PDF Conversion SDK v11

32-bit and 64-bit

API Reference

PDF/PostScript/EPS Conversion Engine for Developers

1. Getting Started

Thank you for purchasing the 32-bit and 64-bit PDF Conversion SDK. With it, you'll have the power and flexibility you need to automatically convert your PDF/PostScript files into industry standard formats. Based on robust parsers, the comprehensive filter framework delivers fast and efficient conversions into vector and image formats.

The Developer Kit is **licensed by input and output format**. To get started, you should have:

PDF Conversion SDK (either downloaded via FTP or on CD ROM)
License Agreement
Conversion SDK API Reference Manuals
License Key

It's a good idea to review the PDF Conversion SDK API Reference manual thoroughly before starting. You should be familiar with all appropriate functions and parameters before beginning to use the Developer Kit.

Function and Flexibility with Intelligent Filter Options

What sets PDF Conversion SDK apart is its ability to parse PostScript files using our Intelligent Filter Options. These allow you closely control your conversion results. We're constantly adding new options based on customer feedback and custom work that we do for clients to help them integrate our technology into their applications. Keep in touch as you use the Developer's Kit and let us know what additions you'd find useful.

General Options apply to any of the destination formats and include (see VgPsLoadOptions function), for example:

Convert Characters to Strings - Uses an algorithm to analyse character spacing to combine characters into word objects and words into text objects. This improves the editability of the text strings.

Ignore Objects- Allows you to decide whether vector, text or image objects should be converted into objects or ignored during conversion

Add White Space – Adds a margin of white space around the boundary (bounding box) of the picture

Cropping – Engine can do the cropping used in the PS/PDF file if the output does not support it

Emulate – Convert text objects into paths to ensure WYSIWYG

Rotation – Specify the rotation angle of the resulting picture.

Page range – Specify the page, page range to convert

Recognition – Of circles/ellipses out of polygons/paths

Format Specific Options are advanced options that apply only to certain formats**.**

Support is at hand
If you have questions or need help using the PDFConversion SDK Developers Kit, please contact us during Central European business hours (8:30am – 6:00pm). Our contact information is:

Square One bv
Oosteinde 34
2361HE Warmond
The Netherlands

☎ +31 71 362 7297

🖹 +31 71 890 0567

🖂 support@visual-integrity.com

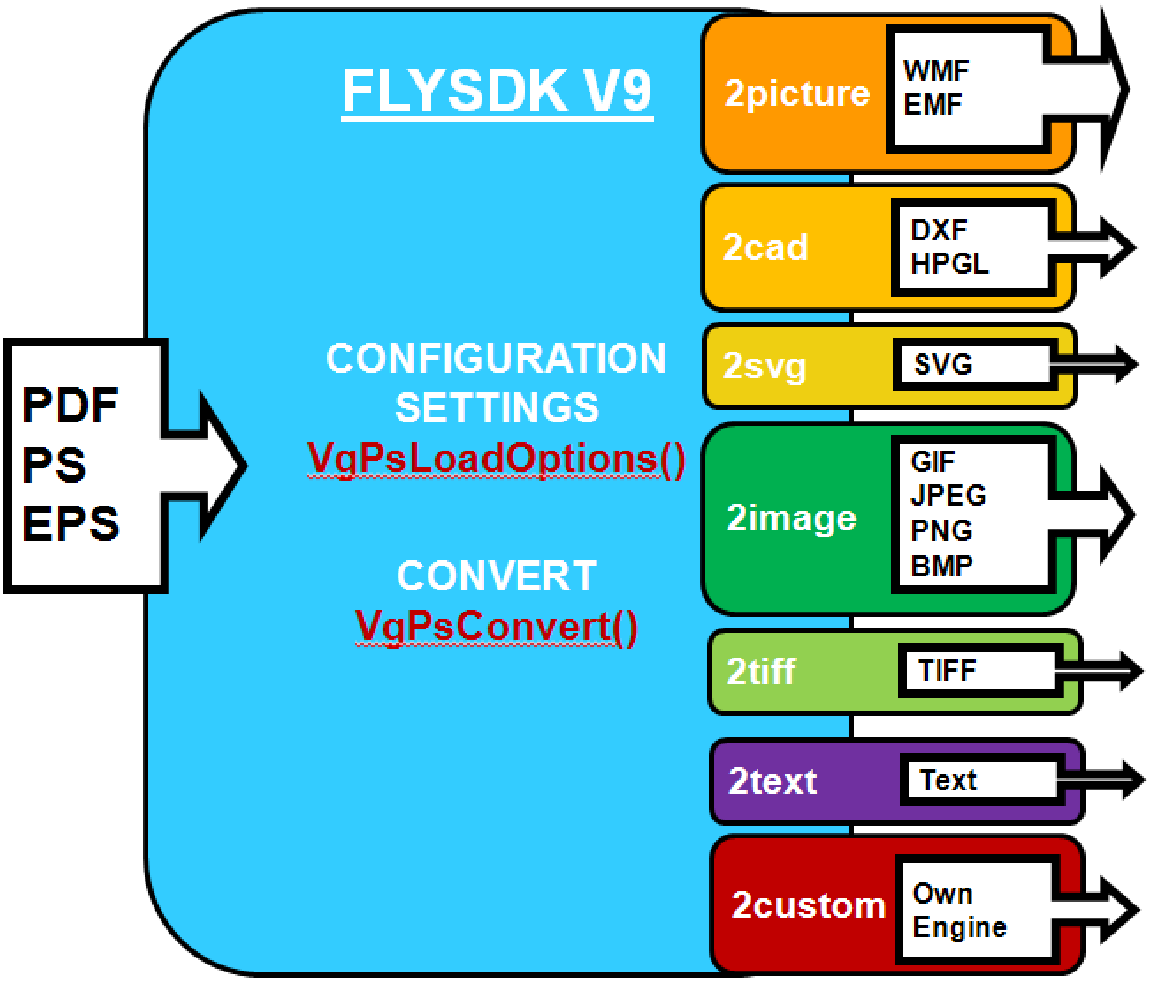
🌍 www.visual-integrity.com

2. The Architecture

The PDF Conversion SDK v11 is a developers kit to convert PDF/PostScript/EPS files into image, vector or text files.

The API engine is based on a very powerful display-list architecture. Before creating the output file, first an internal display-list is built which contains all the graphical objects of the input file. This display list can be manipulated, for example attributes of certain objects can be changed, picture can be rotated, objects can be added, etc., etc.

Saying this, The PDF Conversion SDK Developers Kit is much more than just being a graphics format conversion filter framework !



3. The Conversion SDK PDF API functions

In this section, you will find a description of all the functions of the API (Application Programming Interface) of the PDF/PostScript/EPS Developers Kit which controls the conversion process. The file vgpsflow.h, is included in the developers kit.

VgPsConvert

```
VgPsConvert(char *in, char *out, int format, VgPsProgCB callback );
```

Description

VgPsConvert is the main function to convert a PDF, PostScript or EPS file into one of the vector output formats. Before calling this function, function calls to specify the conversion parameters must be made.

Parameters

in The name of the PDF, PostScript or EPS file

out The name of the output file

format The output formats are:

POSTSCRIPT	1
HPGLII	4
EPS	6
DXF	9
WMF	11
ASCII	12
EMF	16
SVG	22

Callback When VgPsConvert is converting it will call the function callback.
NULL means no function callback. VgPsConvert will invoke callback with 1 parameter: progress of the conversion, range 0 -100.
If the callback returns non zero the conversion is aborted.

Return Value

Zero if conversion succeeded.

Remarks

Example

```
VgPsConvert("foo.ps", "foo.wmf", WMF, NULL)
```

For Unicode filenames use: **VgPsUniConvert2Vector**

VgPsBitmapConvert

```
VgPsBitmapConvert(char *in, char *out, int format, VgPsProgCB callback );
```

Description VgPsBitmapConvert converts PDF, PostScript or EPS file into a high quality anti-aliased image formats. Before calling this function, function calls to specify the conversion parameters must be made.

Parameters

in The name of the PDF, PostScript of EPS file

out The name of the output file

format The output formats are:

TIFF	13
GIF	18
JPEG	21
PNG	27
BMP	28

Callback When VgPsBitmapConvert is converting it will call the function callback. NULL means no function callback. VgPsBitmapConvert will invoke callback with 1 parameter: progress of the conversion, range 0 -100.

Return Value Zero if conversion succeeded.

