

Syntax	long GerberSetFormat(VARIANT_BOOL Incremental, long Preceding, long Succeeding, long Units, VARIANT_BOOL Leading, VARIANT_BOOL Trailing)	
Parameters	Incremental	If true the Gerber files contains incremental coordinates. Set to false for absolute coordinates.
	Preceding	Number of digits before decimal point.
	Succeeding	Number of digits after decimal point.
	Units	Units used in Gerber file. Set to 0 for Inch and 1 for Milimetres.
	Leading	Leading zeros suppressed if set to true.
	Trailing	Trailing zeros suppressed if set to true.
Returns	LONG	Non-zero if success.

2.1.30 *GetBarcodeImage2D*

Encode a text as a 2D Barcode and return as image (DIB). The returned image can be added as markup by using the **MarkupCreatePictureFromDIB** method.

Syntax	long GetBarcodeImage2D(long Type, BSTR Text, OLE_HANDLE* DIB)	
Parameters	Type	Select which Barcode standard the method should use. Following options are available: 0. QR Code 1. Datamatrix 2. Aztec Code

		3. PDF417
	Text	Text to encode into Barcode.
	DIB	Returned DIB handle (Device Independent Bitmap).
Returns	LONG	Non-zero if the image was successfully created.

The text "http://www.softwarecompanions.com" encoded using each available standard:


QR


DataMatrix


Aztec


PDF417

Sample code to create a DIB using DataMatrix encoding, and add it as markup:

```
int dib = 0;
ax.GetBarcodeImage2D(1, "www.softwarecompanions.com", ref dib);
int handle = 0;
    ax.MarkupCreatePictureFromDIB(dib, 100.0, 100.0, 1160.0, 1160.0, "", 1, ref handle);
```

2.1.31 *GetBookmarkName*

Return information name of document bookmark. Document file formats like PDF and Word can contain bookmarks. Use the NumBookmarks property to return number of available bookmarks. The method named GotoBookmark will go to a document bookmark.

Syntax	long GetBookmarkName(long Index, BSTR* BookmarkName)	
Parameters	Index	Index of bookmark. For first bookmark use index 0.
	BookmarkName	Name of bookmark.
Returns	LONG	Non-zero if success.

2.1.32 *GetConfigValue*

Return current value for settings that affect markup and other operations.

Syntax	Long GetConfigValue(long ID)	
Parameter	ID	Unique ID for the setting to be returned. Please see the SetConfigValue method for a description of available ID's.
Returns	LONG	The current value of the selected setting.

2.1.33 *GetDeskewAngle*

Returns the calculated deskew angle. This method will analyze the raster data to find the deskew angle. Note that this method will only work for raster formats: TIFF, PNG, JPEG, BMP, WEBP, HEIC and CALS. This method will also work for PDF files if you load them using the LoadPDFFile method with raster format option.

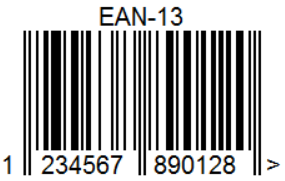
Syntax	LONG GetDeskewAngle(long Page, double *Angle);	
Parameters	Page	Page number to analyze, page numbers start with index 0.
	Angle	Calucalted angle in degrees.
Returns	LONG	Non-zero if successfully calculated.

2.1.34 *GetEAN13BarcodeImage*

Encode a text as an EAN-13 barcode and return as image (DIB). The returned image can for example be added as markup by using the **MarkupCreatePictureFromDIB** method.

Syntax	long GetEAN13BarcodeImage(BSTR BarcodeText, BSTR LabelText, VARIANT_BOOL IncludeText, double Scale, double Ratio, OLE_HANDLE* DIB)	
Parameters	BarcodeText	Text to encode as EAN-13 image.
	LabelText	Optional text to use as label on top of barcode image. Please note that this barcode format only accepts digits, and a maximum of 13 digits can be encoded. The last digit is used for checksum, so the actual number of digits you can encode is 12.
	IncludeText	If set to true the actual text will be displayed below barcode.
	Scale	Scale the size of the returned image.
	Ratio	The ratio between narrow and wide bar. Can be from 2.0 to 3.0. We do recommend to use 3.0 for ratio.
	DIB	Returned DIB handle (Device Independent Bitmap).
Returns	LONG	Non-zero if the image was successfully created.

The text " 1234567890128" encoded using the EAN-13 standard:



Sample code to create a DIB using EAN-13 encoding, and add it as markup:

```
int dib = 0;
ax. GetEAN13BarcodeImage("1234567890128", "EAN-13", true, 1.0, 3, ref dib);
int handle = 0;
ax.MarkupCreatePictureFromDIB(dib, 100.0, 100.0, 1160.0, 1160.0, "", 1, ref handle);
```

2.1.35 *GetFileExtents*

Return the width and height of the current page, in the active document.

Syntax	BOOL GetFileExtent(long *Width, long *Height);	
Parameters	Width	Width in world coordinates for current page.
	Height	Height in world coordinates for current page
Returns	BOOL	TRUE if success.

2.1.36 *GetFileFontInformation*

Return information about fonts used in files. You may use this method to check if a font was not found or not during loading. If the font is not found the display may not be correct.
Use the **FileFontsUsed** property to get the number of fonts in the file.

Syntax	long GetFileFontInformation(long Index, BSTR* FontName, VARIANT_BOOL *FontFound, BSTR *ReplaceFont)	
Parameters	Index	Index of font to query. Fonts are number from 0 to

		FileFontsUsed-1.
	FontName	Name of font as defined in the file.
	FontFound	If false the requested font was not found on the system and have been replaced.
	ReplaceFont	Name of font used to replace the requested font if it was not found.
Returns	LONG	Non-zero if success.

2.1.37 *GetLibraryName*

Return the name of the symbol library with the given index.

Syntax	long GetLibraryName(long LibraryIndex, BSTR* LibraryName);	
Parameters	LibraryIndex	The index for symbol library. Index values start at 0. The GetNumLibraries method will give you number of available symbol libraries.
	LibraryName	Name of the symbol library.
Returns	LONG	Returns non-zero if library index is valid.

2.1.38 *GetMarkupDrawState*

Return the active markup drawing state.

Syntax	long GetMarkupDrawState(long *Mode, long *ElementType)	
Parameters	Mode	Current markup mode, possible values: 0. Idle (no markup action active) 1. Add markup. The control is in add markup mode. ElementType will return the active markup element type. 2. Edit markup. The control is currently editing a markup element. ElementType will return the active markup element type.
	ElementType	Active markup element type. Please see the MarkupDrawElement method for a list of available element types, and the corresponding values.
Returns	LONG	Non-zero if success.

2.1.39 *GetMeasurePnt*

Return a point from the last measurement operation. Number of available measurement points is returned by the NumMeasurePnts property. Points are numbered from 0 to NumMeasurePnts-1. Returned coordinates are given in millimeters unaffected by calibration.

Syntax	long GetMeasurePnt(long Index, double *x, double *y)	
Parameters	Index	Index for point to return.
	x	Returned x coordinate in world coordinates.
	y	Returned y coordinate if world coordinates.
Returns	LONG	Non-zero if success.

2.1.40 *GetMouseAction*

Return the active mouse action.

Syntax	long GetMouseAction(long *Action)	
Parameters	Action	Return the active mouse action. A list of valid action modes follows: 0.No Action (clear active states) 2.Copy as metafile to clipboard mode. 7.Edit markup mode.
Returns	LONG	Non-zero if success.

2.1.41 *GetNamedView*

Return information about a named view. This function is currently only available for DWF and DWFx files. Use the NumNamedViews property to check if any named view is available. All returned coordinates are using world coordinate system.

Syntax	long GetNamedView(long View, BSTR* ViewName, double *x1, double *y1, double *x2, double *y2)	
Parameters	Index	Index of named view. For first view use index 0.
	ViewName	Name of view.
	x1	Left coordinate for view.
	y1	Lower coordinate for view.
	x2	Right coordinate for view.
	y2	Upprt coordinate for view.
Returns	LONG	Non-zero if success.

2.1.42 *GetNumLibraries*

Return the number of available symbol libraries.

Syntax	long GetNumLibraries(long *Libraries);	
Parameters	Libraries	Returns number of available symbol libraries.
Returns	LONG	Returns non-zero if symbols were successfully loaded.

2.1.43 *GetNumPages*

Return the total number of pages in the active document.

Syntax	long GetNumPages()	
Parameters	None	
Returns	LONG	Total number of pages in the active document.

2.1.44 *GetNumPens*

Return the total number of pen entries used in the active document page.
Use the EnumeratePen method to convert pen indexes to get real pen numbers.

Syntax	long GetNumPens()	
Parameters	None	
Returns	LONG	Total number of pens used in the active document page.

2.1.45 *GetNumSymbols*

Return the number of available symbols in a library.

Syntax	long GetNumSymbols(long LibraryIndex, long *Symbols);	
Parameters	LibraryIndex	Symbol library index.
	Symbols	Number of symbols in given library.
Returns	LONG	Returns non-zero if given library index is valid.

2.1.46 *GetPageName*

Return the name of a page. Page names are usually present in DWF and DWFX files. If the page name is not defined the method will return 0 (zero).

Syntax	long GetPageName(long PageIndex, BSTR *PageName)	
Parameters	PageIndex	Page index.
	PageName	Page name.
Returns	LONG	Returns non-zero if the given page have a name. Returns zero if no name is defined for the given page.

2.1.47 *GetPageSize*

Return the width and height of given page. The width and height are returned in millimeters.

Syntax	long GetPageSize(long Page, double *Width, double *Height);	
Parameters	Page	Page index.
	Width	Width for given page in millimeters.
	Height	Height for given page in millimeters.
Returns	LONG	Non-zero if success.

2.1.48 *GetPageThumbnail*

Create and returns a thumbnail image for the given page. The whole page will be scaled to fit the given thumbnail width and height.

Syntax	long GetPageThumbnail(long Page, long Width, long Height, long BitsPerPixel, long Flags, IPictureDisp** Thumbnail)	
Parameters	Page	Page number to create a thumbnail for. The first page is index 0.
	Width	Width of thumbnail image in pixels.
	Height	Height of thumbnail image in pixels.
	BitsPerPixel	The bits per pixel controls number of colors to use in the

		thumbnail image. Set this value to 1 for a monochrome (black & white) image, or 24 for a true color image.
	Flags	Following values are supported: 1. Add border Values can be combined.
	Thumbnail	The returned image (IDispatch).
Returns	LONG	Non-zero if success.

2.1.49 *GetPageThumbnail2*

Create and returns a thumbnail image for the given page. The whole page will be scaled to fit the given thumbnail width and height.

Syntax	long GetPageThumbnail2(long Page, long Width, long Height, long BitsPerPixel, long Flags, OLE_HANDLE* Thumbnail)	
Parameters	Page	Page number to create a thumbnail for. The first page is index 0.
	Width	Width of thumbnail image in pixels.
	Height	Height of thumbnail image in pixels.
	BitsPerPixel	The bits per pixel controls number of colors to use in the thumbnail image. Set this value to 1 for a monochrome (black & white) image, or 24 for a true color image. Set it to 0 to create a bitmap compatible with the display.
	Flags	Following values are supported: 1. Add border Values can be combined.
	Thumbnail	The returned image (HBITMAP). You will need to delete (free) the bitmap handle with DeleteObject when you've finished using it. Sample code: *DeleteObject((HBITMAP)Thumbnail);*
Returns	LONG	Non-zero if success.

2.1.50 *GetPaperSize*

Return the actual paper size if it is defined for the active document.
The width and height will be set to 0 if the size is not defined in the file.

Syntax	long GetPaperSize(double *Width, double *Height)	
Parameters	Width	Width of paper in world coordinates.
	Height	Height of paper in world coordinates.
Returns	LONG	Total number of pages in the active document.

2.1.51 *GetPDFSignatureData*

Return information about a certificate in a PDF file. A PDF file may contain multiple certificates and you can use the Index parameter to return information about a specific certificate. Please check the scViewerDemo sample source code to see how you may use the Windows Crypto API to extract information from returned certificate and signature data. This method is only available if the IsPDFSigned property is returns TRUE (VARIANT_TRUE).

Syntax	LONG GetPDFSignatureData(LONG Index, VARIANT_BOOL* Changed, VARIANT* Certificate, VARIANT* Signature, BSTR* Name, BSTR* Contact, BSTR* Location, BSTR* Time, BSTR* Reason, BSTR* Filter, BSTR* SubFilter)	
Parameters	Index	Index for certificate to return. First certificate is value 0, the number of available certificates can be found with the NumSignatures property
	Changed	Returns TRUE if the PDF file was modified after it was signed.
	Certificate	Returned safe array with x509 certificate data. This will only be valid if the SubFilter is set to adbe.x509.rsa_sha1.
	Signature	Returned safe array with signature data. This is either a DER encoded PKCS#1 binary data object or a DER-encoded PKCS#7 binary data object depending on the used SubFilter.
	Name	Name of the person signing the PDF file.
	Contact	Optional contact information.
	Location	Optional location information.
	Time	Date and time string (for time of signing).
	Reason	Optional reason information.
	Filter	Name of the security handler used.
	SubFilter	A value that describes the encoding of the signature value. This should be adbe.x509.rsa_sha1, adbe.pkcs7.detached, or adbe.pkcs7.sha1.
Returns	LONG	Non-zero if sginature data was successfully returned.

2.1.52 *GetPenState*

Return visibility state for the given pen.

Syntax	BOOL GetPenState(long Pen, long *State)	
Parameters	Pen	Pen number to query.
	State	Current pen status (0 is off and 1 is on).
Returns	BOOL	TRUE if success.

2.1.53 *GetPenTableEntry*

Return width, color and style for the given pen table entry.

Syntax	BOOL GetPenTableEntry(long Pen, double *Width, long *Color, long *Style)	
Parameters	Pen	Pen number to query.
	Width	Current pen width.
	Color	Current pen color.
	Style	Current pen line style.
Returns	BOOL	TRUE if success.

2.1.54 *GetPropertyDouble*

Return a previously stored custom double value for a specific markup element.

Syntax	long GetPropertyDouble (long ID, long Parameter, double *Value)

Parameters	ID	You may define up to 10 custom double values (ID from 0 to 9) per markup element. These custom values will be stored in the markup file and restored when the markup is loaded. Use the SetPropertyDouble method to set these values. Values from 20 and above is used for the following settings: 20. **Line Width**. Return line width for the element with the given handle. Value is line width in millimeters. 21. **Area Height**. Return the measurement area height. The SetConfigValue method controls which coordinate system that should be used. If you want to return a calibrated height you should call SetConfigValue(70,1) before using this method. 22. **Revision Cloud Arc Limit**. Return the arc limit for the given element in millimeters. This limit will be maximum size of radius used to create the revision cloud. 23. **Hatch Line Spacing**. Return the distance between each hatch line in millimeters for the given element. 24. **Hatch Line Width**. Return the line width to be used for hatch lines in millimeters for the given element.
	Parameter	The markup element handle
	Value	Custom double value, please ID for more information.
Returns	LONG	Non-zero if success.

2.1.55 *GetPropertyLong*

Return a property as a long value. This method is a replacement for properties with parameter, which is not supported by all programming languages.

Syntax	long GetPropertyLong(long ID, long Parameter, long *Value)	
Parameters	ID	Identifier of the property value to return. The following values are supported: 0. **Markup Layer State**. Parameter is the layer index. If the returned value is non-zero the layer is visible, else it is invisible. 1. **Markup Element State**. Parameter is the element handle. If the returned value is non-zero the element is visible, else it is invisible. 2. **Markup Element Page**. Parameter is the element handle. The returned value is the page number where the element is located. 3. **Markup Element Lock**. Parameter is the element handle. If the returned value is non-zero the element is locked, else it can be modified. 4. **Markup Layer Color**. Parameter is the layer index. The returned value is the current layer color. 5. **Markup Element Opacity**. Parameter is the element handle. The returned value is the element opacity in the range 0 to 255. 255 is full opacity (no transparency). 6. **Markup Element Color**. Parameter is the element handle. The returned value is the current outline/draw color. 7. **Markup Element Fill Color**. Parameter is the element handle. The returned value is the current fill color. 8. **Markup Element Negative Measure**. Parameter is the element handle. If the returned value is non-zero a markup measurement will be treated as negative. 9. **Markup Element Fill Type**. Parameter is the element handle. The returned value is the current fill type.

		10. **Markup Element Type** (read only). A list of all available element types can be found in the description of the MarkupDrawElement method. Please note that if the markup element was created automatically using mouse action 14, pick element, the returned markup element type will be 1023 instead of 23, to separate it from manually drawn measurement paths. 11. **Markup Element Z-Level**. Parameter is element handle. The value is the new z-level to use for this element. 12. **Markup Element Layer**. Parameter is the element handle. The returned value is the current element layer. 13. **Include Perimeter**. Parameter must be a handle to an area measurement element. If the returned value is non-zero perimeter value is included in the measurement text. 14. **Include Volume**. Parameter must be a handle to an area measurement element. If the returned value is non-zero volume value is included in the measurement text.
	Parameter	Depends on the ID, see ID above for more information.
	Value	Returned value, see ID above for more information.
Returns	LONG	Non-zero if success.

2.1.56 *GetPropertyString*

Return a property as a string value. This method is a replacement for properties with parameter, which is not supported by all programming languages.

Syntax	long GetPropertyString (long ID, long Parameter, BSTR *Value)	
Parameters	ID	Identifier of the property string value to return. The following values are supported: 0. **Markup Element Text**. Parameter is the elment handle. 1. **Markup Element Name**. Parameter is the element handle. 2. **Markup Element Group**. Parameter is the element handle.
	Parameter	Depends on the ID, see ID above for more information.
	Value	Returned value, see ID above for more information.
Returns	LONG	Non-zero if success.

2.1.57 *GetQRText*

Return the text for a QR code found using the DetectQR method.

Syntax	BOOL GetQRText(LONG Page, LONG QRIndex, BSTR* QRText)	
Parameters	Page	Page number. This must be the same page number that was used for DetectQR.
	QRIndex	Return encoded text for QR code with this index. QR codes are numbered from index 0.
	QRText	Returned encoded text.
Returns	BOOL	True if the QR text was returned successfully.

2.1.58 *GetRasterInfo*

Get detailed information about a raster image page.
This method works only for raster formats (for example TIFF, PNG, CALS and others).

Syntax	long GetRasterInfo(long PageNumber, long *PixelWidth, long *PixelHeight, long *DPIx, long *DPIy)	
Parameters	Page	Page number. First page is index 0.
	PixelWidth	Width of raster page in pixels.
	PixelHeight	Height of raster page in pixels.
	DPIx	Horizontal dots per inch used by the image.
	DPIy	Vertical dots per inch used by the image.
Returns	LONG	Non-zero if success.

2.1.59 *GetResolution*

Return the resolution of the active file in dots per inch (DPI).

Syntax	LONG GetResolution()	
Parameters	None	
Returns	LONG	Dots Per Inch. Zero means unknown.

2.1.60 *GetRotatedFileExtents*

Return the width and height of the current page if rotation is used.

Syntax	BOOL GetRotatedFileExtent(long *Width, long *Height);	
Parameters	Width	Rotated width in world coordinates for current page.
	Height	Rotated height in world coordinates for current page
Returns	BOOL	TRUE if success.

2.1.61 *GetSelectedAreaImage*

Return the image created by mouse action 15. You may use this method when the ActionCompleted event is called with action set to 15.

Syntax	long GetSelectecAreaImage(OLE_HANDLE *Image)	
Parameters	Thumbnail	The returned image is a HBITMAP. The caller owns the bitmap handle and will have to free (delete) it when it's no longer needed.
Returns	LONG	Non-zero if success.

2.1.62 *GetSymbolName*

Return the name of a symbol.

Syntax	long GetSymbolName(long LibraryIndex, long SymbolIndex, BSTR* SymbolName);

Parameters	LibraryIndex	Symbol library index.
	SymbolIndex	Symbol index.
	SymbolName	Returns name of the symbol with given library and symbol index.
Returns	LONG	Non-zero if success.

2.1.63 *GetSymbolPicture*

Returns a symbol picture.

Syntax	long GetSymbolPicture(long LibraryIndex, long SymbolIndex, long SymbolSize, OLE_HANDLE* SymbolImage);	
Parameters	LibraryIndex	Symbol library index.
	SymbolIndex	Symbol index.
	SymbolSize	The following values are supported: 0. Return 48x48 preview image. 1. Return an image in original size.
	SymbolImage	The returned symbol image (HBITMAP). You will need to delete (free) the bitmap handle with DeleteObject when you've finished using it. Sample code: *DeleteObject((HBITMAP)Thumbnail);*
Returns	LONG	Non-zero if success.

2.1.64 *GetTextEntry*

Return information about a text entry previously found using SearchTextAll. The current display will not be affected by this method. All coordinates are returned in world units.

Syntax	LONG GetTextEntry(LONG Index, BSTR *FoundText, LONG *Page, double *X1, double *X1, double *X2, double *X2, double *Rotation, LONG *Flags)	
Parameters	Index	Index of text entry to return. First text entry is 0.
	FoundText	The text that was found.
	Page	The page where the text was found.
	X1	Return left coordinate of the text.
	Y1	Return bottom coordinate of the found text.
	X2	Return right coordinate of the text.
	Y2	Return top coordinate of the found text.
	Rotation	Returned the text rotation.
	Flags	Returned text flag. If bit 0 is set the text is hidden.
Returns	LONG	Non-zero if success.

2.1.65 *GetVectorLayerInformation*

Return information about a vector layer.

Syntax	long GetVectorLayerInformation(long Layer, BSTR* LayerName, long *State, OLE_COLOR* Color);	
Parameters	Layer	Layer number, first layer is index 0.

	LayerName	Name of vector layer.
	State	Default visibility state for layer (0=off, 1=on).
	Color	Default layer color.
Returns	LONG	Non-zero if success.

2.1.66 *GetVectorLayerState*

Return the visibility status for a vector layer.

Syntax	long GetVectorLayerState(long Layer, long *State)	
Parameters	Layer	Layer number, first layer is index 0.
	State	Current visibility state for layer (0=off, 1=on).
Returns	LONG	Non-zero if success.

2.1.67 *GetVersion*

Returns the version and build number for the control.

Syntax	BOOL GetVersion(long *Major, long *Minor, long *Build)	
Parameters	Major	Major revision number.
	Minor	Minor revision number.
	Build	Build number.
Returns	BOOL	TRUE if success.

2.1.68 *GetViewRect*

Return world coordinates for the currently displayed area of the active file

Syntax	BOOL GetViewRect(long *x1, long *y1, long *x2, long *y2)	
Parameters	x1	Return left coordinate of the active displayed area.
	y1	Return upper coordinate of the active displayed area.
	x2	Return right coordinate of the active displayed area.
	y2	Return bottom coordinate of the active displayed area.
Returns	LONG	TRUE if success.

2.1.69 *GotoBookmark*

Go to a document bookmark.

Syntax	Long GotoBookmark(long Index)	
Parameters	Index	Bookmark index.
Returns	LONG	Non-zero if success.

2.1.70 *GotoText*

Zoom in and mark a text entry previously found using SearchTextAll.

Syntax	LONG GotoText(LONG Index)	
Parameters	Index	Index of text entry to show. First text entry is 0.
Returns	LONG	Non-zero if success.

2.1.71 *HideToolWindow*

Hide the tooltip window if it is visible. A tooltip window where you may show custom information can be created with the ShowToolWindow method.

Syntax	void HideToolWindow()	
Parameters	None	
Returns	VOID	

2.1.72 *HotspotSettings*

Set hotspot rectangle settings. Hotspots can be present in CGM files.

Syntax	LONG HotspotSettings(BOOL HotspotDraw, OLE_COLOR OutlineColor, LONG LineWidth, LONG ExtraHorzSpace, LONG ExtraVertSpace)	
Parameters	HotspotDraw	If set to true a rectangle with the given color and widths will be drawn when the mouse is above a hotspot.
	OutlineColor	The color to use for the rectangle.
	LineWidth	Width of rectangle outline in pixels.
	ExtraHorzSpace	Extra spacing to add in horizontal direction.
	ExtraVertSpace	Extra spacing to add in vertical direction.
Returns	LONG	Non-zero if success.

2.1.73 *LoadFile*

Load a file for viewing, printing or conversion.

Syntax	BOOL LoadFile(BSTR FileName)	
Parameters	FileName	Name of file to load.
Returns	BOOL	TRUE if the file was successfully loaded.

2.1.74 *LoadFromStream*

Load file from an IStream. Accepts all supported file types, except those formats that are pre converted to PDF, like for example DOCX. For DOCX and similar formats you must use the LoadFromStream2 method.

Syntax	BOOL LoadFromStream(VARIANT Stream)	
Parameters	Stream	VARIANT with IStream interface.

Returns	BOOL	TRUE if the file was successfully loaded.

2.1.75 *LoadFromStream2*

Load file from an IStream. Accepts all supported file types.

Syntax	BOOL LoadFromStream2(VARIANT Stream, BSTR Extension)	
Parameters	Stream	VARIANT with IStream interface.
	Extension	File extension for the stream content. This is required if the stream contains formats like for example RTF, DOCX and other formats that are normally preconverted to PDF.
Returns	BOOL	TRUE if the file was successfully loaded.

2.1.76 *LoadFromString*

Load HPGL/2 data by using a text string.
Since data is given as clear text, the method cannot accept RTL image data or other binary information. Use the LoadFromStream to load file data that contains binary information.

Syntax	BOOL LoadFromString(BSTR Data)	
Parameters	Data	String containing HPGL/2 instructions.
Returns	BOOL	TRUE if the data was successfully loaded.

2.1.77 *LoadPDFFile*

Load a PDF file as raster, vector or native PDF data.

Syntax	long LoadPDFFile(BSTR FileName, long LoadFormat, long BitsPerPixel, long DPI);	
Parameters	FileName	Name of PDF file to load.
	LoadFormat	Select of of the following data formats to use for loading: 0. Load as native PDF data (no conversion). 1. Load as raster data, PDF data will be converted to raster during load. DPI and BitsPerPixel is required for this format. If you load a PDF as raster you may use functions like CleanImage, DefoxImage, DeskewImage and more. 2. Load as vector data. This will extract all vector data from the PDF file. Note that the result may differ visually from native PDF viewing.
	BitsPerPixel	Select number of colors to use for raster format, supported values include: 1. Monochrome (black and white). 24.True color.
	DPI	Resolution of raster image given in dots per inch. Higher DPI values will improve quality, but requires more memory.
Returns	LONG	Non-zero if the file was successfully loaded.

2.1.78 *LoadPenTable*

Load a pen table from file.

Syntax	BOOL LoadPenTable(BSTR FileName)	
Parameters	FileName	Name of pentable file to load.
Returns	BOOL	TRUE if the file was successfully loaded.

2.1.79 *MarkupAddFromFile*

Add markup elements from an existing file to the active document.

Syntax	long MarkupAddFromFile(BSTR FileName)	
Parameters	FileName	Name of markup file to add.
Returns	LONG	Non-zero if the file was successfully loaded.

2.1.80 *MarkupAddLayer*

Add a new markup layer. Markup layers are an optional way to organize markup elements and work equal to layers in CAD files. You may define up to 256 user defined layers per file.

Syntax	long MarkupAddLayer(BSTR LayerName, OLE_COLOR LayerColor, long LayerState)	
Parameters	FileName	Name for new markup layer.
	LayerColor	Default color to use for this layer.
	LayerState	Active visibility state for this layer (0 = off, 1 = on)
Returns	LONG	Non-zero if the layer was successfully added.

2.1.81 *MarkupClearUndo*

Clear the markup undo buffer. After this method is called the end user will not be able to undo any changes.

Syntax	void MarkupClearUndo()	
Returns	Void	No return value.

2.1.82 *MarkupCreateFromXML*

Add one or more elements using XML markup data. You can find a complete description of the XML markup format in Appendix F.

Syntax	long MarkupCreateFromXML(BSTR XML)	
Parameters	XML	XML data containing markup data to add to active file.
Returns	LONG	Non-zero if the XML data was successfully loaded.

