Software Mechanics Pty Ltd

SVGmaker Content Server

Guide for Integrators and Developers

Document ID: svgmaker/specification/SVGMCS_Guide
Version: Version 2.11.07
Revised: 29 September 2006
Author: Mike Kidson

Contents

Introduction

SVGmaker Content Server is a document transcoder framework for Windows server operating systems. The framework harnesses standard Windows applications to perform optimized, concurrent, fault tolerant document transcoding on multi-processor Windows platforms.

API's are provided for C language DLLs, command line, .NET and Java components.

Out-of-the-box transcoding is provided by SVGmaker Content Server Despatcher, a configurable .NET application that works through the file system interface to transcode files on demand from a user defined watch folder.

System Requirements

- Windows 2000, Windows 2000 Server, Windows XP, or Windows Server 2003
- Office 2002 or higher (2003 or higher recommended)
- Java (only required with the Java API)
- .NET Framework (only required for the .NET API and the Despatcher application)

Installation and Configuration

Installation of SVGmaker Content Server requires these manual steps:

1. If you are installing onto Windows Server 2003, set the group policy to enable the installation of kernel-mode printer drivers.

- Open Group Policy: click **Start**, click **Run**, type **gpedit.msc**, and then press ENTER
- Under **Local Computer Policy**, double-click **Computer Configuration**.
- Double-click **Disallow installation of printers using kernel mode drivers**, and then click **Properties**.
- On the **Setting** tab, click **Disabled**, and then click **OK**.

2. Download SVGmaker Content Server:
http://www.svgmaker.com/download/SVGmakerCS.zip

The zip file contains these installation instructions, and an executable file.

3. Install SVGmaker Content Server V2 by running the executable file. One or more SVGmaker Content Server printer drivers will be installed in your Windows Printers Folder. The number installed corresponds to the number of logical processors detected on the machine:

- 1 for a Pentium single CPU machine
- 2 for a dual Intel Xeon machine
- 4 for a dual hyperthreading Intel Xeon machine.

If you already have an existing installation of SVGmaker Content Server it will be upgraded by the installer.

4. Configure Server Properties in the Windows Printers Folder
Printers and Faxes
File > Server Properties > Advanced Tab
Uncheck [ ] Log spooler warning events
Uncheck [ ] Log spooler information events

5. ***Important*** Configure these applications to avoid interuptions and to speed up processing:

5.0 Microsoft Office

Ensure no 3rd party Office addins are installed. If any 3rd party addins are installed, such as Acrobat, uninstall them.

Delete all files from the \TEMP folder regularly to avoid significant performance loss.

Disable Shared WorkSpace updates for Office 2003:

In any Office program, Click Tools > Options > General Tab

Click the Service Options button. In the Shared Workspace category:

Clear [ ] Document is part of a workspace or SharePoint site
Clear [ ] There is important status information regarding the document

Set Workspace Updates to Never: do not get updates

Clear [ ] Show document update Desktop Alerts

Set When closing a document, update the workspace with your changes:
Never: don't update at all.

5.1. For PowerPoint:

Click Tools > Options > Spelling and Style
Clear the checkbox called [ ] Check Spelling As You Type

5.2. For Word:

Click Tools > Options > Spelling and Grammar
Clear the checkbox called [ ] Check Spelling As You Type
Clear the checkbox called [ ] Check Grammar As You Type

Click Tools > Options > Print
Note: Background printing is now set programatically from 2.10 onwards

5.3. For Excel:

Click Tools > Options > Save
Clear the checkbox called [ ] Save AutoRecover info
Check the checkbox called [ ] Disable AutoRecover

5.4. For Adobe Acrobat Reader:

Note: SVGmaker Content Server now works with Adobe Reader **7**. Reader 6 may still work, but Reader 7 is now the supported version and should be deployed from this point onwards because it supports the current PDF file format, revision 1.6.

Adobe Reader 7 launches a program called WISPTIS.EXE ("Windows Ink Services Platform Tablet Input Subsystem"). This program has been observed accumulating GDI handles, a practice that has been observed to coincide with GDI and desktop malfunction after the some number of PDF documents have been processed.

