SwiftView

Configuration Manual

Revised for SwiftView 9.0

Introduction

The SwiftView Imaging Command Set (ICS) puts you in charge of SwiftView's appearance, features, and configuration without complex programming. A simple file placed on a website or a LAN can configure all users instantly, or commands can be placed in files or typed on the command line. No other viewer that we know of offers so much control over the viewing experience. ICS commands can be used to accelerate printing, create your own buttons, convert files in batch mode, and hundreds of other functions.

This manual explains how to do general setup and customization of SwiftView including how to:

Install Swiftview and SwiftView licenses
Properly set up a website for use with SwiftView
Use SwiftView's ICS commands to customize SwiftView.

We will also explain general ICS guidelines, sentence structure, format, and useful examples to point you in the right direction. All ICS commands are listed in the ICS Reference Manual located on our website. If you have any detailed questions regarding the use of ICS commands, please feel free to contact SwiftView Technical Support at www.swiftview.com/support/contact-support/

Table of Contents

Chapter 1: How to install SwiftView

The following section explains the files installed by SwiftView as well as more complicated LAN installation options.

Standalone SwiftView for Windows

Since version 5.2, SwiftView for Windows comes packaged in an automated installer that installs SwiftView and support files in C:\Program Files\SwiftView, registers SwiftView for its usual MIME types and file suffixes, and creates a Start menu entry. The installer is available as either a signed executable (which can be downloaded by IE or copied to a system and run), or an "XPI file" that can be run automatically by Netscape or Firefox.

Upon installation, you will see at least the following files installed in c:\Program Files\SwiftView:

temp	Folder for log and temp files
mime_sample.typ	Sample mime type file
startup.zhp	File shown when opening SwiftView directly.
SVIEW.EXE	SwiftView executable - Win 98SE/NT/2K/Me/XP
sview_sample.ini	Initialization file - optional sample
sview.inf	Setup information accessed during installation
sview.str	Master English translation string file
sview.tra	Default translation string file
sview_Dutch.tra	Dutch translation string file
sview_French.tra	French translation string file
sview_German.tra	German translation string file
sview_Spanish.tra	Spanish translation string file
svinst.exe	Installation file
svsainst.ocx	OCX component for uninstall purposes

SVIEW.INI is optional and may reside either in the same directory as SVIEW.EXE or in your Windows directory (e.g., C:\WINDOWS) or both. This file is described in more detail in Chapter 2: How to Configure SwiftView.

.TRA files allow translation of all SwiftView strings in any Western European (LATIN-1) language. They are selected automatically based on the Windows Regional settings. You can easily create your own for another language, or to modify selected strings. See "Native Language Support" in the spec file at: http://www.swiftview.com/tech/npextspec.txt. This file provides details about translation, and information on controlling the registration of MIME types and file suffixes. Services are available to build custom installation packages, e.g., for custom font files, button icons, logos, native language translations, and file suffixes and MIME types. Contact SwiftView Technical Support at www.swiftview.com/support/contact-support/ for more information.

SwiftView can be deinstalled from Windows Control Panel Add/Remove Programs, or using an uninstall package executable which can be downloaded and run by Netscape or IE. All SwiftView files are removed on a deinstall.

To run a silent install from a batch file or command prompt:

```
installername.exe -P-nosmsg –s
```

LAN Installation Options

There are a number of ways to setup a LAN installation, including some not listed below. The suggest methods are listed in order of ease of use, please contact support www.swiftview.com/support/contact-support/ if you are curious about our other installation methods.

Client Install from LAN location

In this method you simply place the LAN license in the network location, and place the installer in the same location. The client machines should then just browse to that location and run the installer. SwiftView will be installed on the client, and the registry key will be created that points to the LAN license. Note: This install method will require all client machines to reinstall in the same manner when you are upgrading.

Running Swiftview From the Server Without Installing On the Client

To use this mode, simply install SwiftView on a server and run the SVIEW.EXE located on that server from the client machine. At its simplest, this can consist of placing only the program SVIEW.EXE on the server and having clients run it. You can add a shortcut on the client desktop that points to the networked SVIEW.EXE and the users can run it. Note: This method does not associate any file types on the client machine, so users will not be able to "double-click" PCL files and have them automatically open in SwiftView. They will need to always run SwiftView and use the "Open" button.

Thin Client, Thick Server

If you administrate a Windows LAN it is possible to have all the executables for browser-based and standalone SwiftView on the server, with virtually no components installed on the client. Using the following install method will create all of the file association on the client machine, to allow "double-click" opening.

The standard SwiftView installers are used to install in a non-standard location, e.g., on a LAN server. The .exe installer accepts a -P-p[dir] command line option that specifies the location of the SwiftView program directory. If the directory is not given, the installer invokes a dialog to select the location, choosing the standard SwiftView program directory as the default selection. Note: paths longer than 85 characters (including sview.exe) given to the standalone installer will hinder the ability to drop a SwiftView file on a printer icon due to Windows command length limitations.

1. Create a directory on the server where you want to install SwiftView. For explanatory purposes, we will consistently use the following directory: N:\sview\bin\sv_lan_install\

2. If you purchased a LAN license from us, copy the license file (SVIEW.LIC) to N:\sview\bin\sv_lan_install\

3. Download installers: go to http://www.swiftview.com/support/current-customers/ and enter your SwiftView ID, then download your appropriate installer.

4. Run the installers on the server using the -P-p switch: execute each installer chosen in Step 2 followed by -P-p and the name of the directory established in Step 1. It is perfectly acceptable to execute these installers on a client, that is, you don't have to be sitting at the server machine for this step. For example, after changing directories at the DOS prompt to N:\sview\bin\sv_lan_install, type:

```
svinstall_s_libs.exe -P-pN:\sview\bin\sv_lan_install
```

Note that there are no spaces in the above line except between the .exe and the -P. The -P-p switch tells the installer "This is a LAN installation, don't install on the client, install in the following directory".

5. Install on the clients: On each client, do the following:

A) Execute the installer in the directory specified in Step 1. This can be done by double-clicking on it or from the command prompt.

B) When the "LAN License Detected" dialog appears, click the "Use LAN Version" button. This will ensure that the executables remain centralized on the server. If you choose "Install New", SwiftView will be installed on the local C: drive and will not be centralized.

Upgrading a LAN Install

To upgrade SwiftView when a new version is released: simply repeat steps 3-4 above. If the Netscape plug-in is being used, repeat steps 3-5. NOTE: your support and maintenance must be current to upgrade. If you want to get another copy of SVIEW.EXE without the installer package please contact technical support at www.swiftview.com/support/contact-support/

To deinstall SwiftView from a client who installed from the LAN, use the normal means through Add/Remove Programs.

LAN Licensing using Registry Keys

Another way to do the client install and have it find the LAN license is by setting a registry entry. This may be the most useful for administrators who need to role out an installation to a large number of computers on a network. For this type of installation just follow these simple steps:

1. Install SwiftView on each client using the regular installer with no special parameters

2. Run "regedit" and add a String Value named "lanlicense" in this location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SwiftView\
```

3. Make "lanlicense" equal the location of the LAN license file SVIEW.LIC

SwiftView Plug-in/ActiveX (browser based SwiftView)

Netscape Navigator/Communicator 4+ (NOT Netscape 6+) and Microsoft Internet Explorer 4+ are supported. The SwiftView Plug-in/ActiveX installer is much the same as for Standalone SwiftView, available either as an executable (.EXE), or a Cross Platform Installer (.XPI) for Netscape and Firefox. The SwiftView Plug-in/ActiveX installer installs all of the same files as Standalone SwiftView, but also includes SVOCX.OCX, the SwiftView ActiveX control. Note that since Browser-Based SwiftView is intended to be seamlessly invoked when browsing websites it does not add shortcuts to the Windows Start Menu.

The plug-in is installed by/for Netscape and the ActiveX control by/for Internet Explorer. The installation registers the filename suffixes and MIME types supported by SwiftView so that SwiftView will be run in the default browser to view SwiftView-supported documents.

If you would like to set up your own installation page for Browser-based SwiftView, please refer to the Website Configuration chapter later in this manual.

Deinstallation of browser-based SwiftView is also accomplished through Add/Remove Programs in the Windows Control Panel.

Chapter 2: How to Configure SwiftView's Appearance and File Handling

There are many ways to configure SwiftView's appearance and file handling. All of these incorporate SwiftView's ICS commands and can be passed through various configuration files as well as from a command prompt. The following sections detail each of these configuration methods and the formats in which to use them. All ICS commands are found in the ICS Reference Manual, and to point you in the right direction we have provided a number of examples of commonly used ICS commands at the end of this chapter.

Windows Standalone SwiftView Configuration file (sview.ini)

To configure Standalone Swiftview with ICS commands, the SVIEW.INI file needs to be edited. On Windows Systems it is defaulted to c:\Program Files\SwiftView\, however on LAN installations it should be located in the same directory as the SVIEW.EXE.

The SVIEW.INI must begin with the string "[SwiftView]" on the top line, and each ICS command line must begin with "ICS#=", as illustrated here:

```
[SwiftView]
ICS0=firstcommand
ICS1=secondcommand
ICS2=thirdcommand
etc.
```

Only SVIEW.INI for Standalone Windows SwiftView uses this "ICS#" format. We provide an example SVIEW.INI in the installation, which demonstrates the most popular features, such as fast printing, direct printing, HPGL pen colors and markup. You can comment out lines using the # character, but you must renumber the remaining ICS commands so that there are no missing numbers. Once ICS commands are placed in this file, Standalone SwiftView will run these commands as it starts up.

ICS Files (*.zhp or *.ics)

A plain text file, which begins with the letters "ICS" on the top line, can contain a series of ICS commands. This file can be given any filename suffix, however Outlook 2000 claims the suffix ".ics". We recommend the suffix ".zhp" for any client systems that have Outlook 2000 installed. When SwiftView opens this file, the commands are executed. Opening an ICS file does not overwrite existing ICS commands passed from other locations, such as the SVIEW.INI, unless they are in direct conflict to previous commands. Otherwise the new commands are just added as well.

Here is an example of such a file, which will produce a 2-page "virtual document" using single-page files:

```
ICS
onpage 1 use sourcename http://www.swiftview.com/tech/5x7plate.tif
onpage 2 use sourcename http://www.swiftview.com/tech/1page.hpg
draw all
```

NOTE: the format begins with ICS and has each command on the following lines, without the numbering scheme required for the SVIEW.INI. For more information on Virtual Documents, please see Chapter 4: Advanced Features.

UNIX Standalone Configuration File (SVSTART.ICS)

The file SVSTART.ICS should be located in the same directory as the SwiftView executable, and acts on UNIX just as the SVIEW.INI does on Windows: it sets ICS commands to be run when starting SwiftView. The SVSTART.ICS however has a different format then the SVIEW.INI; it is in the same format as a standard ICS file described above. For more information regarding UNIX configuration, please see Chapter 6: UNIX Installation and Configuration.

Browser-based Configuration (NPSVIEW.ICS)

The NPSVIEW.ICS file is an ICS file that globally configures browser-based SwiftView for a website. When this file is placed in the root of the web server, it configures SwiftView for all users while viewing files on the website.

If you are planning on using SwiftView on your website, please refer to Chapter 3: How to Configure a Website. Using the global NPSVIEW.ICS on your website can be a viable solution for your configuration, but customizing SwiftView in other ways, e.g., through Embed or Object tags, can provide much greater flexibility and control over SwiftView's many features.

Embed and Object Tags

SwiftView can also be called in an Embed or Object tag on a web page. Through the scripting on your web page this can allow you to dynamically pass ICS commands via these HTML tags. Because of the complication of this, and the need for sample code displaying how to use it, we have given much more detailed information regarding this in Chapter 3: How to Configure a Website.

Command Line Parameters and Batch mode

The sview executable (sview on UNIX, SVIEW.EXE on Windows) can be passed parameters to configure buttons and settings, as well as sent ICS commands at the command line. This can be used to implement SwiftView in a background process or batch mode. All of the following parameters are used at the command line by typing the parameter after the SwiftView executable.

-ucccc **Button Control**

In the -u command, cccc specifies any combination of the following letters, which may be used in any order to select and order the buttons:

E	Exit
A	Draw all
W	Draw widetop
R	Rotate
P	Print
F	Open (File dialog)
S	Save (Save TIFF file dialog - not part of the default button set)
H	Help

<u>**SwiftView's Default:**</u> sview –uEAWRPFH

If any buttons are defined in SVIEW.INI using the "gui button" ICS command, these -u defined buttons will be replaced.

-wxa,ya,ms,xo,yo **Set viewing area size and shape**

xa	x aspect ratio component of viewing area
ya	y aspect ratio component of viewing area
ms	Maximum amount of screen to use (0-1)

xo Optional horizontal (x) offset.
yo Optional vertical (y) offset.

<u>SwiftView's Default:</u> -w8. 5,11,.8
This default sets the aspect ratio of the SwiftView drawing area to appropriately that of a letter size page (8.5x11 inches), and sets the total size to use 80% of the height of a standard screen. This example could also have been "-w17, 22, 0.7", because only the ratio of width to height (aspect ratio) is used. SwiftView scales whatever page is loaded to best fit the specified drawing window shape and size.

Adding the last two optional parameters to the "-w" command enables control of window position on the screen in a manner similar to the X "-g" parameter. Values are from -1. 0 to + 1.0 where positive numbers give the percentage of screen the upper left corner of the window is from the upper left corner of the screen and negative numbers control the lower right corner in the same manner. "0, 0" places the window at the upper left corner of the screen. "-0, .001, -0, .001" places it at approximately the lower right corner of the screen.

-tcccccc **Set windows title**
In the -t command, "cccccc" specifies the string you want for the title. This command line parameter ends at the first space character, so the title cannot have any spaces in it.

-sboff **Disable scroll bars**
This parameter disables scroll bars.

-pageoff **Disable paging area**
This parameter turns off the current page number and page slider at the right edge of the button bar.

-nologo **Disables "Swift" logo button**
Disables the "Swift" logo button, which appears between the regular buttons and the page control, and takes the user to our SwiftView End User help page on our website. If topbuttons and style labels are configured, the logo is not shown.

-reg **Register standard SwiftView suffixes**
-frcreg **Forcibly register standard SwiftView suffixes**
Registers standard SwiftView suffixes to be viewed with standalone SwiftView. A mime type file (mime.typ) is placed in the directory with SVIEW.EXE to allow automatic registration for specific suffixes. For example, a mime type file containing the following will register SwiftView for PeopleSoft .lis files:

```
application/vnd. SwiftView-ICS:lis:PeopleSoft LIS files
```

The mime type syntax is MIME Type : file suffix : description. The files are registered by placing the mime.typ in the same directory as SwiftView and then running "sview -reg".

-reg does not take over the default registration from another application (e.g., double clicking in Explorer); -frcreg does. –frcreg is particularly useful if you want to use SwiftView with .tif files on Windows98, which comes with another tiff viewer already installed.

-delreg **Unregister standard SwiftView suffixes**
Unregisters standard SwiftView suffixes, but does not re-register programs that may have been formerly registered for the same suffixes.

-host **Show PC hostname**
Some SwiftView products are locked to the hostname of a single computer. This parameter is a convenient way to report the hostname of a computer.

-ics **Pass ICS commands to GUI**
Using the -ics parameter allows the user to pass ICS commands on the command line as follows:

```
sview -ics"...ics... | ...ics... | ... | ...ics..."
```

SwiftView starts, processes the specified ICS commands and remains on the screen.
NOTE: there is no space between -ics and the quoted string. Also note: the "" (quotemarks) are required around the ICS command string.

-c **Pass ICS commands suppressing the GUI**
Using the -c parameter allows the user to pass ICS commands on the command line. Follow the parameter with the ICS commands in quotes, with multiple ICS commands separated by the pipe character in the following format:

```
sview -c"...ics... | ...ics... | ... | ...ics..."
```

SwiftView starts, runs as an icon with dialogs suppressed, processes the ICS commands and exits (background process). Make sure you place a pipe (|) between each ICS command. This is the preferred way to run SwiftView in a batch program.

Batch Mode

To utilize SwiftView in batch mode (no user interface) use Swiftview with the -c parameter, explained above. The following are example lines to be placed in a batch file to use SwiftView in batch mode:

Print all .pcl files in the current directory (example customized for Windows)

```
  FOR %%i IN (*.pcl) DO c:\progra~1\swiftview\sview -c"ldoc %%i| plot 99 all"
```

Convert all .pcl files in a directory to TIFF (example customized for Windows)

```
  FOR %%i IN (*.pcl) DO c:\progra~1\swiftview\sview -c"ldoc %%i|save TF_G4 all1 %%i.tif"
```

Convert all .pcl files in a directory to TEXT (example customized for Windows)

```
  FOR %%i IN (*.pcl) DO c:\progra~1\swiftview\sview -c"ldoc %%i| save TEXT all1 %%i.txt"
```

ICS Examples to Get Started

The following are a number of examples to get you started. In all of these examples we use the format of the ICS file, so just copy them and save them as a file and open it with SwiftView to see the functionality. Remember, you can pass any of these ICS commands through any of the other methods above as well.

These examples show some of the more widely used commands, however there are many more things you can do with ICS commands, please refer to the ICS Reference Manual, or contact Technical Support at www.swiftview.com/support/contact-support/ for assistance.

Basicbuttons.ics

```
### Pound Symbols are used for comments and are ignored ###
### This file clears the buttons and adds an open button and a print button ###
ICS
gui controls clear
gui button label "Open" icon open info "Open document" command "gui dialog open"
gui button label "Print" icon print info "Print document" command "gui dialog print"
```

Printersetup.ics

```
###  This file defines a printer to output a certain way to a specific Windows printer named
LJ4050 ###
###  and it adds a print button to that printer ###
ICS
printer number 23 type PCL5 command none alias "LJ4050"
gui button label "4050print" icon print info "Print to the lj4050" command "plot 23 all"
```

<u>Openandprint.ics</u>

