SwiftConvert User Manual

Updated for Version 9.0.7.2

© Paperbos LLC 2018

Table of Contents

© Paperbos LLC 2018

Introduction

SwiftConvert converts documents from PCL, HPGL, and TIFF into other formats such as PDF or TIFF. It consists of a command line batch converter program and, on Windows, a simple Multi-File Utility which allows converting groups of files without programming. SwiftConvert runs under Microsoft Windows and many UNIX platforms including Linux, HP, Solaris/Sparc, Solaris/x86, IBM RS/6000, and SCO.

SwiftConvert installs the same software as SwiftView, but the license issued to you enables the SwiftConvert functionality. SwiftConvert is a batch only utility. If you would like to view and export files with the SwiftView GUI, in addition to batch conversion, you must purchase a SwiftView Pro+Convert license.

SwiftConvert is built on and extends the SwiftView program. It inherits the easy programmability of SwiftView, which is described in both the *SwiftView Configuration* and *SwiftView ICS Reference Manual*s available from our website here:

http://www.swiftview.com/support/manuals/

For ease of understanding and brevity, this manual concentrates on the capabilities that are most relevant to SwiftConvert.

Chapter 1: Installation

Windows installation

The SwiftConvert installation procedure is identical on all MS Windows platforms: XP, Server 2003/8, Vista, etc. However, the installation procedures are slightly different for the two SwiftConvert licenses – single server or LAN license. Follow the procedure for the product you have purchased.

Step 1: SwiftConvert deliverables

When your purchase is complete you will receive an email from SwiftView containing a link to the SwiftConvert installer, and an attached license file. Click on the link in your email, which will take you to the SwiftView installer web page. Then click to install. You can also download and save the installer for later local execution.

Step 2: Unzip the license files

Your email from SwiftView contains an attachment ZIP file. Detach the ZIP file, save it in the SwiftView program directory (usually C:\Program Files (x86)\SwiftView), and unzip it. This file contains two files: SVIEW.LIC and SVLIC.TXT. SVIEW.LIC is the license file that enables the installed copy of SwiftConvert. SVLIC.TXT is a text version of the SwiftView license agreement. You are now ready to run with a single user or application server license.

> **NOTE:** SVIEW.LIC ***must not be modified in any way***. Doing so will disable it.

UNIX installation

Follow the download instructions on the SwiftView website, choose a directory in which to save the program (usually a "bin" directory included in your $PATH), save the zip file there, and then execute it. Then follow the license file installation procedure (Step 2) described above under "Windows Installation."

Windows Single User Licenses

A Windows single-user license is a file locked to a single computer.

Windows or UNIX Application Server Licenses

An Application Server license allows SwiftView Tools to be installed and executed by any number of users on a single physical Windows server using Citrix, Terminal Server, Remote Desktop, or any other remote access systems, or on a UNIX server or workstation. Each server in a Citrix server farm requires its own license. Any number of users can execute SwiftConvert installed on this server.

© Paperbos LLC 2018

Chapter 2: Primer

This chapter contains a series of examples that illustrate the use of SwiftConvert's command line converter with batch examples that can be automated.

Batch Conversion Using ICS Commands

SwiftConvert processes are controlled with ICS (Imaging Command Set) commands, SwiftView's API. The set of ICS commands is extensive, providing SwiftConvert with a wide range of capabilities. Chapter 3 lists the ICS commands most frequently used for conversion. See the *SwiftView ICS Reference Manual* for a complete list of all ICS commands.

ICS commands can be issued on the command line, in files, through pipes, and DDE. The primer examples in this chapter use the command line because this is the most common method. See the *SwiftView Configuration Manual* for a description of other methods.

There are two command line methods: **-c** and **-ics**. The primer examples use the more common **-c** technique, which starts SwiftConvert, suppresses the GUI, and exits when all of the ICS commands have completed. The **-ics** variant is similar to **-c**, but brings up the SwiftView GUI as it passes the command. SwiftConvert does not include the viewer, so the **–ics** parameter is only useful if you have SwiftView Pro, or SwiftView Pro+Convert license. Here are two common examples. On UNIX, a separate utility SVUTIL is used with the **-c** option.