Stop WISPTIS.EXE from launching by right-clicking on it in Windows explorer, and setting security for "Full Control" to "Deny". WISPTIS.EXE is found in:

c:\winnt\system32 on Windows 2000
c:\windows\system32 on Windows 2003
c:\windows\system32 on Windows XP

Acrobat Reader 7 configuration for SVGmaker Content Server

Edit > Preferences

Forms
Set Autocomplete off

General
Uncheck Automatically save document changes to temporary file

JavaScript
Check Enable Acrobat JavaScript

Uncheck Show console on errors and messages for Javascript Debugger

Multimedia
Uncheck Play dubbed audio when available

Security
Uncheck Verify Signatures when document is opened

Startup
Uncheck Display the Document Status Dialog

Application Startup
Uncheck Display Splash screen
Uncheck Show Messages and automatically update

Trust Manager
Uncheck Allow multimedia operations for Trusted and NonTrusted documents

Updates
Set do not automatically check for critical updates
Uncheck Display notification dialog at startup

6. If you plan to to use SVGmaker Content Server Despatcher make sure the the .NET framework is installed. Despatcher is a Windows server application that automatically transcodes files into SVG when they are copied to a configurable watch folder.

The .NET framework is a free download from Microsoft that enables .NET programs to run on Windows 2000, Windows 2003 and Windows XP systems.

Microsoft .NET Framework Version 1.1 Redistributable Package
http://tinyurl.com/4y9vv

Using SVGmaker Content Server Despatcher

Despatcher is a .NET application that uses the SVGmaker transcoding framework to transcode documents that are placed in a configurable watch folder. When you install SVGmaker Content Server a shortcut is placed on the desktop for launching SVGmaker Content Server Despatcher.

To transcode files with Despatcher:

1. Launch **Despatcher** by double clicking the shortcut installed on the desktop

2. Enter or select a **Watch Folder**. When files are copied to this folder or its child folders, Despatcher will invoke SVGmaker to transcode the document into SVG. By default, SVG output will be placed in the same folder as the original document and the original document will be deleted after transcoding is complete.

3. If you prefer output to be placed in a different folder, enter or select a **Destination Folder**. Any descendent sub-folder heirarchy of the Watch Folder will be reproduced in the Destination Folder.

e.g.

if the Watch Folder is **C:\Test**

and the Destination Folder is **C:\Output**

when a document is placed in the folder **C:\Test\User01\myinvoices\2005**

the transcoded SVG output is placed in **C:\Output\User01\myinvoices\2005**

4. Once you have set a watch folder and a destination, Despatcher is ready to start transcoding. Click the **Activate button**. Despatcher will queue and transcode any files that are in the watch folder or its descendents. Despatcher will also queue and transcode any new files that are copied to the watch folder or its descendents until you click the **Stop button**.

Transcoder Options

[ ] Log folder activity

When checked, transcoding events are logged to `svgmcs.log` in the watch folder where the document originates.

- Job Number, Job status
- Job failure error codes, error messages and removal destination

[ ] Delete original files

When checked, after a file is transcoded the original file is deleted.

[ ] Quarantine failed files

When checked, files that fail to transcode are moved to a failure folder. Files in the failure folder and its descendents are not transcoded by Despatcher so the error will not

be repeated in subsequent transcoding sessions. The log file `svgmcs.log` in the failure folder records the Job Number, Job status and folder of origin for failed files.

The naming convention for failure folders is that each watch folder root begets its own failure folder root. The failure folder heirarchy of the failure folder duplicates the descendent folder heirarchy of the watch folder.

e.g.

if the Watch Folder is **C:\Test**

the failure folder is **C:\Test\failed**

and when a document is placed in the folder **C:\Test\User01\myinvoices\2005**

the failed file is moved to **C:\Output\User01\failed\myinvoices\2005**

Once you have defined a watch folder root on the server and activated Despatcher, users on the same network as the server can transcode files simply by copying them to the watch folder. An easy way to configure access for a number of users is to create a subfolder for each user and place a shortcut to their folder on each user's desktop.

e.g.

Set the Watch Folder to **C:\Test** in Despatcher

Create these folders on the server:

C:\Test\User01
C:\Test\User02
C:\Test\User03
C:\Test\User04
C:\Test\User05
C:\Test\User06
etc.

For each user, create a shortcut to their folder on their desktop.

Users then only have to drag and drop files from Windows explorer onto the folder shortcut to transcode a file into SVG.

Despatcher Implementation Notes

Despatcher is implemented as a .NET application using the FileWatcher class. It transcodes using the following applications:

Extension	Application
.doc	Microsoft Word
.xls	Microsoft Excel
.ppt	Microsoft PowerPoint
.pdf	PrintTo (assumes Adobe Acrobat Reader 7)
.wpd	PrintTo (assumes WordPerfect 12)
All others	PrintTo (uses Windows file association)

API Architecture

The SVGmaker Content Server API Framework consists of six interface layers known collectively as MakeSVG.