Remarks

Example

```
VgPsBitmapConvert("foo.ps", "foo.jpg", JPG, NULL)
```

For Unicode filenames use: **VgPdfUniConvert2Image VgPdfUniConvert2ImagePages or VgPdfUniConvertPage2Image**

VgPsSetLicense

`VgPsSetLicense`(int licvalue)

Description	The VgPsSetLicense function checks the licvalue to license the developers kit. If the filter is not licensed a watermark is added on top of the picture.	
Parameters	*licvalue*	The value of the license (need to get it from Square One bv)
Return Value	None	
Remarks		

VgPsLastError

`VgPsLastError``()`

Description	Returns 0 if the conversion succeeded.
Parameters	`None`
Return Value	Result of the conversion
Remarks	

VgPsLoad

`VgPsLoad``(char *in)`

Description	VgPsLoad interprets the PDF, PostScript or EPS file and builds for each page a displaylist. Function is similar as VgPsConvert except that it does not do a conversion to one of the output formats.
Parameters	*`in`* The name is the PDF, PostScript of EPS file
Return Value	Zero if loading of the file succeeded
Remarks	

VgPsSaveAs

`VgPsSaveAs``(char *out, int format)`

Description	VgPsSaveAs converts the multi-page displaylist, created by VgPsLoad.	
Parameters	*`out`*	The name of the output file.
	`format`	The output format.
Return Value	None	
Remarks	VgPsSaveAs only can be called after VgPsLoad	

VgPsSavePageAs

```
VgPsSavePageAs(char *out, int format, int pageno)
```

Description	VgPsSavePageAs converts 1 page, specified by pageno, of the multi-page displaylist, created by VgPsLoad.	
Parameters	*out*	The name of the output file.
	format	The output format.
	pageno	The page to be converted.
Return Value	pageno if page exists otherwise 0.	
Remarks	VgPsSaveAsPage only can be called after VgPsLoad	

VgPsGetNoPages

```
VgPsGetNoPages()
```

Description	VgPsGetNoPages returns the number of pages of the PDF, PostScript or EPS file
Parameters	None.
Return Value	Number of pages.
Remarks	VgPsGetNoPages only can be called after VgPsLoad

VgPsMapFont

```
VgPsMapFont(char *from, int fromstyle, char *to, int tostyle)
```

Description The Graphics Connection offers the option to do font mapping. By default the TGC developers kit supports the standard 35 fonts specified by the Apple LaserWriter. Both standard and embedded Type 1 fonts can be mapped. The function *VgPsMapFont* maps a font to another font. This mapping is done on the Conversion when generating the output file.

Parameters

from	Fontname to be mapped.
fromstyle	Typestyle to be mapped.
to	Fontname to map to.
tostyle	Typestyle to map to.

Return value None.

Remarks By default no font mappings are done. Styles of a font are:

TGC_NORMAL	0
TGC_BOLD	1
TGC_ITALIC	2
TGC_BOLDITALIC	3

Example Map the font Helvetica to Arial.

```
VgPsMapFont("Helvetica", TGC_NORMAL, "Arial", TGC_NORMAL);
```

VgPsLoadOptions

```
VgPsLoadOptions (int app, char *filename)
```

Parameters	*app*	Type of application, eg PDF2XXX, PS2XXX etc
	filename	The filename of the configuration file, if NULL the default ini file belonging to the app is used

Return value None.

Remarks

Example You will find the *pdf2xx.ini/ps2xxx.ini* file in the root directory of the developers kit. By looking at it, you'll get an overview of all the functions. Below is a sample of the default configuration file.

The Intelligent Filter **options v11 win.pdf** file gives a detailed description of the options.

```
VgPsLoadOptions (PDF2XXX, %INSTALLDIR%\pdf2xxx.ini);
```

VgLoadOptionsFromString(char *string);

```
VgLoadOptionsFromString (char *string)
```

Parameters	*string*	contains one of more options

Return value None.

Remarks

Example You will find the *pdf2xx.ini/ps2xxx.ini* file in the root directory of the developers kit. By looking at it, you'll get an overview of all the functions. Below is a sample of the default configuration file.

The Intelligent Filter **options v9.0 win.pdf** file gives a detailed description of the option

```
VgLoadOptionsFromString ("rotate(90) clipping(0)");
```

VgSetFontPath(char *fname);

```
VgSetFontPath (char *fname)
```

Parameters	*fname*	global path to fonts directory
Return value	None.	
Remarks	Use this function if the fonts directory is not in the same directory as your executable.	

Example

```
VgSetFontPath("C:/Program Files/Visual Integrity/Conversion sdk v9.0/fonts”);
```

Below you will find API functions for PDF files only.

VgPdfNoPages

```
VgPdfNoPages(char *in)
```

Description VgPdfNoPages returns the number of pages of a PDF file

Parameters *in* name of the PDF file

Return Value Number of pages. 0 if file is not a PDF file.

Remarks

For Unicode filenames use **VgPdfUniNoPages**

VgPdfConvertPages

```
VgPdfConvertPages(char *in, char *out, int format, int pageb, int pagee,
VgPsProgCB pgCB);
```

Description VgPdfConvertPages is converts a range of pages of a PDF file into one of the output formats. Before calling this function, function calls to specify the conversion parameters must be made.