Below you will find sample XML data for adding a text element:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<SCMarkupFormat>
<Header>
   <Unit>mm</Unit>
</Header>
<Elements>
   <Element>
      <type>Text</type>
      <layer>0</layer>
      <page>0</page>
      <insertx>10</insertx>
      <inserty>30</inserty>
      <user>Peter</user>
      <color>#FF0000</color>
      <text>This is a text</text>
      <rotation>0</rotation>
      <font height='10' facename='Times New Roman'/>
   </Element>
</Elements>
</SCMarkupFormat>
```

2.1.83 *MarkupCreateFromXMLEx*

This function is equal to MarkupCreateFromXML but can only be used to add a single markup element. The handle for the newly created element will be returned.

Syntax	long MarkupCreateFromXMLEx(BSTR XML, long *Handle)	
Parameters	XML	XML data containing markup data to add to active file.
	Handle	The returned handle of the created element.
Returns	LONG	Non-zero if the XML data was successfully loaded.

2.1.84 *MarkupCreatePictureFromDIB*

Add a DIB (Device Independent Bitmap) as a markup picture element.

Syntax	long MarkupCreatePictureFromStream(OLE_HANDLE DIB, double x1, double y1, double x2, double y2, BSTR ID, long* Handle)	
Parameters	DIB	Device independent bitmap handle
	x1	Left position of added picture in world coordinates.
	y1	Bottom position of added picture in world coordinates.
	x2	Right position of added picture in world coordinates.
	y2	Top position of added picture in world coordinates.
	ID	Optional string ID, this ID is used for caching image data.
	Flags	Following flags are available: 1. Use this value if the DIB is a Barcode image.
	Handle	Returned handle of the created markup element.
Returns	LONG	Non-zero if the markup element was successfully created.

2.1.85 *MarkupCreatePictureFromFile*

Load a picture from a file and add as markup picture.
The file must be using BMP, TIFF, GIF, PNG or JPEG format.

Syntax	long MarkupCreatePictureFromFile(BSTR FileName, double x1, double y1, double x2, double y2, BSTR ID, long* Handle)	
Parameters	FileName	The name of the image file to add.
	x1	Left position of added picture in world coordinates.
	y1	Bottom position of added picture in world coordinates.
	x2	Right position of added picture in world coordinates.
	y2	Top position of added picture in world coordinates.
	ID	Optional string ID, this ID is used for caching image data.
	Handle	Returned handle of the created markup element.
Returns	LONG	Non-zero if the markup element was successfully created.

2.1.86 *MarkupCreatePictureFromStream*

Load a picture from a stream and add it as a markup picture element.
The stream can contain BMP, TIFF, GIF, PNG or JPEG data.

Syntax	long MarkupCreatePictureFromStream(VARIANT InputStream, double x1, double y1, double x2, double y2, BSTR ID, long* Handle)	
Parameters	InputStream	The stream that contains the image data.
	x1	Left position of added picture in world coordinates.
	y1	Bottom position of added picture in world coordinates.
	x2	Right position of added picture in world coordinates.
	y2	Top position of added picture in world coordinates.
	ID	Optional string ID, this ID is used for caching image data.
	Handle	Returned handle of the created markup element
Returns	LONG	Non-zero if the markup element was successfully created.

2.1.87 *MarkupCreateText*

Create a text element using default font and color settings.
This function is faster than using XML if you want to add multiple texts.

Syntax	long MarkupCreateText(double X, double Y, double Rotation, double Height, BSTR Text, long *Handle);	
Parameters	X	Left position of added text in world coordinates.
	Y	Bottom position of added text in world coordinates.
	Rotation	Text rotation in degrees.
	Height	Text height in world coordinates.
	Text	Text string to added.
	Handle	Returned handle of the created markup element
Returns	LONG	Non-zero if the markup element was successfully created.

2.1.88 *MarkupCounterSettings*

Set counter settings and start counting. When counting is ended by the user the number of counted items will be reported by the **CounterCompleted** event. When a count is started the user can end it by pressing the ESC and RETURN keys or use right-click with mouse.

Syntax	long MarkupCounterSettings(long ShapeType, double ShapeSize, BSTR CounterLabel);	
Parameters	ShapeType	Shape to use for this count, the following shapes are available: 0. A filled circle. 1. A filled square. 2. A filled triangle. 3. A filled diamond. 4. A plus symbol (+). 5. A filled 5-pointed star. 6. A Checkmark symbol. 7. Outlined circle. 8. A filled cross.
	ShapeSize	Width and height of the counter shape given in millimeters.
	CounterLabel	Optional parameter. This may be a label that describe what we're counting, e.g.: "Doors", "Windows", etc
Returns	LONG	Non-zero if the counting was started.

2.1.89 *MarkupDefaultFont*

Set default markup font properties like facename, height, weight and character set.

Syntax	long MarkupDefaultFont(BSTR FaceName, long Height, long Weight, long CharSet);	
Parameters	FaceName	Font face name (for example “Arial”).
	Height	Height of font in world coordinates (1016 dpi).
	Weight	Font weight, for example 400 for normal and 700 for bold.
	CharSet	Character set to use for the selected font.
Returns	LONG	Non-zero if the information was successfully set.

2.1.90 *MarkupDeleteElement*

Delete a markup element by handle.

Syntax	long MarkupDeleteElement(long Handle)	
Parameters	FileName	Unique element identifier for element to be deleted.
Returns	LONG	Non-zero if the element was successfully deleted.

2.1.91 *MarkupDeleteLayer*

Delete a markup layer by name.

Syntax	long MarkupDeleteLayer(BSTR LayerName)	
Parameters	LayerName	Name of layer to delete.

Returns	LONG	Non-zero if the layer was successfully deleted.

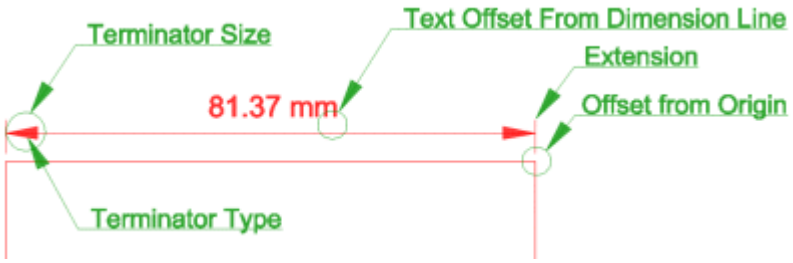
2.1.92 *MarkupDeleteLayerNumber*

Delete a markup layer by layer number.

Syntax	long MarkupDeleteLayerNumber(long LayerNumber)	
Parameters	LayerNumber	Index of layer to delete.
Returns	LONG	Non-zero if the layer was successfully deleted.

2.1.93 *MarkupDimLineSettings*

Set default settings and font to be used for new dimension line elements. All distances and offsets are given in millimeters.
Below is a picture that describes which part of the dimension line each setting will affect:



Syntax	long MarkupDimLineSettings(BSTR FaceName, double FontHeight, long FontWeight, double TermSize, long TermType, double TextOffset, double Extension, double Offset)	
Parameters	FaceName	Facename for font to use for dimension line text.
	FontHeight	Height of font in millimeters.
	FontWeight	Sets how thick or thin characters in the dimension text should be displayed. The weight of the font in the range 0 through 1000. For example, 400 is normal and 700 is bold. If this value is zero, a default weight is used. Below is a list of some predefined values:

Weight	**Value**
FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_HEAVY	900

	TermSize	Size of the terminators in millimeters.
	TermType	Terminator type, one of the following: 0. No terminator used.

		1. Open arrow. 2. Closed, but not filled, arrow. 3. Filled arrow. 4. Filled circle (dot). 5. Slash.
	TextOffset	Distance between the text and the dimension line..
	Extension	Length of the extension line beyond the dimension line.
	Offset	Distance of the offset, or gap, between the extension line and actual measured point (origin).
Returns	LONG	Non-zero if success.

2.1.94 *MarkupDimLineSettings2*

Set default settings and font to be used for new dimension line elements. All distances and offsets are given in millimeters. Please see MarkupDimLineSettings above for more information about the specific parameters.

Syntax	long MarkupDimLineSettings2(BSTR FaceName, double FontHeight, long FontWeight, double TermSize, long TermType, double TextOffset, double Extension, double Offset, double LineDistance)	
Parameters	FaceName	Facename for font to use for dimension line text.
	FontHeight	Height of font in millimeters.
	FontWeight	Sets how thick or thin characters in the dimension text should be displayed.
	TermSize	Size of the terminators in millimeters.
	TermType	Terminator type.
	TextOffset	Distance between the text and the dimension line..
	Extension	Length of the extension line beyond the dimension line.
	Offset	The offset, or gap, between the extension line and actual measured point (origin).
	LineDistance	Distance from the measured point to the actual dimension line.
Returns	LONG	Non-zero if success.

2.1.95 *MarkupDrawElement*

Start drawing of a new markup element.

Syntax	long MarkupDrawElement(long ElementType)	
Parameters	ElementType	Set active markup element draw type. After this method has been called the user can start drawing the selected element type by using the mouse. Following values are supported: 0. Line 1. Freehand 2. Rectangle 3. Ellipse 4. Polygon 5. Arrow 6. Text 7. Picture 8. Arrow with Text

		9. Revision Cloud 10. Note 11. Wave (sound) 12. Stamp (see note 2 below) 13. Eraser 14. Rectangular Eraser 15. Polygon Eraser 16. Dimension Line 17. Area Measurement 18. Polyline 19. Symbol (see note 1 below) 20. Circle 21. Counter (see note 3 below) 22. Measure Line (see note 4 below) 23. Measure Polyline (see note 5 below) 24. Highlighter 25. Rounded Rectangle 26. Shape (predefined shape types) Special draw modes: 1017. Draw a rectangular area measuremenent.. 2017. Draw a rectangular perimeter measurement. Notes: 1. Symbol element cannot be drawn. 2. Stamp element should not be drawn but instead added using XML. 3. Counter element is only included for reference, to draw counter elements you must use the MarkupCounterSettings method. 4. Measure Line is equal to a line element, but with attached measurement text. 5. Measure Polyline is the same as a Polyline element, but with attached measurement text. 6. Type 1017 and 2017 will be stored as an element of type 17 (area) when the drawing is completed.
Returns	LONG	Non-zero if success.

If you want to add markup elements programmatically, you should use the following methods:

2.1.96 *MarkupClearUndo*

Clear the markup undo buffer. After this method is called the end user will not be able to undo any changes.

Syntax	void MarkupClearUndo()	
Returns	Void	No return value.

1. ***MarkupCreateFromXML*** can be used to create all supported types of elements using a XML definition. With this method you may create multiple elements with one call.
2. **MarkupCreateFromXMLEx** can be used to create all supported types of elements using a XML defintion. This method can create only one element per call.
3. **MarkupCreatePictureFromStream** can create a picture element by loading the actual image data from a stream.
4. **MarkupCreatePictureFromFile** can create a picture element by loading the image data from a file.

5. **MarkupCreateText** can create text elements using the current font settings.
6. **MarkupCreatePictureFromDIB** can create a picture element from a device independent bitmap.

2.1.97 *MarkupElementGetHyperlink*

Return active hyperlink settings for the given markup element.

Syntax	long MarkupElementGetHyperlink(long Handle, BSTR* Location, long* LocationType, BSTR* Description)	
Parameters	Handle	Unique element identifier.
	Location	Hyperlink target (file or URL).
	LocationType	One of the following values: 0. No current hyperlink for this element. 1. Local file. 2. URL location.
	Description	Optional description of hyperlink displayed as tooltip.
Returns	LONG	Non-zero if information was returned.

2.1.98 *MarkupElementSetHyperlink*

Change the hyperlink setting for the given markup element. Every markup element can have a hyperlink attached to it. If the user presses an element with a hyperlink, the OpenHyperlink event will be called.

Syntax	long MarkupElementSetHyperlink(long Handle, BSTR Location, long LocationType, BSTR Description)	
Parameters	Handle	Unique element identifier.
	Location	Hyperlink target (file or URL).
	LocationType	One of the following values: 0. Clear active hyperlink (if any). 1. Local file. 2. URL location.
	Description	Optional description of hyperlink displayed as tooltip.
Returns	LONG	Non-zero if information was returned.

2.1.99 *MarkupFileInformation*

Return information about a markup file.

Syntax	long MarkupFileInformtion(BSTR FileName, long *Elements, long *Type)	
Parameters	FileName	Name of markup file to check. The file can be either binary or XML (auto detected).
	Elements	Returns the number of markup elements in the file.
	Type	Returns type of markup file (0 : binary, 1 : xml)
Returns	LONG	Non-zero if the markup file is valid.

2.1.100 *MarkupGetCounterInfo*

Return information about a markup counter element.
This method will only work for counter element types.

Syntax	long MarkupGetCounterInfo (long Handle, long *Count, long *ShapeType, BSTR *Label)	
Parameters	Handle	Unique element identifier for the counter element
	Count	Return the number of counted elements. If negative flag is set the the return number will be negative (See SetPropertyLong with ID 8).
	ShapeType	Return shape type used for this element.
	Label	Return the label used to descrive this counter element.
Returns	LONG	Non-zero if information was returned.

2.1.101 *MarkupGetDefaultFont*

Return the default markup font properties like facename, height, weight and character set.

Syntax	long MarkupGetDefaultFont(BSTR *FaceName, long *Height, long *Weight, long *CharSet);	
Parameters	FaceName	Returns font face name.
	Height	Returns height of font in world coordinates (1016 dpi).
	Weight	Retuns font weight, for example 400 for normal.
	CharSet	Returns character set used for the default font.
Returns	LONG	Non-zero if the information was successfully returned.

2.1.102 *MarkupGetDimensions*

Return dimension information for markup element with given handle.
This method will only work for measurement area and dimension line element types.
The returned values do depend on the active **calibration** and the currently selected **measurement unit**.

Syntax	long MarkupGetDimensions(long Handle, double *Area, double *Length)	
Parameters	Handle	Unique element identifier.
	Area	Returned area value.
	Length	Returned length value. For measurment area element this value is actually the perimeter
Returns	LONG	Non-zero if information was returned.

2.1.103 *MarkupGetDimensionsEx*

Return dimension information for the markup element with given handle. This method is equal to MarkupGetDimension but with an additional volume parameter. Please note that this method will only work for measurement area element type. The returned values do depend on the active **calibration** and the currently selected **measurement unit**.

Syntax	long MarkupGetDimensionsEx(long Handle, double *Area, double *Length,

	double *Volume)	
Parameters	Handle	Unique element identifier.
	Area	Returned area value.
	Length	Returned length value. For measurment area element this value is actually the perimeter
	Volume	Return the calculated volume. Please note that volume will only be returned if the area has a specified height. See SetPropertyDouble method for how to specify an area height.
Returns	LONG	Non-zero if information was successfully returned.

2.1.104 *MarkupGetElementAttributes*

Get markup element attributes like color, with, style and more.

Syntax	long MarkupGetElementAttributes(long Handle, OLE_COLOR *DrawColor, OLE_COLOR* FillColor, long *FillType, double *Width, long *Style, VARIANT_BOOL* Transparent)	
Parameters	Handle	Unique element identifier. You may acquire handles by handling the **NewElementCreated** event, or by enumerating markup elements using the **MarkupGetElementHandle** method.
	DrawColor	Element line or outline color.
	FillColor	Element fill color (only for fillable elements).
	FillType	Element fill style, see **MarkupFillType** for list of styles.
	Width	Element line, or outline, style, see **MarkupLineStyle** for list of supported styles.
	Transparent	Element transparency, if non-zero the element is transparent. If set to zero the element is opaque.
Returns	LONG	Non-zero if information was returned.

2.1.105 *MarkupGetElementAttributes2*

This method is equal to **MarkupGetElementAttributes**, but using LONG* instead of OLE_COLOR* as some environments do have problem with OLE_COLOR* parameters. Return all markup element attributes, like color, with, style and more.

Syntax	long MarkupGetElementAttributes(long Handle, long *DrawColor, long *FillColor, long *FillType, double *Width, long *Style, VARIANT_BOOL *Transparent)	
Parameters	Handle	Unique element identifier. You may acquire handles by handling the **NewElementCreated** event, or by enumerating markup elements using the **MarkupGetElementHandle** method.
	DrawColor	Element line or outline color.
	FillColor	Element fill color (only for fillable elements).
	FillType	Element fill style, see **MarkupFillType** for list of styles.
	Width	Element line width in millimeters.
	Style	Element line, or outline, style, see **MarkupLineStyle** for list of supported styles.
	Transparent	Element transparency, if non-zero the element is transparent. If set to zero the element is opaque.
Returns	LONG	Non-zero if information was returned.

2.1.106 *MarkupGetElementBg*

Return background color information for the given element.
Please note that this function is only used for markup image and text element.

Syntax	long MarkupGetElementBg(long Handle, VARIANT_BOOL *Enabled, OLE_COLOR* BackgroundColor)	
Parameters	Handle	Unique element identifier.
	Enabled	Non-zero if background color is currently used by this element.
	BackgroundColor	Background color.
Returns	LONG	Non-zero if information was returned.

2.1.107 *MarkupGetElementInfo*

Return information about a markup element.

Syntax	long MarkupGetElementInfo(long Handle, long *EntityType, long *Layer, long *Page, DATE *TimeModified, BSTR *ModifiedBy);	
Parameters	Handle	Unique element identifier.
	EntityType	Returned elment type. A list of all available element types can be found in the description of the MarkupDrawElement method.
	Layer	Returned element layer.
	Page	Returned element page.
	TimeModified	Date and time when this element was last modified.
	ModifiedBy	Name of the last user to modify this element.
Returns	LONG	Non-zero if information was returned.

Please note that if the markup element was created automatically using mouse action 14, pick element, the returned markup element type will be 1023 instead of 23, to separate it from manually drawn measurement paths.

2.1.108 *MarkupGetElementInfo2*

Return information about a markup element. This is an alternative to MarkupGetElementInfo which returns the date and time as separate parameters.

Syntax	Long MarkupGetElementInfo2(long Handle, long *EntityType, long *Layer, Long *Page, BSTR *ModifiedBy, long *Day, long *Month, long *Year, Long *Hour, long *Minute, long *Second)	
Parameters	Handle	Unique element identifier.
	EntityType	Returned elment type. A list of all available element types can be found in the description of the MarkupDrawElement method.
	Layer	Returned element layer.
	Page	Returned element page.
	ModifiedBy	Name of last user to modify this element.
	Day	
	Month	

	Year	
	Hour	
	Time	
	Second	
Returns	LONG	Non-zero if information was returned.

2.1.109 *MarkupGetElementXML*

Return XML data for a markup element.

Syntax	long MarkupGetElementXML(LONG Handle, LONG Units, BSTR* XML)	
Parameters	Handle	Unique element identifier.
	Units	Coordinate unit to use for returned XML data. Following values are supported: 0. Native (1016 dpi) 1. Millimeter 2. Inch
	XML	Returned XML data for given element.
Returns	LONG	Non-zero if XML data was returned.

2.1.110 *MarkupGetLayerInfo*

Return information about a markup layer.

Syntax	long MarkupGetLayerInfo(long Layer, BSTR *LayerName, OLE_COLOR *LayerColor, long *LayerState)	
Parameters	Layer	Index of layer to query, first layer index is 0.
	LayerName	Name of layer.
	LayerColor	Default color for this layer.
	LayerState	Current layer state (0=off, non-zero=on).
Returns	LONG	Non-zero if information was returned.

2.1.111 *MarkupGetLayerInfo2*

Return information about a markup layer. This method is equal to MarkupGetLayerInfo, but using LONG* instead of OLE_COLOR* as some environments do have problem with OLE_COLOR* parameters.

Syntax	long MarkupGetLayerInfo(long Layer, BSTR *LayerName, long *LayerColor, long *LayerState)	
Parameters	Layer	Index of layer to query, first layer index is 0.
	LayerName	Name of layer.
	LayerColor	Default color for this layer.
	LayerState	Current layer state (0=off, non-zero=on).
Returns	LONG	Non-zero if information was returned.

2.1.112 *MarkupGetNumPnts*

Return the number of points for given markup element. You can use MarkupGetPnt to obtain the coordinates for each point.

Syntax	LONG MarkupGetNumPnts(LONG Handle)	
Parameters	Handle	Unique element identifier for the counter element
Returns	LONG	Number of points in the element.

2.1.113 *MarkupGetPnt*

Obtain the coordinates for a point in the given markup element. You can use the SetConfigValue method to control what coordinate system should be used. If you want to return calibrated values you should call SetConfigValue(70,1) before using this method.

Syntax	LONG MarkupGetPnt(long Handle, long Index, double* x, double * y);	
Parameters	Handle	Unique element identifier for the counter element
	Index	Index of point to return. First point is zero.
	x	Returned x coordinate for given point.
	y	Returned y coordinate for given point.
Returns	LONG	Non-zero if information was returned.

2.1.114 *MarkupGetPosition*

Get the position for a markup element. Note that this function will only work for elements of following types: rectangle, cloud, circle, stamp, image, ellipse and text.
The coordinates used by this function is given in World coordinate system.

Syntax	long MarkupGetPosition(long Handle, double *x1, double *y1, double*x2, double *y2);	
Parameters	Handle	Unique element identifier.
	x1	Left position.
	y1	Bottom position.
	x2	Right position.
	y2	Top position.
Returns	LONG	Non-zero if information was returned.

2.1.115 *MarkupInsertSymbol*

Add a symbol from one of the available symbol libraries as a markup element.

Syntax	long MarkupInsertSymbol(long LibraryIndex, long SymbolIndex , double x1, double y1, double x2, double y2);	
Parameters	LibraryIndex	Library index of symbol library to use.
	SymbolIndex	Index of symbol to insert as markup.
	x1	Left position of added picture in world coordinates.
	y1	Bottom position of added picture in world coordinates.

	x2	Right position of added picture in world coordinates.
	y2	Top position of added picture in world coordinates.
	ID	Optional string ID, this ID is used for caching image data.
Returns	LONG	Non-zero if the markup element was successfully created.

2.1.116 *MarkupLayerProtection*

Set view and edit restrictions for given markup layer.

Syntax	long MarkupLayerProtection(long Layer, VARIANT_BOOL View, VARIANT_BOOL Edit, BSTR ViewPassword, BSTR EditPassword)	
Parameters	Layer	Index of layer to modify.
	View	If true this layer can always be viewed. If set to false a password for viewing must be provided.
	Edit	If true elements on this layer can always be edited. If set to false a password for editing must be provided.
	ViewPassword	The password a user must enter to be able to view any markup element placed on this layer.
	EditPassword	The password a user must enter to be able to edit any markup element placed on this layer.
Returns	LONG	Non-zero if success.

2.1.117 *MarkupLoad*

Load markup elements from given file. Please note that any current markup elements will be replaced. If you want to add elements from a file to the current ones, you should use MarkupAddFromFile instead.

Syntax	long MarkupLoad(BSTR FileName)	
Parameters	FileName	Name of markup file to load. The file can be either binary or XML (auto detected).
Returns	LONG	Non-zero if markup was loaded.

2.1.118 *MarkupMeasureDefaults*

Set default measurement unit to use for created dimension lines and areas.

Syntax	long MarkupMeasureDefaults(long UnitType, long Digits)	
Parameters	UnitType	Measurement unit to use for new dimension lines, measurement areas, measurement paths: 0. Millimetres 1. Centimetres 2. Decimetres 3. Meters 4. Kilometres 5. Inches 6. Feets 7. Yards 8. Miles
	Digits	Number of digits to use for measurement texts (default is 2).

Returns	LONG	Non-zero if success.

2.1.119 *MarkupMeasureFont*

Set default font to use for measurement functions.

Syntax	long MarkupMeasureFont(BSTR FaceName, long FontHeight, long FontWeight, long CharSet);	
Parameters	FaceName	FaceName of font to use for measurement element text.
	FontHeight	Height of font in millimeters.
	FontWeight	Sets how thick or thin characters in the dimension text should be displayed. The weight of the font in the range 0 through 1000. For example, 400 is normal and 700 is bold. If this value is zero, a default weight is used. Below is a list of some predefined values: **Weight** / **Value** FW_DONTCARE 0 FW_THIN 100 FW_EXTRALIGHT 200 FW_LIGHT 300 FW_NORMAL 400 FW_MEDIUM 500 FW_SEMIBOLD 600 FW_BOLD 700 FW_EXTRABOLD 800 FW_HEAVY 900
	CharSet	Character set to use for measurement text. Should be set to 0.
Returns	LONG	Non-zero if success.

2.1.120 *MarkupMenuDisableItem*

This method allows you to remove an item from the markup right click (context) menu.

Syntax	long MarkupMenuDisableItem(long Command);	
Parameters	Command	The menu item to remove. The following values are available: 32768 Remove the "Properties.." menu item. 32769 Remove the "Change Layer..." menu item. 32770 Remove the "Move to Front" menu item. 32771 Remove the "Send to Back" menu item. 32772 Remove the "Transparent" menu item. 32773 Remove the "Delete" menu item. 32774 Remove the "Copy" menu item.
Returns	LONG	Non-zero if success.

2.1.121 *MarkupMoveElement*

Bring an element to front or send it to back.

Syntax	long MarkupMoveElement(long Handle, long Action)

Parameters	Handle	Unique element identifier
	Action	Format to use for saved file. Supported values for format: 0. Bring element to Front 1. Send element to Back
Returns	LONG	Non-zero if success.

2.1.122 *MarkupPaste*

Paste a markup element from clipboard. Use MarkupCanPaste property to check if the clipboard contains a markup element.

Syntaxl	long MarkupPaste()	
Parameters	None	
Returns	BOOL	Non-zero if an element was successfully pasted.

2.1.123 *MarkupSave*

Save markup elements to given file name using the selected format.

Syntax	long MarkupSave(BSTR FileName, long FormatType)	
Parameters	FileName	Name of file to save markup to.
	FormatType	Format to use for saved file. Supported values for format: 0. Binary Format 1. XML Format
Returns	LONG	Non-zero if markup was saved.

2.1.124 *MarkupSetElementAttributes*

Set markup element attributes like color, with, style and more.

Syntax	long MarkupGetElementAttributes(long Handle, OLE_COLOR DrawColor, OLE_COLOR FillColor, long FillType, double Width, long Style, VARIANT_BOOL Transparent)	
Parameters	Handle	Unique element identifier.
	DrawColor	Element line or outline color.
	FillColor	Element fill color (only for fillable elements).
	FillType	Element fill style, see **MarkupFillType** for list of styles.
	Width	Element line width in millimeters.
	Style	Element line, or outline, style, see **MarkupLineStyle** for list of supported styles.
	Transparent	Element transparency, if non-zero the element is transparent. If set to zero the element is opaque.
Returns	LONG	Non-zero if information was modified.

2.1.125 *MarkupSetElementBg*

Set background color information for the given element.
Please note that this function is only used for markup image and text element.

Syntax	long MarkupSetElementBg(long Handle, VARIANT_BOOL Enabled, OLE_COLOR BackgroundColor)	
Parameters	Handle	Unique element identifier.
	Enabled	If non-zero, background color will be used for this element.
	BackgroundColor	Background color to use.
Returns	LONG	Non-zero if information was modified.

2.1.126 *MarkupSetElementInfo*

Set new page and/or layer for given markup element.

Syntax	long MarkupSetElementInfo(long Handle, long Page, long Layer)	
Parameters	Handle	Unique element identifier.
	Page	New page index for this markup element. You may set page index to -1 for displaying the markup element on all pages in a document.
	Layer	New markup layer to use for this markup element.
Returns	LONG	Non-zero if information was modified.

2.1.127 *MarkupSetLayerInfo*

Modify an existing markup layer.

Syntax	long MarkupSetLayerInfo(long Layer, BSTR LayerName, OLE_COLOR LayerColor, long LayerState)	
Parameters	Handle	Unique element identifier.
	LayerName	New name for layer.
	LayerColor	New default color to use for this layer.
	LayerState	New layer state. Non-zero values will make the layer visible, and zero will make the layer invisible.
Returns	LONG	Non-zero if information was modified.

2.1.128 *MarkupSetLayerInfo2*

Modify an existing markup layer. Same as MarkupSetLayerInfo but using long for color.

Syntax	long MarkupSetLayerInfo(long Layer, BSTR LayerName, long LayerColor, long LayerState)	
Parameters	Handle	Unique element identifier.
	LayerName	New name for layer.
	LayerColor	New default color to use for this layer.
	LayerState	New layer state.
Returns	LONG	Non-zero if information was modified.

2.1.129 *MarkupSetPnt*

Change the coordinates for a point in the given markup element. You can use the SetConfigValue method to control what coordinate system should be used. If you want to use calibrated values you should call SetConfigValue(70,1) before using this method.

Syntax	LONG MarkupSetPnt(long Handle, long Index, double x, double y);	
Parameters	Handle	Unique element identifier for the counter element
	Index	Index of point to return. First point is zero.
	x	New x coordinate for given point.
	y	New y coordinate for given point.
Returns	LONG	Non-zero if success.

2.1.130 *MarkupSetPosition*

Set a new position for a markup element. Please note that this function will only work for elements of following types: rectangle, cloud, circle, stamp, pciture, ellipse and text.

Syntax	long MarkupSetPosition(long Handle, double x1, double y1, double x2, double y2)	
Parameters	Handle	Unique element identifier.
	x1	Left world coordinate position.
	y1	Bottom world coordinate position.
	x2	Right world coordinate position.
	y2	Top world coordinate position.
Returns	LONG	Non-zero if information was returned.

2.1.131 *MarkupShapeSettings*

Set shape settings used by MarkupDrawElement type 26.

Syntax	long MarkupShapeSettings(long ShapeType, double ShapeSize, OLECOLOR ShapeColor);	
Parameters	ShapeType	Shape to use for this count, the following shape types are available: 0. Checkmark (approved) 1. Reject
	ShapeSize	Width and height of the shape given in millimeters.
	ShapeColor	Color to use for the shape element.
Returns	LONG	Non-zero if the counting was started.

2.1.132 *MarkupShowLayerDialog*

Open the markup layer dialog where you can add and edit markup layers.

Syntax	long MarkupShowLayerDialog(long ParentWindow)	
Parameters	ParentWindow	Handle to parent window, set to 0 to use active window.
Returns	LONG	Non-zero if dialog was displayed.

2.1.133 *MarkupUndo*

Undo last markup change. You can undo all changes by setting AllChanges to non-zero.

Syntax	long MarkupUndo(long AllChanges)	
Parameters	AllChanges	If set to zero the last change will undone, can be called multiple times until all changes are undone. Set to non-zero to undo all changes in one call.
Returns	LONG	Non-zero if markup change(s) was undone.

2.1.134 *MergeFile*

Merge an external file with the currently viewed document page, either horizontally or vertically. It is possible to merge multiple files in for example a 3 x 3 tile pattern. To do this you will need to do the following: Load the first file and use MergeFile for the two remaining files for the first row with horizontal set to true, and then call MergeFile the first file in the next row with horizontal set to false. Follow up by calling MergeFile for the next 2 files on second row with horizontal set to true. Repeat the procedure for the third row.

Syntax	long MergeFile(BSTR FileName, VARIANT_BOOL Horizontal)	
Parameters	FileName	File to be merged with currently viewed file. Note that you can only merge **vector** with **vector** and **raster** with **raster**. You may use this method for PDF files, but doing so will force the PDF files to be loaded as raster format.
	Horizontal	If true the files will be merged horizontally (side by side). If false the new file will be placed above the active file.
Returns	LONG	Non-zero if merge was successful.

2.1.135 *MirrorImage*

Mirror an image horizontally or vertically. This method will only work for raster formats.

Syntax	LONG MirrorImage(long Page, long Axis)	
Parameters	Page	Page number to mirror. Page numbers start at index 0.
	Axis	Axis to mirror: 0 : Horizontally 1 : Vertically
Returns	BOOL	Non-zero if the page was successfully mirrored.

2.1.136 *PaperBinName*

Return the name of a paper bin by index. The number of available paper bins is returned by the **PaperBins** property.

Syntax	BSTR PaperBinName (long Index)	
Parameters	Index	Index of paper bin name that should be returned.
Returns	BSTR	Name of paper bin.

2.1.137 *PDFConform*

Conform, or convert, an existing PDF file into selected PDF/A standard.

Syntax	long PDFConform(BSTR InputFile, long ConformType, BSTR OutputFile, BSTR *Errors)	
Parameters	InputFile	The PDF file you want to conform into PDF/A. If you set this parameter to an empty file name (“”) the currently loaded file will be used as original.
	ConformType	Select the conformation standard to use: **Value** / **PDF comform standard** 0 Normalize 1 PDF/A-1b 2 PDF/A-2b 3 PDF/A-3b 4 PDF/A-2u 5 PDF/A-3u 6 PDF/A-4
	OutputFile	Conformed output file.
	Errors	Will contain error message(s) if the conformation fail.
Returns	LONG	Non-zero if conformation was successful.

2.1.138 *PDFEncrypt*

Encrypt a PDF file using password(s) and restriction settings.

Syntax	long PDFEncrypt(BSTR OriginalFile, BSTR EncryptedFile, BSTR OpenPassword, BSTR OwnerPassword, long Restrictions);	
Parameters	OriginalFile	The PDF file you want to create an encrypted copy of. If you set this parameter to an empty file name (“”) the currently loaded file will be used as source file.
	EncryptedFile	The encrypted PDF file. This will be an exact copy of the orignal file but encrypted.
	OpenPassword	Optional password required to open the file.
	OwnerPassword	Optional owner password.
	Restrictions	Set optional user restrictions for the encrypted PDF file. The following values are available: **Value** / **Restriction** 0 No restrictions. 4 Deny printing. 8 Deny modification of contents. 16 Deny copying of contents. 32 Deny commenting. 3900 Deny all. The flags can be combined, for example you may set restrictions to 20 (4+16) to deny both printing and copying.
Returns	LONG	Non-zero if encryption was successful.

2.1.139 *PDFExportToCAD*

Export the active PDF file to vector formats like DXF, DWF, PLT, CGM and more.

Syntax	long PDFExportToCAD(BSTR FileName, BSTR Format, long Page)

Parameters	FileName	Name of the exported vector file.
	Format	Export format to use for conversion. Supported formats: • CGM (Computer Graphics Metafile) • DWF (Drawing Web Format) • DXF (Autodesk Drawing Exchange Format) • GBR (Gerber RS274X) • PLT (HPGL/2) • SVG (Scalable Vector Format)
	Page	Document page to convert
Returns	LONG	Non-zero if success.

2.1.140 *PDFMergeAddFile*

Add a PDF file to the currently merged PDF file.
All pages from the given PDF file will be added to the merged output file.
PDFMergeInit must be called before any PDF file is added.

Syntax	long PDFMergeAddFile(BSTR PDFFileName)	
Parameters	PDFFileName	Full path name to the PDF file to add to the merged PDF.
Returns	LONG	Non-zero if file was added.

2.1.141 *PDFMergeAddFileEx*

Add a PDF file to the currently merged PDF file.
The PageInformation parameter controls which pages from the given PDF file that will be added to the merged output file. PDFMergeInit must be called before any PDF file is added.

Syntax	long PDFMergeAddFile(BSTR PDFFileName, BSTR PageInformation)	
Parameters	PDFFileName	Full path name to the PDF file to add to the merged PDF.
	PageInformation	Control which pages that should be imported from the given PDF file. Use ";" to separate pages, e.g.: setting PageInformation to "1,2,10,11" will import pages 1,2,10 and 11 and add them to the merged PDF file.
Returns	LONG	Non-zero if file was added.

2.1.142 *PDFMergeClose*

Close the currently merged PDF file and output all pages to given file name.

Syntax	long PDFMergeClose(BSTR PDFOutputName)	
Parameters	PDFOutputName	Full path name of the new PDF file to create.
Returns	LONG	Non-zero if success.

2.1.143 *PDFMergeInit*

Start a new empty PDF file prepared for merging.
Use PDFMergeAddFile or PDFMergeAddFileEx to add files, and finally call PDFMergeClose to complete the merge.

Syntax	long PDFMergeInit()	
Parameters	None	
Returns	LONG	Non-zero if success.

2.1.144 *PDFOptimize*

Optimize a PDF file for faster loading and rendering. The optimize process will rescale images and rebuild the structure of the input file. The function rebuilds the content streams of all pages, templates and, annotations. Useless operators as well as errors in content streams will be removed.