Example: Convert PCL to PDF (Windows and UNIX)

This example reads a PCL file TEST.PCL and converts it to a PDF file DOC.PDF:

Windows: >sview -c"ldoc test.pcl | save PDF all doc.pdf onefile"

UNIX: >svutil -c"ldoc test.pcl | save PDF all doc.pdf onefile"

Example: Convert PCL to a TIFF

This example reads a PCL file TEST.PCL and converts it to a TIFF file TEST.TIF:

```
>sview -c"ldoc test.pcl | save TIFF all test.tif onefile"
```

Notice the quotes around the ICS commands; these are required. These examples use two ICS commands separated by the pipe | character. Any number of ICS commands may be linked together in this fashion. For example: >sview -c"cmd | cmd | cmd | cmd"

Two common ICS commands are used: **ldoc** identifies the source file, and **save** identifies the output format. Note that save has multiple parameters because you have to specify the output format. You do not have to specify the input format with ldoc because ldoc detects the input format automatically. These are the save parameters:

TF_G4 or PDF	Use TIFF Group 4, or PDF output
all	Convert all pages in TEST.PCL
test.tif *or* doc.pdf	Path/filename; the output data is written to this file
onefile	All output pages are stored in one single file. NOTE: without onefile, the output pages will be stored in multiple files, one page per file.

PDF Conversion

The example before the one above showed the most common conversion between PCL and PDF**.** The following examples demonstrate other useful ways to convert to PDF. These examples can also be used to output other types, such as TIFF.

Example: Convert a subset of PCL pages into a PDF file

This example extracts pages 2 - 3 from the PCL FILE TEST.PCL and converts them to the PDF file DOC.PDF.

```
>sview -c"ldoc test.pcl | save PDF 2-3 doc.pdf onefile"
```

Example: Convert to PDF and force letter size output

A source PCL file may contain a combination of paper sizes, most frequently, mixed letter and legal size paper.

© Paperbos LLC 2018

The output type PDF conserves the original paper sizes. The output type PDF_LETTER causes all pages to be output as letter size. The legal size pages will be downscaled to fit on a letter size. NOTE: there is no comparable TIFF example, because TIFF always saves at the original document size.

```
>sview -c"ldoc test.pcl | save PDF_LETTER all doc.pdf onefile"
```

Example: Convert legacy text file to PDF
Text files are ordinary ASCII files with lines of text terminated by line-feeds (UNIX) or carriage return and line feed (DOS, MS Windows). This is a very common report format from mainframe computers and from legacy applications. This example converts a legacy text file into a PDF file.

```
>sview -c"ldoc report.txt | save PDF all doc.pdf onefile"
```

Example: Convert legacy text report to PDF, using landscape orientation
Many older text reports use 100 characters per line and are designed to print in landscape mode; however the landscape command is not in the text file – instead the printer operator manually sets the printer to landscape, using the printer's front panel controls. The command pcl orientation landscape does the same thing for SwiftConvert: it tells it to assume landscape mode in its PCL print-rendering engine.

```
>sview -c"pcl orientation landscape | ldoc legacy.txt | save PDF all legacy.pdf onefile"
```

Example: Convert legacy text report to PDF, using portrait orientation
Some older reports used 132 characters per line, which, on letter size paper in portrait orientation requires a pitch of 16.66 characters per inch (the PCL default is 10). They also use 80 lines per page (the PCL default is 60). Use the command pcl to set SwiftConvert's rendering engine to the needed default state.

```
>sview -c"pcl fontpitch 16.66 linesperpage 80 | ldoc legacy.txt | save PDF all legacy.pdf onefile"
```

Converting Multi-page and Single Page Files

Most source documents consist of multiple pages. For most output types (PNG is an exception), SwiftConvert can save all of the converted source pages into a single output file (a single, multi-page file). However, you may want to have each file written out individually (multiple single page files). In these cases, SwiftConvert writes each page to a file, appending a sequential numerical suffix to each output file name, for example: DOC.001, DOC.002, DOC.003, etc. This function is not very useful for PDF, but can be quite useful for TIFF, since it allows the user to delete, add or move individual pages. In addition, for TIFF, SwiftConvert can reassemble the individual files back into a single multi-page file.

