The SCO™ CGI™

Development System

Release and Installation Notes

Version 1.1.0

The Santa Cruz Operation, Inc.

SCO Document Number: CGI/286/386-3-17-89-1.1.0E

Processed Date: 3/17/89

SCO CGI Development System Release and Installation Notes

Release and Installation Notes
Release 1.1.0
SCO Computer Graphics Interface
for Personal Computers
Development System
March 17, 1989

1. Preface

Thank you for purchasing the SCO XENIX Development System with the SCO Computer Graphics Interface (CGI) for personal computers.

These *Release and Installation Notes* contain instructions for installing the SCO CGI software on your system, and information on the software and documentation distributed with this release. They also explain how to select the options appropriate to your configuration and how to interface application programs with SCO CGI.

SCO CGI is a software library of graphics subroutines and device drivers that enables you to develop device-independent graphic application programs. The subroutines are specified in Chapter 3 of the *SCO CGI Programmer's Guide*, "SCO CGI Functions," and in the *Language Reference Guides*. To use SCO CGI with a graphics application program, include the subroutine calls in your source code and compile as usual. See the *Language Reference Guides* for information on linking your application program with the desired language-binding library.

We always appreciate hearing about users' experience with our product, as well as their recommendations for making it even more useful. All written suggestions are given serious consideration.

1.1 A Note to Developers of XENIX Graphics Applications

Continuing in our tradition of expanding XENIX environments, The Santa Cruz Operation has adapted the popular GSS*CGI™ to work closely with SCO XENIX. At this time, the SCO CGI product is included with the SCO XENIX Development System at no extra charge. With SCO CGI, developers now have the tools to create high-quality graphics applications packages for the XENIX environment.

Note

To prevent clashes between SCO CGI applications, developers should avoid using hard-coded pathnames in applications. Instead, use environment variables that can be reset if needed.

Included with the SCO CGI package, also at no additional cost, are the selected graphics device drivers listed in the section "Supported Devices" in these *Release and Installation Notes*. Developers can use these drivers to test the functionality of their graphics applications on various peripherals. These drivers are not, however, for further distribution without the prior written consent of The Santa Cruz Operation, Inc.

SCO offers the SCO Graphics Run Time Package to run graphics applications developed with the SCO CGI. Please call The Santa Cruz Operation regarding the availability of the the SCO Graphics Run Time Package, as well as for additional information.

2. Your Software Package

Your SCO CGI software package includes the following items:

- these *Release and Installation Notes*,
- the *SCO CGI Programming Guide*, *Device Driver Supplement*, *C Language Reference Guide*, *FORTRAN Language Reference Guide*, and *Pascal Language Reference Guide*,
- the distribution media that you use to install SCO CGI on your hard disk,
- a software license agreement,
- a serialization card containing your serial number and activation key, both of which are needed to install SCO CGI, and
- a Customer Registration Form, which you should complete and return to us within five days of receiving your software package. To be eligible for our support services, please supply all information requested on the form.

2.1 Supported Environments

SCO CGI runs on the media types and host operating systems listed below:

Media
3½" 135 tpi dsdd diskettes
5¼" 48 tpi dsdd diskettes
5¼" 96 tpi dshd diskettes

Operating System
SCO XENIX 386 System V Release 2.2 and later
SCO XENIX 286 System V Release 2.2 and later

2.2 Disk Usage

The contents of this distribution consume approximately 4000 Kbytes of disk space. Most files are extracted into directories beginning with */usr*. Be sure there is enough space in the filesystems, as indicated in the table below, before you attempt to install SCO CGI.

Filesystem	Usage
/usr	3800 Kbytes
/tmp	200 Kbytes

2.3 Memory Usage

It is difficult to give exact memory usage because each computer configuration is different. Therefore, these are the approximate minimum-memory requirements for one user and for each additional user:

System Load	Memory Requirements
One user	128 Kbytes
Each additional user	64 Kbytes

2.4 Supported Devices

The graphics device drivers included with the SCO CGI package are:

PostScript™
Enhanced Graphics Adapter (EGA)
Color Graphics Adapter (CGA)
CGABW
Video Graphics Array (VGA)
Hercules™

HP® Plotter
HP LaserJet®
HP ThinkJet®
HP PaintJet®
SCO Metafile
Epson® MX™/FX™ 80/100 printers

3. Installing SCO CGI

Before you can install SCO CGI on your hard disk, you need the following items:

- the SCO CGI distribution media,
- your serial number, which is an alphanumeric code located on your serialization card, and
- your activation key, which is an alphabetic code located on your serialization card.

To install the SCO CGI software, you must be logged into the *root* account as the "super user." The super user has access to all the system files, so be careful not to overwrite, corrupt, or delete those files by accident.