The resulting content streams are error free and usually smaller.
How much the optimization takes effect depends on the quality of the original content streams.

Syntax	long PDFOptimize(BSTR InputFile, BSTR OutputFile, long MonoDPI, long ColorDPI, long MonoCompression, long lColorCompression);	
Parameters	InputFile	The PDF file you want to optimize. If you set this parameter to an empty file name (“”) the currently loaded file will be used as original.
	OutputFile	Optimized output file.
	MonoDPI	Resolution to use for monchrome images in output file. Any image larger than the set resolution will be rescaled to fit this setting.
	ColorDPI	Resolution to use for color images in output file. Any image larger than the set resolution will be rescaled to fit this setting.
	MonoCompression	Set compression method to use for monochrome images: **Value** / **Compression method** 0 – Flate Compression (ZIP) 2 – CCITT Group 3 Fax 3 – CCITT Group 4 Fax 8 – JBIG2
	ColorCompression	Set compression method to use for color images: **Value** / **Compression method** 0 – Flate Compression (ZIP) 1 – JPEG 4 – LZW 7 – JPEG2000
Returns	LONG	Non-zero if Optimization was successful.

2.1.145 *PDFReplaceFont*

Replace a PDF font with a Windows Truetype font. Some PDF fonts are not available in Windows and must be replaced with a Truetype font. For example "Helvetica" can be replaced by "Arial".

Syntax	long PDFReplaceFont(BSTR PDFFont, BSTR WindowsFont)

Parameters	PDFFont	Name of PDF font to replace.
	WindowsFont	Windows Truetype font name to use as replacement.
Returns	LONG	Non-zero if success.

2.1.146 *PDFRotatePage*

Rotate a PDF page by the given angle. Supported rotation factors are 90, 180 and 270 degrees. This method is only supported for PDF files.

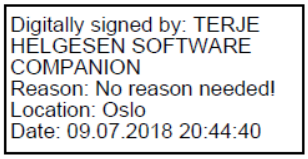
Syntax	LONG PDFRotatePage(long Page, long Angle)	
Parameters	Page	Page number to rotate. Page numbers start at index 0.
	Angle	Rotation angle in degrees (90, 180 and 270).
Returns	BOOL	TRUE if the page was successfully rotated.

2.1.147 *PDFSignImage*

Digitally sign a PDF file and display a signature image. To be able to sign a PDF file with scViewerX you will need a certificate stored as a PKCS#12 file (PFX).
Please note that the image position is relative to lower left corner of the page.

Syntax	LONG PDFSignImage(BSTR OriginalFile, BSTR SignedFile, BSTR CertFile, BSTR CertPassword, BSTR Reason, BSTR Location, LONG Page, BSTR ImageFile, DOUBLE x, DOUBLE y, DOUBLE w, DOUBLE h)	
Parameters	OriginalFile	The PDF file you want to create a signed copy of. If you set this parameter to an empty file name (“”) the currently loaded file will be used as original.
	SignedFile	The signed PDF file. This will be an exact copy of the orignal file but digitally signed.
	CertFile	Name and path for the certificate file (PFX).
	CertPassword	The password to use for the certificate.
	Reason	A string describing the reason for signing this file (optional).
	Location	A string describing the location (optional).
	Page	The page index where the signature image should be placed (pages start at index 0).
	ImageFile	Name of the image file to use. This can be a scanned signature or any other image. Supported formats include PNG, JPEG and TIFF. If no image file is provided the control will create an automatic image which contain information about the certificate.
	x	The x position in mm where the image should be located.
	y	The y position in mm where the image should be located.
	w	The width of the signature in mm. If you have an image file you may set this parameter to 0 to use the actual physical dimension of the image.
	h	The height of the signature in mm. If you have an image file you may set this parameter to 0 to use the actual physical dimension of the image.
Returns	LONG	Non-zero if signing was successful.

Here is an example on how an automatic created signature image may look:



2.1.148 *PDFSplit*

Split a multi-page PDF file into smaller files based on the parameters.

Syntax	long PDFSplit(BSTR InputFile, BSTR OutputFolder, BSTR FileLabel, long PagesPerFile)	
Parameters	InputFile	The multi-page PDF file that will be split into smaller files.
	OutputFolder	A valid folder name for the output files.
	FileLabel	Optional label to add to output filenames. By default the output files will be named "inputfilename_1.pdf", "inputfilename_2.pdf" and so on. By adding "part" as label, the output files will be named "inputfilename__part_1.pdf", "inputfilename__part_2.pdf" and so on.
	PagesPerFile	Number of pages per output file. If the input document contains 20 pages and you set page per file to 1, the control will create 20 files. If you set pages per file to e.g. 2, the control will output 10 files with 2 pages each.
Returns	LONG	Non-zero if success.

2.1.149 *Print*

Print the loaded file. You can control the result using the following properties: PrintScale, PrintFitToPaper and PrintCenterOnPage.

Syntax	BOOL Print()	
Parameters	None	
Returns	BOOL	TRUE if success.

2.1.150 *PrintDisplay*

Print the current display portion of the loaded file. You can control the result using the following properties: PrintScale, PrintFitToPaper and PrintCenterOnPage.

Syntax	BOOL PrintDisplay()	
Parameters	None	
Returns	BOOL	TRUE if success.

2.1.151 *PrintMarkup*

Print a single markup element, or all elements. This method will not include the loaded file, only markup data will be printed.

Syntax	BOOL PrintMarkup(long PrintType, long ElementHandle)

Parameters	PrintType	0. Print single element. If handle is set to 0 the currently selected element will be printed. 1. Print all elements.
	ElementHandle	Handle of element to print if single element is selected. Set to 0 for printing the currently selected elements.
Returns	BOOL	TRUE if success.

2.1.152 *PrintRegion*

After calling this method the user should select the region of the drawing that should be printed. You may choose to include, or exclude, both drawing and markup from the print job.

Syntax	long PrintRegion (VARIANT_BOOL Drawing, VARIANT_BOOL Markup)	
Parameters	Drawing	Include drawing in print job if non-zero.
	Markup	Include markup in print job if non-zero
Returns	LONG	Non-zero if success.

2.1.153 *PrinterName*

Return the name of the printer selected by given index. You may use this method together with the **InstalledPrinters** property to enumerate all installed printers on a system. The returned printer name can be used as parameter to the **SelectPrinter** method.

Syntax	BSTR PrinterName(long Index)	
Parameters	Index	Index of printer name that should be returned.
Returns	BSTR	Name of printer.

2.1.154 *PrintMultiPageOnSheet*

Print two or more pages on a single sheet of paper. This is also known as n-up printing (2-up, 4-up and so on). For example, to use 4-up printing you should set both horizontal and vertical pages parameter to 2 (2x2).

Syntax	long PrintMultiPageOnSheet(OLE_HANDLE DC, long Options, long HorzPages, long VertPages, long FromPage, long Pages)	
Parameters	DC	Printer device context. Set this to NULL to either use the default printer device, or let the user select a printer.
	Options	Choose from the available options, values may be combined. 1. Print page delimiters. Display grid lines separating each page on the sheet. 2. Select and use the default printer device. If this flag is not given, a printer selection dialog will be displayed. 4. Auto-toggle paper orientation. The control will select the paper orientation for maximum use of paper, based on the document extents. 8. Include markup. 16. Use PrintPageRange property to select pages to include in the print. If this flag is enabled, the FromPage and Pages parameters are not used.

	HorzPages	Number of horizontal pages per sheet.
	VertPages	Number of vertical pages per sheet.
	FromPage	Start page for printing (set to 0 for first page).
	Pages	Number of pages to print, use the value returned from GetNumPages property to print all pages
Returns	BOOL	TRUE if success.

2.1.155 *PrintPoster*

Print a file using multiple sheets of paper. This method is useful for printing a large format drawing on a normal printer, for example a drawing in A0 format on a A4 printer in original size (100% scaling). You can control the number of sheets by using the **PrintScale** property.

Syntax	BOOL PrintPoster()	
Parameters	None	
Returns	BOOL	TRUE if success.

2.1.156 *PrintToDC*

Print the loaded file using given printer device context (hdc). You can control the result using the following properties: PrintScale, PrintFitToPaper and PrintCenterOnPage.

Syntax	BOOL PrintToDC(OLE_HANDLE PrintDC, long PrintType)	
Parameters	PrintDC	Printer device context.
	PrintType	0. Normal print. 1. Print the displayed portion of the loaded file. 2. Print poster mode. See PrintPoster for more information.
Returns	BOOL	TRUE if success.

2.1.157 *RasterUndo*

Undo last raster operation (for example DeskewImage).

Syntax	long RasterUndo(long Page)	
Parameters	Page	Undo last raster operation for given page number.
Returns	LONG	Non-zero if success.

2.1.158 *RotateImage*

Rotate an image by the given angle. Supported rotation factors are 90, 180 and 270 degrees. This method is only supported for raster image formats. You must check the IsRasterFormat property to see if the current file can be rotated.

Syntax	LONG RotateImage(long Page, long Angle)	
Parameters	Page	Page number to rotate. Page numbers start at index 0.
	Angle	Rotation angle in degrees (90, 180 and 270).
Returns	BOOL	TRUE if the page was successfully rotated.

2.1.159 *SaveDisplay*

Save the currently displayed portion of the loaded file to a new image file. The SetMetaFileDPI method can be used to change the resolution of the saved image file.

<table>
<tr><td>Syntax</td><td colspan="2">LONG SaveDisplay(BSTR FileName, BSTR Format, BOOL IncludeMarkup)</td></tr>
<tr><td>Parameters</td><td>FileName</td><td>Filename to use for the created file.</td></tr>
<tr><td></td><td>Format</td><td>Format to use for the created file. Following formats are supported by this method:
<table>
<tr><th>Format ID</th><th>Description</th></tr>
<tr><td>BMP</td><td>Windows Bitmap Format</td></tr>
<tr><td>HPRTL</td><td>HP-RTL</td></tr>
<tr><td>JPEG</td><td>JFIF Compliant JPEG Format</td></tr>
<tr><td>PCX</td><td>Paintbrush Format</td></tr>
<tr><td>PDF</td><td>Acrobat PDF</td></tr>
<tr><td>PLT</td><td>HPGL/2 Plotter Format</td></tr>
<tr><td>PNG</td><td>Portable Network Graphics</td></tr>
<tr><td>TIFF</td><td>Tagged Image File Format</td></tr>
<tr><td>WEBP</td><td>Google WebP Image Format</td></tr>
</table></td></tr>
<tr><td></td><td>IncludeMarkup</td><td>Set to true (non-zero) to include markup if present. If set to zero the markup will not be included.</td></tr>
<tr><td>Returns</td><td>LONG</td><td>Returns non-zero if success.</td></tr>
</table>

2.1.160 *SaveThumbnail*

Create and save a thumbnail image for the selected page to the given output file. The whole page will be scaled to fit the given thumbnail width and height.

<table>
<tr><td>Syntax</td><td colspan="2">LONG SaveThumbnail(long PageNo, long Width, long Height, long BitsPerPixel, long Flags, BSTR Format, BSTR OutputFile)</td></tr>
<tr><td>Parameters</td><td>Page</td><td>Page number to create a thumbnail for. The first page is 0.</td></tr>
<tr><td></td><td>Width</td><td>Width of thumbnail image in pixels.</td></tr>
<tr><td></td><td>Height</td><td>Height of thumbnail image in pixels.</td></tr>
<tr><td></td><td>BitsPerPixel</td><td>The bits per pixel parameter controls number of colors to use in the thumbnail image. Set this value to 1 for a monochrome (black & white) image, or 24 for a true color image. Set it to 0 to create a bitmap compatible with the display.</td></tr>
<tr><td></td><td>Flags</td><td>Following values are supported:
1. Add border
Values can be combined.</td></tr>
<tr><td></td><td>Format</td><td>The file format to use for the thumbnail image file. The following file format identifiers are supported:
<table>
<tr><th>Format ID</th><th>Description</th></tr>
<tr><td>BMP</td><td>Windows Bitmap Format</td></tr>
<tr><td>JPEG</td><td>JFIF Compliant JPEG Format</td></tr>
<tr><td>PDF</td><td>Acrobat PDF</td></tr>
<tr><td>PNG</td><td>Portable Network Graphics</td></tr>
<tr><td>TIFF</td><td>Tagged Image File Format</td></tr>
</table></td></tr>
</table>

	OutputFile	Name of the created thumbnail image file.
Returns	LONG	Non-zero if success.

2.1.161 *ScanToFile*

Scan to file using an installed WIA scanner. This method can only be used if the HaveScanner property returns true.

Syntax	LONG ScanToFile(BSTR FileName, BSTR Format, LONG Flags)	
Parameters	FileName	Filename to use for the scanned raster document.
	Format	Format to use for the saved file, either “TIFF”or “PDF”.
	Flags	If set to 1 the scanned file will be loaded for viewing after the scan is complete.
Returns	LONG	Non-zero if success.

2.1.162 *ScreenToWorld*

Scale from screen coordinate to world coordinates. World coordinates are the loaded file's native coordinate system.

Syntax	BOOL ScreenToWorld(long ScreenX, long ScreenY, long *WorldX, long *WorldY)	
Parameters	ScreenX	Screen X coordinate.
	ScreenY	Screen Y coordinate.
	ScreenX	Transformed World X coordinate.
	ScreenX	Transformed World Y coordinate.
Returns	BOOL	TRUE if success.

2.1.163 *SaveCompareResult*

Save the result of the current comparison to a file.

Syntax	long SaveCompareResult(BSTR FileName, BSTR Format, double Scale, long ExportDPI)	
Parameters	FileName	Name of saved file
	Format	Format to use for created file. The following formats are available for this feature: • PDF • TIFF • PNG • JPEG • PNG
	Scale	Scale to use for the created file, set to 1.0 for original size.
	ExportDPI	DPI to use for the converted file, higher dpi will give higher quality but also larger files.
Returns	LONG	Non-zero if success.

2.1.164 *SearchText*

Search for text in the active document. If the text is found the current display will be changed to make the text visible (and marked).

Syntax	LONG SearchText(BSTR Text, LONG Page, VARIANT_BOOL MatchCase, VARIANT_BOOL MatchWord)	
Parameters	Text	The text you want to search for, it may contain wildcards.
	Page	The page you wish to search in.
	MatchCase	Set to true if the search is case sensitive.
	MatchWord	Set to true if you search for whole words.
Returns	LONG	Non-zero if success.

2.1.165 *SearchTextAll*

Search for all text in the active document that do match the input parameters. The function will return number of text entries found. Use the GetTextEntry method to obtain information about an individual text entry. Use the GotoText method to zoom in and mark a found text entry.

Syntax	LONG SearchTextAll(BSTR Text, LONG Page, VARIANT_BOOL MatchCase, VARIANT_BOOL MatchWord)	
Parameters	Text	The text you want to search for, it may contain wildcards.
	Page	The page you wish to search in.
	MatchCase	Set to true if the search is case sensitive.
	MatchWord	Set to true if you search for whole words.
Returns	LONG	Returns number of found text entries. Returns 0 if no text entries are found.

2.1.166 *SearchTextEx*

Search for text in the active document. If the text is found the location of the text and the actual text will be returned by the method. The current display will not be affected by this method. All coordinates are returned in world units.

Syntax	LONG SearchText(BSTR Text, LONG Page, VARIANT_BOOL MatchCase, VARIANT_BOOL MatchWord, LONG *X1, LONG *X1, LONG *X2, LONG *X2, BSTR *FoundText)	
Parameters	Text	The text you want to search for, it may contain wildcards.
	Page	The page you wish to search in.
	MatchCase	Set to true if the search is case sensitive.
	MatchWord	Set to true if you search for whole words.
	X1	Return left coordinate of the text.
	Y1	Return bottom coordinate of the found text.
	X2	Return right coordinate of the text.
	Y2	Return top coordinate of the found text.
	FoundText	The actual text that was found, this is useful if you use wildcards.

Returns	LONG	Non-zero if success.

2.1.167 *SelectPrinter*

Select printer to use by giving Driver, Printer and Port names.
Sample usage: SelectPrinter("", "HP DeskJet 895C Series Printer", "");
Note that this method will only work if **UseDefaultPrinter** property is set to False.

Syntax	BOOL SelectPrinter(BSTR DriverName, BSTR PrinterName, BSTR PortName)	
Parameters	DriverName	DriverName. Just pass an empty string for this one.
	PrinterName	Name of printer
	PortName	Name of printer port.
Returns	BOOL	TRUE if success.

2.1.168 *SetConfigStringValue*

This method can be used to change settings that affect markup and other operations.

Syntax	long SetConfigStringValue(long ID, BSTR Value)	
Parameters	ID	Unique ID for the setting to be changed. Please see table below of a description of the available settings.
	Value	Value to set, see specific ID for accpeted values.
Returns	BOOL	TRUE if success.

Available configuration ID values and their meaning:

ID	Description
0	**Note Element Image File:** Set path to a user defined bitmap file that will be used Markup Note elements, instead of the built in image. This setting only accepts Windows BMP files.
1	**Text File Font Name:** Set name of font to use for text file viewing. The default font name is “Courier New”.
2	**Word Converter:** Select Word to PDF converter to use. The default value is OfficeToPDF.exe. You can change it to scWordToPDF.exe if you want to use the old converter.
3	**Excel Converter:** Select Excel to PDF converter to use. The default value is scExcelToPDF.exe.
4	**Powerpoint Converter:** Select PowerPoint to PDF converter to use. The default value is scPPTToPDF.exe.
5	**LibreOffice Converter:** Select LibreOffice to PDF converter to use. The default value is scLibreToPDF.exe.
6	**Chinese Replacement Font.** Select a font to use for Chinese text if the requested font is not present on the system. Default font name is “SimSun.

2.1.169 *SetConfigValue*

This method can be used to change several settings that affect markup and other operations.

Syntax	long SetConfigValue(long ID, long Value)	
Parameter	ID	Unique ID for the setting to be changed, please check the table below for available values.
	Value	Value to set, please see the specific ID for accepted values.

Returns	LONG	Returns non-zero if the setting was changed.

Available configuration ID values and their meaning:

ID	Description
0	**UseGDIPlus**: Use GDI+ instead of GDI for markup drawing. GDI+ is by default enabled, set to 0 to disable it.
1	**CenterMeasureAreaText**: If set to 1 the measurement text will be placed in center of new measurement area elements.
2	**MeasureAreaUseLineWith**: Set this value to 1 to use current line width as outline for created measurement area elements. If you set this value to 0 only a one pixel outline will be drawn. The default value is 0.
3	**ShowMeasurePathText**: Set this value to 1 to display text for measurement line and polyline elements. If this value is set to 0, no measurement text will be displayed. The default value is 1.
4	**GdiPlusTransparency**: Set default transparency value to use for new elements. This value must be between 0 and 255, where 0 is full transparency and 255 is no transparency (opaque). The default value is 200.
5	**LineCapStyle**: Set line end cap style to use for new line, polyline and arrow markup elements. Supported values: 0 Round (default) 1 Flat 2 Square
6	**MeasurePathCapStyle**: Set end line cap style to use for markup measurement path and line elements. Supported values: 0 Round 1 Flat (default) 2 Square
7	**PDFVectorUseRotation**: If this value is set to non-zero (true) will PDF files that are loaded as vector, be loaded using the rotation defined in the PDF file. The default value is 0.
8	**MeasureAreaTextBackgroundColor**: Set color to use for measurement area text background. By default the current background color is used.
9	**MouseHoverDelay**: Set mouse the hovering delay in milliseconds. Set to 0 to use the system default value which is normally 400 milliseconds. The default value is 0.
10	**MeasurePathTextBackgroundColor**: Set color to use for measurement path and line text background. By default the current background color is used.
11	**UseLastSavedView**: If enabled the displayed view and rotation will be stored in the markup file. When the markup file is loaded the current view including rotation will be restored. The default value is 1.
12	**MarkupFileNameConvention**: Choose naming convention to use when loading markup files. The following options is available: 0 FILENAME.VCM - This will replace any existing file extension with vcm, e.g.: test.vcm 1 FILENAME.EXT.VCM - This will append the vcm extension to any existing extension, e.g.: test.pdf.vcm Please note that you will have to make sure that the correct file name convention is used when you save the markup with the MarkupSave method.
13	**HoverMarkupOnly**: If enabled the MarkupMouseHover event will only be called if the mouse is over an markup element. The default value is 1 (enabled).
14	**FillPolygonWhileDrawing**: If enabled, polygonal shapes will be filled while drawing. Set this option to 0 to only display the outline while drawing. The default value is 1.
15	**EnableUCCommand**: If enabled, UC commands will be handled. Set to 0 to disable the use of HPGL UC command. The default value is 1.
16	**LinePointEditHandleSize**: Set handle size for linepoint editing in pixels. The default value is 15.

17	**HidePlacementText**: Used only by the AddImagePlaceholder function. Set to 0 to keep text after the function is used. Default value is 1 (text will be hidden).
18	**MarkupImageCache**: If set to 1 the control will use a cache for markup pictures loaded from file (XML). Enabling the cache will save a lot of memory if you need to load multiple copies of the same image. Default value is 1.
19	**MarkupAutoRedraw**: Set to 0 to disable automatic redraw after a markup property has been changed. By setting this to 0 the application would need to call Invalidate() to redraw the markup. The default value is 1.
20	**UseFileBackColor**: Set to 1 to use the background color defined by the loaded file. Many DWF files have a predefined background color, and you should set this configuration value to 1 if you want the control to use this color. The default value is 0.
21	**LoadDWFMarkup**: Set to 1 to load Autodesk Design Review markup from DWF files. If set to 0 all DWF markups will be ignored. The default value is 1.
22	**RestoreMouseAction**: Set to 1 to restore previous mouse action after zoom window action is completed. The default value is 1.
23	**AutoUpdateMeasureText**: Set to 1 to automatically recalculate all measurement text if calibration or measurement unit is changed. The default value is 1.
24	**OutlinePolygons**: Set to 1 to force all polygons to be outlined with a line.
25	**KeepAreaTextCentered**: Set to1 to make sure the measuremnt area text is always centered when edited in line mode.
26	**MarkupEndEditModeOnFinish**: Set to 1 to end edit mode completely when the current element is completed.
27	**DisablePDFPrintAntiAliasing**: Set to 1 to disable anti-aliasing of PDF files during printing. Use this option if your printouts look faint.
28	**PDFHighResFloats**: Set number of digits to use in exported PDF files. Default value is 2.
29	**PDFCompareResolution**: PDF compare are done using image representations of the PDF data. You may improve the quality by increasing the DPI (the default is 200).
30	**DWFPlotInfoHack**: Some older plot files may be loaded incorrectly due to an error in early Autodesk products. Set this option to 0 if your files are loaded with extreme extents or other errors.
31	**Markup Double Buffering**: Use a double buffer image during editing of markup elements to reduce flickering. May cause a slight delay on markup selection, if you have many (100's) markup element for the active document. Set his setting to 0 if you want to disable this feature.
32	**ZoomOutShadowWidth**: The width of the shadow added when an image is displayed in zoomed out state. Set this value to 0 to remove this shadow completely.
33	**Screen Resolution:** The DPI value retuned by Windows API does sometimes not match the actual screen resolution. You may override this DPI value by using this method. Typical screen DPI retuned by Windows drivers is 96, but you could for example adjust it to 128, or any other value, if that's a better value for your monitor.
34	**Two Point Arrow Element:** Enable this setting to enable two arrow drawing mode. In this mode an arrow will be drawn as a single line, limited to two points. If disabled you may draw many pointed arrows. The default setting is 1 (enabled).
35	**Note Icon Bitmap Handle:** Set bitmap handle (HIBTMAP) to use when drawing note elements. The image can for example be loaded from your applications resources.
36	**Keep Page Size after Deskew:** A deskew will normally change the page size. Enable this setting to keep the page size after deskew. The default setting is 1 (enabled).
37	**DXF Ignore White Areas:** If set to non-zero, all filled white areas will be ignored when a file is converted to DXF. This may be useful if you're converting PDF files to DXF, because white areas may give unexpected results in AutoCAD as they normally are inverted to black on white background. Default value is 1.
38	**Allow Markup Resize:** If set to non-zero the user can resize markup elements and edit line points. If set to false the user may only move a markup element. The default value is 1.
39	**Enable PDF Geometry Extraction:** If set to non-zero the control will extract vector information from PDF files to allow snap to geometry for measurements and markup drawing. Depending on the file this may slow down the loading of the file. If you don't need this feature you may disable geometry extraction by setting this value to 0.
40	**Use Temporary File:** If set to non-zero a temporary file will be used while viewing the file.

	The default value is 0.
41	**Set Compression for Monochrome TIFF:** Set the compression method to use for monochrome TIFF files created by the control. Default value is 4 (CCITT-G4). See table below for more information.
42	**Set Compression for Color TIFF:** Set the compression method to use for color TIFF files created by the control. The default value is 5 (LZW).
43	**Set Compression for True Color TIFF:** Select one of the available compression methods to use for true color (24 bit) TIFF files created by the control. The default value is 5 (LZW).
44	**Set PDF Print Resolution:** Set the resolution (DPI) to use when printing PDF files. The default value is 300.
45	**Enable PDF Add Spaces:** If set to non-zero additional spaces will be added during text extraction.
46	**Enable Markup Element Z-Level:** If non-zero you can set individual z-level for each markup element. Enabling Z-level gives improved control of the draw order. The default value is 0 (disabled)
47	**Print Auto Rotate:** Enable this option to allow a page to be rotated if that fits the paper orientation better. This option is only used when printing multi-page documents, and if PrintFitToPaper is enabled too. Use a non-zero value to enable this option. This is a very useful option to use if a document (for example PDF) contains pages of different size and rotation.
48	**Text Font Height:** Set height of font to use for text file viewing. The default height is 10 points (10/72 inch).
49	**Set Compression for Monochrome Images in PDF:** Select one of the available compression methods to use for monochrome images added to PDF files.
50	**Set Compression for Color Images in PDF:** Select one of the available compression methods to use for color images added to PDF files.
51	**Set Compression for True Color Images in PDF:** Select one of the available compression methods to use for true color (24 bit) images added to PDF files.
52	**Set Page Size millimeter:** Set to non zero to force GetPageSize to return size in mm. Set to 0 to return page size in inches.
53	**DWF Exporter Resolution:** Set resolution to use for exported DWF files, the default value is 1000.
54	**PDF Extract Method:** Select one of two available text extraction method (0 or 1).
55	**Print Use Display Rotation:** If enabled (non-zero) the current display rotation will be used during printing too. Set to 0 to ignore display rotation during printing.
56	**Single Background Color:** Set to 1 to use the selected background color for the whole display area.
57	**Invert Mousewheel Zoom:** Set to non-zero to invert the mouse wheel zoom behavior.
58	**Show Hotspot Tooltips:** Set to non-zero to show a tooltip when the mouse is over a hotspot (CGM).
59	**Set DWF Exporter Unit:** Set to non-zero for metric output and zero if you want to use inch.
60	**Skip Content on Hidden Layers during Export:** Set to non-zero to skip content on hidden layers (layers that are turned off). Set to zero to include this content.
61	**Add Markup Layers to Exported File:** If set to non-zero all markup layers will be added to the exported file (PDF, DWF and DXF). If set to zero all markup data will be added as a single layer (merged).
62	**Zoom In and Out Percentage:** Control scale factor change when the user zooms in or out. The default value is 50 (50%). You may change it for example 25 to get a smaller change. Any value between 10 and 100 is valid.
63	**Save Picture Reference Only in XML Markup Files:** Set to true tif you want to save only the file reference in XML markup files. Please note that if you use this you have to make sure that the referenced picture files are included if you send the markup file to another user. If you set this to false the XML file will include binary image data (no need for referenced files). Default setting is false (picture binary data included in XML files).

<table>
<tr><td>64</td><td>Measurement Path Text Location: Set text position for measurement path and line elements. The following values are supported:
<table>
<tr><td>0</td><td>Text is placed on the line.</td></tr>
<tr><td>1</td><td>Text is placed below or left to the line.</td></tr>
<tr><td>2</td><td>Text is placed above or right to the line.</td></tr>
</table>
</td></tr>
<tr><td>65</td><td>JPEG2000 Compression Level: Set compression level to use for JPEG2000 files created by the control. Accepted values are from 0 to 100. 0 is lossless compression. The default value is 20.</td></tr>
<tr><td>66</td><td>Tool Window Transparency: Set transparency to use for the tool window displayed by the ShowToolWindow method. 0 is full transparency and 100 is no transparency.</td></tr>
<tr><td>67</td><td>Export PDF Layer States: If set to non-zero changed layer states will be written to the exported PDF files.</td></tr>
<tr><td>68</td><td>DXF Metric Flag: If set to non-zero exported DXF files will be created using metric coordinates, even if the system is set to use inch (US/Imperial). Set to zero to write DWF files using inch instead.</td></tr>
<tr><td>69</td><td>Load Hidden Text from DWF files: If set to non-zero all hidden text will be loaded from DWF files for text extract and search purpose. Set this flag to zero to skip hidden text, this may improve performance for some files.</td></tr>
<tr><td>70</td><td>Configure Measure Point Coordinate System: Set this value to 0 if you want MarkupGetnt to return world coordinates. Set this value to 1 if you want to return calibrated coordinate or 2 if you want to get coordinates in mm or inch (based on system settings).</td></tr>
<tr><td>71</td><td>Transparent Measurement Area Text Background: Set to non-zero to enable transparent text background for area measurements.</td></tr>
<tr><td>72</td><td>Create Bookmarks from DWF Named Views: If enabled each named in a DWF file will be added as bookmark during PDF conversion</td></tr>
<tr><td>73</td><td>Enable Raster Display: If enabled all raster data found in CAD files, for example DWF and PLT, will be displayed.</td></tr>
<tr><td>74</td><td>Map DWF Measurement Viewports in PDF: If enabled all measurement viewports found in DWF will be converted to a PDF measurement viewport.</td></tr>
<tr><td>75</td><td>Set Metric Measurement: If set to non-zero the control will use metric measurements even if the system regional setting is set to US/Imperial. Set this value to 0 to force use of US/Imperial measurements.</td></tr>
<tr><td>76</td><td>Set Edit Line Mode Cursor. Select cursor to use while editing line mode points. Following values are available:
<table>
<tr><td>0</td><td>Default cursor (size all)</td></tr>
<tr><td>1</td><td>Arrow</td></tr>
<tr><td>2</td><td>Cross</td></tr>
<tr><td>3</td><td>Arrow + small cross (same as default edit markup)</td></tr>
</table>
</td></tr>
<tr><td>77</td><td>Enable Drag and Drop. Set this option to non-zero to allow drag and drop of files. Set this value to 0 to disable drag and drop of files. Default value is 1 (enabled).</td></tr>
<tr><td>78</td><td>Enable Dimension Line Text Confirmation. If this option is enabled the control will ask the user to confirm or modify the text displayed for the dimension line after it has been drawn.</td></tr>
<tr><td>79</td><td>Screen Update Control. This flag controls how the screen will be periodically updated during redraw operations (progressive rendering), following settings are available:
<table>
<tr><td>0</td><td>Never update screen during a redraw.</td></tr>
<tr><td>1</td><td>Only update screen on zoom all redraws.</td></tr>
<tr><td>2</td><td>Always update screen.</td></tr>
</table>
</td></tr>
<tr><td>80</td><td>DXF Importer Default Unit. Set default unit to use for imported DXF files. The default value depends on Windows regional settings. Following values are available:
<table>
<tr><th>Value</th><th>Unit</th></tr>
<tr><td>0</td><td>Inch</td></tr>
<tr><td>1</td><td>Metric</td></tr>
</table>
</td></tr>
<tr><td>81</td><td>Enable Markup Element Copy using CTRL+Select. Enable or disable the possibility of</td></tr>
</table>

	copying a markup element by pressing CTRL key while selecting an existing element. Default value is 1 (enabled). Set this value to zero (0) to disable this feature.
82	**Revision Cloud Arc Limit.** Set the maximum size of radius for each arc used to create the revision cloud. This value is given in 1/10th millimeters. Default value is 0 which means no limit.
83	**Select Gerber Loader.** Select internal (legacy) or external Gerber loader (scrwGBX.dll). The external Gerber loader do support conversion together with markup, but the internal does not. Set this value to non-zero for internal, or to zero for external Gerber loader. The preferred setting is zero (external).
84	**Enable Compare Color Information Box.** Toggle the visibility of the color information box in compare mode.
85	**Set Compare Background Color.** Set the color to use for background file during compare.
86	**Set Compare Foregound Color.** Set the color to use for foreground file during compare.
87	**Set Color to use for Unchanged Content during File Compare.** Set the color to use for unchanged content, the default color is black. You can for example use this settings to use a lighter gray color to make the different changes easier to spot in larger drawings.
88	**Enable Printing of Transparent Images.** If your file contains transparent images you should enable this option. For example HPGL/2 files may contain transparent images.
89	**Enable Raster Image Processor for Printing.** Some printers doesn't support all the different raster operations that can be used by some file formats, for example HPGL/2. If you have such a printer you can enable this option to prepare a raster image for printing.
90	**Set Default Bits Per Pixel for HPGL/2 Device.** Set default bits per pixel to use for HPGL/2 files with raster images. Some HPGL/2 files doesn't include this setting and you have to set to the correct value before you load the file. Default value is 1 bits per pixel.
91	**Set unit for saved XML markup files.** Set unit to use for XML markup files saved using the SaveToFileXML method. Supported values: **Value** / **Unit** 0 – Native (1016 dpi) – default 1 – Millimeters 2 – Inch
92	**Set HEIC Compression Level:** Set compression level to use for HEIC files created by the control. Accepted values are from 0 to 100. 0 is lossless compression. The default value is 50.
93	**PDF TrueType Fonts.** Enable or disable the use of TrueType fonts for created PDF files, if this is possible (for example conversion from DWF with TrueType text to PDF).
94	**PDF Resolution for QR search.** Set the resolution to use when searching for QR codes in a PDF file. Default value is 200.
95	**Create Progressive JPEG files.** Enable or disable creation of progressive JPEG. Set to a non-zero value to enable progressive files. Default value is 0 (non-progressive).
96	**JPEG Compression Quality.** Set compression quality for created JPEG files. The given value can be between 1 and 100. Default value is 75.

TIFF compression methods supported by SetConfigValue are based on bits per pixel in the exported TIFF file:

Bits Per Pixel	ID	Value	Compression method
1	41	1 4 5 8 32773 32946	No compression CCITT Group 4 Fax LZW Deflate Compression (ZIP) Packbits Deflate (same as 8)
4 8	42	1 5 8 32773 32946	No compression LZW Deflate Compression (ZIP) Packbits Deflate (same as 8)