Example: Convert PCL to multiple single page TIFF files
Assume for this example that the PCL file TEST.PCL is a three-page file. The command below converts the PCL data into three individual TIFF files: DOCS.001 (page 1), DOCS.002 (page 2), and DOCS.003 (page 3). When SwiftConvert loads a TIFF file with the naming convention, NAME.001, it assumes that it is the first page of a multi-page file and automatically scans for files with the same name and sequential suffixes. SwiftConvert stops scanning when it finds the first missing sequential suffix.

```
>sview -c"ldoc test.pcl | save TF_G4 all docs"
```

Example: Convert PCL to multiple single PDF files
This converts the PCL file into multiple PDF files: PDFPAGE.001, PDFPAGE.002, etc. NOTE: SwiftConvert writes PDF files, but does not read (display) PDF files unless you have purchased SwiftView Pro+PDF or SwiftConvert+PDF. Thus, the base SwiftConvert product cannot convert multiple single page PDF files into a single, multi-page PDF file, as it does in the TIFF examples that follow this example.

```
>sview -c"ldoc test.pcl | save PDF all pdfpage"
```

Example: Convert multiple single page TIFF files into a single multi-page TIFF file
SwiftConvert takes the 3 individual TIFF files DOCS.001 through DOCS.003, and writes out a single 3-page TIFF file DOCS.TIF.

```
>sview -c"ldoc docs.001 | save TF_G4 all docs.tif onefile"
```

Example: Convert a multi-page TIFF file into multiple single page TIFF files
The reverse of the previous example, this one writes files NEWDOCS.001 through NEWDOCS.003.

© Paperbos LLC 2018

```
>sview -c"ldoc docs.tif | save TF_G4 all newdocs"
```

Text Extraction

NOTE: The following discussion applies only to PCL and HPGL source files.

SwiftConvert can extract text from a source PCL file and convert it into ordinary text. Ordinary text is mono-spaced courier Latin 1 symbol set with carriage return and/or line feed line terminations. Because it has one character size, SwiftConvert can only approximate the original text layout. However, using spaces, it tries to maintain the original column alignment. This is the same text that SwiftView searches and copies to the Windows clipboard.

SwiftConvert has two extraction modes. TEXT tries to keep the original paragraph and page layout. TEXTPOS makes no attempt to preserve page layout; it writes one word at a time with that word's physical bounding rectangle.

SwiftConvert does not attempt to OCR text, it extracts text that resides in PCL (or HPGL) format in the source file. If the PCL file does not use Latin-1 (or ASCII) encoding, then the text is not readable. This is unfortunately common in files using embedded download fonts. If you extract text and see gibberish, then your file has non-standard ASCII text. For example, if you expect to see "dog" but get "!#s" or some other random sequence, then your file has garbled, or scrambled, text. Another common problem is text that is a raster image. When displayed, a raster (or bit-map) file looks like it has ordinary text, but the text is only a graphic image, not actual text.

The main source of garbled text problems is in files produced by Microsoft Windows printer drivers. We have never seen a case of garbled text from mainframe report writers or application programs that generate their own PCL output. No application that uses printer-resident fonts has a problem. For these applications, TEXT output can be extremely useful. You can use it to produce text suitable for searching, indexing, etc.

Example: Convert PCL file to TEXT

This produces an ordinary text file output. Individual pages are separated by form-feeds.

```
>sview -c"ldoc test.pcl | save TEXT all test.txt onefile"
```

Example: Convert PCL file to TEXT with position information

This command extracts the text form a PCL file.

```
>sview -c"ldoc test.pcl | save TEXTPOS all test.txt onefile"
```

The text below is an example of the output syntax when using TEXTPOS.

```
page 1
4.437 0.543 5.067 0.726 Chapter
5.113 0.543 5.207 0.726 6
5.347 0.543 5.743 0.726 UNIX
```

A page number appears at the beginning of every page. Each line presents two coordinate pairs followed by a word. Each coordinate pair is a diagonal corner of a bounding box followed by the word that is bounded by the box. Coordinates are in inches, measured from the upper left corner of the page. The upper left corner is given first, followed by the lower right corner.