Interface Layer		API Documentation	Description
Highest level			
	`MakeSVGAsync.dll`	`Doc\dotNet\makesvgasync.txt`	Asynchronous enqueing and job status
	`makesvg.jar`	`doc\java\index.html`	Java interface
	`makesvgdotnet.dll`	`doc\dotNet\documentation.chm`	.NET interface
	`makesvghl.dll`	`include\makesvghl.h`	C language DLL with multiprocessor optimization and document transcoding API (requires makesvg.dll and makesvg.exe to be present)
	`makesvg.dll`	`include\makesvg.h`	C language DLL that provides a low-level API for the document transcoding system (requires makesvg.exe to be present)
Lowest level	`makesvg.exe`	`run with no parameters to display usage help`	Windows command-line executable

and a rendering layer:

Rendering Layer	API Documentation	Description
SVGmaker Content Server Printer Driver	See the Printer Driver Registry API documentation below.	Windows GDI system printer used by the interface components tabled above.

Depending on the requirements of your application, any one of these levels can be programmed against.

For files of type .doc, .ppt and .xls MakeSVG invokes Word, PowerPoint or Excel respectively under programatic control to print with an SVGmaker Content Server printer driver. Other filetypes are rendered using the application associated with the Windows shell generic PrintTo function.

API Documentation

The SVGmaker Content Server API documentation for Java, .Net and C developers is installed on your computer as part of the standard SVGmaker Content Server install.

Java interface:
```
...\doc\java\index.html
```

.NET interface:
```
...\doc\dotNet\documentation.chm
...\doc\dotNet\makesvgasync.txt
```

makesvghl.dll:
```
...\include\makesvghl.h
```

makesvg.dll:
```
...\include\makesvg.h
```

makesvghl.exe:
```
run with no parameters to display command-line switches.
```

where `...` is the root of the SVGmaker Content Server installation. By default this is

C:\Program Files\SVGmaker Content Server V2

See section “Architecture” for a description of the above components.

Sample Programs

The following usage notes refer to sample programs that are installed in
Program Files\MakeSVG Content Server V2\samples

Sample files are provided for three types of transcoding applications:

- On-demand asynchronous transcoding
- Batch process transcoding
- Single file transcoding

In the documentation that follows, <printerCount> is the number of SVGmaker printers installed (one per logical CPU as described in the Installation section below). The Windows environment variable `%NUMBER_OF_PROCESSORS%` is used in batch files etc. since it corresponds to the number of SVGmaker Content Server printers installed.

On-demand Asynchronous Transcoding Samples

.NET Interface—Sample Code for on-demand asynchronous transcoding with MakeSVGAsync.DLL using MakeSVGDotNet.DLL

System requirement: .NET Framework 1.1.

A C# sample project demonstrates on-demand enqueing with asynchronous job status reporting.

The sample is located by default in this folder:

C:\Program Files\SVGmaker Content Server V2\samples\MakeSVGDotNet-CSharp

The example command-line program is called CSharpAsyncSample.exe has parameters following this pattern:

<printerCount> <avg.time> <inputPattern> <outputDir>

example:

```
CSharpAsyncSample.exe  %NUMBER_OF_PROCESSORS%  30 c:\docs\* c:\out\
```

transcodes all the documents of the pattern c:\docs* with an average interval of 30 seconds between enqueing of each file to simulate variable demand.

Output goes to **c:\out**.

Job status reporting occurs real time as job events happen.

To transcode sample files with CSharpAsyncSample please run **CSharpAsyncSample.bat**.

Java—Sample Code for on-damand asynchronous transcoding using MakeSVGJava

System requirement: Java 1.4.2.

A Java example demonstrates on-demand enqueing with asynchronous job status reporting. The sample is located by default in this folder:

C:\Program Files\SVGmaker Content Server V2\samples\MakeSVGJavaAsync

The example program is called AsyncSample.class and has parameters following this pattern:

<printerCount> <avg.time> <sourceDir> <destinationDir>

example:

```
cd  C:\Program Files\SVGmaker Content Server V2\samples\MakeSVGJavaAsync

java -cp .;..\..\lib\MakeSVG.jar AsyncSample %NUMBER_OF_PROCESSORS% 5 c:\ppt\
c:\out\ppt\
```

transcodes all the documents in the **c:\ppt** folder with an average interval of 5 seconds between enqueing of each file to simulate variable demand.