24	43	1	No compression

		5 7 8 32773 32946	LZW JPEG Deflate Compression (ZIP) Packbits Deflate (same as 8)

Sample usage:
SetConfigValue(41,1) – all exported 1-bit TIFF files will be compressed using CCITT G4.
SetConfigValue(42, 8) – all exported 4 and 8 bit TIFF files will be compressed using Deflate.
SetConfigValue(43, 7) – all exported 24 bit TIFF files will be compressed using JPEG.

PDF compression methods supported by SetConfigValue based on bits per pixel:

Bits Per Pixel	**ID**	**Value**	**Compression method**
1	49	0 2 3 8	Flate Compression (ZIP) CCITT Group 3 Fax CCITT Group 4 Fax JBIG2
4 8	50	0 4	Flate Compression (ZIP) LZW
24	51	0 4 1 7	Flate Compression (ZIP) LZW JPEG JPEG 2000

2.1.170 *SetFooterFont*

Set the font to be used for printing of footer information. A footer is a text that will be displayed at the bottom of the printed page.

Syntax	Long SetFooterFont(BSTR FaceName, long Height, long Weight, long CharSet);	
Parameters	FaceName	Font name.
	Height	Font height in points.
	Weight	The weight of the font in the range 0 through 1000. For example, 400 is normal and 700 is bold. If this value is zero, a default weight is used.
	CharSet	The character set to use. You will find more information about character sets in the Windows SDK documentation.
Returns	LONG	Non-zero if success.

2.1.171 *SetFooterText*

Set the text to be displayed in the footer area at the bottom of the printed page. . Calling this function with a valid text string will set the EnableFooter property to true.

Syntax	long SetFooterText (BSTR FooterText, OLE_COLOR TextColor, long Position, double BottomDistance);

<table>
<tr><td>Parameters</td><td>FooterText</td><td>The text to be displayed in the footer area. You may use predefined constants to include information like scaling, pages and more:
<table>
<tr><th>Type this</th><th>To print this</th></tr>
<tr><td>&d</td><td>Date in short format as specified by Regional Settings in Control Panel.</td></tr>
<tr><td>&f</td><td>File name.</td></tr>
<tr><td>&F</td><td>Full file path.</td></tr>
<tr><td>&n</td><td>Line break. You can have a maximum of three text lines.</td></tr>
<tr><td>&P</td><td>Total number of pages in document.</td></tr>
<tr><td>&p</td><td>Currently printed page number.</td></tr>
<tr><td>&s</td><td>Print scaling. For example 100%.</td></tr>
<tr><td>&t</td><td>Time in the format specified by Regional Settings in Control Panel.</td></tr>
<tr><td>&u</td><td>Name of the currently logged on user.</td></tr>
</table></td></tr>
<tr><td></td><td>TextColor</td><td>Color to use for the text (use 0 for black)</td></tr>
<tr><td></td><td>Position</td><td>Horizontal position for the footer text:
0. Left
1. Center
2. Right</td></tr>
<tr><td></td><td>BottomDistance</td><td>Distance from the paper edge to the bottom of the footer text.</td></tr>
<tr><td>Returns</td><td>LONG</td><td>Non-zero if success.</td></tr>
</table>

2.1.172 *SetHeaderFont*

Set the font to be used for printing of header information. A header is a text that will be displayed at the top of the printed page.

Syntax	Long SetHeaderFont(BSTR FaceName, long Height, long Weight, long CharSet);	
Parameters	FaceName	Font name.
	Height	Font height in points.
	Weight	The weight of the font in the range 0 through 1000. For example, 400 is normal and 700 is bold. If this value is zero, a default weight is used.
	CharSet	The character set to use. You will find more information about character sets in the Windows SDK documentation.
Returns	LONG	Non-zero if success.

2.1.173 *SetHeaderText*

Set the text to be displayed in the header area at the top of the printed page. Calling this function with a valid text string will set the EnableHeader property to true.

Syntax	long SetHeaderText (BSTR HeaderText, OLE_COLOR TextColor, long Position, double TopDistance);

<table>
<tr><td>Parameters</td><td>FooterText</td><td>The text to be displayed in the header area. You may use predfined constant to include information like scaling, pages and more:
<table>
<tr><th>Type this</th><th>To print this</th></tr>
<tr><td>&d</td><td>Date in short format as specified by Regional Settings in Control Panel.</td></tr>
<tr><td>&f</td><td>File name.</td></tr>
<tr><td>&F</td><td>Full file path.</td></tr>
<tr><td>&n</td><td>Line break. You can have a maximum of three text lines.</td></tr>
<tr><td>&P</td><td>Total number of pages in document.</td></tr>
<tr><td>&p</td><td>Currently printed page number.</td></tr>
<tr><td>&s</td><td>Print scaling. For example 100%.</td></tr>
<tr><td>&t</td><td>Time in the format specified by Regional Settings in Control Panel.</td></tr>
<tr><td>&u</td><td>Name of the currently logged on user.</td></tr>
</table></td></tr>
<tr><td></td><td>TextColor</td><td>Color to use for the text (use 0 for black)</td></tr>
<tr><td></td><td>Position</td><td>Horizontal position for the footer text:
3. Left
4. Center
5. Right</td></tr>
<tr><td></td><td>TopDistance</td><td>Distance from the paper edge to the top of the header text.</td></tr>
<tr><td>Returns</td><td>LONG</td><td>Non-zero if success.</td></tr>
</table>

2.1.174 *SetIdleImage*

Set filename to an image file that will be displayed in the control if no file is loaded.
This image could be a splash screen image or a logo.

Syntax	long SetIdleImage(BSTR ImageFileName)	
Parameters	ImageFileName	UNC or URL to a valid image file. Formats supported: TIFF, PNG, BMP and JPEG.
Returns	LONG	Non-zero if success.

2.1.175 *SetLicenseOwner*

Set the name of the license owner. Will be shown in the About dialog as "Licensed to owner".

Syntax	BOOL SetLicenseOwner(BSTR Owner)	
Parameters	Owner	Name of license owner. This is a user defined value and is only used for display purpose.
Returns	BOOL	TRUE if success.

2.1.176 *SetMetaFileDPI*

Set resolution to use for metafiles copied to clipboard. This setting is also used for the SetMouseAction(15) and GetSelectedAreaImage methods. The default resolution is 96 DPI.

Syntax	BOOL SetMetaFileDPI(long DPI)	
Parameters	DPI	Metafile, or image, resolution given as dots per inch (DPI)

Returns	BOOL	TRUE if success.

2.1.177 *SetMouseAction*

Set active mouse action. Use this method to configure the mouse to work in different modes, for example to pan while left button is pressed, select a region that will be saved as an image and much more.

Syntax	long SetMouseAction(long Action)	
Parameters	Action	Set new active mouse action. See table below for available actions.
Returns	LONG	Non-zero if success.

Available mouse actions:

Value	**Action**
0	No action. This will clear all active states.
2	Set copy as metafile to clipboard mode. Use the mouse to drag a rectangle and the area will be copied to the clipboard.
4	Activate panning hand mode.
7	Activate markup edit mode.
10	Activate area measurement mode.
11	Activate measurement calibration mode.
12	Activate distance measurement mode.
13	Set save selected area to an image file mode. Use the mouse to drag a rectangle and you will then be prompted for file name and file format.
14	Activate a special mode where you can select lines and polylines in a drawing. When a line or polyline is selected a markup measurement element will be automatically created using the same coordinates as the original drawing line or polyline.
15	Set a mode where user can select an area on the screen. The area will be copied to a bitmap. After the bitmap is created the ActionCompleted event will be called with ID set to 15. You may then use the GetSelectedAreaImage method to obtain the actual image (as a HBITMAP). You can use the SetMetaFileDPI method to control the resolution of the returned image.
16	Select crop area mode. The user will select an rectangular area and the current document will be cropped to fit the selected area.
17	Activate a special mode where you can select line segmenets in a drawing. When a line segment is selected a markup measurement element will be automatically created using the same coordinates as the original drawing line.
19	Select a rectangular area with the mouse that will be cleared with white color. This tool will only work for raster type file formats (for example TIFF, PNG and JPEG).
20	Set select and copy text to clipboard mode. In this mode the user can select an area on the screen. Any text found within the area will be highlighted. When the user releases the mouse button, the selected text will be copied to clipboard.

2.1.178 *SetNamedView*

Set a named view as currently displayed portion.

Syntax	long SetNamedView(BSTR ViewName)	
Parameters	ViewName	The named view to set as currently displayed portion. The view name must be one of the views obtained using GetNamedView.
Returns	LONG	Non-Zero if success.

2.1.179 *SetPanPosition*

Set pan values. Control which part of the loaded file that is currently displayed.

Syntax	BOOL SetPanPosition(long PosX, long PosY)	
Parameters	PosX	New horisontal position.
	PosY	New vertical position.
Returns	BOOL	TRUE if success.

2.1.180 *SetPDFCustomProperty*

Add a user defined property to exported PDF file. You can add as many custom properties as you want. Please note that custom properties will be valid only for one exported PDF file, so you will have to define custom properties for each PDF file you want to create. A custom property can be any string that is not equal to one of the standard PDF properties. The standard properties, like for example Author, can be changed by using the SetPDFProperty method.

Syntax	HRESULT SetPDFCustomProperty(BSTR Name, BSTR Value)	
Parameters	Name	User defined property name.
	Value	The string to set for user defined property.
Returns	HRESULT	Returns S_OK if success.

2.1.181 *SetPDFProperty*

Set one of the predefined PDF properties to given value (strings only). These properties will be added to all PDF files created by the control.

Syntax	HRESULT SetPDFProperty(LONG Key, BSTR Value)	
Parameters	Key	The following values are supported by this method: 0. Author 1. Creator 2. Keywords 3. Producer 4. Subject 5. Title 6. Company
	Value	The string to set for the given key.
Returns	HRESULT	Returns S_OK if success.

2.1.182 *SetPDFRasterLoadSettings*

Set resolution and number of colors to be used when a PDF is loaded as raster image.

Syntax	long SetPDFRasterLoadSettings(long BitsPerPixel, long DotsPerInch)	
Parameters	BitsPerPixel	Number of bits per pixel, following values are supported: 1. Monochrome (black and white only). 4. 16 colors. 8. 256 colors. 24. True color (16.7 million colors).

	DotsPerInch	Resolution in pixels (dots) per inch, also known as DPI.
Returns	LONG	Non-zero if success.

2.1.183 *SetPDFResourcePath*

Set the directory where all the PDF font resource files are located.

Syntax	long SetPDFResourcePath(BSTR ResourcePath)	
Parameters	ResourcePath	Set folder where the PDF font resource files are located.
Returns	LONG	Non-zero if success.

2.1.184 *SetPenState*

Set the visibility state for given pen index.

Syntax	BOOL SetPenState(long Pen, long State)	
Parameters	Pen	Pen number to modify.
	State	New pen state (0 = off, 1 = on).
Returns	BOOL	TRUE if success.

2.1.185 *SetPenTableEntry*

Set width, color, and style for the given pen table entry.

Syntax	BOOL SetPenTableEntry(long Pen, double Width, long Color, long Style)	
Parameters	Pen	Pen number to modify.
	Width	Pen width in inches.
	Color	Pen Color (Windows COLORREF). When specifying an RGB color, the color value has the following hexadecimal form: 0x00bbggrr.

	Style	Windows GDI pen style.
		The following styles are supported:
		`#define PS_SOLID 0`
		`#define PS_DASH 1 /* ------- */`
		`#define PS_DOT 2 /* ..... */`
		`#define PS_DASHDOT 3 /* _._._._ */`
		`#define PS_DASHDOTDOT 4 /* _.._.._ */`

Returns	BOOL	TRUE if success.

2.1.186 *SetPrintCopies*

Use this method to set the number of copies to use for next print job.
Please note that this method may only be used together with the **SelectPrinter** method.

Syntax	long SetPrintCopies(long Copies)	
Parameters	Copies	Number of copies for the next print job.
Returns	LONG	Non-zero if success.

2.1.187 *SetPrintPaperSize*

Set paper size and orientation for selected printer.
Please note that this method may only be used together with **SelectPrinter**.

Syntax	long SetPrintPaperSize(long Size, long Orientation)	
Parameters	Size	Paper format index defined by Windows. You can find a list of available values in Appendix G.
	Orientation	Page Orientation to use. Accepted values: 1.Portrait mode. 2.Landscape mode.
Returns	LONG	Non-zero if success.

2.1.188 *SetPrintPaperSizeMM*

Set paper size and orientation for selected printer based on the given size in millimeters.
Please note that this method may only be used together with **SelectPrinter**.

Syntax	long SetPrintPaperSizeW(double Width, double Height)	
Parameters	Width	Minimum width of paper to select.
	Height	Minimum height of paper to select.
Returns	LONG	Non-zero if success.

2.1.189 *SetPropertyDouble*

Set a custom property double value for markup element.

Syntax	long SetPropertyDouble (long ID, long Parameter, double Value)	
Parameters	ID	You may define up to 10 custom double values (ID from 0 to 9) per markup element. These custom values will be stored in the markup file and can be obtained later using the GetPropertyDouble method. Values from 20 and beyond is used for the following settings: 20. **Line Width**. Change line width for the element with the given handle. Value is line width in millimeters. 21. **Area Height**. Set the measurement area height. The SetConfigValue method controls which coordinate system that should be used. If you want to set a calibrated height you should call SetConfigValue(70,1) before using this method. 22. **Revision Cloud Arc Limit**. Set the arc limit for the given element in millimeters. This limit will be maximum size of radius used to create the revision cloud. 23. **Hatch Line Spacing**. Set the distance between each hatch line in millimeters for the element with the given handle. 24. **Hatch Line Width**. Set line width to be used for hatch lines in millimeters for the element with the given handle.
	Parameter	Markup element handle.
	Value	Double value to set, please see the different ID's for more information.

Returns	LONG	Non-zero if success.

2.1.190 *SetPropertyLong*

Set a property long value.

Syntax	long SetPropertyLong(long ID, long Parameter, long Value)	
Parameters	ID	Identifier of the property value to set. The following values are supported: 0. **Markup Layer State**. Parameter is the layer index. If the value is non-zero the layer will be visible, else it is invisible. 1. **Markup Element State**. Parameter is the element handle. If the value is non-zero the element is visible, else it is invisible. 2. **Markup Element Page**. Parameter is the element handle. The value is the page number where the element will be located (use -1 to display the item on all pages). 3. **Markup Element Lock**. Parameter is the element handle. If the value is non-zero the element is locked, else it can be modified. 4. **Markup Layer Color**. Parameter is the layer index. The value is new layer color. 5. **Markup Element Opacity**. Parameter is the element handle. The value is the new element opacity in the range 0 to 255. 255 is full opacity (no transparency). 6. **Markup Element Color**. Parameter is the element handle. The value is the new line color. 7. **Markup Element Fill Color**. Parameter is the element handle. The value is the new fill color. 8. **Markup Element Negative Measure.** Parameter is the element handle. If the value is non-zero a markup measurement will be treated as negative. 9. **Markup Element Fill Type**. Parameter is the element handle. The value is the new fill type. 10. **Markup Element Type**. This is a read only value. 11. **Markup Element Z-Level**. Parameter is element handle. The value is the new z-level to use for this element. Z-levels can be given between -5 to 5. By using z level you have full control of the draw order for all elements. Z level 0 is default value for all new elements. Note that z-level can only be used if it is enabled by SetConfigValue(46,1). 12. **Markup Element Layer**. Parameter is the element handle. The value is the new layer. 13. **Include Perimeter**. Parameter must be a handle to an area measurement element. If the given value is non-zero the perimeter will be included in the measurement text. 14. **Include Volume**. Parameter must be a handle to an area measurement element. If the given value is non-zero the calculated volume will be in the measurement text. Please note that this requires that a valid height has been set, see SetPropertyDouble id 21 for more information.
	Parameter	Depends on the ID, see ID above for more information.
	Value	Value to set, please see ID above for more information.
Returns	LONG	Non-zero if success.

2.1.191 *SetPropertyString*

Set a property string value. This method is a replacement for properties with parameter, which is not supported by all programming languages.

Syntax	long SetPropertyString (long ID, long Parameter, BSTR Value)	
Parameters	ID	Identifier of the property string value to set. The following values are supported: 0. **Markup Element Text**. Parameter is the elment handle. 1. **Markup Element Name**. Parameter is the element handle. 2. **Markup Element Group**. Parameter is the element handle.
	Parameter	Depends on the ID, see ID above for more information.
	Value	Returned value, see ID above for more information.
Returns	LONG	Non-zero if success.

2.1.192 *SetSerialNumber*

Unlock the control to use all its functionality and remove all evaluation messages.
The serial number to use with this method will be provided when you purchase a license.

Syntax	BOOL SetSerialNumber(BSTR SerialNumber)	
Parameters	SerialNumber	Serial number to unlock control.
Returns	BOOL	TRUE if a valid serial number is provided.

2.1.193 *SetTIFFTag*

Override one of the available TIFF tags to given value (strings only). These tags will be written to all TIFF files created by the control.

Syntax	HRESULT SetTIFFTag (LONG Index, BSTR Value)	
Parameters	Index	The following values are supported by this method: 0. Software 1. Artist 2. Copyright
	Value	The string to set for the given key.
Returns	HRESULT	Returns S_OK if success.

2.1.194 *SetToolWindolStyle*

Set font styles and colors to be used by the tooltip window displayed using the **ShowToolWindow** method.

Syntax	long SetToolWindowStyle(OLE_COLOR BackColor, OLE_COLOR TextColor, BSTR FaceName, long Size, long CharSet, long Weight, long Italic)	
Parameters	BackColor	Background color used for the tooltip window.
	TextColor	Color to use for the text inside the tooltip window.
	FaceName	Facename of the font to use.
	Size	Size of the font in points (1/72 dpi).

	CharSet	Character set to use for the font.
	Weight	Character weight, e.g.: 0 for normal, 700 for bold.
	Italic	Set this value to non-zero to use an italic style font.
Returns	LONG	Non-zero if success.

2.1.195 *SetTranslationFile*

Set full path to the translation file. You can have several translated text resource files with different languages and can use this method to switch between them dynamically.
The default English translation file, named **scViewerX.TextResources**, can be found in the redistribution folder and may be modified using a text editor (for example Notepad).

Syntax	long SetTranslationFile(BSTR FileName)	
Parameters	SerialNumber	Full path to an existing translation file.
Returns	LONG	Non-zero if file was successfully loaded.

2.1.196 *SetTXTParams*

Set font, margins and page dimension to be used for loading and displaying text files.

Syntax	long SetTXTParams(BSTR Fontname, double Fontheight, double Pagewidth, double Pageheight, double MarginH, double MarginV);	
Parameters	Fontname	Name of font to use for text.
	Fontheigth	Font height in points. Default value is 10.
	Pagewidth	Width of page in mm. Default value is 210.
	Pageheight	Height of page in mm. Default value is 297.
	MarginH	Horizontal margin in mm. Default value is 20.
	MarginV	Vertical margin in mm. Default value is 20.
Returns	LONG	Non-zero if success.

Sample settings for loading text files:
A4: SetTXTParams(“Arial”, 10, 210, 297, 20, 20);
Letter: SetTXTParams(“Arial”, 12, 8.5 * 25.4, 11 * 25.4, 25.4, 25.4);

2.1.197 *SetUserCursor*

Set your own cursor for the given mouse action type.

Syntax	long SetUserCursor(enumCursorType CursorType, OLE_HANDLE Cursor)	
Parameters	CursorType	Select cursor to change: 0. Default cursor 1. Zoom window cursor 2. Pan hand cursor 3. Pressed pan hand cursor 4. Zoom in/out cursor 5. Save Area as Image selection cursor.
	Cursor	Handle to a cursor loaded by LoadResPicture or LoadCursor.

Returns	LONG	Non-zero if the cursor was successfully changed.

2.1.198 *SetVectorLayerState*

Set visibility state for given vector layer.

Syntax	long SetVectorLayerState(long Layer, long State)	
Parameters	Layer	Vector layer number.
	State	New visibility state for the layer (0=off, 1=on)
Returns	LONG	Nonzero if success.

2.1.199 *SetWatermarkFont*

Set the font that will be used to print watermarks.

Syntax	long SetWatermarkFont(BSTR FaceName, long Height, long Weight, long CharSet);	
Parameters	FaceName	Font name.
	Height	Font height in points.
	Weight	The weight of the font in the range 0 through 1000. For example, 400 is normal and 700 is bold. If this value is zero, a default weight is used.
	CharSet	The character set to use. You will find more information about character sets in the Windows SDK documentation.
Returns	LONG	Non-zero if success.

2.1.200 *SetWatermarkText*

Set watermark text, rotation, transparency and color. Calling this function with a valid text string will set the EnableWatermark property to true. You will find a batch print sample in the SDK that demonstrates how to use this method.

Syntax	long SetWatermarkText(BSTR WatermarkText, OLE_COLOR TextColor, DOUBLE Opacity, long Rotation);	
Parameters	WatermarkText	Text to use for watermark.
	TextColor	Color to use for the watermark text (use 0 for black)
	Opacity	Watermark opacity from 0 to 100. 0 is no opacity (invisible) and 100 is full opacity (no transparency). The default value is 50.
	Rotation	Rotation gvein in degrees to use for the watermark text. The default value is 45 degrees.
Returns	LONG	Non-zero if success.

2.1.201 *SetZoomWindowColors*

Customize the colors and line style that will be used for the zoom window rectangle. SetZoomWindowColors(RGB(101, 114, 224), RGB(199, 216, 248), PS_SOLID) will change the zoom window rectangle to a Windows styled zoom window.

Syntax	BOOL SetZoomWindowColors(OLE_COLOR Border, OLE_COLOR Fill, long Border)	
Parameters	Border	Color to use for the rectangle border.
	Fill	Fill Color to use for the rectangle. Set to 0 for no fill.
	Border	Linestyle to use for the rectangle. One of the predefined Windows values,for example PS_SOLID.
Returns	BOOL	TRUE if success.

2.1.202 *SetViewRect*

Select a rectangular area of the file that should be viewed. This function uses world coordinates.

Syntax	BOOL SetViewRect(long x1, long y1, long x2, long y2)	
Parameters	x1	Left coordinate of area to show.
	y1	Upper coordinate of area to show.
	x2	Right coordinate of area to show.
	y2	Lower coordinate of area to show.
Returns	LONG	TRUE if success.

2.1.203 *SetZoomType*

Set zoom type or action.

Syntax	BOOL SetZoomType(long ZoomType)	
Parameters	ZoomType	Set Zoom Type, one of the following: 1. Zoom All 2. Zoom To Width 3. Zoom In 4. Zoom Out 5. Zoom Window 6. Pan Center 7. Panning Hand 8. Zoom In/Out mode. 9. Zoom to Height. 10.Zoom to 100% (actual size). 11. Clear any selected markup and redraw.
Returns	LONG	TRUE if success.

2.1.204 *ShowToolWindow*

Show a tooltip window containing a text message that you want to display to the user.
You may modify the font and colors used by the text with the **SetToolWindowStyle** method.

Syntax	long ShowToolWindow(long X, long Y, BSTR MessageText, long TimeOut)	
Parameters	X	Horizontal position of the window in screen coordinates.
	Y	Vertical position of the window in screen coordinates.
	MessageText	The text you want to display in the tooltip window. The text may contain several lines, use LF (\n) to separate lines.

	TypeOut	Number of milliseconds the window will stay visible until it is automatically removed.
Returns	LONG	Non-zero if file was successfully loaded.

2.1.205 *TextExtract*

Extract text from a single page or whole document. You must use the HaveText property to see if the document has text or not.

Syntax	long TextExtract(LONG PageNo, BSTR* TextContent)	
Parameters	PageNo	Page to extract text from, page numbers start at index 0. Set PageNo to -1 to extract text from all pages in the document.
	TextContent	The extracted text.
Returns	LONG	Non-zero if success.

2.1.206 *TIFFMergeAddFile*

Add a file to the currently merged TIFF file.
TIFFMergeInit must be called before any file can be added.

Syntax	long TIFFMergeAddFile(BSTR FileName, BOOL AllPages)	
Parameters	FileName	Full path name of the file that will be added to the currently merged TIFF.
	AllPages	Include all pages, or just the first page in the given file.
Returns	LONG	Non-zero if the file was added.

2.1.207 *TIFFMergeAddImage*

Add an image to the currently merged TIFF file. The image handle must be HBITMAP.
TIFFMergeInit must be called before any image can be added.

Syntax	long TIFFMergeAddImage(OLE_HANDLE Image)	
Parameters	Image	The HBITMAP handle of the image to add as a new page.
Returns	LONG	Non-zero if the image was added.

2.1.208 *TIFFMergeClose*

Close the currently merged TIFF file and output all pages to the given file name.

Syntax	long TIFFMergeClose(BSTR OutputName)	
Parameters	OutputName	Full path name of the new TIFF file to create.
Returns	LONG	Non-zero if success.

2.1.209 *TIFFMergeInit*

Start a new empty TIFF file prepared for merging.

Use TIFFMergeAddFile or TIFFMergeAddImage add pages, and finally call TIFFMergeClose to complete the merge.

Syntax	long TIFFMergeInit(long BitsPerPixel, long DPI)	
Parameters	BitsPerPixel	Default bitsperpixel setting to use for the added pages. This setting will be used if an added file doesn't have bitsperpixel settings.
	DPI	Default resolution to use for the added pages. This setting will be used if an added file doesn't have dpi settings.
Returns	LONG	Non-zero if success.

2.1.210 *TIFFSplit*

Split a multi-page TIFF file into multiple single paged TIFF files (one per page).

Syntax	long TIFFSplit(BSTR InputFile, BSTR OutputFolder, BSTR FileLabel)	
Parameters	InputFile	The multi-page PDF file that will be split into smaller files.
	OutputFolder	A valid folder name for the output files.
	FileLabel	Optional label to add to output filenames. By default the output files will be named "inputfilename_1.tif", "inputfilename_2.tif" and so on. By adding "part" as label, the output files will be named "inputfilename__part_1.tif", "inputfilename__part_2.tif" and so on.
Returns	LONG	Non-zero if success.

2.1.211 *WorldToScreen*

Scale from world to screen coordinates. World coordinates are the file's coordinate system.

Syntax	BOOL WorldToScreen(long WorldX, long WorldY, long *ScreenX, long *ScreenY)	
Parameters	WorldX	World X coordinate.
	WorldY	World Y coordinate.
	ScreenX	Transformed screen X coordinate.
	ScreenY	Transformed screen Y coordinate.
Returns	LONG	TRUE if success.

2.2 scViewerX properties

2.2.1 *ActivePaperBin*

Can only be used together with the SelectPrinter and associated methods. You can list all available paper bins by using the PaperBins and PaperBinName properties.	
Type	LONG
Access	Read and write
Default value	-1 (default)

2.2.2 *AdjustColorFlag*

If this property is set to TRUE all colors equal to, or near, the background color will be inverted to make sure the geometry is visible. This will make sure that CAD drawings using white lines will be displayed as black on white background.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.3 *AntiAliaseRaster*

If this property is set to TRUE all raster images will be drawn with better quality when the image is scaled down.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.4 *ArcResolution*

Set the number of line segments used to draw circles and arcs.	
Type	LONG
Access	Read and write
Default value	60

2.2.5 *BackColor*

Set or get the active background color.	
Type	OLE_COLOR
Access	Read and write
Default value	WHITE

2.2.6 *BkImageMemoryLimit*

Set or get the maximum size of memory to use for display double buffering. Double buffering is used to reduce the number redraws, and make panning smoother. The default value is 150 MB.	
Type	LONG
Access	Read and write
Default value	150

2.2.7 *BorderStyle*

Set or get the active border style.	
Type	SHORT
Access	Read and write
Default value	0

2.2.8 *Calibration*

Set or get the calibration value to use for markup measurement functions. If you have a drawing which is e.g.: 1:10 (1 measured millimeter equals 10 millimeters in the real world, you should set the value to 10.0	
Type	DOUBLE
Access	Read and write
Default value	1.0

2.2.9 *CalibrationDefaultUnit*

Set or get default calibration unit for the calibration dialog. Following values are supported if you are using metric measurement: 0.mm 1.cm 2.dm 3.m 4.km Following values are supported if you are using imperial measurement: 0.inch 1.feet 2.yard 3.mile	
Type	LONG
Access	Read and write
Default value	0

2.2.10 *CenterImage*

If this option is enabled the image will be centered on the screen when zoomed out. The area outside the image will be filled with the color defined by **WorkspaceColor** property.

Type	LONG
Access	Read and write
Default value	1 (TRUE)

2.2.11 *CurrentPage*

Set or get the currently displayed page index. Pages are numbered from 0.	
Type	LONG
Access	Read and write
Default value	0

2.2.12 *DialogEnableFlags*

A flag for disabling or enabling dialogs (for example calibration dialog).
By default, all dialogs are enabled.
If you disable the calibration dialog an event will be called instead.
The following flags are supported:

Bit	Value	Controls
1	1	If bit is set the calibration dialog will be enabled.
2	2	If bit is set the markup element property dialog will be displayed when double-clicking an element.
3	4	If bit is set the state page will be displayed in the markup element property sheet dialog.
4	8	If bit is set the xml page will be displayed in the markup element property sheet dialog.
5	16	If bit is set the metadata page will be displayed in the markup element property sheet dialog.

Type	LONG
Access	Read and write
Default value	0xFFFFFFFF (all bits enabled, all control dialogs enabled).

Sample usage:
1. To disable only XML and STATE property pages set DialogEnableFlags to 0xFFFFFFF3.
2. To disable only the calibration dialog: set DialogEnableFlags to 0xFFFFFFFE

2.2.13 *DXFWriterMM*

If enabled all DXF files be written using coordinates in millimeter. If disabled DXF files will use Inch coordinates.	
Type	LONG
Access	Read and write
Default value	1 (TRUE)

2.2.14 *DXFLineWeightFlags*

This property controls how lineweights should be written to a DXF file.

Please note that AutoCAD only support specific values for lineweights, but the control can be configured to write any value. The following flags are supported: 0. Only write values supported by AutoCAD. All other values will be rounded up/down to nearest valid value. 1. Only write values supported by AutoCAD. All other values will be set to lineweight value 0. 2. Write any lineweight value. Using this option may cause the DXF to be incompatible with AutoCAD, but it will load fine in many other applications. AutoCAD do only support the following lineweight values: 0, 5, 9, 13, 15, 18, 20, 25, 30, 35, 40, 50, 53, 60, 70, 80, 90, 100, 106, 120, 140, 158, 200 and 211.

Type	LONG
Access	Read and write
Default value	0

2.2.15 *DXFLineWeights*

If enabled the control will write line weights to the DXF file (DXF code 370), if set to FALSE (zero) no line weights will be written. Please look at the DXFLineWeightFlag property for more information.	
Type	LONG
Access	Read and write
Default value	1 (TRUE)

2.2.16 *EnableAboutBox*

The About command will be visible in the context menu if set to TRUE. Set to FALSE to hide the About command. Only valid if control is licensed.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.17 *EnableContextMenu*

Allows the user to use the right-click context menu if set to TRUE. Set to FALSE to disable context menu.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.18 *EnableContextMenuPrint*

The Print item will be visible in the context menu if this property set to TRUE. Set to FALSE to remove Print option from the context menu.	
Type	BOOL

Access	Read and write
Default value	TRUE

2.2.19 *EnableFooter*

Enable or disable footer information during printing. A footer is a user defined text that will be printed at the bottom margin of the paper sheet. See the SetFooterFont and SetFooterText methods for more information about how to setup footer text.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.20 *EnableHeader*

Enable or disable header information during printing. A header is a user defined text that will be printed at the top margin of the paper sheet. See the SetHeaderFont and SetHeaderText methods for more information about how to setup header text.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.21 *EnableMarkupContextMenu*

Disable or enable the context menu which is displayed when a user right-clicks a selected markup element.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.22 *EnableWatermark*

Enable or disable printing of watermarks. A watermark is a text that will be displayed on top of the document, usually with transparency. Please check the SetWatermarkFont and SetWatermarkText methods for more information about how to define watermarks.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.23 *ExportAlignmentH*

Controls placement of exported drawing if the exported "page" width is larger than the drawing width. Used by the ExportExtents method. Available values: hLeft, hCenter and hRight.	
Type	LONG

Access	Read and write
Default value	hLeft

2.2.24 *ExportAlignmentV*

Controls placement of exported drawing if the exported "page" height is larger than the drawing height. Used by the ExportExtents method. Available values: vTop, vMiddle and vBottom.	
Type	LONG
Access	Read and write
Default value	vTop

2.2.25 *FileFontsUsed*

Return number of fonts that are used in the active document. Use the **GetFileFontInformation** method to obtain information about document fonts.	
Type	LONG
Access	Read
Default value	Depends on the loaded file

2.2.26 *Flip*

Flip the image vertically if set to TRUE.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.27 *ForeColor*

Set foreground color to use when monochrome drawing is enabled.	
Type	OLE_COLOR
Access	Read and write
Default value	0

2.2.28 *Grayscale*

Set or get grayscale mode. In grayscale mode all colors will be replaced by shades of gray.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.29 *HaveComplexROP*

This property will return true if the file contains complex raster operations. Such files may be correctly converted to other vector formats, like for example DWF. You may instead convert the file to a raster format (for example DWFRASTER).	
Type	BOOL
Access	Read
Default value	Depends on the loaded file.

2.2.30 *HaveRaster*

This property will return TRUE if the loaded file contains raster images.	
Type	BOOL
Access	Read
Default value	Depends on the loaded file.

2.2.31 *HaveScanner*

This property will return nonzero if a WIA compatible scanner is installed.	
Type	BOOL
Access	Read
Default value	Depends on the system.

2.2.32 *HaveText*

This property will return nonzero if the loaded file contains text information.	
Type	LONG
Access	Read
Default value	Depends on the loaded file.

2.2.33 *InstalledPrinters*

This property will return the number of installed printers in a system. Use this property together with the **PrinterName** method to enumerate all printers on a system.	
Type	BOOL
Access	Read
Default value	System dependent

2.2.34 *IsPDFA*

This property will return a non-zero value if the loaded file is stored using a PDF/A standard.	
Type	BOOL
Access	Read
Default value	Depends on the loaded file

2.2.35 *IsPDFSigned*

This property will return a non-zero value if the loaded file contains digital signatures.	
Type	BOOL
Access	Read
Default value	Depends on the loaded file

2.2.36 *IsRasterFormat*

This property will return a non-zero value if the loaded file is a raster (image) file. Raster files include formats like TIFF, JPEG, CALS, PNG, GIF and others. You may also load any PDF file as a raster using the LoadPDF. If you have a raster format you may use the different raster processing methods, for example the **DeskewImage** method.	
Type	BOOL
Access	Read
Default value	Depends on the loaded file

2.2.37 *LineEditModeDefault*

If this property is set to TRUE the default markup element edit mode will be line editing, if the selected markup element can be edited in this mode. You will then have to press ALT key to move such an element. If you set this property to FALSE, the default edit mode will be move mode for elements than can be edited in line mode (and you must press ALT key to enter line edit mode).	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.38 *LoadDWFPaperSize*