The Printer Command and SwiftConvert's Print Capacity

Above are some examples that use SwiftConvert's **save** command. The most common outputs formats are PDF and TIFF. The following examples use the **printer** command for output formats that are associated with printers, including PCL4, PCL5, PostScript, and HPGL. These commands can send the converted data either to a file or directly to a printer, which can be very useful. For complete command syntax, see Chapter 3: Printer Formats and the ICS Printer Command

Example: Convert a TIFF file to HPGL format

This example reads a TIFF file and converts it into HPGL, a format suitable for an HP Plotter. This example tells SwiftConvert to output a raster HPGL image using the closest US paper size (A through E) that fits the original TIFF drawing size.

© Paperbos LLC 2018

```
>sview -c"ldoc drawing.tif | printer number 1 type RHPGL_US command FILE alias none | set
filename drawing.hpg | plot 1 all"
```

The ICS commands must set up a number of things for SwiftConvert:

printer number 1 type RHPGL_US command FILE alias none
: This defines a printer and gives it the ID number 1. The output format for the printer is RHPGL_US. The printer port is command FILE, i.e., output to a file, and alias none indicates no alias name for the printer.

set filename drawing.hpg
: Sets the path/filename to which the data is sent.

plot 1 all
: Use printer number 1 to print all pages.

Windows and UNIX Printer Examples

These examples show how to send SwiftConvert data directly to a printer. Most of the examples apply only to MS Windows installations, because they involve using MS Windows drivers to produce driver-specific output.

Example: Convert a TIFF file to PCL and output to a printer (Windows)

This is the first example that does not write to a file; instead, SwiftConvert writes directly to a printer.

```
>sview -c"ldoc document.tif | printer number 1 type PCL5 command none alias ""Lexmark
Optra"" | plot 1 all"
```

Notice the changes in the printer command:

type
: PCL5 indicates PCL5 output. This command retains the original source page sizes. If the source file contains mixed letter and legal sizes, then the PCL output will have mixed sizes.

command
: none because we are not outputting a file and do not need the FILE parameter

alias
: ""Lexmark Optra"" the printer friendly name*

*NOTE: double quotes are required here first, because the printer name has a space in it.

The alias parameter selects a specific printer. The alias name is unique to your installed set of printers. This example uses the printer's "friendly name" (Microsoft's terminology). This is the name associated with a printer in MS Windows' Printers menu. To see this menu, use the following Windows clicks: Start Button → Settings → Printers. In this example, "Lexmark Optra" is the name of a PCL5 compatible printer.

Example: Convert a TIFF file to PCL and output to a printer (UNIX)

This example shows how to use SwiftConvert to stream data directly to a UNIX printer. SwiftConvert writes each page to the system temp place (e.g., /usr/tmp) and immediately executes the print command specified by command. Please see the *SwiftView ICS Reference Manual* for more information regarding print files.

```
>svutil -c"ldoc document.tif | printer number 1 type PCL5 command \"lpr –r\"
alias none | plot 1 all"
```

Example: Convert a PCL file to PostScript and output to printer port LPT1 (Windows)

This example reads a PCL file, converts the output to PostScript and writes the data to printer port LPT1.

```
>sview -c"ldoc document.pcl | printer number 1 type POST2 command none alias
""na,na,LPT1: "" | plot 1 all"
```

The example above this one used the printer's *friendly name* [Lexmark Optra] but what if you don't know the *friendly name*? We have seen cases where the system manager did not know ahead of time what a printer name was going to be, but knew the printers were always on LPT1. Using the alias "na,na,LPT1: " the command above always outputs the PCL generated by SwiftConvert to LPT1.