Parameters

in	The name of the PDF file
out	The name of the output file
format	See output formats VgPsConvert
pageb	Start page number
pagee	End page number
pgCB	When VgPdfConvertPages is converting it will call the function callback. NULL means no function callback. VgPdfConvertPages will invoke callback with 1 parameter: progress of the conversion, range 0 -100. If the callback returns non zero the conversion is aborted.

Return Value Zero if conversion succeeded.

Remarks

Example Convert page 1 and 2 of the PDF file foo.pdf

```
VgPdfConvertPages("foo.pdf", "foo.emf", EMF, 1, 2, NULL)
```

For Unicode filenames use: **VgPdfUniConvertPage2Vector or VgPdfUniConvert2VectorPages**

VgPdfBitmapConvertPages

```
VgPdfConvertPages(char *in, char *out, int format, int pageb, int pagee,
VgPsProgCB pgCB);
```

Description VgPdfBitmapConvertPages is converts a range of pages of a PDF file into one of the output formats. Before calling this function, function calls to specify the conversion parameters must be made.

Parameters

in	name of the PDF file
out	name of the output file
format	see output formats VgPsBitmapConvert
pageb	start page number
pagee	end page number
pgCB	When VgPdfBitmapConvertPages is converting it will call the function callback. NULL means no function callback. VgPdfBitmapConvertPages will invoke callback with 1 parameter: progress of the conversion, range 0 - 100 If the callback returns non zero the conversion is aborted.

Return Value Zero if conversion succeeded.

Remarks

Example Convert page 1 and 2 of the PDF file foo.pdf to JPEG

```
VgPdfBitmapConvertPages("foo.pdf", "foo.jpg", JPEG, 1, 2, NULL)
```

For Unicode filenames use: **VgPdfUniConvert2Image VgPdfUniConvert2ImagePages or VgPdfUniConvertPage2Image**

VgPdfCheckAnnotationsOnPage

```
VgPdfCheckAnnotationsOnPage (char *in, int pageno)
```

Description VgPdfCheckAnnotationsOnPage returns TRUE/FALSE if the specified page of a PDF file contains annotation objects.

Parameters

in	name of the PDF file
pageno	page number

Return Value TRUE if annotation objects have been found, otherwise FALSE

Remarks

VgPdfGetLayerInfo

```
VgPdfGetLayerInfo (char *in, VgPsLayerInfo *layerinfo)
```

Description VgPdfGetLayerInfo checks the number of PDF layers used in a PDF file and stored this info in the VgPsLayerInfo struct.

Parameters

in	name of the PDF file
layerinfo	contains the layer data of the PDF file

Return Value

Remarks See vgpsflow.h for the definition about the VgPsLayerInfo struct

VgPdfGetLayerInfoOnPage

```
VgPdfGetLayerInfoOnPage(char *in, int pageno, VgPsLayerInfo *layerinfo)
```

Description VgPdfGetLayerInfoOnPage checks the number of PDF layers used on the specified page of the PDF file and stored this info in the VgPsLayerInfo struct.

Parameters

in	name of the PDF file
layerinfo	contains the layer data of the PDF file

Return Value

Remarks See vgpsflow.h for the definition about the VgPsLayerInfo struct

VgPdfConvert2Image

```
VgPdfConvert2Image(char *in, char *out, int format, VgPsProgCB pgCB,
                       VgPsLayerInfo *layerInfo)
```

Description VgConvertPdf2Image converts the objects on the specified PDF layers of a PDF file into a high quality anti-aliased image format.

Parameters

in	name of the PDF file
out	name of the output file
format	the output formats are:

TIFF	13
GIF	18
JPEG	21
PNG	27
BMP	28

pgCB	when VgPdf2Image is converting it will call the function callback. NULL means no function callback. VgPdf2Image will invoke callback with 1 parameter: progress of the conversion, range 0 -100.
`layerInfo`	contains the list of layers

Return Value Zero if conversion succeeded.

Remarks See vgpsflow.h for the definition about the VgPsLayerInfo struct

For Unicode filenames use: **VgPdfUniConvert2Image VgPdfUniConvert2ImagePages or VgPdfUniConvertPage2Image**

Similar functions:

```
VgPdfConvert2ImagePages(char *in, char *out, int format, int pageb, int
pagee, VgPsProgCB pgCB, VgPsLayerInfo *layerInfo);
```

```
VgPdfConvertPage2Image(char *in, char *out, int format, int pageno,
VgPsLayerInfo *layerInfo);
```

VgPdfConvertPage2DIB

```
VgPdfConvertPage2DIB(char *in, int pageno, VgPsLayerInfo *layerInfo,
              BITMAPINFOHEADER *bmi, unsigned char **imagedata)
```

Description VgPdfConvertPage2DIB converts the objects on the specified PDF layers of a PDF file into a windows BMP memory based 24-bit (BGR) image.