Setting this property to TRUE will force the DWF reader to load all files with paper size settings if available. If you set this property to FALSE, all DWF files will be loaded with drawing extents.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.39 *LoadedFormatType*

Return the format identifier for currently loaded document.
One of the following values is returned:

0. Unknown file format – cannot be loaded
1. HPGL/2
2. Gerber RS-274D
3. Gerber RS-274X
4. Calcomp plotter format
5. CALS
6. TIFF

7. PNG 8. Windows BMP 9. JPEG 10. CGM (Computer Graphics Metafile) 11. Autodesk DWF or DWFx 12. Adobe PDF 13. JEDMICS C4 (tiled group 4) 14. WebP Image Format 15. Microsoft Word 16. Microsoft Excel 17. Microsoft Powerpoint 18. GIF 19. JPEG 2000 20. Text 21. LibreOffice Draw Format 22. Intergraph Raster Format 23. Autodesk DXF 24. Excellon Drill Format

Type	LONG
Access	Read
Default value	Depends on the loaded file

2.2.40 *LoadEmbeddedRaster*

Control if embedded raster (RTL) should be loaded or not from plotter files.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.41 *MarkupCanPaste*

Returns TRUE if a markup element has been copied to clipboard. If this property returns TRUE you can paste the markup element by using the MarkupPaste method.	
Type	BOOL
Access	Read
Default value	FALSE

2.2.42 *MarkupCanUndo*

Returns TRUE if any markup element has been modified and undo action is possible. If this property returns TRUE you can undo the last change by calling the **MarkupUndo** method.	
Type	BOOL
Access	Read
Default value	FALSE

2.2.43 *MarkupDimLineWidth*

Get or set line width in millimeter to use for new markup dimension line elements.	
Type	DOUBLE
Access	Read and write
Default value	0.125

2.2.44 *MarkupDrawColor*

Get or set line color to use for new markup elements. This color will also be used to outline filled elements (like for example polygon, rectangles and more)	
Type	OLE_COLOR
Access	Read and write
Default value	0x00FF0000 (RED)

2.2.45 *MarkupElementGroup*

Set, or read, a user defined group name that will be associated with the given markup element. The same name may be given to several different elements, to show that these elements belong to the same group. Sample usage: "Room 1", "Room 2" etc.	
Parameter	Handle for markup element.
Type	STRING
Access	Read and write
Default value	None

2.2.46 *MarkupElementLock*

Lock or unlock the given markup element. Use value 1 to lock an element and 0 to unlock it. A locked element cannot be modified until it is unlocked.	
Parameter	Handle for markup element.
Type	LONG
Access	Read and write
Default value	0

2.2.47 *MarkupElementName*

Set, or read, a unique user defined name that will be associated with the given markup element.	
Parameter	Handle for markup element.
Type	STRING
Access	Read and write
Default value	None

2.2.48 *MarkupElementOpacity*

Set, or get, the opacity for the given markup element. Opacity values are given in the range from 0.0 to 1.0, where 1.0 is 100% opacity (no transparency at all).	
Parameter	Handle for markup element.
Type	DOUBLE
Access	Read and write
Default value	1.0

2.2.49 *MarkupElementPage*

Set, or get, the page number (zero index based) for the given markup element. You may set the page number to -1 to display the element on all pages in a multi-page document.	
Parameter	Handle for markup element.
Type	LONG
Access	Read and write
Default value	Page number for which the markup element was created

2.2.50 *MarkupElementState*

Set, or get, the display state for the given markup element.	
Parameter	Handle for markup element.
Type	LONG
Access	Read and write
Default value	1 (Visible)

2.2.51 *MarkupElementText*

Set, or get, the text for given markup element. You can modify change text for the following element types: Text, Note, Arrow Text, Stamp and Dimension Line. If you change the Dimension Line text, it will be used as custom text and override the measured value. To reset Dimension Line text back to measured value use an empty string (“”).	
Parameter	Handle for markup element.
Type	STRING
Access	Read and write
Default value	None

2.2.52 *MarkupEnableLineWidth*

Get or set line markup line width display flag. This property will allow you enable or disable line width for displayed and drawn markup elements. Set this flag to false to draw all markup elements with 1 pixel linewidth.	
Type	BOOL

Access	Read and write
Default value	True: Draw elements with assigned line width.

2.2.53 *MarkupEraserWidth*

Get or set the width for the markup eraser in millimeters.	
Type	DOUBLE
Access	Read and write
Default value	10.0

2.2.54 *MarkupFillColor*

Get or set fill color to use for new filled markup elements.	
Type	OLE_COLOR
Access	Read and write
Default value	0x00FF0000 (RED)

2.2.55 *MarkupFillType*

Get or set fill type to use for new filled markup elements. Following values are valid: 0. Outlined 1. Solid 2. Horizontal Hatch 3. Vertical Hatch 4. Cross Hatch 5. Diagonal Hatch 6. Diagonal Cross Hatch	
Type	LONG
Access	Read and write
Default value	1 (Solid)

2.2.56 *MarkupFolder*

Get or set the current markup folder. Use this property to setup the control to use a common folder for all markup files.	
Type	STRING
Access	Read and write
Default value	None

2.2.57 *MarkupHatchSpacing*

Set hatch pattern distance in millimeters. This distance is used for MarkFillTypes from 2 to 6 (hatch patterns).	
Type	DOUBLE

Access	Read and write
Default value	2.5

2.2.58 *MarkupHighlightColor*

Get or set color to use for new markup highlight elements (type 24).	
Type	OLE_COLOR
Access	Read and write
Default value	0x00FFAD5B

2.2.59 *MarkupIsModified*

Returns TRUE if any markup elements has been modified, or are added.	
Type	BOOL
Access	Read
Default value	FALSE

2.2.60 *MarkupLayer*

Get or set layer number to use for new markup elements.	
Type	LONG
Access	Read and write
Default value	0

2.2.61 *MarkupLayerColor*

Get or set active color for given markup layer.	
Parameter	Markup layer index
Type	OLE_COLOR
Access	Read and write
Default value	None

2.2.62 *MarkupLayerState*

Get or set visibility state for given markup layer.	
Parameter	Markup layer index
Type	LONG
Access	Read and write
Default value	None

2.2.63 *MarkupLineStyle*

Get or set current markup line style. Following styles are supported: 0. Solid 1. Dash 2. Dot 3. Dash dot 4. Dash dot dot	
Type	LONG
Access	Read and write
Default value	0 (Solid)

2.2.64 *MarkupLineWidth*

Get or set line width in millimeters to use for added markup elements.	
Type	DOUBLE
Access	Read and write
Default value	0.125

2.2.65 *MarkupMTextColor*

Get or set the color to be used for markup measurement element text content. By default, all measurement elements will use the current draw color for its text. You can use this property to set a separate text color for all created measurement elements, independent of the current draw color. If you want to use the active draw color for text as well, you should set this property to 0xFFFFFFFF.	
Type	OLE_COLOR
Access	Read and write
Default value	0xFFFFFFFF (the current draw color will be used for measurement text)

2.2.66 *MarkupNumElements*

Return number of markup elements for the active document.	
Type	LONG
Access	Read and write
Default value	0

2.2.67 *MarkupNumLayers*

Return number of available markup layers. User defined layers can be added by using the **MarkupAddLayer** method.	
Type	LONG
Access	Read and write
Default value	1

2.2.68 *MarkupSelectedHandle*

Return handle for currently selected markup element. If the returned value is 0 there is no currently selected markup element.	
Type	LONG
Access	Read
Default value	0 (no element selected)

2.2.69 *MarkupShow*

Toggle display of markup elements on/off.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.70 *MarkupShowDimText*

Toggle display of markup dimension and measurement path text on/off.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.71 *MarkupShowAreaText*

Toggle display of markup area measurement text on/off.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.72 *MarkupSnapEnable*

Enables or disables snap to drawing geometry when drawing markup elements.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.73 *MarkupTransparency*

Get or set transparency flag for added markup elements.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.74 *MarkupUser*

Get or set name of active commenter.	
Type	STRING
Access	Read and write
Default value	Currently logged on user.

2.2.75 *Mirror*

Mirror the image horizontally if set to TRUE.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.76 *Monochrome*

If set to TRUE the loaded file will be shown in monochrome (black and white only) mode only.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.77 *MouseWheelZoomPan*

Enable or disable the use of the mouse wheel button for zooming and panning operations. If TRUE the mouse wheel button with be used for zooming by scrolling the wheel to zoom in out, and for panning by pressing the wheel button down while moving the mouse.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.78 *NumBookmarks*

Return the number of bookmarks available in the current active file. Use the **GetBookmarkName** method to get name of a bookmark and use **GotoBookmark** to go to a bookmark.	
Type	LONG
Access	Read only
Default value	Depends on the loaded file.

2.2.79 *NumMeasurePnts*

Return the number of points available after the latest measurement operation. To get measurement points you may use the **GetMeasurePnt** method.	
Type	LONG

Access	Read
Default value	0

2.2.80 *NumNamedViews*

Return the number of named views in the current active file. Use the **GetNamedView** method to get information about the available views.	
Type	LONG
Access	Read only
Default value	Depends on the loaded file.

2.2.81 *NumSignatures*

Return the number of signatures in a PDF file. Use the GetPDFSignatureData to extract information about each signature.	
Type	LONG
Access	Read only
Default value	Depends on the loaded file.

2.2.82 *NumVectorLayers*

Return the number of vector layers in the currently active document.	
Type	LONG
Access	Read
Default value	Depends on loaded file.

2.2.83 *OCRAvailable*

Returns a true if the Tesseract OCR engine is installed on the system. If Tesseract is installed you can use the CreateSearchablePDF method to create searchable PDF files from scanned PDF, TIFF, PNG and other file formats. See appendix K for more information about Tesseract.	
Type	BOOL
Access	Read
Default value	TRUE if Tesseract is installed, else FALSE.

2.2.84 *OrthoMode*

Enable this property to draw and measure with lines using 45 degrees steps.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.85 *PageViewMode*

Set the page view mode for multi-page documents. The following values are supported: 0. Single page mode 1. Page scroll mode – all pages will be displayed continuously.	
Type	LONG
Access	Read and write
Default value	TRUE

2.2.86 *PaperBins*

This property will return the number of available paper bins for the currently selected printer device. Use SelectPrinter to select printers. You may use ActivePaperBin property to change the paper bin to use for the next print job. You can obtain name of paper bins with the PaperBinName property.	
Type	BOOL
Access	Read
Default value	System dependent

2.2.87 *PDFLargeFormat*

Enables or disables creation of large format PDF files. Large format are files with width or height beyond 5x5m.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.88 *PDFReadEnabled*

Enables or disables loading of PDF files.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.89 *PDFTransparency*

Enable or disable use of transparency in exported PDF files. Some printer drivers don't support transparency, and turning off transparency for exported PDF files may make your files print faster.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.90 *PDFWriterFormat*

Select PDF standard to use for exported PDF files, following options are available: 0. PDF 1.6 (editable) 1. PDF/A-1B 2. PDF/A-2B	
Type	LONG
Access	Read
Default value	2 (PDF/A-2B).

2.2.91 *PrintAllPages*

If TRUE all pages in document will be printed. If you set this property to FALSE only the current page will be printed.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.92 *PrintCenterOnPaper*

If TRUE, the printed drawing will be centered on the paper.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.93 *PrintFitToPaper*

If TRUE, the printed document will be scaled to fit the paper. If you set this property to FALSE, you will have to provide a valid **PrintScale**.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.94 *PrintPageRange*

Print a selection of pages. Set string to empty to print all pages. You may use – to set a range of pages, for example “1-20” will print all pages from 1 up to and including 20. Other samples on usage: “1,2,3,6,8” – print pages 1,2,3,6 and 8. “1-5,9,11-14” – will print pages 1,2,3,4,5,9,11,12,13,14	
Type	BSTR
Access	Read and write
Default value	“” – empty string

2.2.95 *PrintScale*

Scaling to use when printing the drawing. The scaling is a percentage. You can use 100.0 to print in original drawing scale (1:1). Note that if **PrintFitToPaper** is enabled, this scale factor is ignored.	
Type	DOUBLE
Access	Read and write
Default value	100.0 (100%)

2.2.96 *RasterBitsPerPixel*

Return number of bits per pixel for a raster image page. This method works only for raster formats, which includes TIFF, PNG, JPEG, GIF, CALS, EDMICS, WEBP and BMP.	
Parameters	Page number. First page is index 0.
Type	LONG
Access	Read and write
Returns	Bits per pixel used for given raster page.

2.2.97 *RasterCanUndo*

Use this property to check if there are any undoable raster operations available. If it returns TRUE, you can undo operations like DeskewImage with the RasterUndo method.	
Parameter	Page number to check for raster changes.
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.98 *Rotation*

The new displayed rotation to use for the active document in degrees. Legal values are between 0.0 and 360.0.	
Type	DOUBLE
Access	Read and write
Default value	0.0

2.2.99 *ScalePenTable*

If TRUE, the pen table widths will be scaled together with the drawing.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.100 *ShowScrollBars*

Enable or disable use of scrollbars. If set to FALSE, the scrollbars will be hidden.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.101 *SnapEnable*

Enables or disables snap to drawing geometry. Snap to geometry is available for calibration, measurement and markup draw modes. By enabling snap, you will get more accurate measurements.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.102 *SymbolLibraryPath*

Get or set the folder where the symbol libraries are installed. Changing this property with a valid folder will reload all symbol libraries.	
Type	STRING
Access	Read and write
Default value	“C:\Users\Public\Documents\ScViewerX SDK\Symbols”

2.2.103 *TIFFSingleStripMode*

Controls if created TIFF files created with a single strip, or multiple strips. Single strip TIFF files are usually smaller (in terms of file size), but may require an application to use more memory when working with the TIFF file.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.104 *TransparentRaster*

If TRUE transparent raster images will be drawn using transparency. Some printers may have problems with transparent image and for these you can set this property to FALSE.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.105 *TrueSize*

If TRUE all polygons will be filled, and thick lines will be displayed using their real width.	
Type	BOOL

Access	Read and write
Default value	TRUE

2.2.106 *UseDefaultPrinter*

If this property is set to TRUE (non-zero) the control will use the default printer when printing. If FALSE (zero), a Printer Setup Dialog box will be shown when printing is invoked.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.107 *UseFilePaperSize*

If TRUE, the original paper size, if available, will be used when converting, instead of the calculated extents of all drawing entities. Use this property to maintain any white space borders around the drawing.	
Type	BOOL
Access	Read and write
Default value	FALSE

2.2.108 *UsePenTable*

If TRUE, the current pen table will be used for display, print and conversion.	
Type	BOOL
Access	Read and write
Default value	TRUE

2.2.109 *WorkspaceColor*

Color to use for filling the area outside the actual image area.	
Type	OLE_COLOR
Access	Read and write
Default value	Depending on the active Windows desktop theme.

2.2.110 *ZoomFactor*

Set/get current zoom factor. The zoom factor is used to scale the document in the active window.	
Type	DOUBLE
Access	Read and write
Default value	A dcoument initial scale is set to zoom all, and the actual ZoomFactor depend highly on the document.

2.3 scViewerX Events

2.3.1 *ActionCompleted*

This event is called after a mouse action has been completed. The ActionID parameter will contain the same value as the one given to SetMouseAction.

Syntax	void ActionCompleted(long ActionID)	
Parameters	ActionID	ID for completed mouse action.
DispatchID	19	

2.3.2 *AfterPrint*

This event is called after a page has been printed but before the page is sent to the printer. This event allows client application to add information to the page by using the provided GDI device context. The Page parameter contains the number for the page that is printed.

Syntax	void AfterPrint(long DC, long Page)	
Parameters	DC	Windows GDI printer device context
	Page	Document page currently printed (first page is index 0)
DispatchID	2	

2.3.3 *CalibrationComplete*

This event is called after user calibration is completed. The user has selected two points in the drawing, and the parameter MeasuredValue is the distance between these points in millimeters.

Syntax	void CalibrationComplete(double MeasuredValue)	
Parameters	MeasuredValue	Measured distance in millimeters.
DispatchID	4	

2.3.4 *CounterComplete*

This event is called after an item counting has been completed.
The number of counted items is available in the Count parameter. The CounterLabel parameter will be the same text that was passed as parameter to **MarkupCounterSettings** when the counting started.

Syntax	void CounterCompleted(long Count, BSTR CounterLabel)	
Parameters	Count	Measured distance in millimeters.
	CounterLabel	A string describing what's being counted.
DispatchID	8	

2.3.5 *DblClick*

This event is called each time user has double clicked with the mouse.

Syntax	void DblClick()	
Parameters	None	
DispatchID	Stock property	

2.3.6 *DeleteElementConfirm*

This event is called before an element is about to be deleted.
You may show a dialog to the user where you ask him or her to confirm the deletion.

Syntax	void DeleteElementConfirm(long Handle, long *Action)	
Parameters	Handle	An unique identifier for the element to be deleted.
	Action	Tell scViewer how to proceed with this request, you may return one of the following values: 0. Do not delete element. The element will stay selected. 1. Delete the item and add it to the undo buffer. 2. Delete the item but do not add it to the undo buffer.
DispatchID	11	

2.3.7 *ElementDeleted*

This event is called each time a markup element is deleted.

Syntax	void ElementDeleted(long Handle)	
Parameters	Handle	An unique identifier for the deleted element.
DispatchID	5	

2.3.8 *ElementModified*

This event is called each time a markup element is modified.

Syntax	void ElementModified(long Handle, long Action)	
Parameters	Handle	An unique identifier for the modified element.
	Action	Describes how the element was modfiied: 0. Element moved or sized. 1. Line point edited.
DispatchID	12	

2.3.9 *ElementSelected*

This event is called each time a markup element is selected or deselected.

Syntax	void ElementDeleted(long Handle)	
Parameters	Handle	An unique identifier for the deleted element. If the value is 0 no element is selected.
DispatchID	16	

2.3.10 *ElementUndo*

This event is called each time a markup undo action has been performed.

Syntax	void ElementUndo (long Handle, long UndoAction)	
Parameters	Handle	An unique identifier the handled element.
	UndoAction	Describes how the element was modfiied: 0. New element undone (latest added element will be deleted). 1. Move of element undone. 2. Sizing of element undone. 3. Delete of element undone. 4. Any other change to element undone (like change of color, width and so on).
DispatchID	15	

2.3.11 *FileDragOpen*

This event is called when a file is opened using drag and drop on the control window.

Syntax	void FileDragOpen(BSTR FileName)	
Parameters	FileName	Name of file dragged and dropped on the control window.
DispatchID	20	

2.3.12 *FileLoadComplete*

This event is called when a file is completely loaded.

Syntax	void FileLoadComplete(void)	
Parameters	None	
DispatchID	14	

2.3.13 *FileOpenPassword*

This event is called when if a password is required to continue to load a file. PDF and DWF files may be protected by a password. You must handle this event to be able to provide the control with a password during file load. The event will be called 3 times, or until a correct password is provided.

Syntax	void FileOpenPassword(BSTR *Password)

Parameters	Password	Returned password.
DispatchID	9	

2.3.14 *HotspotClick*

This event is called when the user is pressing left mouse button while hovering over a hot spot. Hot spots can be present in CGM files. All coordinates are given in screen coordinates.

Syntax	void HotspotClick(long x, long y, BSTR ID, BSTR Tooltip)	
Parameters	x	Screen x coordinate for mouse cursor.
	y	Screen y coordinate for mouse cursor.
	ID	ID defined for this hotspot.
	Tooltip	A tooltip describing this hotspot.
DispatchID	22	

2.3.15 *HotspotHover*

This event is called when the cursor is hovering over a hot spot. Hot spots can be present in CGM files. All coordinates are given in screen coordinates.

Syntax	void HotspotHover(long x, long y, long x1, long y1, long x2, long y2, BSTR ID, BSTR Tooltip)	
Parameters	x	Screen x coordinate for mouse cursor.
	y	Screen y coordinate for mouse cursor.
	x1	Left coordinate for hotspot rectangle.
	y1	Top coordinate for hotspot rectangle.
	x2	Right coordinate for hotspot rectangle.
	y2	Bottom coordinate for hotspot rectangle.
	ID	ID defined for this hotspot.
	Tooltip	A tooltip describing this hotspot.
DispatchID	21	

2.3.16 *MarkupMouseHover*

This event is called when the mouse is hovering on top of a markup element.
You may pass the handle parameter to functions like **MarkupGetElementInfo** to obtain more information about the markup element.

Syntax	void MarkupMouseHover(long x, long y, long Handle)	
Parameters	x	Screen x coordinate for mouse cursor.
	Y	Screen y coordinate for mouse cursor.
	Handle	Unique markup element identifier.
DispatchID	10	

2.3.17 *MeasureComplete*

This event is called after a user measurement is completed. The MeasureType parameter will tell if this is an area or a distance measurement. All measurement values are world coordinates scaled using the active calibration.

Syntax	void MeasureComplete(long MesureType, double Length, double Area, double Perimeter)	
Parameters	MeasuredType	Type of measurment done: 1. Area Measurement (Mouse action type 10) 2. Distance Measurement multiple points (Mouse action type 12) 3. Distance Measurement two points (Mouse action type 9)
	Length	Measured length.
	Area	Measured area.
	Perimeter	Measured perimeter (only for area measurements).
DispatchID	6	

2.3.18 *MeasuredValue*

This event is called during measurement each time the mouse is moved. You may use this event to display the current measured value in for example the status bar, or as a tooltip.. The MeasureType parameter will tell if this is an area or a distance measurement. All measurement values are world coordinates scaled using the active calibration.

Syntax	void MeasuredValue(long MeasureType, double Length, double Area, double Perimeter)	
Parameters	MeasureType	Type of measurement used: 0. Area Measurement (Mouse action type 10) 1. Distance measurement defined by multiple points (Mouse action type 12) 2. Distance measurement between two points (Mouse action type 9)
	Length	Measured length.
	Area	Measured area.
	Perimeter	Measured perimeter (only for area measurements).
DispatchID	6	

2.3.19 *MouseDown*

This event is called each time user has pressed a mouse button.

Syntax	void MouseDown(short Button, short Shift, OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y)	
Parameters	Button	Mouse button indicator.
	Shift	Shift key indicator.
	X	Mouse x screen position.
	y	Mouse y screen position.
DispatchID	Stock property	

2.3.20 *MouseMove*

This event is called each time user has moved the mouse.

Syntax	void MouseMove (short Button, short Shift, OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y)	
Parameters	Button	Mouse button indicator.
	Shift	Shift key indicator.
	X	Mouse x screen position.
	Y	Mouse y screen position.
DispatchID	Stock property	

2.3.21 *MouseUp*

This event is called each time user has released a pressed mouse button.

Syntax	void MouseUp (short Button, short Shift, OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y)	
Parameters	Button	Mouse button indicator.
	Shift	Shift key indicator.
	X	Mouse x screen position.
	y	Mouse y screen position.
DispatchID	Stock property	

2.3.22 *NewElementCreated*

This event is called each time a new markup element is created.

Syntax	void NewElementCreated(long Handle)	
Parameters	Handle	An unique identifier for the new element.
DispatchID	3	

2.3.23 *OpenHyperlink*

This event is called each the user presses a hyperlink which points to a file, URL or a certificate.

Syntax	void OpenHyperlink(BSTR Location, LONG LocationType)	
Parameters	Location	Location to open, either a local file or an URL.
	LocationType	One of the following values: 1. Local file. 2. URL (http or https). 3. Hotspot 4. Signature. The Location parameter is a string with the certificate index (“0”, “1” and so on).
DispatchID	17	

2.3.24 *PageChanged*

This event is called when the control changes the active page. A page change may occur if the user presses hyperlink that points to another page inside the same document.

Syntax	void PageChanged(long NewActivePage)	
Parameters	NewActivePage	The new active page.
DispatchID	18	

2.3.25 *Progress*

This event is called during load time. This event allows the client to show a progress meter.

Syntax	void Progress(long Percent)	
Parameters	Percent	Progress given as percentage (0 to 100).
DispatchID	1	

2.3.26 *ZoomChanged*

This event is called each time scaling and pan parameters are changed.

Syntax	void ZoomChanged()	
Parameters	None	
DispatchID	13	

3 Appendixes

In the Appendix section you will find additional information about scViewerX.

3.1 Appendix A System Requirements

Supported Operating Systems

- Windows XP SP3, 32 and 64 bit.
- Windows Vista, 32 and 64 bit.
- Windows 7, 32 and 64 bit.
- Windows 8, 32 and 64 bit.
- Windows 8.1, 32 and 64 bit.
- Windows 10, 32 and 64 bit.
- Windows Server 2003
- Windows Server 2008
- Windows Server 2012
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022
- Supports Terminal Server and Citrix.

Minimum System Requirements

Display
1024 x 768

Processor
1 GHz

Memory
64-bit Windows: 4 GB
32-bit Windows: 2 GB

Hard Disk / SSD
250MB available space

3.2 Appendix B Supported File Formats

3.2.1 *Input Formats*

scViewerX can open the following formats:

The scViewerX control can open the following formats for viewing, annotation, printing and conversion:

Format Description	**Extension(s) used by Format**
Adobe PDF, PDF/A	PDF
Autodesk DWF 2D	DWF
Autodesk DWFx 2D	DWFx
Calcomp Plotter Format	PLT,907
CALS Type 1 CCITTG-4 Raster	CAL,CG4
Computer Graphics Metafile [1]	CGM
EDMICS C4 Tiled Raster	CG4, TG4
Excellon Drill Format	DRL
Gerber RS-274D, RS-274X, X2	GBR, GBX, and many others
Graphics Interchange Format	GIF
HEIC Image Format	HEIC
HPGL Plotter Format	PLT, HP, Many different
HPGL/2 Plotter Format	PLT, HP2, Many different
HP-RTL Plotter Format	PLT, Many different
Intergraph Raster Format (type 9, 24, 27 and 65)	CIT, TG4, COT, RLE, RGB
JPEG Image Format	JPG
JPEG 2000 Image Format	JP2, J2K, and others
KIPGL Plotter Format	PLT, HP2, and many others
PNG Image Format	PNG
TIFF Image Format [2]	TIF, TIFF
WebP Image Format	WEBP
Windows Bitmap Format	BMP

Notes:
1. CGM Binary and Clear Text encodings are supported.
2. Supported TIFF compression methods: Uncompressed, Packbits, LZW, Inflate, CCITT G3 and G4.

Optional formats

If the target system have Microsoft Office 2007 or newer installed you can in addition open the following formats for view, printing and conversion

Format Description	**Extension(s) used by Format**
Microsoft Word	DOC, DOCX. RTF, TXT
OpenDocument (handled by Word)	ODT
Microsoft Excel	XLS, XLSX
Microsoft PowerPoint	PPT,PPTX

From version 6.10 you may use LibreOffice version 5.x instead of Microsoft Office, to enable viewing, printing and conversion of Office formats.

3.2.2 *Output Formats*

scViewerX can convert to the following formats:

The scViewerX control can export the loaded document into the following output formats:

Format Description	**Default Extension**
Adobe PDF and PDF/A	PDF
Adobe Postscript	PS
Autodesk DWF 2D	DWF
Autodesk DXF	DXF
CALS Type 1 CCITTG-4 Raster [1]	CAL
Computer Graphics Metafile	CGM
Gerber RS-274X	GBX
HEIC Image Format	HEIC
HPGL/2 Plotter Format	PLT
HP-RTL Plotter Format	PLT
JPEG Image Format	JPG
JPEG 2000 Image Format	J2K
Paintbrush PCX Format	PCX
Portable Network Graphics (PNG)	PNG
Tagged Image File Format (TIFF) [2]	TIF
WebP Image Format	WEBP
Windows Bitmap Format	BMP
Windows Enhanced Metafile	EMF
Windows Metafile	WMF

Notes:
1. MIL-PRF-28002B Type 1.
2. Supported compression methods for TIFF includes: CCITT-G3, CCITT-G4, LZW, and Packbits.
3. No additional applications or drivers are needed for conversion.

3.3 Appendix C Redistributables

You will find all the files you need to redistribute with your application under the folder named **"C:\Users\Public\Documents\scViewerX SDK\redist"**.

If you have installed 32-bit SDK the redistributable are in a subfolder named x86.
If you have installed 64-bit SDK the redistributable are in a subfolder named x64.

None of the support DLL's requires registration on the target system, only the control, scviewerx.ocx, needs to be registered.
All DLL files need to be installed to the same folder as scviewerx.ocx.
Here is a List of all DLL's and their purpose:

Name	Description	Optional	Register
scviewerx.ocx	The ActiveX control	No	Yes
scCoder.dll	Contain functions for compression, decompression and barcode generation.	No	No
dynapdf.dll	PDF reader and writer	No	No
scrdCalcomp.dll	Calcomp plotter format reader	Yes	No
scrdCGM.dll	CGM importer (Computer Graphics Metafile)	Yes	No
scrdDRL.dll	Excellon drill format reader	Yes	No
scrwDWF.dll	Autodesk DWF and DWFx 2D reader	Yes	No
scrdDXF.dll	Autodesk DXF reader	Yes	No
scwrDWF6.dll	Autodesk DWF writer	Yes	No
scrwGBX.dll	Gerber file format reader (RS-274X / RS-274D)	Yes	No
scrwGIF.dll	GIF file format reader	Yes	No
scrdHEIC.dll	HEIC file format reader	Yes	No
scrdINGR.dll	Intergraph raster file format reader	Yes	No
scrwJPEG200.dll	JPEG 2000 reader and writer	Yes	No
scrwWEBP.dll	Google WebP image format reader and writer	Yes	No

If you want to view and convert PDF files you will need to install the PDF font resource files. These files are in a subfolder of the redistribution folder named “**Resource**”.
All files and subfolders to this folder should be installed as subfolder to the folder where you place your copy of scviewerx.ocx.

For example, if you install scviewerx to a folder named “c:\programfiles\scviewerx” you must create a subfolder to this folder which is named “c:\programfiles\scviewerx\Resource”.
This folder must be an exact copy of the “Resource” folder content.
Other files to include with your installation:

Name	Description	Optional
sRGB.icc	PDF/A output intent profile	No
ScViewerx.TextResources	Translation file	Yes
OfficeToPDF.exe	Microsoft Office formats to PDF converter. This	Yes

	converter requires Microsoft Office version 2007 or newer to be installed on the target system.	
scExcelToPDF.exe	Microsoft Excel format to PDF converter. This converter requires Microsoft Excel version 2007 or newer to be installed on the target system.	Yes
ScLibreToPDF.exe	LibreOffice formats to PDF converter. This conversion will only work if LibreOffice is installed on the target system.	Yes
scPPTToPDF.exe	Microsoft PowerPoint format to PDF converter. This converter requires Microsoft PowerPoint version 2007 or newer to be installed on the target system.	Yes
scWordToPDF.exe	Microsoft Word format to PDF converter. This converter requires Microsoft Word version 2007 or newer to be installed on the target system.	Yes
Heif.dll	Required if you want to read or write HEIC files.	Yes
Libx265.dll	Required if you want to read or write HEIC files.	Yes
Libde265.dll	Required if you want to read or write HEIC files.	Yes

3.4 Appendix D Menus

Context Menu

The context menu is displayed when the user clicks the right mouse button inside the control's window when a file is loaded.
Use the **EnableContextMenu** property to disable or enable this menu.
Here is a brief description of each item in the context menu:

Zoom All	Show the whole image inside the window.
Zoom In	Increase scaling factor by 50%.
Zoom Out	Decrease scaling factor by 50%.
Zoom In/Out	Zoom in and out mode. Move the mouse while pressing the left mouse button to zoom in or out.
Zoom Window	Select a region with the mouse that will fit the window.
Pan Hand	Move around the image by pressing left mouse button and moving the cursor.
Print	Print the image using current settings.
About	Shows information about the program.

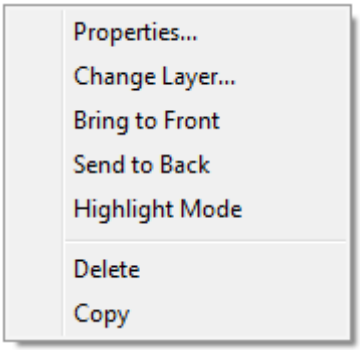
You can remove two of the command items in this menu:

Print	Set the **EnableContextMenuPrint** property to FALSE to hide this entry from the menu.
About...	Set the EnableAboutBox property to FALSE to hide this entry.

Markup Context Menus

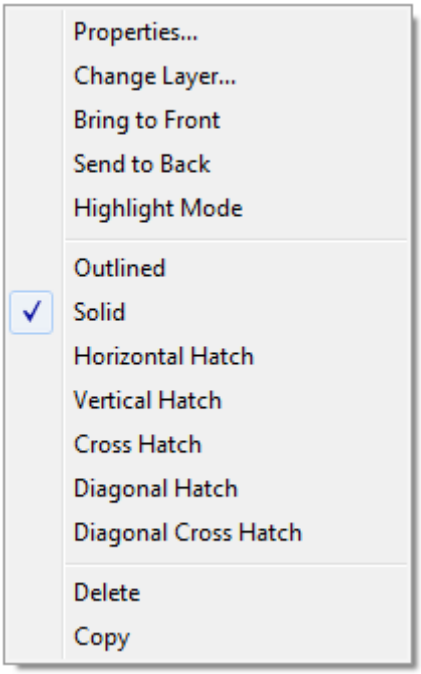
There are two variant of markup context menus, depending on the selected element type. If you right click a selected line, polyline, freehand, arrow, note, text, stamp, dimension line or a measurement path the following context menu will appear:

Properties	Open the markup property dialog.
Change Layer	Display a dialog that will allow you to change the layer for the element.
Bring To Front	Move the markup element to top, which make sure it's draw on top of others.
Send to Back	Send the markup element to back , which make sure it's draw below of others.
Highlight Mode	Toggle transparency on/off.
Delete	Delete this element.
Copy	Copy this element to clipboard.



If you right click a selected polygon, rectangle, circle, ellipse, revision cloud or picture the following context menu will appear:

Properties	Open the markup property dialog.
Change Layer	Display a dialog that will allow you to change the layer for the element.
Bring To Front	Move the markup element to top, which make sure it's draw on top of others.
Send to Back	Send the markup element to back , which make sure it's draw below of others.
Highlight Mode	Toggle transparency on/off.
Outlined	Set fill type to outlined.
Solid	Set fill type to solid.
Horizontal Hatch	Set fill type to horizontal hatch.
Vertical Hatch	Set fill type to vertical hatch.
Cross Hatch	Set fill type to cross hatch.
Diagonal Hatch	Set fill type to diagonal hatch.
Diagonal Cross Hatch	Set fill type to diagonal cross hatch.
Delete	Delete this element.
Copy	Copy this element to clipboard.



You may use the **EnableMarkupContextMenu** property to disable or enable these menus when the user right clicks a selected markup element.

3.5 Appendix E Keyboard Shortcuts

Keys for general navigation

Function	Shortcut
Go to next page	Page Down
Go to previous page	Page Up
Go to top of first page	Home
Go to bottom of last page	End
Go back to previous view after following a link.	Backspace
Zoom In	+
Zoom out	-

Keys used during markup drawing and measurement

Function	Shortcut
Toggle line edit mode during editing	Press ALT when selecting the element
Regret last added point for polyline and polygons	Backspace
End a counter session	ESC or ENTER
Finish a polyline or polygon	ENTER
End measurement and show result	ENTER
Toggle orthogonal mode	Press SHIFT while drawing

Keys used during markup editing

Function	Shortcut
End edit of selected element	ESC
Rotate selected element	'R'
Delete selected element	'D' or DELETE

3.6 Appendix F Markup XML Format

Software Companions XML Markup Format Description

The Software Companions XML markup format can be easily created by a third-party application or by using a text editor. By using this format you can create your own markup files from scratch. Most of the different markup element types available in scViewerX, can be created from a XML file. A XML formatted markup file may be added calling the **AddFromFile** method. It's also possible to load XML markup data by using the **MarkupCreateFromXML** method. The default extension for a Software Companions XML Markup file is .scmx, but .xml may also be used.

File Format Specification

The file must start with the following header:

```
<?xml version="1.0" encoding="UTF-8"?>
<SCMarkupFormat>
```

Only UTF-8 encoding is currently supported.
The header and elements section must then follow. A layers section may optionally be included.
A complete xml markup file example is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<SCMarkupFormat>
<Header>
   <Unit>mm</Unit>
</Header>
<Layers>
   <layer name="Markup Layer 1" color="#0000FF" enabled='1'/>
   <layer name="Markup Layer Two" color="#00ff00" enabled='1'/>
</Layers>
<Elements>
   <Element>
      <type>Text</type>
      <layer>0</layer>
      <page>0</page>
      <center x="50%" y="50"/>
      <user>Peter</user>
      <color>#FF0000</color>
      <text>This is a text</text>
      <rotation>45</rotation>
      <font height='10' facename='Times New Roman'/>
   </Element>
</Elements>
</SCMarkupFormat>
```

This xml markup file defines a single text element. The text center position is at 50% of document width and 50& of document height.

The markup coordinate system origin is always the lower left corner of the document. The coordinates can be given in millimeters, inches or native (1016 DPI) values. All color values are coded as red, green and blue intensities in hexadecimal notation (HTML standard - #RRGGBB). The Layers section defines two layers. Please see the different section descriptions below for more information.

Header Section

This section contains settings that are common for all markup elements.
The supported entries in this section are:

Unit

Accepted values:

Value	Description
mm	Coordinates and sizes are given in millimeters.
inch	Coordinates and sizes are given in inches.
mil	Coordinates and sizes are given in 1/1000 inch.
native	Coordinates and sizes are given in native coordinates (1016 DPI).