```
### This file opens a specified file (c:\testfile.pcl) and then instantly brings up the print dialog ###
### This can be utilized on a website, passing ldoc commands to open files ###
ICS
ldoc c:\testfile.pcl
gui dialog print
```

Chapter 3: How to Configure a Website

One of SwiftView's greatest abilities is how it can be incorporated into your own website. In this section we will discuss how you should set up your web server, make a custom SwiftView installer page for your users, as well as give examples on how to configure SwiftView using Object and Embed tags in HTML.

Basic Server Setup

Mime Types

In order for SwiftView to be initiated when a user views a file on your website, you must first configure the mime types on your web server. For detailed instructions on this, please refer to the section "HTTP Server Setup" documented in our Plug-in Release notes at: http://www.swiftview.com/tech/svreadme.txt

The Plug-in Release Notes also have good information regarding known bugs, and Browser specific problems.

NOTE: IIS 4 has a different means of setting mime types (don't use regedit to change the registry.) Consult your IIS documentation.

Most servers require a total reboot for these mime types to take effect. After you attempt to configure the mime types, view your SwiftView document with Netscape (not IE), click View, Page Info and read the information returned for File Mime Type. If the correct mime type is not returned, check your web server documentation and try again.

Website Licensing

To license all files on your website the SwiftView Web-License file, npsview.lic, needs to be placed at the root of your website. It needs to have full read permissions to the extent that if you go to the exact URL of the npsview.lic file it displays the license. Here is the location of our license at our main SwiftView website: http://www.swiftview.com/npsview.lic

NOTE: for Windows Server 2003 the mime type of the NPSVIEW.LIC must be specified as text. For all other servers, in our experience, the NPSVIEW.LIC mime type does not matter.

Web-based SwiftView Customization

As noted above in the How to Configure SwiftView section, SwiftView can be customized in a few ways through your website. SwiftView can be configured globally for all files on your site by setting commands in the NPSVIEW.ICS file on the root of the server. Configurations can also be based on what directory files are located in, or by using Object and Embed tags in your HTML pages.

Global Configuration (npsview.ics)

The NPSVIEW.ICS file is an ICS file which is used to globally configure browser-based SwiftView for a website. When this file is placed in the root of the web server, it configures SwiftView for all users while viewing files on the website. We provide an example NPSVIEW.ICS, which demonstrates the most popular features, such as fast printing, direct printing, HPGL pen colors and markup. It is located at: http://www.swiftview.com/tech/npsview_ics.txt

Simply comment out any undesired features using the # character. NOTE: The file had to be renamed to NPSVIEW_ICS.TXT so that it could be viewed, otherwise, with its original name of NPSVIEW.ICS it would simply be processed by browser-based SwiftView and would not show on screen.

Configuration by Directory

SwiftView can be configured to act differently depending on what directory of your website the files are located in. This can be especially useful when applying markup. For example, all files in one directory can display a watermark saying "CONFIDENTIAL" and another directory can contain files that don't.

To apply SwiftView configuration by directory the NPSVIEW.ICS located at the root of your web directory must be manipulated. The file must contain this line:

```
set ldocinit "set nofileerror n| ldoc \"\$docdir()CONFIG.ICS\"| set nofileerror y"
```

The "set ldocinit" command defines commands to be run when an ldoc is completed (initdoc). This setting loads the file CONFIG.ICS from the directory of the current document. "set nofileerror n" disables the nofile error, so that all directories do not have to contain CONFIG.ICS.

The CONFIG.ICS file that you would place in each directory can contain any ICS commands you wish, including markup. This CONFIG.ICS file must be in the format of an ICS File described previously in Chapter 2.

Configuration by Web Page

Because SwiftView is an ActiveX control, it can be easily placed in your HTML pages using object and embed tags. Using modern scripting languages such as PHP, ASP, and JavaScript, you can utilize these objects and embed tags to dynamically change ICS commands that you want to have passed to a web page.

Using EMBED tags

In this first example, we use the html embed tag to embed SwiftView into an HTML table cell. Make sure to set the following EMBED attributes:

SRC
- The source file you want to load, this would be the same as the filename you would use in an ldoc
WIDTH and HEIGHT
- Set the percentage of the table cell you want to use
COMMANDS
- Any ICS commands you wish to pass to SwiftView

Below is sample code of an embed, where it loads the file docs/pcl/phonebill.pcl, and passes commands to display buttons on the right hand side:

```
!-- Copyright (c) 2000 SwiftView, Inc.                    -->
<!-- This example is provided WITHOUT WARRANTY either expressed or implied.-->
<!-- You may study, use, modify, and distribute it for any purpose.        -->
<!--                                                      -->
<HTML>
<HEAD>
        <TITLE>SwiftView embedded in an HTML page using the EMBED tag</TITLE>
</HEAD>
<BODY>
<CENTER>
<IMG SRC="images/trabanna.gif">
<TABLE BORDER="0" WIDTH="100%" COLS="1">
   <TR>
   <TD>
        <CENTER><B><FONT FACE="Book Antiqua"><FONT COLOR="#0105AF"><FONT
          SIZE="+2">Generous Telephone Corporation</FONT></FONT></FONT></B>
<BR></CENTER>
        <HR> <BR>
   </TD>
   </TR>
   <TR>
   <TD WIDTH="100%" HEIGHT="450" ALIGN="Center">
          <EMBED
             SRC="docs/pcl/phonebill.pcl"
             WIDTH="100%"
             HEIGHT="100%"
             COMMANDS="gui controls display rightbuttons"
          >
   </TD>
   </TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

To see what this source code does when put in a web page, go to:
http://www.swiftview.com/howto/embed1.htm

Using OBJECT tags

This second example uses the html object tag to embed SwiftView. Remember to set the CLASSID and CODEBASE attributes within the object tag. IE will use this information to install SwiftView if it is not installed or is outdated.

CLASSID
-Is set to "CLSID:7DD62E58-5FA8-11D2-AFB7-00104B64F126", the registered ID for our SwiftView ActiveX control. Without this information, IE does not know which ActiveX control to associate with this OBJECT.
CODEBASE
-Is set to "../product/current/svinstall_a_stat.exe#Version=", the location of the EXE file to use when installing the ActiveX control and the version number. This attribute may be omitted; however, IE will not be able to install the ActiveX control if missing or outdated.
WIDTH and HEIGHT
-Are set to what percentage of the cell you want the SwiftView Object to take up

In this example we specify the document to display using the PARAM tag, setting its NAME attribute to "SRC" and VALUE to "docs/pcl/phonebill.pcl". We also use the PARAM tag to set COMMANDS as we did in the first example to display a SwiftView toolbar.

```
<!-- Copyright (c) 1999,2000 SwiftView, Inc.                              -->
<!-- This example is provided WITHOUT WARRANTY either expressed or implied.-->
<!-- You may study, use, modify, and distribute it for any purpose.        -->
<!--                                                                        -->
<?php
require("curversion.php3");
// We place current version information into the PHP3 file curversion.php3
// and use the variables specified in that file to eliminate the need to
// edit numerous files on this website.
//   The variable $Cur_softupdate_version is set to a string like "5,1,0,1".
//   The variable $Cur_version is set to a string like "5.1.1".
// We then use the PHP3 echo command with these variables to write out
// the version strings.
?>
<HTML>
<HEAD>
   <TITLE>SwiftView embedded in an HTML page using the OBJECT tag</TITLE>
```

```
</HEAD>
<BODY>
<CENTER>
<IMG SRC="images/trabanna.gif">
<TABLE BORDER="0" WIDTH="100%">
   <TR>
   <TD>
          <CENTER><B><FONT FACE="Book Antiqua"><FONT COLOR="#0105AF"><FONT
            SIZE="+2">Generous Telephone Corporation</FONT></FONT></FONT></B>
<BR></CENTER>
          <HR> <BR>
   </TD>
   </TR>
   <TR>
   <TD WIDTH="100%" HEIGHT="450" ALIGN="center">
          <p>
          <OBJECT
             ID="embed2"
             WIDTH="100%"
             HEIGHT="100%"
             CLASSID="CLSID:7DD62E58-5FA8-11D2-AFB7-00104B64F126"
             CODEBASE="../product/current/<?php echo $SvExeFilename;?>#Version=<?php
echo                                                 $Cur_softupdate_version; ?>"
          >
             <PARAM NAME="SRC" VALUE="docs/pcl/phonebill.pcl">
             <PARAM NAME="COMMANDS" VALUE="gui controls style icons | gui logo enable |
gui controls display rightbuttons">
          </OBJECT>
   </TD>
   </TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

To see what this source code does when put in a web page, go to:
http://www.swiftview.com/howto/embed2.htm

Chapter 4: Advanced Features

Documents, Files, and Virtual Documents

SwiftView presents "documents" to the user. The user can go to any page using standard keyboard and mouse controls or SwiftView's page counter and then adjust the view on that page. Documents can be defined by a single file or by combinations of files, so it is important to separate the concept of "document" from "file".

SwiftView can view multipage PCL, HPGL, and TIFF files. It can also emulate a multipage document with a series of single-page files. We sometimes refer to a document composed of many separate files as a "virtual document", since the single document that appears in SwiftView does not actually exist in the world as a single file.

ICS-defined Virtual Documents: one file per page

ICS commands can be used to define an aggregate of individual, single-page files as a single virtual document to the user. This can be done on the fly by extracting the pages of interest from a database, writing a short ICS file identifying the pages and starting SwiftView with the ICS file name on the command line. The user then sees what appears to be a single multi-page document for viewing. This feature is the functional replacement of SVP files, which should no longer be used. SVP files are still supported but will be removed in a future release. This is also a more powerful alternative to the ".001 page numbering" method which follows. Direct Printing is not supported with virtual documents.

For simplicity, the examples in this section are in "ICS file" format. However, these commands are not limited to ICS files; they can be sent to SwiftView in a number of ways, including command line, from web sites, through pipes and via DDE.

Here is a simple ICS file defining a 3-page document composed of 3 files. Remember that all standard rules for ICS files apply.