Parameters

in	name of the PDF file
pageno	page number
layerInfo	contains the list of layers
bmi	info about the bitmap
imagedata	handle to the image data

Return Value Zero if conversion succeeded.

Remarks See BMP related articles for more information.

VgPdfConvertPage

```
VgPdfConvertPage(char *in, char *out, int format, int pageno, VgPsLayerInfo
              *layerInfo);
```

Description *VgPdfConvertPage* converts the objects on the specified PDF layers of a PDF file into a one of the supported vector formats.

Parameters

in	name of the PDF file
out	name of the output file
format	the output formats are:

HPGLII	4
EPS	6
DXF	9
WMF	11
EMF	16
SVG	22

layerInfo	contains the list of layers

Return Value Zero if conversion succeeded.

Remarks See vgpsflow.h for the definition about the VgPsLayerInfo struct

For Unicode filenames use: **VgPdfUniConvertPage2Vector or VgPdfUniConvert2VectorPages**

VgPdfConvert2Layers

```
VgPdfConvert2Layers(char *in, char *out, int format, VgPsProgCB pgCB)
```

Description VgPdfConvert2Layers converts the objects on each PDF layer of a PDF file into one of the supported vector formats. Output file for each layer.

Parameters

in	name of the PDF file
out	name of the output file
format	the output formats are:

HPGLII	4
MIF	5
EPS	6
DXF	9
WMF	11
EMF	16
SVG	22

pgCB	callback function

Return Value Zero if conversion succeeded.

Remarks Layer name is appended to the output file name.

Similar Functions:

```
STDAPI VgPdfConvertPage2Layers(char *in, char *out, int format, int pageno)
```

VgPsConvert2Custom

```
VgPsConvert2Custom(char *in, CU_OBJECT_CB cbfunc)
```

Description VgPsConvert2Custom walks through objects in the PDF/PostScript file and for each object the cbfunc is called.

Parameters

in	name of the PDF/PostScript file
cbfunc	callback func for each object

Return Value Zero if conversion succeeded.

Remarks See custom.h for the CU_OBJECT_CB and object data type definitions

Similar functions:

```
STDAPI VgPdfConvertPage2Custom(char *in, CU_OBJECT_CB cbfunc, int pageno,
VgPsLayerInfo *layerInfo, void *userData)
```

VgPdfGetPageSize

```
VgPdfGetPageSize (char *in, int pageno, double *extent, int cropbox);
```

Description	VgPdfGetPageSize stores the dimension in mm of the page-size of a specified page of the PDF file in the extent array.	
Parameters	*in*	name of the PDF file
	pageno	page number
	extent	array of 4 doubles (lower-left corner, upper-right corner)
	cropbox	flag to get the cropping or page dimension
Return Value		
Remarks	Cropping and page dimensions are often the same	

For Unicode filenames use: **VgPdfUniGetPageSize**

VgPdfGetMeasureDictInfo

```
VgPdfGetMeasureDictInfo(char *in, int pageno, double *bbox, double *xScale,
                        double *yScale)
```

Description	VgPdfGetMeasureDictInfo retrieves the page + scaling dimensions for a page from the dictionary	
Parameters	*in*	name of the PDF file
	pageno	page number
	bbox	array of 4 doubles (lower-left corner, upper-right corner)
	xScale	scale factor in x-direction
	yScale	scale factor in y-direction
Return Value		
Remarks	If entries are not found the returning values are set to 0.0.	

4. VS201x PDF Conversion C++, C# and Visual Studio 2010 Examples

In sub-directories **VS201x PDF Conversion -** ***C++, c# and VB*** you will find demo source code which demonstrates how to use the PDF Conversion SDK API. The demo programs are developed with Visual Studio 2010, but also work fine in Visual Studio 2017.

You can use these projects as a start for your own application or just copy/paste the code you want to reuse.

The function calls to the Conversion SDK developers kit are very simple !

For example in vcexdlg.cpp of the **VS201x PDF Conversion - C++ Example** the following calls are made:

```
if (format == JPEG || format == GIF)

 VgPsBitmapConvert(m_Source, dest, format, NULL);

else

 VgPsConvert(m_Source, dest, format, NULL);
```

5. Running your program

In order to run your program using the Conversion SDK libraries you need to add the following files/directories into the same directory as your program:

Vgflow.dll
Vgpsflow.dll
Fonts directory