Example: Convert a TIFF file to PCL and output to the default printer (Windows)

In this example, the alias "na,na,na" does not specify a specific printer or port. SwiftConvert writes to the default MS Windows printer.

```
>sview -c"ldoc document.pcl | printer number 1 type PCL5 command none
alias ""na,na,na"" | plot 1 all"
```

© Paperbos LLC 2018

Example: Read a TIFF file and output using an MS Windows printer driver
In this example, the type MS_WIN tells SwiftConvert to use the MS Windows printer driver to produce the output format.

```
>sview -c"ldoc document.tif | printer number 1 type MS_WIN command none
alias ""Lexmark Optra"" | plot 1 all"
```

Example: Read an HPGL file and save to file using an MS Windows printer driver
This example also uses type MS_WIN to select the MS Windows driver. It reads an HPGL file DRAWING.HPG and saves an output file generated by an MS Windows driver, DRAWING.PRN. Note that the driver used is the driver associated with the *friendly name* in the alias parameter.

```
>sview -c"ldoc drawing.hpgl | printer number 1 type MS_WIN command FILE
alias ""Lexmark Optra"" |  set filename drawing.prn | plot 1 all"
```

More Complex ICS commands

The following examples illustrate how ICS commands can be linked together. Up to now, examples have used two ICS commands, ldoc and save. The following examples use five ICS commands to load, markup, and convert to TIFF.

Example: A more complex ICS command

```
>sview -c"ldoc test.pcl |
onpage all markup attributes coordinates device xorigin 0 yorigin -0 |
onpage all markup text font ""face serif size 14"" |
onpage all markup text rxloc 0.25 ryloc -0.25 string ""Preliminary"" |
save TF_G4 all marked_doc.tif onefile"
```

The example above is written on multiple lines for clarity – on a real command line it would be one long string. This series of commands tells SwiftConvert to 1. read the file DOCS.001, 2. mark the bottom of every page with the word “Preliminary” one quarter of an inch from the lower left corner, and 3. save the result as a TIFF file. Don't worry about understanding this example; its primary purpose is to show a more complex ICS command. But, for those interested – here is exactly what happens:

ldoc test.pcl
: Loads the imaginary example file, TEST.PCL.

onpage all markup attributes coordinates device xorigin 0 yorigin -0
: Establishes a coordinate system on all pages. DEVICE indicates that coordinates are relative to the paper (page). XORIGIN 0 YORIGIN -0 establish coordinates at the lower left corner of the paper, regardless of paper size (letter, legal, etc.)

onpage all markup text font ""face serif size 14"" |
: Note that the use of double quotes "" is required. This sets the font to the default serif face (Times Roman) at 14 points.

onpage all markup text rxloc 0.25 ryloc -0.25 string ""Preliminary""
: Writes the word “Preliminary” in the font just established, at a quarter inch right of and above the lower left corner.

save TF_G4 all marked_doc.tif onefile
: Saves the resulting marked up pages in the single multi-page TIFF file called MARKED_DOC.TIF.

Example: ICS command file
ICS commands are not limited to the command line. You can also create files that contain ICS commands. For example, below we create an ICS file called MARK.ICS that has the ICS commands from the previous example:

```
ICS
# Example ICS file
ldoc test.pcl
onpage all markup attributes coordinates device xorigin 0 yorigin -0
onpage all markup text font "face serif size 14"
onpage all markup text rxloc 0.25 ryloc -0.25 string "Preliminary"
save TF_G4 all marked_doc.tif onefile
```

© Paperbos LLC 2018

The following command line reads and executes the entire ICS file (MARK.ICS) above.

```
>sview -c"ldoc mark.ics"
```

Note that all ICS files have ICS as the first line. Lines beginning with #are treated as comments. Each ICS command is on a separate line, so there is no need for the pipe | separator. And since this is not a command line, single quotes are used.

Error log examples

Normally SwiftConvert runs silently. However you may want to log error messages, especially when you are writing complex ICS commands.

Example: Error reading a file

You cannot load (ldoc) a PDF file, so if you run the command below, SwiftConvert exits silently because it cannot read PDF.

```
>sview -c"ldoc doc.pdf | save PDF all docs"
```

Example: Seeing errors in a log file

If you want information about the file error, use this example:

```
>sview -c"callback filename log.txt | ldoc doc.pdf | save PDF all docs"
```

Once you run the above command line and after it exits, examine the file LOG.TXT to see the following text: SVC:ERROR: SwiftView does not read Postscript files. Note: SwiftConvert does not distinguish between PDF and PostScript and reports the same error for both. There is a wealth of information returned in the callback file. For a comprehensive discussion, please see the *SwiftView ICS Reference Manual*.