```
ICS
# clear any prior document
onpage reset
# define 3 pages
onpage 1 use sourcename l:\documents\file1. pcl
onpage 2 use sourcename m:\photos\photo. tif
onpage 3 use sourcename k:\drawings\drawing. tif
# set default rotation
rotate 0
# draw at default scale
draw
```

The following example adds 3 blank pages in the middle. It also adds the command "scale widetop" which is recommended for cases that mix different size and shape pages (e. g., mixed PCL and HPGL).

```
ICS
onpage reset
onpage 1 use sourcename file1. pcl
onpage 2-4 use blankpage letter
onpage 5 use sourcename file2. hpg
rotate 0 | scale widetop
draw
```

Note that in this example pages 2 through 4 are blank letter size pages. If a page is defined multiple times, the last definition wins.

The example above could also be written as:

```
ICS
onpage reset
onpage 1-5 use blankpage letter
onpage 1 use sourcename file1. tif
onpage 5 use sourcename file2. hpg
rotate 0 | scale widetop
draw
```

Simple one file per page - "*.001 " page numbering

SwiftView treats any file with the suffix ".001" as the first page of a multipage virtual document. To create a virtual document using this feature, simply give a series of one-page files an identical name with sequentially numbered suffixes (e.g., report.001, report.002, report.003, etc.) The first missing number in the sequence terminates the virtual document. If the document name requested does not exist, but the same name with the suffix ". 001" does exist, the multipage scheme is also assumed. All files must be of the same type. Starting a document with "filenameame.002" or any other page does nothing: a virtual document will not be recognized.

Because of the way browsers use (or don' t use) file suffixes, this method of creating virtual documents is not reliable for browser-based SwiftView. For reliable use of virtual documents in browser-based SwiftView, ICS "onpage #" commands should be used.

Documents with more than 999 pages can be both viewed and created. Pages 1-999 are still defined as above to ensure backward compatibility. Higher page numbers use 4-digit suffixes:
".1000", ".1001", ... ".9999".

Page Serving

Internet File access and ICS-defined documents combine to allow a complete document to be defined from single-page files on a web server. This allows quick access to an individual page without downloading the entire document. For example, a set of files of the form "test.001" – "test. 023" can be accessed via an ICS file on the server as such:

```
ICS
onpage reset
onpage 1-23 use sourcename "http ://www. myserv. com/test.%03d"
initdoc
draw
```

Page serving is currently operational only with Win32 clients. Please contact us if you have a UNIX requirement. www.swiftview.com/support/contact-support/

Associating ICS Files With a Document

SwiftView supports a mechanism to associate ICS files with one or more document files. You can specify an ICS file to be loaded for all documents in a directory, or an individual file to be loaded with each document. This allows, for example, applying a set of markups (overlays) or reconfiguring the SwiftView user interface for a given set of documents.

Load an ICS for all documents in a directory:

```
set ldocinit "set nofileerror n| ldoc \"\$docdir()config. zhp\"| set nofileerror y| rotate 0| scale default"
```

Load an ICS file of the same name as the document but with a different suffix:

```
set ldocinit "set nofileerror n| ldoc \"\$docdir()\$docbase().mrk"| set nofileerror y| rotate 0| scale default"
```

"set ldocinit" defines commands to be run when an ldoc is completed (initdoc).
"rotate 0 | scale default" are the default actions performed by initdoc; this setting loads the file "CONFIG.ZHP" from the directory of the current document.
"set nofileerror n" disables the nofile error, so that all directories do not have to contain config.zhp. See "Environment Variables and ICS Macros" for a discussion of the "$docxxx" macros.

Typically, you will put these commands in a web site configuration file (NPSVIEW.ICS on the root of the web server), a local configuration file (c:\Program Files\Swiftview\sview.ini), or a LAN configuration file (sview. ini in the same directory as sview. exe)

PCL Macros and Additional Font Support

PCL Macro Support

SwiftView versions 5.3.0 and higher contain a feature that mimics printer resident PCL macros. For Windows, macros can be placed in c:\Program Files\SwiftView\macros\ when opening files that require these printer resident macros. SwiftView will look in this folder and if it finds the appropriate macro it will display accordingly. For UNIX SwiftView, resident PCL macros are stored in the directory $NDGMACRO or, if not defined, in $NDGUTIL\macros. If the environment variable $NDGUTIL is not set, SwiftView looks for the macros directory in the location from which SVIEW.EXE was invoked. $NDGMACRO may be set to look in a directory on a web server.

In a PCL file, macros are identified by a positive number between 0 and 32767 inclusive. SwiftView macro files use this number along with the extension ".mac" as a name. For example, the macro with ID= 503 would be stored in file "503.mac". If an overlay macro is used, it must be called for in the pcl file being used, otherwise SwiftView has no knowledge that an overlay macro is required.

Scalable TrueType Font Support

Beginning in version 5.3.0, SwiftView looks for TrueType font files in the directory c:\Program Files\SwiftView\ttf in Windows, and $NDGUTIL\ttf for UNIX. We currently have all of the TrueType Fonts that we have made available can be found at: http://www.swiftview.com/tech/supportfiles.htm

TrueType fonts have the advantage of providing all point sizes in a single file as opposed to bitmap fonts. Very few files actually require these TrueType fonts, however in some cases files will not view correctly if they are not present. Please contact us www.swiftview.com/support/contact-support/ if you have any questions.

p300x300 Bitmap Font Directory

This p300x300 directory can also be used to hold font files for Windows in cases where supplemental fonts will improve the appearance of a document. In this case you can provide your own font files to be used. These files must be placed in c:\Program Files\SwiftView\p300x300, and must follow the conventions below.

The file "fonts.txt" must be present containing the list of all font files (one per line of text) in the directory in any order. It can be generated on UNIX by typing "ls > fonts. txt", placing each name on a separate text line. Each font file contains the bitmaps for a single combination of the characteristics reflected in the name. Each name is of the form: TTWSSLXX. PPP

TT	Typeface:
	UN = Universe - LJ 3 internal font similar to HV
	TR = Times Roman or equivalent serif
	CR = Courier or equivalent monospace typewriter font
	LP = Line printer
	HV = Helvetica or equivalent sanserif
W	Style:
	R = Regular
	B = Bold
	I = Italic
SS	Symbol set:
	AU = HP PC-8
	8U = HP Roman-8
	0N = Latin 1 (ECMA-94)
	BU = HP PC-8 Danish/Norwegian
L	Orientation - only portrait fonts are required:
	L = Landscape
	P = Portrait
XX	Numeric value indicates fixed pitch:
	10 = 10 cpi
	12 = 12 cpi
	16 = 16. 67 cpi
	PS = Proportional spacing
PPP	Point size. Examples are:
	095 = 9. 5 point
	120 = 12 point

Examples:

CRR8UP10. 120 = Courier regular, Roman-8, 10 cpi, 12 point
UNB8UPPS. 180 = Universe bold, Roman-8, proportional spaced, 18 point

Web-based Font Support

For website customers that have files requiring extra fonts, SwiftView can be configured to look for fonts on the web server, instead of locally. This can be done by passing an ICS command that sets the environment variable NDGTTFONT or NDGBMFONT, depending on whether your files require TTF or p300x300 fonts. To do this pass an ICS command from your website like this:

```
set env "NDGTTFONT=http://www.swiftview.com/ttf/"
```

In addition to the font files, this web directory must also contain a text file named DIRLIST (no file extension). DIRLIST must list all of the ttf files listed in that directory such as:

```
sv0p0s0b6t.ttf
sv0p0s3b6t.ttf
sv0p1s0b6t.ttf
sv1p0s0b52t.ttf
```

```
sv1p0s0b5t.ttf
sv1p0s3b52t.ttf
sv1p0s3b5t.ttf
sv1p1s3b5t.ttf
sv1p4s0b52t.ttf
sv1p4s3b52t.ttf
etc.
```

NOTE: This can be a little tricky, if you have any questions please contact SwiftView technical support www.swiftview.com/support/contact-support/.

Environment Variables and ICS Macros

SwiftView uses a number of environment variables to change its operation from the default behavior, or to configure its installation. These variables are listed in the ICS Reference Manual.

SwiftView also allows you to insert the value of an environment variable into an ICS command using a “macro” syntax, and to define any environment variable with the ICS command “set env". Macros can contain any number of commands, parameters, and additional macros nested to any depth. Using this facility, you can effectively pass parameters into ICS command files by setting an environment variable and calling it as a macro.

Any occurrence of "$str()" in an ICS command is replaced by the value of the environment variable "str". The '"$" character can be escaped ("\$" or "\$$" per above) to defer replacement in nested quoted ICS command strings or to prevent expansion entirely. A '"$" character without a terminating "()" in the token is not replaced.

The following environment variables are preset based on the current document for use in ICS commands, and should normally not be overridden by “set env”:

"docname"	The URL name of the current document, if any.
"docdir"	The directory containing this current document, with a trailing slash as required.
"docbase"	The simple filename of the current document, if any, without any directory path prefix or trailing dot and suffix. This was released with Ver 5. 1. 1.
"curpage"	The current page number.
"numpages"	The current page count.

See the reload command for the definition of "current document".

For example, you can display the name of the current document in the standalone SwiftView title bar with the following command:

```
set ldocinit "gui wintitle \"\$docname()\" | rotate 0 | scale default"
```

SwiftView Markup

One of SwiftView's more interesting features is the ability to add markup to documents. Using markup you can add watermarks, overlay images, and add valuable information to your documents. Because the markups are added using ICS commands, the integrity of the document is kept intact. This gives the ability of adding many different markups without manipulating the original file.

This section of the manual explains how to apply markup using ICS commands, allowing for use on websites or applying via a SwiftStamp. All markup ICS commands are defined in the ICS Reference Manual starting with the word "markup".

General markup concepts

Each page of a multipage document is a separate "current drawing" and only one page is displayed at a time by SwiftView. Markups are normally applied to a page or pages using the "onpage…" command. Markups applied in this manner are displayed on the designated page(s) as the user changes pages. Markup commands issued without a preceding "onpage" ("transient" markups) are applied to the current page, and are discarded if the page changes. All markups are discarded if an "onpage reset" or "ldoc" command is issued.

Markup attributes (color, etc.) affect only subsequent "graphics" commands, including lines, arrows, rectangles, polygons, etc. All "markup attributes" parameter data pairs may be used in combination. The attribute control commands "reset", "push" and "pop" must appear alone.

Whenever a new page is loaded, all transient markups are destroyed, and all markup attributes, text fonts, and current drawing position are reset to defaults. Markup attributes are stored and processed inline with the markup entities, so they must be prefixed with the same "onpage" range as the entities they control.

Markups work very naturally with ICS commands and ICS files. The following ICS files can be used to markup "standard" and "out of revision" documents or drawings:

<u>dwg_std. zhp</u>

```
ICS
onpage all markup text font "face ascii size 24 pitchcpi 5"
onpage all markup attributes display no
onpage all markup text rxloc 0 ryloc 0 string "CONTROLLED DOCUMENT PRINTED
$strftime(%X %x )"
```

<u>dwg_outrev. zhp</u>

```
ICS
onpage all markup text font "face ascii size 24 pitchcpi 5"
onpage all markup attributes display no
onpage all markup text rxloc 0 ryloc 0 string "CONTROLLED DOCUMENT PRINTED
$strftime(%X %x )"
onpage all markup attributes display yes
onpage all markup text rxloc 0 ryloc 17 string "CAUTION: THIS DOCUMENT IS OUT OF
REVISION; SEE DOCUMENT CONTROL FOR DETAILS"
```

Standard documents would be displayed in this manner:

```
sview -ics"ldoc somedrawing.hpg | ldoc dwg_std. zhp | draw"
```

Out of revision documents would be displayed in this manner.

```
sview -ics"ldoc somedrawing.hpg | ldoc dwg_outrev. zhp | draw"
```

Markups don't necessarily need to be placed in a separate ICS file, they can be passed directly. So Standard documents could also be displayed as follows:

```
sview -ics"ldoc somefile. hpg | onpage all markup attributes display no | onpage all markup
   text font \"face ascii size 24 pitchcpi 5\" | onpage all markup text rxloc 0 ryloc 0 string
   \"CONTROLLED DOCUMENT PRINTED $strftime(%X %x )\" | draw"
```

Markup coordinates

SwiftView commands and callbacks use three coordinate systems:

Drawing coordinates, where the units are the actual physical coordinates on the document as it would be when normally printed. (When document files lack dimensions, SwiftView makes an assumption.) Values are real numbers and may have whole and fractional parts, e.g., “1. 23” This is the default for markup.

Pixel, or "display" coordinates, where the units are in pixels on the display. This coordinate method is not used in markup. Values are integers only.

Device coordinates, where the units are the physical coordinates on the display or printed paper output. This is used only in markup. Values are real numbers.

In all cases, the coordinate "0, 0" is the upper left corner of the page and coordinates increase toward the lower right. Drawing and device coordinates are specified, and returned in callbacks, in inches or centimeters per the "units" and "markup units" commands.

Many markup commands can omit position information and the "current drawing position" (CDP) is substituted by SwiftView. CDP is initialized to (0, 0), but is changed by each markup drawing command to the last drawing coordinate given.

The "markup attributes coordinates …" command can be used to shift the origin of the coordinates, as well as make the coordinates relative to the drawing or the output device.

By default, drawing coordinates are used, so if you print an entire page of a document the markup text location and size will be based on the upper left hand corner of the original document. If you printed only a selection of the document, the string "CONTROLLED DOCUMENT PRINTED" would not show unless you happened to select the area with the markup.

The following example shows how to print relative to the paper, causing the string "CONTROLLED DOCUMENT PRINTED $strftime(%X %x)" to always be printed at the same place and in the same size on the paper regardless of which part of the original document or page is printed:

```
dwg_std. zhp
ICS
onpage all markup attributes coordinates device xorigin 1 yorigin 1
onpage all markup text font "face ascii size 24 pitchcpi 5"
onpage all markup attributes display no
onpage all markup text rxloc 0 ryloc 0 string "CONTROLLED DOCUMENT PRINTED
$strftime(%X %x )"
```

By using "device" coordinates, the string "CONTROLLED DOCUMENT PRINTED $strftime(%
X %x)" is printed at the same size in the upper left hand corner of all pages.

Markup "hotspots"

Hotspots can be used to associate any group of ICS commands with any entity. Since entities can be made invisible or transparently colored, the result is a powerful hotspot generation capability. This can make it so a user clicks on a certain area of the file and a pop-up window appears, a website is opened, or any other set of ICS commands are run. The following examples show how many use hotspots:

Transparent, filled, "hot" rectangle that pops a message
All pages of an invoice package are given a transparent yellow rectangle which, when clicked, gives the name of the person responsible for that package.

```
ICS
ldoc invoices.tif
```

```
onpage all markup attributes fgcolor rgb:ff/ff/0 fillpattern solid
onpage all markup attributes bgcolor rgb:ff/ff/0
onpage all markup attributes drawmode transparent
onpage all markup filledrectangle 0. 1 0. 1 0. 6 0. 6 hotspot "message \"This invoice prepared by
Betty Boop on Nov 10, 1992 \" " status "Click here for prepared info"
draw
```

Invisible "hot" rectangle that pops a message

```
ICS
ldoc invoices.tif
onpage all markup attributes push
onpage all markup attributes print no
onpage all markup attributes display no
onpage all markup filledrectangle 0. 1 0. 1 0. 6 0. 6 hotspot "message \"This invoice prepared by
Betty Boop on Nov 10, 1992 \" " status "Click here for prepared info"
onpage all markup attributes pop
draw
```

For more examples of SwiftView Markups, visit the Website Integration page at:
http://www.swiftview.com/howto/webhowto1.htm

Integrated Programming, Callbacks, and Text Extraction

Using SwiftView's ActiveX control, or our SwiftInside DLL, you can easily integrate SwiftView into your own application using Visual C++, Visual Basic, or other Win32 programming methods. Because of the complexity regarding programming integration, all of our documentation for this is located on our website. Please start here if you want to integrate SwiftView into your own application:
http://www.swiftview.com/tech/svint.htm

If you are looking into integration, please feel free to contact us at www.swiftview.com/support/contact-support/. We will be glad to assist you and answer any questions you may have.

Callbacks

SwiftView is a powerful tool for building customized user interfaces for PCL, HPGL, and TIFF file viewing, and for analysis of PCL files. Its simple textual callbacks allow you to collect information about the SwiftView program state and about the file you are viewing. All callbacks are listed in the ICS Reference Manual in Chapter 3: ICS Callbacks.

In conjunction with the ICS commands, callbacks put you in charge of SwiftView's appearance, features, and configuration without complex programming. Your application can control all aspects of the SwiftView user interface, including (in the ActiveX control) mouse interaction. Callbacks also provide the mechanism for returning detailed information from PCL files, including text strings and the location of text on the page.

The simplest way to begin seeing callbacks is to use the ICS command "callback filename" and setting it to a local output file. Here is one way to do this from a command line:

```
sview -ics"callback filename c:\callbackfile.txt"
```

When running SwiftView all callbacks will be outputted to this text file, and you will see the basic functionality.

Callbacks are intended for use in integration. Using varying programming methods, the callbacks can be redirected to function calls in your own application that can then parse through the output and extract the information you need. Callbacks can also be integrated into websites using JavaScript and ASP.

Text Extraction

One of SwiftView's more powerful features is the ability to extract text from PCL documents. Using ICS commands you can tell SwiftView to convert entire PCL files to text, or simply extract certain pages or portions of a document. This functionality allows users to both extract and store information contained in PCL files, as well as use that information for indexing purposes.

There are two main ICS commands to use when extracting text, "save TEXT" and "save TEXTPOS". They are both defined under the Save command in the ICS Reference Manual. "save text" saves as a simple text file, "save textpos" outputs a file that separates each individual word in a document along with it's corresponding positioning on the page. Below are 2 simple examples to run from the command line:

```
sview -c"ldoc yourfile.pcl| save TEXT 1-1 c:\outputfile.txt onefile"

sview -c"ldoc yourfile.pcl| save TEXTPOS 1-1 c:\outputfile2.txt onefile"
```

In these examples, the first command, "save TEXT" will output c:\outpufile.txt that contains all of the text on the first page(1-1) of yourfile.pcl. The second command "save TEXTPOS" will output c:\outputfile2.txt, which will contain all of the words and their corresponding positions for page 1 of the file. If you run these commands separately you will notice the difference in outputs.
These ICS commands can also be fine tuned by adding coordinates, to return only the text in a certain location of the page. The most common way this is used is if someone wants to extract things such as an address that always appears at the top corner of a document. These coordinates are inputted exactly the same way as the coordinates are outputted using "save TEXTPOS". The following is an example of SwiftView returning the text only in a specific location of a file:

```
sview -c"ldoc yourfile.pcl| save TEXT all c:\outputfile.txt rcut 7.25 10.25 7.50 10.75 onefile"
```

This command saves the text in the specified location of the page on all pages.

Integrated into your own application SwiftView can do many things with this functionality. It can be used to search through PCL files, determine their content, and dynamically perform actions depending on the contents.

Chapter 5: Other Windows Details

Print Methods

SwiftView can output to a printer using three different methods. Some methods are faster or more accurate, but can only be used with certain combinations of file types and printer types. The three methods are Windows Printing, Fast Printing and Direct Printing.

Windows Printing is used by most Windows programs, and will print to any type of printer, however it is the slowest method of the three.

Fast Printing will only print to a PCL5 (LaserJet-compatible) printer or PostScript printer.

Direct Printing will only print to a printer that is compatible with the original file type, thus if the original file is PCL the printer must be PCL, or if the original file is an HPGL file, the plotter must be an HPGL plotter. Direct Print offers the user no options whatsoever-- it simply copies the entire file directly to the printer. If the file is compatible with the printer being used, it works perfectly and very quickly. If there are printer/file incompatibilities, Direct Print sometimes results in odd fonts or unexpected paper sizes being selected. In those cases, **Fast Print** often produces better results.

The following table shows the process that occurs when each method of printing is used. Fewer steps in the process indicate faster printing.

Windows Printing	Your File > Rendered by SwiftView > Windows Printer Driver > Printer
Fast Printing	Your File > Rendered by SwiftView > Printer
Direct Printing	Your File > Printer

How to specify a Print Method

Print method can be specified in three ways: by the end user in the Universal Print Dialog, by defining a numbered printer using ICS commands (see below), or by calling a print method dialog directly (using the commands gui dialog winprint, gui dialog fastprint, or gui dialog directprint). NOTE: Calling a print method dialog directly is deprecated and will disappear in a future version. We advise using gui dialog print and setting printmenudefault to control the options available to users, or else defining a numbered printer.

Defining a numbered printer using ICS commands

In SwiftView, printing from the command line is a two-step process. A printer selection is defined by the ICS "printer" command and then the print job is started by the ICS "plot" command. In addition, any paper output size, duplexing, resolution, and color output settings are set using the set printsize, set printduplex, set printres, and set printcolor commands listed in the ICS Reference Manual.

All three print methods can be controlled directly through ICS commands. The examples below demonstrate ICS commands that define printers. These definitions are associated with numbers between 0 and 89 inclusive; numbers 90 to 99 are reserved for SwiftView. The examples below use numbers arbitrarily starting at 20. The printer "type" parameter selects the print model; MS_WIN indicates MS Windows printing (through the Windows printer driver), HPLJ3_ORIGINAL indicates PCL Fast Printing (P2_300 would be used for PS Fast Printing), and DIRECT indicates Direct printing.

```
printer number 20 type MS_WIN command none alias "HP LaserJet 4050"
printer number 21 type MS_WIN command none alias Optra
printer number 22 type MS_WIN command FILE alias "HP LaserJet 4050"
printer number 23 type MS_WIN command FILE alias "na, na, na"
printer number 24 type PCL5 command none alias "HP LaserJet 4050"
printer number 25 type PCL5 command FILE alias none
printer number 26 type PCL5 command none alias "na, na, na"
printer number 27 type DIRECT command FILE alias Optra
printer number 28 type DIRECT command none alias "na, na, na"
printer number 29 type DIRECT command none alias "na, na,LPT1"
```

These commands define the following: Printer number 20 prints using the MS Windows driver associated with the printer name "HP LaserJet 4050". Note that quotation marks are used in the name because the name contains spaces. Printer 21 outputs via the MS windows printer diver associated with the name Optra. Printer number 22 also uses the "HP LaserJet 4050" driver but the "FILE" parameter directs the output into a file instead of to the printer. The use of the alias "na, na, na" in printers 23, 26, and 28 indicate that the printer name is not known. In this case the default Windows printer is used. Printer 24 selects PCL fast printing, directed at the "HP LaserJet 4050" printer, while printer number 25 outputs to a file. Printer number 27 selects Direct printing, in this case copying the file directly to the Optra printer. Printer number 28 selects direct printing to the default Windows printer. Printer number 29 selects direct printing to the printer on port LPT1.

Notice the use of the "friendly" printer names, "Optra" and "HP LaserJet 4050". These are the names found on the Windows Start-> Settings-> Printers Menu. Names that use space characters require quotation marks, but those without spaces do not. Of course, your printer names will likely be different from these examples.

The special case "na, na, na" is used to select the current default Windows printer. Prior to SwiftView release 5.2.0, the old fashion triplet style names were required (these looked like "Optra, Winspool,LPT1"). In general, using the old triplet style is not recommend; however beginning with ver. 5.3.0, the old triplet style can be used to print to the port by name. Some users know the printer port name (LPT1: for example) but do not know the "friendly" name, and in this case the printer definition would resemble printer 29 above. However, if both the friendly name and the port number are given in the triplet, the friendly name takes precedence over the port. In the example below, SwiftView will print to the printer named Optra:

```
Printer number 30 type PCL5 command none alias "Optra, na, LPT1"
```

The commands above define printers, but the ICS command "plot" must be used to actually print. For example to print an entire document to printer number 22, use the command:

```
plot 22 all
```

The printer number 99 is reserved for SwiftView, it selects MS Windows printing to the default Windows printer driver (no printer command required) and is used like so:

```
plot 99 all
```

The command parameter uses the entry "FILE" to redirect the output from a printer to a file. The ICS command. "set filename name" is used to select the file name, for example:

```
set filename L:\stuff\foo. prn
```

ICS commands can be sent to SwiftView in a number of ways; however, the examples below demonstrate the use of the printer ICS commands with the SwiftView command line. This is chosen both because it is a common case and also because it demonstrates the use of nested quotation marks. The following examples print the fictional file "foo.pcl" to our example printers. This example assumes that you are invoking SwiftView from the current directory, loading and printing the example file.

The entire command must be issued on a single line, but the space restraints of this page require showing it below on multiple lines. Notice the use of the "\" symbols. This custom quote can now be used nested within the outer quotes used for the –ics command line entry. Also notice the use of the pipe character "|" used to concatenate multiple ICS commands together.

```
sview -ics"printer number 20 type MS_WIN command none alias \"HP LaserJet 4050\" | ldoc foo.
pcl | plot 20 all"
```

Here is an example of a printer that does not have spaces in its name:

```
sview -ics"printer number 20 type MS_WIN command none alias Optra | ldoc foo. pcl | plot 20 all"
```

Universal Print Dialog

A universal print dialog, which allows users to select between the three methods of printing, was introduced in version 5.5.0. As discussed above, whether a particular print method will work is dependent on the type of file and the type of printer. The default is to prevent users from being able to select methods that will fail for their file/printer combination.

Print Range

With the Universal Print Dialog a user can specify to print the whole document (ALL), a page range (Pages), and a "Cut" area of the file (Selection).

Any rectangle can be "cut to the printer" by clicking "Selection" and pressing OK. The user will then see a "scissor cursor" indicating "cut mode". Simply use the left mouse button to select the area to be printed with a "rubber band rectangle".

NOTE: Page Range and Selection are not available using the Direct Print Method because Direct Print copies the file directly to the printer without any manipulation.

Print Method

In order to determine if a printer is PCL-compatible, the first time it is used, SwiftView silently prints to file and tests the output. That information is saved in the registry. It is then used to determine which Print Methods should be available. The print methods are explained in detail above in the section title Print Methods.

Selecting "Reset To Defaults" tells SwiftView to recheck the available print methods for the selected printer by silently printing to file using your printer's driver and evaluating the resulting file. In some cases this allows SwiftView to enable print methods that were not previously available.

Selecting "Allow All" enables all Print Methods regardless of SwiftView's tests. In some cases SwiftView is unable to determine a good output type for a printer and disables all but Windows Print. "Allow All" allows you to override this setting and gives all print methods. NOTE: this can allow using print methods with printers that are not compatible, which may cause invalid output or other problems.

Options

The Options dialog is available from the Universal Print Dialog and allows the user to set Print Method, Paper Selection, Scaling, and other options.

Paper Selection allows the user to use the Document Paper Size specified in the file (use this option to print documents that mix page sizes, such as letter and legal), or to force the entire document to Use the One Size set in the Printer Properties. This size is shown in parentheses below the radio buttons and can be changed from the Properties button on the main Universal Print Dialog. Also under Paper selection one can choose whether or not to pass existing duplexing commands contained in the file.

Scaling options allow printing at a scale of 1:1 (some of the drawing may be missing if your paper isn't large enough), scaling to fit the printable area (try this if your printed document is missing data from the edges), and scaling to fit the entire area of the paper (try this if your document's margins are too large).

Other options include aligning the page by the longest edge when printing or printing as rotated in the display. It also allows you to position the output to the top left of the page. This can be useful when doing things such as printing a letter size page on legal paper. If the check box is selected to Position at the upper left, the data will begin at the top of the page. Otherwise the data will print centered on the legal size page.

There is also a checkbox that allows the printing or suppression of Markup if there is SwiftView Markup contained in the file.

Copy and Paste Features

Text

SwiftView supports the standard Windows ability to copy text from documents in SwiftView and paste into other applications. Copied text can be up to a page in length and by default is selected in "document mode", that is, as continuous sentences. Selecting text contained within a user-defined area only (not entire lines or paragraphs) can be done by changing the default "document" to "drawing" mode. See the "set select …" command in the ICS Reference Manual for details.

Some documents do not contain selectable text. Sometimes this occurs because what appears to be text contains no ASCII data but is actually an image. Another cause is mis-encoded text generated by Windows NT and 2000. On Windows NT and 2000, when an application in conjunction with a PCL printer driver downloads fonts, it does not assign ASCII characters to the standard (ASCII) encoded values. Instead, the first letter they download is given the value 1; the next is given the value 2, etc. Therefore, if ' A' is the first letter used in the font, it might be 1 instead of the standard 65. However, since the downloaded fonts match their use in the file, it's completely readable and printable - just not searchable or selectable. Print drivers onWin98SE simply offset the standard ASCII numbers in downloaded bitmap fonts by a small constant value, and we compensate for that. Unfortunately, Win2000 and NT appear to completely scramble the encoding.

When a file is opened, SwiftView tests all downloaded fonts used in processing the document, and if one contains a glyph less than 21, a warning message about mis-encoded text is invoked. Some special fonts may trigger this test improperly; if this is a problem you can disable this message with the "set checktext disable" command. This command also disables character offset adjusting for Win98SE. Some files from other sources may download partial fonts, improperly triggering this offset and scrambling the text for search and copy.

There are two possible solutions to mis-encoded downloaded fonts on NT/2000:

- Print your files on Win98SE instead of NT/2000. Be sure to select "Download TrueType fonts as Bitmap soft fonts", as SwiftView cannot currently unscramble outline soft fonts.

- Have your application use printer-resident rather than downloaded fonts. Modern printers have lots of fonts in them so this is certainly possible, if you have control of your app.

Graphics

It is possible to copy graphics from SwiftView and paste them into other applications as a Windows Metafile (WMF). For example, to copy an HPGL drawing into Microsoft Word, do this:

1) In SwiftView, **select a graphic** by holding down **Alt** and selecting a rectangle with the mouse.

2) **Copy the graphic** to clipboard. The copy keystroke depends on whether you are using standalone SwiftView (standard keys) or browser-based SwiftView (where browsers prevent us from using the standard key sequence). Click the '?' button to learn the correct keystroke. Switch to Word and paste using the Edit, Paste Special, Picture (Enhanced Metafile) to paste the graphics. Note: using the standard paste keystrokes or button in Word will paste only selected text as before, not graphics.

3) If you are copying mixtures of text and graphics from a PCL file, both selection features are used. The text will be placed on the clipboard and simultaneously a bitmap of all graphics and text selected will also be made available with WMF as noted above. If you want editable text, paste with the standard paste process as before. "Paste special" is required only for graphics.

Chapter 6: UNIX Installation and Configuration

Standalone SwiftView® for UNIX X Window systems is delivered with the following files:

In . /bin_* (e.g., bin_solar):

sview	SwiftView user program (executable)
svutil	SwiftView "no user interface" utility program (executable)
svstart. ics	Sample startup ICS file with sample printer definitions
NDGSview	Optional (example) SwiftView X resource file
p300x300/*	PCL 300 dpi font files - Roman 8 symbol set only
SwiftBM	SwiftView icon bitmap

NOTE: each system type is delivered into a different bin_* directory. This enables extracting of each system release into the same directory, maintaining both the system specific subdirectories and the common subdirectories. The system specific binary subdirectories are:

bin_hpux	HP-UX V9 and V10 on PA_RISC_1. 1
bin_solar	Sparc Solaris 4. 3+ systems
bin_r6k	RS-6000 AIX, 3. 2+
bin_linux	Linux
bin_sunos	Sparc SunOS 4. 1.2+
bin_sco3	SCO ODT 3
bin_sgi	Silicon Graphics 5.3+
bin_dec	Alpha DEC UNIX

Once the files are extracted, the only other requirement is to define the environment variable NDGUTIL to point to the bin_* directory where sview and svutil were extracted. If NDGUTIL is not defined, SwiftView gives warnings when starting. SwiftView is still fully functional without NDGUTIL defined, except that printers will not be defined and font files required for viewing PCL will not be found.

Also, in order to configure printing you must modify the supplied sample $NDGUTIL/SVSTART.ICS to define printers as described below.

Configuring UNIX Swiftview

There are many ways to configure UNIX SwiftView's appearance and file handling. All of these incorporate SwiftView's ICS commands and can be passed through various configuration files, as well as from a command prompt. The following sections detail each of these configuration methods; since most are very similar to the Windows settings there will not be a re-explanation of formatting.

UNIX Standalone configuration file (SVSTART.ICS)

SVSTART.ICS acts on UNIX SwiftView just as the SVIEW.INI does for Windows Standalone: it sets the default configuration for SwiftView. SVSTART.ICS is in the same format as an ICS file described earlier in Chapter 2 and can be used to pass any ICS commands in the same way.

Printer definition using svstart.ics

The following example shows how to configure printers using SVSTART.ICS. In this example we assume that the system manager places all SwiftView files in /usr/ndg/bin_solar. The following would be placed in /etc/profile or $HOME/. profile (for most shells except csh):

```
PATH= $PATH:/usr/ndg/bin_solar; export PATH
NDGUTIL= /usr/ndg/bin_solar; export NDGUTIL
```

The system manager could write $NDGUTIL/SVSTART.ICS as:

```
ICS
printer number 0 type PCL5 command "lpr -Pprint1 -r -h" alias "Ljet"
printer number 1 type POST2 command "lpr -Pprint2 -r -h" alias "Pscript"
printer default 0
set filename sviewout
```

The above environment variable definitions, and SVSTART.ICS file display the following functionality:

1) SwiftView can be executed by typing "sview", because its location is in the user's PATH.
2) The default printer is 0, a 300 dpi PCL 5 (LaserJet 3/4 compatible) printer.
3) The default output file name is "sviewout"
4) The command "lpr -Pprint1 -r -h" is used to print on the default printer. SwiftView does not delete this temporary file, but the "-r" in the print example above does delete it.
5) Each page is submitted as a separate print job. The "-h" in the example removes the header that would otherwise appear on each page.
6) NOTE: lpr may not accept files above a certain size (e.g., 1 MB), and image files, especially PostScript level 1 files, can be much larger than 1 MB. The -s option should be used in this case.
7) Users can also select printing on "Pscript", a Postscript Level 2 printer, specified as the system device "print2".

ICS Files

ICS files are the same in UNIX as in Windows. To see a detailed explanation, as well as formatting, please refer to Chapter 2: How to Configure SwiftView.

Command Line Parameters

Simply type "sview" to run SwiftView. "sview filename" runs SwiftView and loads "filename" for viewing. Type "sview -h" to see the following command line information:

```
SwiftView(R)
Version 7.0.2
SwiftView, Inc.
 Copyright (c) 1989-2004 SwiftView, Inc.
 All Rights Reserved

USAGE: sview xxxxx
        where:  xxxxx is optional file name to view
        (first param without leading '-' is file name)
OPTIONAL FLAGS:
 -uccccc    -> c may be any combination of: E=Exit, A=Draw all, W=Draw wide
              P=Print, S=Convert, F=Open, R=Rotate, H=Help.
              (Default:  -uEAWRPFH)
 -wf1,f2,f3 -> Set viewing window.
              f1=xaspect, f2=yaspect, f3=max screen used
              (Ex: -w8.5,11,.7 -> 8.5x11 using 70% of screen)
              Optional f4,f5 (-1.0 to 1.0) control window position
 -aXXX      -> XXX is the name of ICS pipe
 -AXXX      -> XXX is the name of ICS pipe - newline terminated commands
 -pN        -> N is file descriptor of ICS pipe
 -prN       -> N is file descriptor of ICS pipe - newline terminated commands
 -iconic    -> Startup iconic
 -ics"..."  -> Initial commands separated by ';' or '|'
 -licLICSTR -> customer-specific license
 -logo      -> Turn on SwiftView logo button
 -lserv host[:port] -> SwiftServe license server
 -nopc      -> Turn off check for parent process death (Default: on)
 -noshow    -> Startup hidden (neither iconic nor normal)
 -pageoff   -> Turn off paging controls
 -sboff     -> Turn off drawing area scrollbars
 -visual truecolor  -> Use X server truecolor visual
 -visual N  -> Use X server visual ID=N
 -windialogs-> Dialogs iconify independently of main window
 -h         -> This help message.
 Standard X command line options are valid.
 Log file $HOME/.sviewerror or ./sviewerror will be written
SwiftView(R) Version 7.0.2 Copyright (c) 1989-2004 SwiftView, Inc.
 All Rights Reserved
```

Any or all of the characters representing user interface buttons may be used with the -uccc flag.
Buttons available are:

```
   E: Exit
   A: All (redraw all of current page)
   W: Wide (spread top of page across the full window)
   P: Print (current document on default printer)
   O: Output (general printing and TIFF output)
   F: File Open dialog
   R: Rotate the current page' s image
```

The "-pageoff" flag turns off the "paging group" in the upper right hand corner of the SwiftView window.

Buttons are displayed in the order specified starting from the left, except for the "paging group" which is always in the upper right corner. Only those specified are displayed. For example, "-uA" gives "All" and the "paging group" only. "-uAP" gives "All" "Print", and "paging group".

The default is equivalent to -uEAWOFR (Exit, All, Output, Open, Rotate). The ICS commands associated with buttons defined this way are fixed and may not be modified. Use the X resource file as described below if more modifications are required.

Initial size of the document viewing area is set using three real numbers with no embedded spaces as follows:

```
-wf1,f2,f3,f4,f5
   f1 X component of aspect ratio
   f2 Y component of aspect ratio
   f3 Max % of either screen dimension used
   f4 Optional X position
   f5 Optional Y position
```

"-w8. 5,11. 0, 0. 7" sets the aspect ratio of the SwiftView drawing area to that appropriate for a letter size page, and sets the total size to use 70% of the height of a standard screen. This example could also have been "-w17, 22, 0.7", because only the ratio of width to height (aspect ratio) is used. SwiftView scales whatever page is loaded to best fit the specified drawing window shape and size.

Adding the last two optional parameters to the "-w" command enables control of window position on the screen in a manner similar to the X "-g" parameter. Values are from -1. 0 to + 1.0 where positive numbers give the % of screen the upper left corner of the window is from the upper left corner of the screen and negative numbers control the lower right corner in the same manner. "0, 0" places the window at the upper left corner of the screen. "-0. 001,-0. 001" places it at approximately the lower right corner of the screen.

This feature is fully functional only with Motif 1. 2 compatible window managers and can be used to simultaneously control window size and position better than using the separate "-g" parameter.

"-nopc" (no parent check) turns off the following default function. SwiftView occasionally checks for existence of the parent process that launched it. If that process does not exist, then SwiftView terminates. This precludes runaway SwiftView processes caused by improper exit of certain well known IS application development products.

"-A" ("-pr") defines the named pipe (pipe) through which ICS commands are submitted. Each command must end with a "newline" (\n) character. "-a" ("-p") provide the same functions without requiring newline termination. We recommend requiring new lines.

UNIX Batch Mode (svutil)

If you want to run SwiftView for UNIX in batch mode you need to use SVUTIL, which is included in the standard installation package. SVUTIL is the non-gui execution of SwiftView for UNIX, and is run from a command line just like SVIEW, using the -c parameter to specify ICS commands. Here is a small example of using SVUTIL to convert a file to TF_G4:

```
svutil -c"ldoc doc1| save TF_G4 all1 doc1. doc"
```

This runs SwiftView without any type of GUI and converts the file doc1 to a tiff.

Other UNIX Details

Interprocess communications - UNIX named pipes

Normal systems

Named pipes are a simple method of UNIX inter-process communication. This example shows how to use SwiftView with named pipes from the UNIX prompt. The identical commands could be sent using "system" calls in your database "4GL" or other environment.

A unique named pipe is required for each process using SwiftView. Using the process ID is one simple way of doing this:

```
/etc/mknod /tmp/fifo$$ p
```

NOTE: if you are using "system" calls you must use the process ID of your own program in order to use it later. Doing a system call will give a different process ID each call, because $$ gives the process ID of the invoked shell.

Start SwiftView with the named pipe and any other desired command line parameters:

```
sview -a/tmp/fifo$$ -uAE
```

PLEASE NOTE THAT THERE IS NO SPACE between "-a" and the name of the named pipe. If a space is inserted, the name will be used as a document to be viewed and the named pipe will not be functional.

Send commands to load a document called "docfile" and display its first page as follows:

```
echo "ldoc docfile\ndraw" > /tmp/fifo$$
```

Terminate SwiftView with:

```
echo "quit" > fifo$$
```

Certain commands that occur in pairs (e.g., "ldoc filename" " draw") should be written to the named pipe from a shell script or from a "4GL" environment in a single write operation separated with "new lines", as shown in the example above. SunOS systems require use of "/usr/5bin/echo" for the ' \n' processing to be handled properly by the shell.

Special fix for SYS V. 3 and other older systems

The examples above work for all SYS V. 4 or SunOS 4. 1+ compatible UNIX systems. However, SYS V. 3 systems, such as SCO Open Desktop 2, IBM AIX and HP-UX, as well as Solaris 2.x don' t properly implement use of named pipes in X Window applications. We supply the source code for a small program called "ndgstart" which converts "named pipes" into "pipes" to solve the problem. If there is any doubt about your system, use "ndgstart". "ndgstart" executes the example above as follows:

```
mknod /tmp/fifo$$ p
ndgstart /tmp/fifo$$ sview "-uAE -title Documents" &
echo "ldoc docfile\ndraw" > /tmp/fifo$$
```

Note that nothing has changed except that the program NDGSTART is used in addition to SVIEW. The second parameter (SVIEW) is a keyword for the program to execute (not an arbitrary name) which executes the program $NDGUTIL/ SVIEW.

Contact us at www.swiftview.com/support/contact-support/ for the source code for NDGSTART.

Interprocess communications - UNIX pipes

1. Command line
2. Start SwiftView: `sview -p0`
3. Now type commands into the original window that you started it from as follows:
   ```
   ldoc docfile
   draw
   ```

This is most useful for testing and demonstrating SwiftView in order to determine how you wish to use it. NOTE: some window managers "kill" the original window when SwiftView closes its STDIN upon EXIT. Type "cat | sview -p0" to prevent this.

Compiling and linking "svio. c"

Use of pipes from a program is simplified by compiling and linking the supplied C module "./ipc/svio. c" with your application. This is the same module discussed under "Special Fix for SYS V. 3" above and connects your program to "sview" or "svutil" using pipes. "svio. c" provides the following basic entry points:

```
SYNOPSIS: int StartSview(cmdline)
  char *cmdline; SwiftView command line options
FUNCTION: Start SwiftView process attached through pipe. The file name $NDGUTIL/sview
isused.
RETURNS:  -1 -> Error (sview already started or can' t start)
  0 -> OK
```

```
SYNOPSIS: int CmdSview(pcmd)
  char *pcmd; SwiftView API text command
 FUNCTION: Send commands to attached SwiftView process via open pipe. If user has stopped
SwiftView, it is restarted with the original command line.
RETURNS:  -1 -> Error (sview not started or can' t restart)
  0 -> OK
```

```
SYNOPSIS: int StopSview()
```

```
FUNCTION: Stop attached SwiftView process.
RETURNS:  -1 -> Error
  0 -> OK
```

Oracle SQL*Forms 3 can' t write to files except by "echo" in a "system" call. If you don' t want to do a "system" call each time you send a command, install "svio. c" as a "user exit" as documented in the SQL*Forms demo provided by SwiftView. Use the same calls described above, but starting with "o3_": o3_StartSview(), etc.

When using INFORMIX-4GL, just define svio. c as a C file in your project with the -DUSE_INFORMIX compile option. These 4GL compatible calls named as above, but starting with "fgl_": fgl_StartSview(), etc.

Technical notes about printing

SwiftView writes each page being printed to the system "temp" place (e. g., /usr/tmp, defined by a call to tmpnam(NULL)) and immediately executes the print command specified in the installation. Default operation makes each page a separate print job. Image print files are large and writing a complete document before calling the print spooler may overflow available "temp" space on many systems. Setting the environment variable NDGONEPRINTJOB to any value will cause the entire print job to be submitted but this uses considerably more system temp space.

SwiftView does not remove the "temp" print file, because it cannot know when the print spooler no longer requires it. As a result, the system print command must remove these files or they will "build up" in the system "temp" place. This is done with the "lpr" remove flag, "lpr -r ...", as shown in the example above under "Printer definition using svstart. ics".

When using "lp" SwiftView' s print command should be a "shell script" which removes the temporary print file as follows:

```
ICS
printer number 0 type HPLJ3 command "$NDGUTIL/ svpr jetdpcl" alias "Ljet"
printer number 1 type POST2 command "$NDGUTIL/svpr jetdps" alias "Pscript"
printer default 0
set filename sviewout
```

Then place the following in the shell script $NDGUTIL/svpr and make sure it is executable:

```
lp -c -d $1 -o nobanner $2
rm $2
```

The "-c" flag forces lp to make its own copy of the print file which it deletes when it is done.
The "temp" copy generated by SwiftView is removed immediately after submission to lp.

Problem Solving - Printing

Typical printing problems are:

1. The intermediate print command file (svpr above) is not executable.
2. The print spooler modifies the print file by inserting carriage returns, form feeds, or other information. Most spoolers do this incorrectly for image printing, so you should attempt to provide SwiftView with a raw (100% unmodified data stream) printer device.
3. The SwiftView printer type is not set correctly.
4. The printer does not have enough memory (1. 5 MB or more in most laser printers).
5. Data is lost between the system and the printer due to the large amount being sent and an improper communications configuration.

If you still have problems, set the environment variable NDGDBUG to TRUE, run the simplest test case possible, and e-mail the complete test conditions and the file $HOME/.sviewerror to SwiftView Technical Support at support@paperbos.com. You can also set NDGDBUG_STDOUT to TRUE in order to see the same results on your screen as the program executes.

Problem Solving - Viewing

Typical viewing problems are:

1. The DISPLAY environment variable is not set. DISPLAY is an environment variable generally set by default by UNIX environments, however if it is not set SwiftView cannot start its window.
2. The SwiftView process does not have privilege to read the file to be viewed.
3. The TIFF file has been transferred with PC-DOS ftp using a "text" setting instead of the "binary" setting. This inserts carriage returns that corrupt the file.
4. You are trying to view an unsupported or improperly created file format.
5. Your OS version is not compatible with the current SwiftView build. See the Downloads page for supported operating systems.
6. Your system setting for dynamic libraries (e. g., LD_LIBRARY_PATH) is not correct or your system does not have those libraries.