Layers Section

This section contains layer definitions. Each layer definition can have three different attributes.
The first defined layer will have index 0, the next one 1, etc. The layer index is used when elements are defined.

The description of each attribute is given below:

Attribute Name	Description
Name	Name to use for the layer.
Color	Default color to use for elements placed on this layer.
Enabled	Enter a value different from 0 to make the layer visible.

Sample layer entry:

<Layer name="Markup Layer 1" color="#0000FF" enabled='1'/>

This entry will create a layer that is named "Markup Layer 1", with default color blue and it will be visible.

Elements Section

This section contains one or more Element entries.

```
<Elements>
   <Element>
       //data for first element
   </Element>
   <Element>
       //data for second element
   </Element>
   <Element>
       //data for third element
   </Element>
</Elements>
```

3.6.1 Element Section Keywords

Each element entry defines a new markup element. Some of the entry keywords are common for all element types, and some are type specific.
The following table lists the keywords that are common for all elements:

Keyword	**Description**
type	Markup element type keyword. The following element types are available: **Arrow** **Barcode** **Circle** **Ellipse** **Erase Polygonal Area** **Erase Rectangular Area** **Eraser** **Line** **Picture** **Polygon** **Polyline** **Rectangle** **Revision Cloud** **Rounded Rectangle** **Rubber Stamp** **Shape** **Symbol** **Text**
handle	A unique element numerical ID to be applied to the element. If not provided the control will allocate an ID.
hyperlink	Link to a document or an URL.
layer	Layer index to use for the markup element. If not provided the default layer will be used.
locked	Set the element lock flag. A locked element (non-zero value) cannot be moved or modified.
page	Page index where the markup element will be placed. Page indexes are zero based. If not provided page 0 is assumed. You can set page to -1 to force the markup element to be displayed on all pages.
transparent	If defined, the markup element will be transparent.
user	Name of the user that have created the element. If not provided the Windows user name will be applied to the markups.

3.6.2 Arrow Element

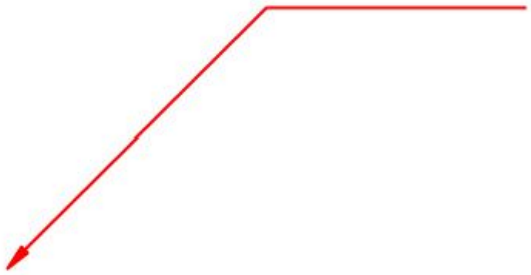
The arrow element requires the following keywords, in addition to the common keywords:

Keyword	Description
points	This section contains one point entry for each point in the arrow polyline. Each point is defined with an x and y attribute. See the xml definition below for an example on how to describe point entries.
color	Define draw color to use for the element.
arrowhead	Define type of arrow header to use. Supported arrow types: 0: open, 1: closed, 2: filled.
arrowsize	Define size of arrow header.
linewidth	Define width of the arrow lines. The width is defined using the active unit. This entry is optional.

Sample arrow element xml definition:

```
<element>
   <type>Arrow</type>
   <layer>0</layer>
   <page>0</page>
   <color>#FF0000</color>
   <arrowhead>2</arrowhead>
   <arrowsize>5</arrowsize>
   <linewidth>0.3</linewidth>
   <points>
      <point x='100' y='100'/>
      <point x='150' y='150'/>
      <point x='200' y='150'/>
   </points>
</element>
```

The xml data to the left will create the following arrow element:



3.6.3 Barcode Element

The barcode element requires the following keywords, in addition to the common keywords:

Keyword	**Description**
center	Barcode center coordinates. The center coordinate is defined by setting x and y to a percentage of the document extents, or absolute values. Sample usages: * * Add the barcode centered at document center. * * Add the barcode horizontally centered and 40 millimeters above the bottom of the page.
size	Set the dimensions for the barcode in percentage of current document extents, or absolute values. The extents are defined using two attributes named w and h. Sample definitions: *<size w="10%" h="10%"/>* Create a barcode with width equal to 10% of document page width, and the height will be 10% of document page height. *<size w="40" h="40"/>* Create a barcode with width and height set to 40 units.
barcodetype	The type of barcode encoder to use. The following barcode standards are supported: • aztec • datamatrix • code128a • code128b • code128c • code39 • code93 • ean13 • pdf417 • qr
text	The text to encode using the selected barcode type.
barcodeincludetext	Include the actual text as part of the barcode image.
Barcodelabel	Optional label text added to the barcode image.

Sample barcode element xml definition:

```
<element>
   <type>Barcode</type>
   <barcodetype>QR</barcodetype>
   <layer>0</layer>
   <page>0</page>
   <center x="50%" y="50"/>
   <size w="40" h="40"/>
   <text>This is a text</text>
</element>
```

The xml data to the left will create the following barcode element:

3.6.4 Circle Element

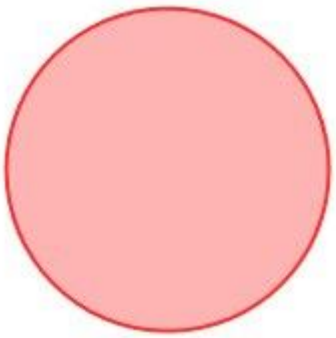
The circle element requires the following keywords, in addition to the common keywords:

Keyword	Description
center	Circle center coordinates. The center coordinate is defined by setting x and y to a percentage of the document extents, or absolute values. Sample usages: Add the circle centered at document center. Add the circle with center 50 millimeters from the left and 40 millimeters above the bottom of the page.
radius	The radius of circle in active the unit.
fillcolor	Define the fill color to use for the element.
color	Define the outline color to use for the element.
linewidth	Define the width of the outline. The width is defined using the active unit. This entry is optional.
linestyle	Define line style to use for the outline. This entry is optional.
fillstyle	Define fill style to use for the element. This entry is optional.

Sample circle element xml definition:

```
<element>
   <type>Circle</type>
   <center x=”50%” y=”50”/>
   <radius>50</radius>
   <layer>0</layer>
   <page>0</page>
   <color>#FF0000</color>
   <fillcolor>#FFA0A0</fillcolor>
   <fillstyle>1</fillstyle>
   <linewidth>1.0</linewidth>
   <transparent/>
</element>
```

The xml definition to the left will create the following circle:



3.6.5 Erase Polygonal Area Element

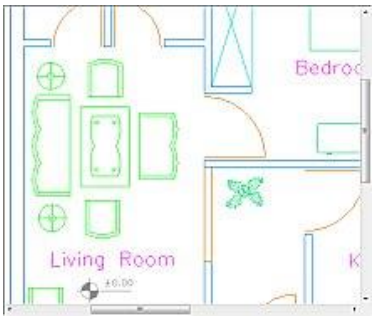
Define an area to hide (erase) using a polygon definition. This feature is also known as wipeout. The erase polygonal area element requires the following keywords, in addition to the common keywords:

Keyword	**Description**
points	This section contains one point entry for each point in the polygon. Each point is defined with an x and y attribute. See the xml definition below for an example on how to describe point entries.

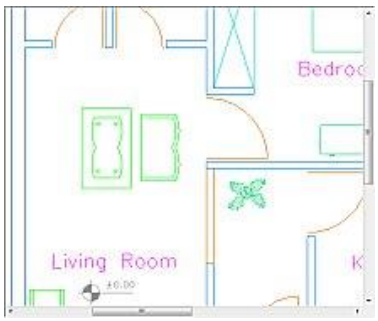
Sample erase polygonal area element xml definition:

```
<element>
   <type>Erase Polygonal Area</type>
   <layer>0</layer>
   <page>0</page>
   <points>
      <point x='150' y='190'/>
      <point x='150' y='290'/>
      <point x='200' y='290'/>
      <point x='200' y='265'/>
      <point x='175' y='265'/>
      <point x='175' y='215'/>
      <point x='200' y='215'/>
      <point x='200' y='190'/>
      <point x='150' y='190'/>
   </points>
</element>
```

The xml data to the left will erase the polygonal area as shown below when used for sample file *compare_revA.plt*:



Before xml markup is loaded



After xml markup is loaded.

3.6.6 Erase Rectangular Area Element

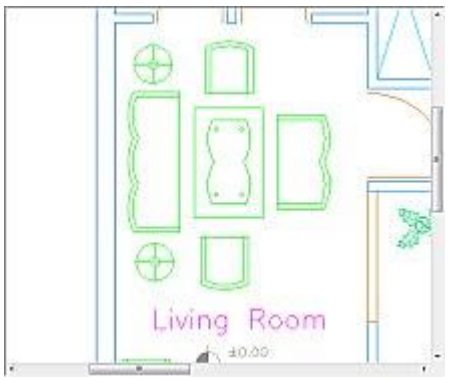
Define an area to hide (erase) using a rectangle. This feature is also known as wipeout. The erase rectangular area element requires the following keywords, in addition to the common keywords:

Keyword	**Description**
Boundary	Define boundary for the created rectangle. The boundary is defined using four attributes. These attributes are named: x1, y1, x2 and y2. Sample boundary entry: *<boundary x1='150' x2='190' y1='230' y2='290'/>*

Sample erase rectangular area element xml definition:

```
<element>
<type>Erase Rectangular Area</type>
   <layer>0</layer>
   <page>0</page>
   <boundary x1='150' y1='190'
    x2='230' y2='290'/>
</element>
```

The xml data to the left will erase the rectangular area as shown below when used for sample file *compare_revA.plt*:



Before xml markup is loaded.

After xml markup is loaded.

.

3.6.7 Eraser Element

Define a polyline that will hide drawing contents. The polyline defined will be drawn using the current background color and with the width defined by the

linewidth setting. The eraser element requires the following keywords, in addition to the common keywords:

Keyword	Description
points	This section contains one point entry for each point in the polyline. Each point is defined with an x and y attribute. See the xml definition below for an example on how to describe point entries.

Sample eraser element xml definition:

```
<element>
   <type>Eraser</type>
   <layer>0</layer>
   <page>0</page>
   <linewidth>5.0</linewidth>
   <points>
      <point x='150' y='190'/>
      <point x='150' y='290'/>
      <point x='200' y='290'/>
      <point x='200' y='265'/>
      <point x='175' y='265'/>
      <point x='175' y='215'/>
      <point x='200' y='215'/>
      <point x='200' y='190'/>
   </points>
</element>
```

3.6.8 Line Element

The line element uses the following keywords, in addition to the common keywords:

Keyword	Description
boundary	Define boundary for the created line. The boundary is defined using four attributes. These attributes are named: x1, y1, x2 and y2. Sample boundary entry: *<boundary x1='80' x2='180' y1='10' y2='110'/>*. This will draw a line will between x1,y1 and x2,y2.
color	Define line color to use for the element.
linewidth	Define the width of the line in active units. This entry is optional.
linestyle	Define line style to use for the line. This entry is optional.

Sample line element xml definition:

```
<element>
   <type>Line</type>
   <layer>0</layer>
   <page>0</page>
   <boundary x1='80' x2='180' y1='10' y2='110'/>
   <color>#00FF00</color>
   <linewidth>1.0</linewidth>
   <linestyle>0</linestyle>
</element>
```

3.6.9 Picture Element

The picture element requires the following keywords, in addition to the common keywords:

Keyword	Description
center	Picture center coordinates. The center coordinate is defined by setting x and y to a percentage of the document extents, or absolute values. Sample usages: * * Add the picture centered at document center. * * Add the picture horizontally centered and 40 millimeters above the bottom of the page.
filename	Full path to the image file that will be added as markup. Please note that either double back slash (\\), or a single slash (/) must be used. If you skip the directory part and just enter the filename, the control will search for the image file in the same folder as the markup file.
imagerotation	Rotation in degrees that should be applied to the picture. The following rotation values are valid: 0, 90, 180 and 270.
size	Set the dimensions for the picture in percentage of current document extents, or absolute values. The extents are defined using two attributes named w and h. Sample definitions: *<size w="10%" h="10%"/>* Add the picture with width equal to 10% of document page width, and the height will be 10% of document page height. *<size w="40" h="40"/>* Add the picture with width and height set to 40 units. If you do not define size the actual picture dimension will be used, based on the image file dpi settings.

Sample picture element xml definition:

```
<element>
   <type>Picture</type>
   <layer>0</layer>
   <page>0</page>
   <center x="50%" y="50%"/>
   <size w="100" h="30"/>
   <filename>sclogo.png</filename>
</element>
```

The xml data to the left will add the following picture:

3.6.10 Polygon Element

The polygon element uses the following keywords, in addition to the common keywords:

Keyword	Description
points	This section contains one point entry for each point in the polygon. Each point is defined with an x and y attribute. Please see the polygon element definition below for an example on how to describe point entries.
color	Define line color to use for the element.
fillcolor	Define fill color to use for the element.
color	Define outline color to use for the element.
linewidth	Define width of the outline. The width is defined using the active unit. This entry is optional.
linestyle	Define line style to use for the outline. This entry is optional.
fillstyle	Define fill style to use for the element. This entry is optional.

Sample polygon element xml definition:

```
<element>
   <type>Polygon</type>
   <layer>0</layer>
   <page>0</page>
   <user>Bill</user>
   <color>#FF0000</color>
   <fillcolor>#CCFFCC</fillcolor>
   <linewidth>0</linewidth>
   <fillstyle>1</fillstyle>
      <points>
         <point x='10' y='10'/>
         <point x='20' y='20'/>
         <point x='30' y='10'/>
         <point x='20' y='0'/>
      </points>
</element>
```

The xml data to the left will create the following 4-point polygon:

3.6.11 Polyline Element

The polyline element uses the following keywords, in addition to the common keywords:

Keyword	Description
points	This section contains one point entry for each point in the polyline.

	Each point is defined with an x and y attribute. See the polyline element definition below for an example on how to describe point entries.
color	Define line color to use for the element.
fillcolor	Define fill color to use for the element.
color	Define outline color to use for the element.
linewidth	Define width of the polyline. The width is defined using the active unit. This entry is optional.
linestyle	Define line style to use for the outline. This entry is optional.
fillstyle	Define fill style to use for the element. This entry is optional.

Sample polyline element xml definition:

```
<element>
   <type>Polyline</type>
   <layer>0</layer>
   <page>0</page>
   <color>#FF0000</color>
   <linewidth>0.5</linewidth>
      <points>
         <point x='10' y='10'/>
         <point x='15' y='15'/>
         <point x='20' y='10'/>
         <point x='25' y='15'/>
      </points>
</element>
```

The xml data to the left will create the following 4-point polyline:

3.6.12 Rectangle Element

The rectangle element uses the following keywords, in addition to the common keywords:

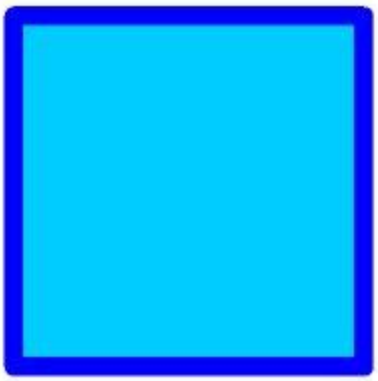
Keyword	**Description**
boundary	Define boundary for the created rectangle. The boundary is defined using four attributes. These attributes are named: x1, y1, x2 and y2.

	Sample boundary entry: *<boundary x1='80' x2='180' y1='10' y2='110'/>*
fillcolor	Define fill color to use for the element.
color	Define outline color to use for the element.
linewidth	Define width of the outline. The width is defined using the active unit. This entry is optional.
linestyle	Define line style to use for the outline. This entry is optional.
fillstyle	Define fill style to use for the element. This entry is optional.

Sample rectangle element xml definition:

```
<element>
   <type>Rectangle</type>
   <layer>0</layer>
   <page>0</page>
   <boundary x1='10' x2='90' y1='10' y2='90'/>
   <color>#0000FF</color>
   <fillcolor>#00CCFF</fillcolor>
   <fillstyle>1</fillstyle>
   <linewidth>0</linewidth>
   <transparent/>
</element>
```

The xml data to the left will create the following rectangle:



3.6.13 Revision Cloud Element

The revision cloud element uses the following keywords, in addition to the common keywords:

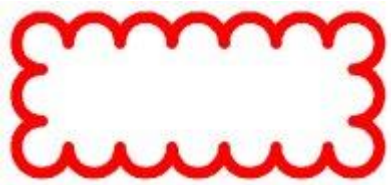
Keyword	**Description**
boundary	Define boundary for the created revision cloud. The boundary is defined using four attributes. These attributes are named: x1, y1, x2 and y2. Sample boundary entry: *<boundary x1='80' x2='180' y1='10' y2='110'/>*

fillcolor	Define fill color to use for the element.
color	Define outline color to use for the element.
linewidth	Define width of the outline. The width is defined using the active unit. This entry is optional.
linestyle	Define line style to use for the outline. This entry is optional.
fillstyle	Define fill style to use for the element. This entry is optional.
Arclimit	Set the maximum radius for the arcs in the revision cloud. The radius is given in current units. This entry is optional.

Sample revision cloud element definition:

```
<element>
   <type>Revision Cloud</type>
   <layer>0</layer>
   <page>0</page>
   <boundary x1='10' x2='110' y1='10' y2='50'/>
   <color>#FF00FF</color>
   <fillstyle>0</fillstyle>
   <linewidth>3</linewidth>
   <transparent/>
</element>
```

The xml data to the left will create the following revision cloud:



3.6.14 Rounded Rectangle Element

The rounded rectangle element uses the following keywords, in addition to the common keywords:

Keyword	**Description**
boundary	Define boundary for the created rounded rectangle. The boundary is defined using four attributes. These attributes are named: x1, y1, x2 and y2. Sample boundary entry: *<boundary x1='80' x2='180' y1='10' y2='110'/>*
fillcolor	Define fill color to use for the element.

color	Define outline color to use for the element.
linewidth	Define width of the outline. The width is defined using the active unit. This entry is optional.
linestyle	Define line style to use for the outline. This entry is optional.
fillstyle	Define fill style to use for the element. This entry is optional.

Sample rectangle element xml definition:

```
<element>
   <type>Rounded Rectangle</type>
   <layer>0</layer>
   <page>0</page>
   <boundary x1='50' x2='90' y1='50' y2='60'/>
   <color>#FF0000</color>
   <fillcolor>#CCCCCC</fillcolor>
   <fillstyle>1</fillstyle>
   <linewidth>2</linewidth>
   <transparent/>
</element>
```

The xml data to the left will create the following rectangle:

3.6.15 Rubber Stamp Element

The rubber stamp element uses the following keywords, in addition to the common keywords:

Keyword	Description
center	Define center point for the rubber stamp in percentage of document extents, or using absolute coordinates. The center point is defined using two attributes and these are named x and y. Sample definitions: * * This will place the stamp centered on the document page. * * This will place the stamp with center 100 millimeters from the left and 100 millimeters above the bottom of the page.
size	Set the extents for the rubber stamp in percentage of current document page extents, or absolute values. The extents are defined using two attributes named w and h. Sample definition: *<size w="20%" h="10%"/>* This will create a stamp with width equal to 20% of document page width, and the height will be 10% of document page height. *<size w="80" h="20"/>* This will create a stamp with width of 80 units and height of 20 units.
rotation	Define stamp rotation in degrees.
backcolor	Define stamp background color.
textcolor	Define color to use for the stamp text.
text	Stamp text.
font	Define font to use for the stamp text. The font is defined using several different attributes, each of them correspond to the Windows LOGFONT definition. The following font attributes are supported: height - Same as LOGFONT lfHeight. This value will recalculated.. orientation - Same as LOGFONT lfOrientation. weight - Same as LOGFONT lfWeight. italic - Same as LOGFONT lfItalic. underline - Same as LOGFONT lfUnderline. strikeout - Same as LOGFONT lfStrikeout. charset - Same as LOGFONT lfCharset. outprecision - Same as LOGFONT lfOutPrecision. clipprecision - Same as LOGFONT lfClipPrecision. quality - Same as LOGFONT lfQuality. pitch - Same as LOGFONT lfPitchAndFamily. facename - Same as LOGFONT lfFaceName. Sample font definition: *<font italic='1' facename='Times New Roman'/>*
linewidth	Define width of the outline. The width is defined using the active unit. This entry is optional.

The XML description below will add a stamp with its center at document center (50% of the document width, and 50% of the document height).
The stamps width will be 20% of document width, and the height will be 10% of the document height.
The stamp rotation is set to 0 degrees.

Sample rubber stamp element definition:

```
<element>
   <type>Rubber Stamp</type>
   <layer>0</layer>
   <page>0</page>
   <center x="50%" y="50%"/>
   <size w="20%" h="10%"/>
   <text>Draft</text>
   <rotation>0</rotation>
   <textcolor>#FF0000</textcolor>
   <backcolor>#FFCCCC</backcolor>
   <font italic='0'
      facename='Times New Roman'/>
   <transparent/>
</element>
```

The xml data to the left will create the following rubber stamp:

3.6.16 Shape Element

The shape element requires the following keywords, in addition to the common keywords:

Keyword	**Description**
center	Shape center coordinates. The center coordinate is defined by setting x and y to a percentage of the document extents, or by using absolute values. Sample usage: - add the shape centered at document center. - add the shape 50 millimeters from the left and 40 millimeters above the bottom of the document, if millimeter is the active unit.
shapetype	One of the predefined shape types: 0. Approved/Checkmark 1. Rejected
shapesize	Size of the shape element in current units.

Sample symbol element XML definition:

```
<element>
   <type>Shape</type>
   <layer>0</layer>
   <page>0</page>
   <center x=”50%” y=”50%”/>
   <shapetype>0</shapetype>
   <shapesize>10</shapesize>
</element>
```

The xml data to the left will add the following shape to the file:

3.6.17 Symbol Element

The symbol element requires the following keywords, in addition to the common keywords:

Keyword	**Description**
boundary	Define boundary for the created symbol. The boundary is defined using four attributes. These attributes are named: x1, y1, x2 and y2. Sample boundary entry: *<boundary x1='100' x2='150' y1='200' y2='250'/>*
symbollibrary	The name of the symbol library to use.
symbolname	The name of the symbol to use.

Sample symbol element XML definition:

```
<element>
   <type>Symbol</type>
   <layer>0</layer>
   <page>0</page>
   <boundary x1='100' x2='150' y1='100' y2='150'/>
   <symbollibrary>Sample</symbollibrary>
   <symbolname>Copyright</symbolname>
</element>
```

The xml data to the left will add the following symbol to the file:

3.6.18 Text Element

The text element requires the following keywords, in addition to the common keywords:

Keyword	**Description**
center	Text center coordinate. Text center coordinate is defined by setting x and y to a percentage of the document extents, or absolute coordinates. Sample usage: * * This will add the text centered at document center. * * This will add the 100 millimeters from the left and 20 millimeters above the bottom of the page.
insertx	Text insert x coordinate. This value is the absolute coordinate for the horizontal text origin. You may either use text center as described above or use insertx together with inserty to define the text origin.
inserty	Text insert y coordinate. This value is the absolute coordinate for the vertical text origin. You may either use text center as described above or use inserty together with insertx to define the text origin.
alignment	The alignment setting defines the origin of the text to insert. Please note that this setting can only be used together with the insertx and insert keywords. The following values are available: 0. Origin is center of the text. 1. Origin is top left of the text. 2. Origin is bottom left of the text. 3. Origin is bottom right of the text. 4. Origin is top right of the text.
rotation	The text rotation in degrees.
textcolor	Define color to use for the text.
text	The text string to display.
font	Define font to use for the text. The font is defined using several different attributes, each of them correspond to the Windows LOGFONT defintition. The following font attributes are supported: height - Same as LOGFONT lfHeight. width - Samer as LOGFONT lfWidth. orientation - Same as LOGFONT lfOrientation. weight - Same as LOGFONT lfWeight. italic - Same as LOGFONT lfItalic. underline - Same as LOGFONT lfUnderline. strikeout - Same as LOGFONT lfStrikeout. charset - Same as LOGFONT lfCharset. outprecision - Same as LOGFONT lfOutPrecision. clipprecision - Same as LOGFONT lfClipPrecision. quality - Same as LOGFONT lfQuality. pitch - Same as LOGFONT lfPitchAndFamily. facename - Same as LOGFONT lfFaceName. Sample font defintion: *<font italic='1' height='10' facename='Times New Roman'/>*

Sample text element definition:

```
<element>
   <type>Text</type>
   <layer>0</layer>
   <page>0</page>
   <center x=”50%” y=”50%”/>
   <textcolor>#FF0000</textcolor>
   <text>This is a text</text>
   <rotation>45</rotation>
   <font height='10'
    facename='Times New Roman'/>
</element>
```

The xml data to the left will create the following text element:

This description will add a red text element with its center at document center and rotated 45 degrees. The text height is set to be 10 units.

3.6.19 Line and Fill Style Settings Used for Element Definitions

Linestyle

The linestyle can be defined using one of the following values:
0 Solid line.
1 Dashed line.
2 Dotted line.
3 Dash-dot line.
4 Dash-dot-dot line.