Chapter 3: Reference

Common ICS Commands for SwiftConvert: save and ldoc

This manual does not duplicate the ICS definitions in the *SwiftView ICS Reference Manual*, but does add explanatory material relevant to conversion. The two most used ICS commands are ldoc, which identifies an input path/filename for conversion, and save, which outputs the file.

```
ldoc filename
save type range filename [onefile] [other parameters]
```

Details of the save parameters are as follows:

type selects the output file format

Type	Description
PDF Types	
PDF	PDF, retain original size
PDF_LETTER	PDF, scale pages to letter
TIFF Types	
TF_G4	TIFF, group 4 compression, original size
TF_UNC	TIFF, uncompressed, original size
TF3_96	TIFF, group 3 compression, low resolution fax, scale to letter
TF3_192	TIFF, group 3 compression, high resolution fax, scale to letter
PNG Types	
PNG_MONO	PNG, retain original size
PNG_MONO_LETTER	PNG, scale pages to letter
CALS Types	
CALS	CALS, Raster type 1 file, original size. NOTE: CALS does not support the onefile option
ASCII Text types	
TEXT	ASCII text file

© Paperbos LLC 2018

TEXTPOS	ASCII text with text bounding boxes

range indicates which pages you want to convert

all	Convert all pages
m-n	Page ranges separated by commas, e.g., 2-7 or 2-7, 20-25, 80-100
odd	odd pages only
even	even pages only
reverse	reverse page order

filename is the output filename.

onefile (optional) assures that all pages will be stored in one file: filename. The default, if onefile is absent, puts each page in a single file with sequential names, e.g., filename.001, filename.002, filename.003, etc.

other parameters there are many other infrequently used parameters; see the *SwiftView ICS Reference Manual* for descriptions. Other parameters allow you to select a rectangular cut area for output. You can also convert less than the whole page, which is useful for TEXT output modes where you are looking for text on a particular area of the page, such as address.

Other related issues with Save commands

Color

With the release of SwiftConvert 8.0 we can convert color input files (PCL5C, PCL6, JPEG, and TIFF) into color output files (PDF, TIFF, PCL5C). Color files can also be converted down to monochrome.

TIFF output paper sizes

The TIFF output types TF_G4 and TF_UNC both save at the paper size of the original document. There is no facility to force the paper size to letter. The fax types T3_96 and T3_192 always output to letter size. These files use Group 3 compression and fax resolutions, because they are optimized for use by fax server software.

Output Resolution

SwiftConvert renders all PCL files at their original resolution, if the file is 300DPI we display it at 300DPI, if 600DPI it is displayed at 600DPI. By default SwiftConvert will also output at the documents original resolution. If you need to manually set the output resolution, you can use the ICS command "set printres n", where n is the desired DPI. Here is an example:

```
>sview -c"ldoc 600dpifile.pcl | set printres 300| save PDF all docs"
```

The above command opens a 600DPI file, the saves it as a 300DPI PDF. Lowering the output resolution can help lower the output file size, but keep in mind that the clarity is reduced as well.

TIFF output resolution

TIFF files are commonly used to store large engineering drawings. SwiftConvert is frequently used to convert HPGL plotter drawings to TIFF format. However, HPGL is a vector representation, which means that data such as lines are stored as a series of numerical coordinate pairs. These coordinates have a default accuracy of 1016 pixels per inch. Since this is not a raster representation of data, the accuracy does not contribute to file size. However, to store the TIFF file at 1016 pixels per inch would result in huge raster files. For example: an E size drawing, which is 34x44 inches would have a raster of 194 MB uncompressed (34*44*1016*1016)/8. Because such large rasters are undesirable, SwiftConvert uses a set of rules to choose output resolution, as follows:

> If the HPGL drawing is less than 11x11 inches, 300dpi is used, else 200 dpi.
> For HPGL or any other input source: if the resolution results in a raster buffer>32 MB, one of the following resolutions is tried until the raster buffer<32 MB:
> 300, 200, 100, 75.