Output goes to to **c:\out\ppt**.

Job status reporting ocurs real time as job events happen.

To transcode sample files with AsyncSample.java please run **AsyncSample.bat**.

Batch Process Transcoding Samples

.NET Interface—Sample Code for batch processing using MakeSVGDotNet.DLL

System requirement: .NET Framework 1.1.

C# and VB.NET sample projects are provided. Both these projects are command-line programs and are functionally identical to each other (the C# and VB.NET code are straight translations of each other).

The C# sample is located by default in this folder:
C:\Program Files\SVGmaker Content Server V2\samples\MakeSVGDotNet-CSharp

The VB.NET sample is located by default in this folder:
C:\Program Files\SVGmaker Content Server V2\samples\MakeSVGDotNet-VB

The example command-line programs are called CSharpSample.exe and VBSample.exe respectively and have parameters following this pattern:

<printerCount> <baseDir> <inputPattern> <outputDir> [<inputPattern> <outputDir> [...]]

example:

```
CSharpSample.exe  %NUMBER_OF_PROCESSORS%  c:\  docs\*.ppt  c:\out\
```

transcodes all the documents of the pattern docs*.ppt that are inside the **c:** folder.

Output goes to **c:\out**.

Any failed files are listed at the end of the transcoding run.

When more than one pair of inputPattern, outputDir are specified, the documents are managed in a queue and transcoded in parallel. For an example of this see **CSharpSample.bat** and **VBSample.bat**. These batch files call the appropriate executable to transcode files from three separate data folders.

Java—Sample Code for batch processing using MakeSVGJava

System requirement: Java 1.4.2.

The example command-line program is called Sample1.class and has parameters following this pattern:

<printerCount> <inputDir> <outputDir> [<inputDir> <outputDir> [...]]

example:

```
cd  C:\Program Files\SVGmaker Content Server V2\samples\MakeSVGJava

java -cp .;..\..\lib\MakeSVG.jar Sample1 %NUMBER_OF_PROCESSORS% c:\ppt\  c:\out\ppt\
```

transcodes all the documents in the **c:\ppt** folder.

Output goes to to **c:\out\ppt**.

Any failed files are listed at the end of the transcoding run.

When more than one pair of inputDir, outputDir are specified, the documents are managed in a queue and transcoded in parallel. For an example of this see **JavaSample1.bat**. This batch file calls sample1.exe to transcode files from three separate folders.

MakeSVGHL.DLL—Sample Code for MakeSVGHL.DLL

The example command-line program is called sample.exe and has parameters following this pattern:

<printerCount> <inputPattern> <outputDir> [<inputPattern> <outputDir> [...]]

Pairs of inputPatterns and outputDirs follow printerCount. The program attempts to print every file matching inputPattern to its corresponding outputDir.

example:

```
sample.exe %NUMBER_OF_PROCESSORS% c:\docs\*.ppt c:\out\ppt\
```

transcodes all the .ppt files in the folder **c:\docs**

Output goes to to **c:\out\ppt**

When more than one pair of inputPattern, outputDir are specified, the documents are managed in a queue and transcoded in parallel. For an example of this see **MakeSVGHLSample1.bat**. This batch file calls sample.exe to transcode files from three separate folders.

Single File Transcoding Sample

MakeSVG.DLL—Sample Code for MakeSVG.DLL

The example command-line program is called dlltest.exe and has parameters following this pattern:

Usage: dlltest <input-path> <output-path>.\n

The input-path and the output-path must each specify a unique filename.

Calls to MakeSVG.DLL transcode one job at a time.

Output File Format

SVGmaker Content Server produces output in Scalable Vector Graphics (SVG) format. One file is generated for each page of the original document.

The filenames for page-per-file print jobs are constructed using the following scheme:

filename.001.svg for page 1
filename.002.svg for page 2
filename.003.svg for page 3

and so on.

If the file extension is .svg the file contents will be uncompressed SVG, whilst if the extension is .svgz the file contents will be compressed SVGZ. In addition to one output file per page, an index file is generated. The index file is called filename.html and contains links to all the individual page files for the document.

Technical Notes

Multiprocessing Support

MakeSVGHL.DLL is a queue manager that runs multiple print jobs at the same time on multiproccessor hosts, taking account of interoperability constraints of Microsoft Office programs.

Processor utilization is maximized when heterogeneous types of documents are printed at the same time by MakeSVGHL.