Fillstyle

The fillstyle can have one the following values:
0 Outlined (no fill).
1 Solid fill.
2 Horizontal hatch.
3 Vertical hatch.
4 Crosshatched.
5 Diagonal hatch.
6 Diagonal cross hatching.

3.7 Appendix G Paper Sizes

The **SetPrintPaperSize** method accepts following predefined paper format constants:

Type	Value	Meaning
DMPAPER_LETTER	1	US Letter 8 1/2 x 11 in
DMPAPER_LETTERSMALL	2	US Letter Small 8 1/2 x 11 in
DMPAPER_TABLOID	3	US Tabloid 11 x 17 in
DMPAPER_LEDGER	4	US Ledger 17 x 11 in
DMPAPER_LEGAL	5	US Legal 8 1/2 x 14 in
DMPAPER_STATEMENT	6	US Statement 5 1/2 x 8 1/2 in
DMPAPER_EXECUTIVE	7	US Executive 7 1/4 x 10 1/2 in
DMPAPER_A3	8	A3 297 x 420 mm
DMPAPER_A4	9	A4 210 x 297 mm
DMPAPER_A4SMALL	10	A4 Small 210 x 297 mm
DMPAPER_A5	11	A5 148 x 210 mm
DMPAPER_B4	12	B4 (JIS) 257 x 364 mm
DMPAPER_B5	13	B5 (JIS) 182 x 257 mm
DMPAPER_FOLIO	14	Folio 8 1/2 x 13 in
DMPAPER_QUARTO	15	Quarto 215 x 275 mm
DMPAPER_10X14	16	10 x 14 in
DMPAPER_11X17	17	11 x 17 in
DMPAPER_NOTE	18	US Note 8 1/2 x 11 in
DMPAPER_ENV_9	19	US Envelope #9 3 7/8 x 8 7/8
DMPAPER_ENV_10	20	US Envelope #10 4 1/8 x 9 1/2
DMPAPER_ENV_11	21	US Envelope #11 4 1/2 x 10 3/8
DMPAPER_ENV_12	22	US Envelope #12 4 3/4 x 11 in
DMPAPER_ENV_14	23	US Envelope #14 5 x 11 1/2
DMPAPER_CSHEET	24	C size sheet
DMPAPER_DSHEET	25	D size sheet
DMPAPER_ESHEET	26	E size sheet
DMPAPER_ENV_DL	27	Envelope DL 110 x 220mm
DMPAPER_ENV_C5	28	Envelope C5 162 x 229 mm
DMPAPER_ENV_C3	29	Envelope C3 324 x 458 mm
DMPAPER_ENV_C4	30	Envelope C4 229 x 324 mm
DMPAPER_ENV_C6	31	Envelope C6 114 x 162 mm
DMPAPER_ENV_C65	32	Envelope C65 114 x 229 mm
DMPAPER_ENV_B4	33	Envelope B4 250 x 353 mm
DMPAPER_ENV_B5	34	Envelope B5 176 x 250 mm
DMPAPER_ENV_B6	35	Envelope B6 176 x 125 mm
DMPAPER_ENV_ITALY	36	Envelope 110 x 230 mm

DMPAPER_ENV_MONARCH	37	US Envelope Monarch 3.875 x 7.5 in
DMPAPER_ENV_PERSONAL	38	6 3/4 US Envelope 3 5/8 x 6 1/2 in
DMPAPER_FANFOLD_US	39	US Std Fanfold 14 7/8 x 11 in
DMPAPER_FANFOLD_STD_GERMAN	40	German Std Fanfold 8 1/2 x 12 in
DMPAPER_FANFOLD_LGL_GERMAN	41	German Legal Fanfold 8 1/2 x 13 in
DMPAPER_ISO_B4	42	B4 (ISO) 250 x 353 mm
DMPAPER_JAPANESE_POSTCARD	43	Japanese Postcard 100 x 148 mm
DMPAPER_9X11	44	9 x 11 in
DMPAPER_10X11	45	10 x 11 in
DMPAPER_15X11	46	15 x 11 in
DMPAPER_ENV_INVITE	47	Envelope Invite 220 x 220 mm
DMPAPER_LETTER_EXTRA	50	US Letter Extra 9 1/2 x 12 in
DMPAPER_LEGAL_EXTRA	51	US Legal Extra 9 1/2 x 15 in
DMPAPER_TABLOID_EXTRA	52	US Tabloid Extra 11.69 x 18 in
DMPAPER_A4_EXTRA	53	A4 Extra 9.27 x 12.69 in
DMPAPER_LETTER_TRANSVERSE	54	Letter Transverse 8 1/2 x 11 in
DMPAPER_A4_TRANSVERSE	55	A4 Transverse 210 x 297 mm
DMPAPER_LETTER_EXTRA_TRANSVERSE	56	Letter Extra Transverse 9 1/2 x 12 in
DMPAPER_A_PLUS	57	SuperA/SuperA/A4 227 x 356 mm
DMPAPER_B_PLUS	58	SuperB/SuperB/A3 305 x 487 mm
DMPAPER_LETTER_PLUS	59	US Letter Plus 8.5 x 12.69 in
DMPAPER_A4_PLUS	60	A4 Plus 210 x 330 mm
DMPAPER_A5_TRANSVERSE	61	A5 Transverse 148 x 210 mm
DMPAPER_B5_TRANSVERSE	62	B5 (JIS) Transverse 182 x 257 mm
DMPAPER_A3_EXTRA	63	A3 Extra 322 x 445 mm
DMPAPER_A5_EXTRA	64	A5 Extra 174 x 235 mm
DMPAPER_B5_EXTRA	65	B5 (ISO) Extra 201 x 276 mm
DMPAPER_A2	66	A2 420 x 594 mm
DMPAPER_A3_TRANSVERSE	67	A3 Transverse 297 x 420 mm
DMPAPER_A3_EXTRA_TRANSVERSE	68	A3 Extra Transverse 322 x 445 mm
DMPAPER_DBL_JAPANESE_POSTCARD	69	Japanese Double Postcard 200 x 148 mm
DMPAPER_A6	70	A6 105 x 148 mm
DMPAPER_JENV_KAKU2	71	Japanese Envelope Kaku #2
DMPAPER_JENV_KAKU3	72	Japanese Envelope Kaku #3
DMPAPER_JENV_CHOU3	73	Japanese Envelope Chou #3
DMPAPER_JENV_CHOU4	74	Japanese Envelope Chou #4
DMPAPER_LETTER_ROTATED	75	Letter Rotated 11 x 8 1/2 11 in
DMPAPER_A3_ROTATED	76	A3 Rotated 420 x 297 mm
DMPAPER_A4_ROTATED	77	A4 Rotated 297 x 210 mm
DMPAPER_A5_ROTATED	78	A5 Rotated 210 x 148 mm

DMPAPER_B4_JIS_ROTATED	79	B4 (JIS) Rotated 364 x 257 mm
DMPAPER_B5_JIS_ROTATED	80	B5 (JIS) Rotated 257 x 182 mm
DMPAPER_JAPANESE_POSTCARD_ROTATED	81	Japanese Postcard Rotated 148 x 100 mm
DMPAPER_DBL_JAPANESE_POSTCARD_ROTATED	82	Double Japanese Postcard Rotated 148 x 200 mm
DMPAPER_A6_ROTATED	83	A6 Rotated 148 x 105 mm
DMPAPER_JENV_KAKU2_ROTATED	84	Japanese Envelope Kaku #2 Rotated
DMPAPER_JENV_KAKU3_ROTATED	85	Japanese Envelope Kaku #3 Rotated
DMPAPER_JENV_CHOU3_ROTATED	86	Japanese Envelope Chou #3 Rotated
DMPAPER_JENV_CHOU4_ROTATED	87	Japanese Envelope Chou #4 Rotated
DMPAPER_B6_JIS	88	B6 (JIS) 128 x 182 mm
DMPAPER_B6_JIS_ROTATED	89	B6 (JIS) Rotated 182 x 128 mm
DMPAPER_12X11	90	12 x 11 in
DMPAPER_JENV_YOU4	91	Japanese Envelope You #4
DMPAPER_JENV_YOU4_ROTATED	92	Japanese Envelope You #4 Rotated
DMPAPER_P16K	93	PRC 16K 146 x 215 mm
DMPAPER_P32K	94	PRC 32K 97 x 151 mm
DMPAPER_P32KBIG	95	PRC 32K(Big) 97 x 151 mm
DMPAPER_PENV_1	96	PRC Envelope #1 102 x 165 mm
DMPAPER_PENV_2	97	PRC Envelope #2 102 x 176 mm
DMPAPER_PENV_3	98	PRC Envelope #3 125 x 176 mm
DMPAPER_PENV_4	99	PRC Envelope #4 110 x 208 mm
DMPAPER_PENV_5	100	PRC Envelope #5 110 x 220 mm
DMPAPER_PENV_6	101	PRC Envelope #6 120 x 230 mm
DMPAPER_PENV_7	102	PRC Envelope #7 160 x 230 mm
DMPAPER_PENV_8	103	PRC Envelope #8 120 x 309 mm
DMPAPER_PENV_9	104	PRC Envelope #9 229 x 324 mm
DMPAPER_PENV_10	105	PRC Envelope #10 324 x 458 mm
DMPAPER_P16K_ROTATED	106	PRC 16K Rotated
DMPAPER_P32K_ROTATED	107	PRC 32K Rotated
DMPAPER_P32KBIG_ROTATED	108	PRC 32K(Big) Rotated
DMPAPER_PENV_1_ROTATED	109	PRC Envelope #1 Rotated 165 x 102 mm
DMPAPER_PENV_2_ROTATED	110	PRC Envelope #2 Rotated 176 x 102 mm
DMPAPER_PENV_3_ROTATED	111	PRC Envelope #3 Rotated 176 x 125 mm
DMPAPER_PENV_4_ROTATED	112	PRC Envelope #4 Rotated 208 x 110 mm
DMPAPER_PENV_5_ROTATED	113	PRC Envelope #5 Rotated 220 x 110 mm
DMPAPER_PENV_6_ROTATED	114	PRC Envelope #6 Rotated 230 x 120 mm
DMPAPER_PENV_7_ROTATED	115	PRC Envelope #7 Rotated 230 x 160 mm
DMPAPER_PENV_8_ROTATED	116	PRC Envelope #8 Rotated 309 x 120 mm
DMPAPER_PENV_9_ROTATED	117	PRC Envelope #9 Rotated 324 x 229 mm
DMPAPER_PENV_10_ROTATED	118	PRC Envelope #10 Rotated 458 x 324 mm

3.8 Appendix H Merging PDF Files

scViewerX makes it easy to merge any number of PDF files into a single multi-page PDF file.
The PDF files may have different page sizes and any number of pages.

To merge the files you will first have to tell scViewerX that you want to start a merge by using the **PDFMergeInit**() method.

You may after this call start to add files to the new merged document.
To add all pages from a PDF file you may use:

PDFMergeAdd("c:\mergein\file1.pdf");
This call will include all pages from the file named file1.pdf

To add only certain pages from a file you can use **PDFMergeAddEx**.
Here are some samples on how it's used:

PDFMergeAddEx("c:\mergein\file2.pdf", "1;2;3;");
This call will include page 1,2 and 3 from the file named file2.pdf

PDFMergeAddEx("c:\mergein\file3.pdf", "1;20;)
This call will include page 1 and 20 from the file named file3.pdf

When you have added all the files and pages you need you finally call

PDFMergeClose("c:\mergeout\output.pdf");

All files and pages will be written to new file, in this case, named output.pdf.

Please note that any filename sent to scViewerX using the above methods have to include the full path including folder names.

To merge two files and include selected pages, to a new PDF, all you have to do is to add the following 4 lines to your code:

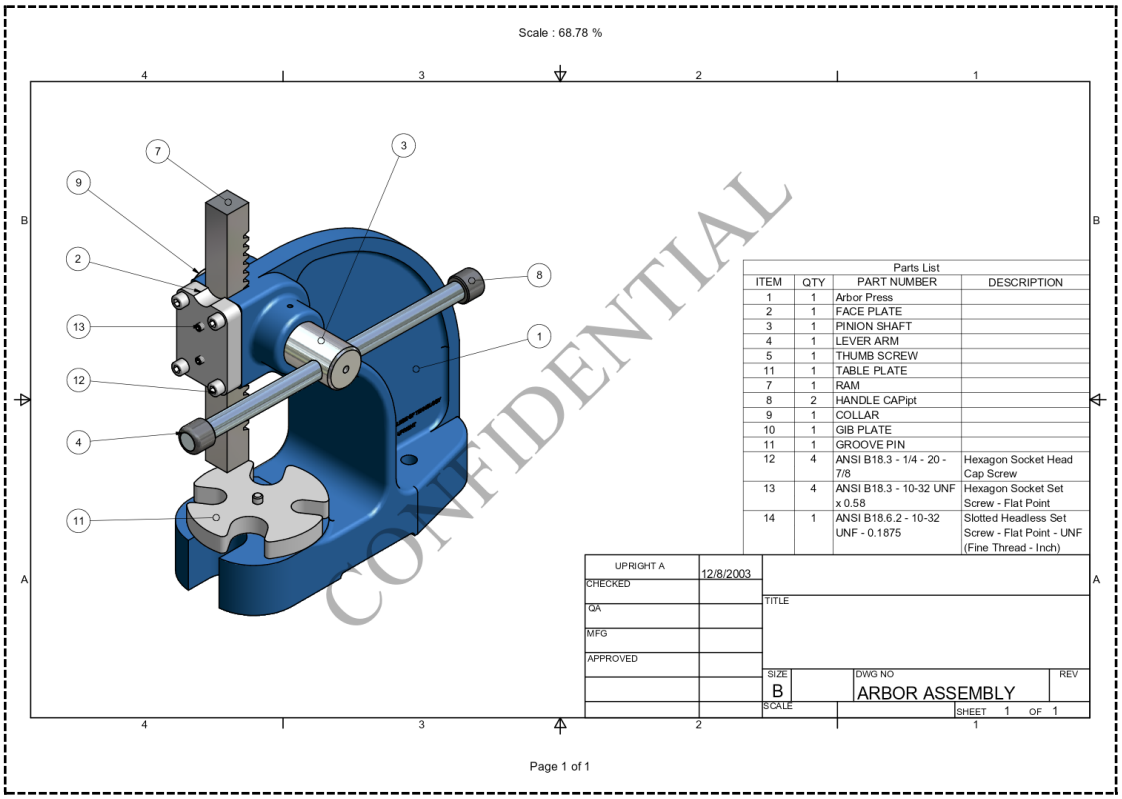
```
scViewer.PDFMergeStart();
scViewer.PDFMergeAddEx( "c:\mergein\file1.pdf", "1;2;3;");
scViewer.PDFMergeAddEx( "c:\mergein\file2.pdf", "1;20;”);
scViewer.PDFMergeClose("c:\mergeout\output.pdf");
```

If you want to include all pages from the source same documents you can use:

```
scViewer.PDFMergeStart();
scViewer.PDFMergeAdd( "c:\mergein\file1.pdf"  );
scViewer.PDFMergeAdd(  "c:\mergein\file1.pdf" );
scViewer.PDFMergeClose( "c:\mergeout\output.pdf" );
```

3.9 Appendix I Printing with Header, Footer and Watermark

Header, footer and watermark are optional text information you can add to each printed sheet. To make your printed documents more useful and professional-looking, you can add information such as your company name, page numbers, date and time, file name, or project-specific information to the header and/or footer.
A watermark can be used to show a text on top of the printed page. This could be any text that will give the document reader additional information. You could for example use the watermark text “CONFIDENTIAL” to inform everyone that this document should be kept secret.
Here is a printed sheet with header, footer and watermark:



Here is the source code that was used to create these:

```
scViewer.SetHeaderFont( L"Arial", 10, FW_NORMAL, ANSI_CHARSET );
scViewer.SetHeaderText( L"Scale : &s", RGB(0,0,0), 1, 5.0 );

scViewer.SetFooterFont( L"Arial", 10, FW_NORMAL, ANSI_CHARSET );
scViewer.SetFooterText( L"Page &p of &P",  RGB(0,0,0), 1, 5.0 );

scViewer.SetWatermarkFont( L"Times New Roman", 100, FW_NORMAL,
ANSI_CHARSET );
scViewer.SetWatermarkText( L"CONFIDENTIAL", RGB(0,0,0), 0.3f, 45 );
```

The header and footer may contain up to 3 lines of text. You can use a predefined constant to insert dynamic information, like for example date and time. Below is a list of supported constants:

Type this	**To print this**
&d	Date in short format as specified by regional settings in the Control Panel.
&f	File name for printed document.
&F	Full file path for printed document.
&n	Line break. You can have a maximum of three text lines.

&P	Total number of pages in printed document.
&p	Currently printed page number.
&s	Print scaling. For example 100%.
&t	Time in the format specified by regional settings in the Control Panel.
&u	Name of the currently logged on user.

A watermark may contain only one line of text. You have full control of the color, transparency and rotation used for the watermark. You may configure set the font name, size, boldness and character set used for the text.

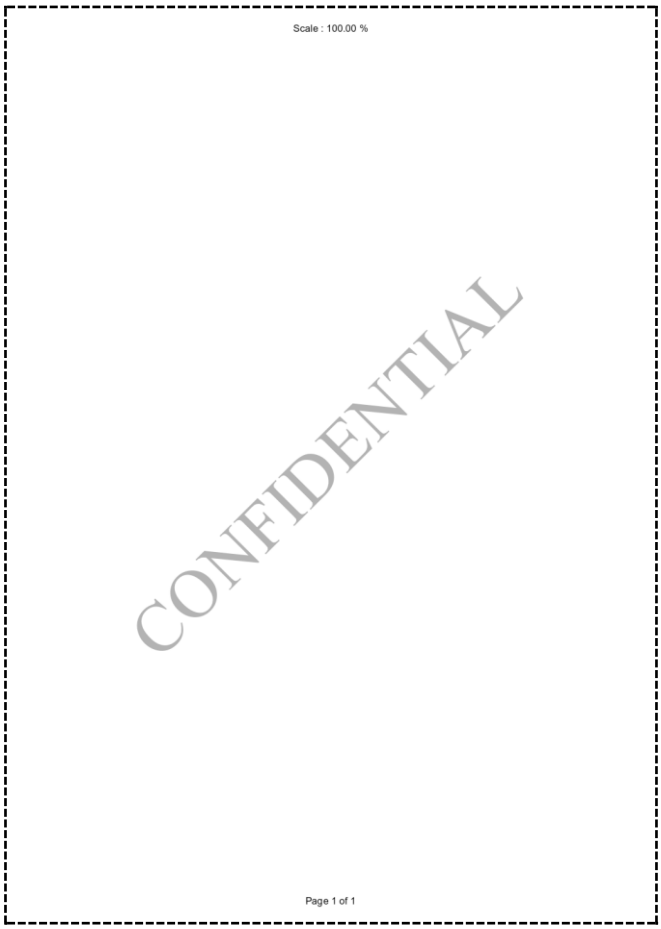
For more information please look at the available methods and properties:

SetFooterFont
SetFooterText
EnableFooter

SetHeaderFont
SetHeaderText
EnableHeader

SetWatermarkFont
SetWatermarkText
EnableWatermark

Sample header, footer and watermark on a blank sheet without any drawing information:

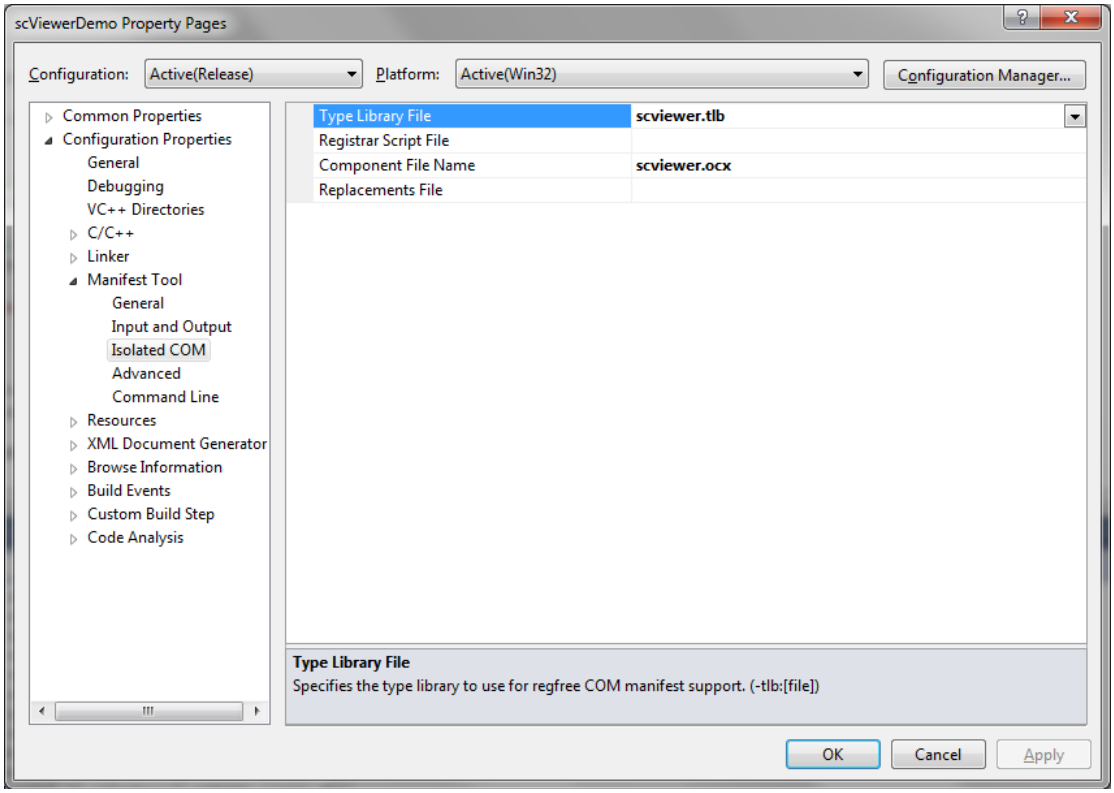


3.10 Appendix J Isolated COM

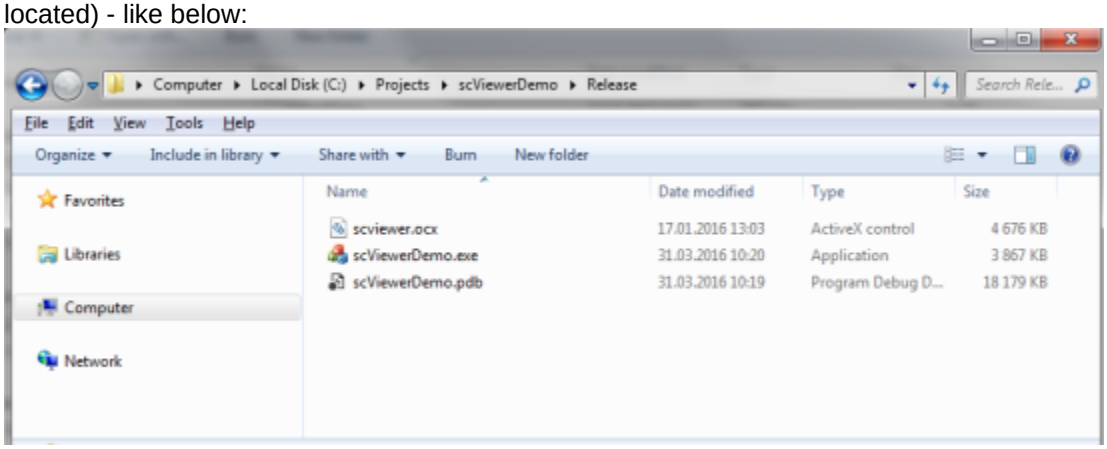
Most environments will allow you to use the control in Isolated COM mode. By using this mode, the scviewer.ocx does not have to be registered on the system, and the installed ocx will be private to your application.
With Isolation you may even have applications using different versions of scviewer.ocx running on the same system.

For this example, we will use the MFC/C++ sample application scViewerDemo, and we will show how to use the scviewer.ocx isolated.

First you need to copy the file named scviewer.tlb to the source folder for the project (where the C++ files are placed). The tlb file can be found in the SDK installation folder.
Then open the project and go project settings and locate the "Manifest Tool" settings.
Add two entries to the Isolated COM section as shown below:

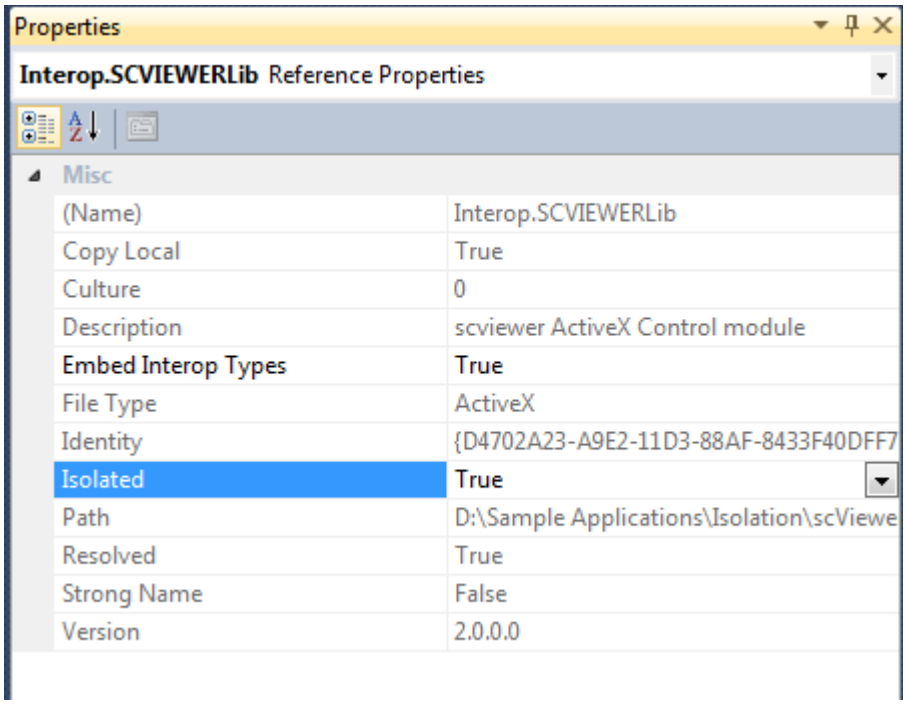


Finally, you must copy the scviewer.ocx file to the output folder (where is executable is located) - like below:



Just recompile and then run the application!

ScViewCSharp sample
If you want to enable Isolation for the C# sample, you'll need to locate the SCVIEWERLib entry under References. Open the properties for this item and you will find an entry named "Isolated", change it to true:



Then recompile the project, and you may now run the scViewCSharp application without registering scviewer.ocx on the target system.
During installation you must copy the scviewer.ocx file to the same folder as your executable, in addition to the other redistributable files that are listed in Appendix C.

Please note that the ActiveX must always be registered on the system used to compile your application.

3.11 Appendix K Tesseract OCR Engine

Tesseract OCR Engine Information

Tesseract OCR (Optical Character Recognition) engine is a free open-source software developed by Google. It is designed to recognize and extract text from images or scanned documents. Tesseract was initially developed at Hewlett-Packard Laboratories in the 1980s and later released as open source in 2005. Google took over the project in 2006 and has been actively maintaining and improving it since then.

Tesseract has undergone significant improvements over the years and is known for its high accuracy in recognizing printed text. However, its performance may vary depending on the quality of the input image, the complexity of the text, and the language being used.

The Tesseract OCR engine is a powerful tool for extracting text from images and documents. It has gained popularity due to its accuracy, language support, and active development community. However, it's important to note that while Tesseract performs well on printed text, it may not be as effective in recognizing handwriting or text with complex layouts.

Tesseract is an open-source project, which means that its source code is freely available for anyone to use, modify, and distribute. The project has a dedicated community of developers who actively contribute to its development, bug fixes, and feature enhancements.

You can always download the latest available Tesseract installation from the following site:

https://github.com/UB-Mannheim/tesseract/wiki