HPGL/2

SwiftConvert reads HPGL/2 files, as defined in *The HPGL/2 and HP RTL Reference Guide, Third Edition*. This book actually defines two separate specifications: HPGL/2 and HP RTL. Historically, HPGL/2 was the format used by pen plotters, whereas RTL was the format used by large format raster plotters. The distinction between these plotters has blurred in recent years. Also adding to the confusion are MS Windows drivers that are called

© Paperbos LLC 2018

HPGL drivers but use RTL. SwiftConvert only reads HPGL/2. SwiftConvert cannot render RTL properly, and there is currently no plan to add full RTL support. Contact SwiftView Support if you need RTL support.

Printer Formats and the ICS Printer Command

Several output formats are related to printing. These include Postscript and PCL. The printer ICS command, mentioned earlier, gives you access not only to the formats that SwiftConvert generates itself, but also to all the formats generated by any printer driver on your system. It also lets you choose between saving the data in a file or streaming it directly to the printer. Below is the printer command syntax and common parameters.

printer number pn **type** pt **command** cs **alias** as

number - pn is a printer ID, an arbitrary number between 0 and 89, inclusive.

type - **pt** selects the printer type; below are the most common and when to choose them:

Conditions for Using	Type	Details
To copy source file directly to the output	DIRECT	Copies the input format directly to the output
To generate **Postscript** output	POST2	Postscript level 2, 300 dpi, preserves the original paper sizes
To generate **PCL5** output	PCL5	PCL5, preserves the original paper size
To generate **HPGL**/RTL output	RHPGL_US	HPGL/RTL, uses paper sizes between A - E
or	RHPGL_EURO	HPGL/RTL, uses paper sizes between A0 - A4
To have MS Windows drivers generate the output	MS_WIN	Uses MS Windows driver

command for **cs**, use the parameter FILE if you want to redirect the print output to a file, and use NONE if the output is going to a printer.

Alias – **as** identifies a printer from one of these:

friendly name	The MS Windows name displayed to users, e.g., HP LaserJet 4050 Series PCL
na,na,port	The MS Windows port name: LPT1:, LPT2:, COM1:, COM2:, etc.
na,na,na	The MS Windows default printer

NOTE: In ancient times printers were identified by a triplet: name,driver,port. The alias method still supports this mode, e.g., Optra RT+,LMPCL5C,LPT1; however, today this form is almost never used.

Other common ICS commands

plot printer_id range [other] [onefile]

The printer command defines the characteristics of a specific print process. SwiftConvert maintains a list of these printer commands – up to 90 entries. Each entry is identified by an arbitrary number in the range of 0 – 89, inclusive. In a batch application it would be unusual to ever define more than a single print process. The plot command invokes one of these predefined print processes; it actually causes printing to happen. The command is called plot instead of the more logical "print" for obscure historical reasons.

printer_id	This is a number 0-89 which links to a printer ICS command.
range	Indicates which pages are to be printed: all Convert all pages m-n Page range(s) separated by commas, e.g., 2-7 or 2-7, 20-25, 80-100 odd odd pages only even even pages only reverse reverse page order
other	less common parameters; see the *SwiftView ICS Reference Manual*
onefile	all output pages are stored in one single file

set filename path/name

This identifies the path and filename when printing to file. Print to file is selected by the command FILE parameter in the printer command.

© Paperbos LLC 2018

callback filename path/name

Use this ICS command when you want to log callbacks associated with SwiftConvert ICS processes.

Controlling the PCL rendering process

There are several ICS commands that, in effect, act like the front panel controls of a printer. They set up default conditions. Normally you do not have to use these commands, because most PCL files contain control sequences that specify their requirements explicitly.

As an example, consider the printer default for page size: letter. A PCL file that needs legal size paper will, in almost all cases, contain a command that sets the paper size to legal. But if the file does not have a command setting legal size, the printer will truncate the bottom part of the page. To make the printer choose legal size, the user must manually set the paper size to legal on the printer's front panel. SwiftConvert must also be told to assume legal size instead of letter size, because it acts just like the printer.