In batch conversion patterns, this sequence of documents:

PPT DOC XLS PPT DOC XLS PPT DOC XLS PPT DOC XLS PPT DOC XLS

transcodes faster than this sequence of documents:

PPT PPT PPT PPT PPT DOC DOC DOC DOC DOC XLS XLS XLS XLS XLS

on a multi-processor host. To transcode an entire corpus of documents most efficiently, integrators should queue a mix of different document types in preference to monotonous runs of individual document types.

Fault Tolerance

1. Combinations of Office applications
Certain combinations of Microsoft Office applications have been shown empirically to result in operating system instability when run at the same time, and certain combinations of Office applications are acknowledged by Microsoft to be unstable together.

See http://support.microsoft.com/default.aspx?scid=kb;EN-US;257757

> "they use global resources (such as memory mapped files, global add-ins or templates, and shared Automation servers), which can limit the number of instances that can run concurrently and lead to race conditions if they are configured in a multi-client environment. Developers who plan to run more then one instance of any Office Application at the same time need to consider "pooling" or serializing access to the Office Application to avoid potential deadlocks or data corruption."

MakeSVGHL.DLL uses a set of built-in rules that prevent these unstable combinations from occuring. MakeSVGHL.DLL only schedules stable combinations of applications, and maximizes processor utilization in the context of the job queue.

2. Watchdog Thread
MakeSVG includes a watchdog thread to detect stalled applications. The error code returned is ERR_JOB_APPHUNG. It is the integrator's responsiblity to respond to this error condition by terminating stalled applications. See sample code for usage examples.

Known Problems and Limitations

- Japanese vertical text is not rotated correctly
- Despatcher allows multiple instances to launch but should be single instance only
- Despatcher doesn't check elegantly whether .NET framework is present

This issue will be addressed in further releases.

SVGmaker Content Server Printer Driver Registry API

The SVGmaker Content Server printer driver supports a registry API so that SVG print jobs can be generated under programmatic control without operator intervention. This registry API is used internally by the MakeSVG interface layers.

Note: If you are programming against any of the MakeSVG interface layers you do not need to consider the printer driver registry API because this is already managed for you by MakeSVG components.

With the regisry API, SVGmaker suppresses file selection and error dialogs to achieve unattended generation of SVG files from standard Windows applications. Parameters including the output filename and configuration options may be specified on a per-job basis and status and error messages can be optionally logged to a job-specific status file. The status file survives the SVG print job for potential use by audit processes or for the control of downstream applications. Real time status information for the print job is available via system messages.

A typical application of the registry API involves invoking a Windows application through its COM interface to print using SVGmaker. The controlling application sets SVGmaker filename and configuration options through the registry interface prior to the commencement of each print job.

DOCINFO Parameters

When printing through the Win32 API, job parameters are normally passed to a printer driver via the DEVMODE structure specified when a print job is started. SVGmaker uses the DEVMODE information but also allows extra parameters to be specified in the Windows registry and the DOCINFO structure passed to the Win32 StartDoc function for the print job.

Using the Windows registry allows output filename and a number of non-standard configuration options to be specified. Using DOCINFO allows only the output filename to be specified. Job parameters derived from the Windows registry override those specified in the DEVMODE. Job parameters derived from the DOCINFO override those specified in the DEVMODE as well as the registry.

Windows Registry Parameters

SVGmaker loads job parameters from the Windows registry at the start of the print job. Consequently any parameters specified must be set up prior to commencement of printing. The following registry entries are recognized. If any entry is absent or contains an invalid data value the default value shown will be used.

```
Registry key: HKEY_CURRENT_USER\Software\Software Mechanics\<printername>
Notes:        <printername> = Windows printer name as displayed in Printers folder

Value name:   OutputFile
Type:         REG_SZ
Data:         fully qualified path spec of output file
              file extension is used to determine output format:
               .svg  uncompressed
               .svgz compressed
               .<anything else> compressed output
Default:      if not specified the output file will be derived from either
              DOCINFO filename if specified or file open dialog if not
Notes:        this entry will be cleared once the print job has started
              this entry will be overridden by a filename specified via DOCINFO

Value name:   LogFile
Type:         REG_SZ
Data:         fully qualified path spec of status log file
Default:      output file with extension .log appended
```

```
Notes:          this entry is only used when status logging to file is enabled via
                the StatusReporting parameter

Value name:     CropToImageArea
Type:           REG_DWORD
Data:           0 = use full page-size
                1 = crop to image area
Default:        DEVMODE setting

Value name:     InitialView
Type:           REG_DWORD
Data value:     0 = 1:1
                1 = fit to window
                2 = cover window
Default:        DEVMODE setting

Value name:     MultipageFormat
Type:           REG_DWORD
Data value:     0 = navigation control (JavaScript)
                1 = vertical tiling (JavaScript)
                2 = HTML index + 1 file per page
                3 = HTML frames + 1 file per page
                4 = no JavaScript
                5 = HTML index + 1 file per page no subdirectories
Default:        DEVMODE setting
Value name:     HtmlWrapper
Type:           REG_DWORD
Data value:     0 = no HTML wrappers
                1 = generate HTML wrappers
Default:        DEVMODE setting

Value name:     ZipOutput
Type:           REG_DWORD
Data value:     0 = output is not zipped
                1 = zip archive containing all output files is produced
Default:        output is not zipped

Value name:     TextMode
Type:           REG_DWORD
Data value:     0 = draw text
                1 = use system fonts
                2 = embed as SVG fonts
Default:        DEVMODE setting

Value name:     TextPositioning
Type:           REG_DWORD
Data value:     0 = basic
                1 = viewer adjusted using textLength
                2 = per word
                3 = per character
Default:        DEVMODE setting

Value name:     ImageEmbedding
Type:           REG_DWORD
Data value:     0 = generate images files external to the SVG document
                1 = embed images in SVG document
Default:        DEVMODE setting

Value name:     ImageConstruction
Type:           REG_DWORD
Data value:     0 = reconstruct original images using strict joining heuristics
                1 = reconstruct original images using less-strict joining heuristics
                2 = construct images by joining contiguous bitmap data
Default:        reconstruct original images using strict joining heuristics

Value name:     ImageResampleDpi
Type:           REG_DWORD
Data value:     0 = resample images to 96 dpi
                <value> = resample resolution
Default:        resample images to 96 dpi
Notes:          best output will result when the resample resolution is an
                integral fraction of the driver output resolution (288/576 dpi)
                suggested values are 48,96,144 or 288 dpi
```

```
Value name:   StatusReporting
Type:         REG_DWORD
Data value:   0 = errors displayed using dialog boxes
              1 = errors display is suppressed
              2 = errors and job status information logged to file
              3 = errors and job status information logged to file, file is appended to
Default:      errors displayed using dialog boxes
Note:         file name for logging to file is derived from the LogFile parameter
              for log file format details see Job Status Log File section below

Value name:   NotificationMessages
Type:         REG_DWORD
Data value:   0 = no progress notification messages
              1 = progress notification messages posted
Default:      no progress notification messages
Notes:        for further details see Progress Notification Messages section below

Value name:   NotificationId
Type:         REG_DWORD
Data value:   0 = use spooler job id
              <value> = use value as id
Default:      use spooler job id
Notes:        the id is used as a notification message parameter
              the id is combined into the notification object name
              this entry will be cleared once the print job has started

Value name:   DocumentTags
Type:         REG_DWORD
Data value:   0 = no document tags
              1 = generate document tags in the SVG document
Default:      no document tags
Notes:        inserts following tags at appropriate positions in document:
               <svgmaker:startdoc/>
               <svgmaker:startpage/>
               <svgmaker:endpage/>
               <svgmaker:enddoc/>

Value name:   NamePrefix
Type:         REG_SZ
Data:         text to be prepended to all ids used in the SVG document
Default:      no prefix
Notes:        prefix is limited to a maximum of 31 characters

Value name:   DocumentTitle
Type:         REG_SZ
Data:         name of document
Default:      use spooler document title
Notes:        title is limited to a maximum of 255 characters
              this entry will be cleared once the print job has started
```

DOCINFO Parameters

The DOCINFO structure is passed to SVGmaker via the Win32 StartDoc function during printing. It may be used to specify the output file via the DOCINFO.lpszOutput. The value specified will override any corresponding registry entry.

User filename entry

If neither DOCINFO.lpszOutput nor the corresponding registry entry are specified the output filename will be requested from the user via an open file dialog box.

Questions and Support

Send questions to support@svgmaker.com with a subject heading of

MakeSVG Support: <yourname>

When you report a problem please include the following information:

(a) What version of Windows you are using.

(b) What versions of Office programs you are using.

(c) The text of any error message(s) you see.

(d) A description of the problem and how to produce it.

May 2006
Brisbane, Australia