Text files are another source of data that may need PCL ICS commands. The term *text* refers to files that use only ASCII text and are terminated with Carriage Return and/or Line Feed characters. They may also (but often do not) use Form Feed characters between pages. These are valid PCL files, but contain no controls on fonts, paper size, orientation etc. They are most often associated with legacy printing applications, such as large multi-page reports. We have customers with files containing hundreds of thousands of pages in this form.

A few of the most common PCL ICS commands for controlling rendering are listed below; see the *SwiftView ICS Reference Manual* for a list of the less common ones. First is the **pcl** command, with basic syntax:

pcl [pagesize n] [orientation o] [linesperpage n] [fontpitch p] [reset]

pagesize n	Sets the default paper size. The values of n are taken from the *PCL Printer Language Technical Reference* manual. The most common values are: 2 Letter 3 Legal 6 B size 26 A4 27 A3
orientation o	Selects paper orientation. Allowed values for o are: landscape and portrait
lineperpage n	Sets the number of lines per page (integer value). The default value for PCL is 60.
fontpitch p	Sets the font characters per inch, e.g., fontpitch 16.66
reset	Sets all values back to factory default.

pcl edgetoedge on/off

Normally, a PCL printer has a 0.25-inch margin that it cannot print on, due to older printers' rubber grips. Some high-end printers can print all the way to the edge of the paper. Use pcl edgetoedge on when you want SwiftConvert to emulate those printers.

HPGL Pen Colors and sizes

SwiftConvert renders HPGL/2 vector files. Most files specify pen color and size, however some legacy files do not. This is because HPGL selects pens by numbers – such as pen number 1. The actual color of pen number 1 was whatever the operator put in the plotter's pen carousel. SwiftConvert has command for this default data:

hpgl pen [n|all] red n green n blue n width w

pen n	a pen number form 1 to 255 inclusive, e.g., pen 8
red n	0 – 255, the red component of an RGB pixel
green n	0 – 255, the green component of an RGB pixel
blue n	0 – 255, the blue component of an RGB pixel
width w	a generic pen width, or a real number specifying the width in mm or as follows: **normal** generic average pen **bold** generic bold pen **verybold** generic heavy pen **fine** 1 pixel wide **0.3** Any real number, in mm

© Paperbos LLC 2018

ICS commands that can affect SwiftConvert

The following is a list of ICS commands that can affect SwiftConvert, but are not commonly used. You may want to look at these commands in the *SwiftView ICS Reference Manual* if you have more complex issues related to topics below.

markup	Markup capability is extensive
pcl oldfonts	Legacy font support
search	Extensive text search capability for PCL/HPGL files that contain searchable text.
set invert image	Permits inverting an image: black becomes white, white becomes black
set allpensblack	Assumes all HPGL pens are black

Commands that control text:

set fractionallinefeed	Sets the minimum vertical text advance interpreted as a linefeed
set fractionalspace	Sets the distance at which words are divided

The following commands all set how the print area is used when the source is not an exact (or close) fit. Refer to the *SwiftView ICS Reference Manual* for details on their use.

set printsize
set printarea
set printscale
set printautorotate
set printxposition
set printyposition

There are also a number of commands that affect color, resolution, searchable text, PJL handling, duplex, and tray handling:

set printres
set printwintray
pcl defaultres
pcl forceres
set printcolor
set printbitonalscaling
set printpjl
set printpjlcmds
set printtext

There are many available functions in SwiftConvert that may not be easily found in the reference manual. If you have any questions or thoughts on features please Contact SwiftView Sales, and we may be able to easily point you the right direction.

© Paperbos LLC 2018

Chapter 4: Integration

SwiftConvert can be easily integrated into your own application using Visual C++, Visual Basic, or other Win32 programming methods, by running it using functions such as CreateProcess(). You can also use SwiftView's ActiveX control, or our SwiftInside DLL, however for basic conversion in most cases this is unnecessarily complicated.

If you are looking into integration, please feel free to SwiftView Sales. We will be glad to assist you and answer any questions you may have.

© Paperbos LLC 2018

Index of Examples

© Paperbos LLC 2018