3.1 Using custom

1. Log in as *root*.
2. Type **custom** and then press <Return>. You see:

```
1. Operating System
2. Development System
3. Text Processing System
4. Add a Supported Product
```

3. Select option 4 — Add a Supported Product (SCO CGI). The system displays:

```
Installing custom data file...

Insert distribution volume 1
and press <Return> or enter q to quit.
```

4. Insert SCO CGI volume 1 into the primary drive and then press <Return>.

5. The system displays the following menu:

```
1. Install one or more packages
2. Remove one or more packages
3. List the files in a package
4. Install a single file
5. Select a new set to customize
6. Display current disk usage
7. Help
```

6. Choose option 1 — Install — to install any of the packages in the product.

7. The packages contained in SCO CGI are displayed on your screen. You see:

Name	Inst	Size	SCO CGI Package
ALL	No	8000	SCO CGI Extended Package
286	No	618	SCO CGI 286 Libraries
386	No	414	SCO CGI 386 Libraries
DRIVER	No	5356	SCO CGI Drivers
TEST	No	1120	SCO CGI Test Files
EGA	No	378	SCO CGI EGA Driver

8. You are prompted to enter the packages you wish to install. Type **all**, for all of the SCO CGI packages, then press <Return>. If you later discover you do not need all of the SCO CGI packages, you can easily remove them using **custom**.

9. You see:

```
Insert SCO Computer Graphics Interface volume 1
and press <Return> or enter q to return to the menu:
```

Volume 1 should still be in the drive. Press <Return>. You see:

```
Extracting Files...
```

If this release of SCO CGI has more than one volume, you are asked to load each additional volume in turn.

A printed list of the distribution media files is included in the section "The SCO CGI Files" in these *Release and Installation Notes*.

10. After all the files are extracted, a restricted rights legend is displayed on your screen.

11. You are prompted to enter your serial number. The system displays:

```
SCO CGI serialization

Enter your serial number or enter q to quit:
```

Enter your serial number exactly as it appears on the serialization card, including the three-letter prefix. Press <Return>.

12. The system then displays:

```
Enter your activation key or enter q to quit:
```

Enter your activation key exactly as it appears on the serialization card. Press <Return>. If you mistype your serial number or activation key, you are prompted to enter them both again.

After you enter your activation key, the system displays:

```
Checking file permissions...
```

You are returned to the **custom** menu. Enter **q** to have a system prompt returned to you. Remember to remove the distribution media from the disk drive.

For the safety of your files, do not use SCO CGI when logged in as the super user. File permissions protect you from unintentionally overwriting certain files when using SCO CGI as a normal user. However, when logged in as the super user, you can overwrite any file. Therefore, leave super-user mode by logging out of the root account before you actually use SCO CGI.

SCO CGI is now installed and ready to use. To begin working with it, log into a user account and refer to the SCO CGI documentation for further instructions.

3.2 The SCO CGI Files

All of the SCO CGI files are listed below. Some of the files used for installing SCO CGI are automatically removed after the installation is complete. Each of these is marked with an asterisk (*). Other files, not originally found on the distribution media, but created during installation, are included in this list.

./etc/perms/cgi

./tmp/brand *
./tmp/fixperm *
./tmp/init.cgi *
./tmp/init.ega *
./tmp/install *

./usr/lib/386/Fbindcgi.a
./usr/lib/386/Pbindcgi.a
./usr/lib/386/Slibccgi.a
./usr/lib/Llibccgi.a
./usr/lib/Mlibccgi.a
./usr/lib/Slibccgi.a

./usr/lib/cgi/cgiprep
./usr/lib/cgi/ddmeta

./usr/lib/cgi/fonts/fontlist.dat
./usr/lib/cgi/fonts/ibmbw.bld
./usr/lib/cgi/fonts/ibmbw.mon
./usr/lib/cgi/fonts/ibmbw.std
./usr/lib/cgi/fonts/ibmco.bld
./usr/lib/cgi/fonts/ibmco.mon
./usr/lib/cgi/fonts/ibmco.std
./usr/lib/cgi/fonts/ibmega.bld
./usr/lib/cgi/fonts/ibmega.mon
./usr/lib/cgi/fonts/ibmega.std
./usr/lib/cgi/fonts/instfont
./usr/lib/cgi/fonts/1ja.1
./usr/lib/cgi/fonts/1ja.2
./usr/lib/cgi/fonts/1ja.3
./usr/lib/cgi/fonts/1jb.1
./usr/lib/cgi/fonts/1jb.2
./usr/lib/cgi/fonts/1jb.3
./usr/lib/cgi/fonts/1jb.4
./usr/lib/cgi/fonts/1jb.5
./usr/lib/cgi/fonts/1jb.6
./usr/lib/cgi/fonts/1jf.1
./usr/lib/cgi/fonts/1jf.2
./usr/lib/cgi/fonts/1jf.3
./usr/lib/cgi/fonts/1jf.4
./usr/lib/cgi/fonts/1jf.5
./usr/lib/cgi/fonts/1jf.6
./usr/lib/cgi/fonts/1jl.1
./usr/lib/cgi/fonts/1jl.2
./usr/lib/cgi/fonts/1jl.3
./usr/lib/cgi/fonts/1jm.1
./usr/lib/cgi/fonts/1jm.2
./usr/lib/cgi/fonts/1jm.3
./usr/lib/cgi/fonts/1jn.1
./usr/lib/cgi/fonts/1jn.2
./usr/lib/cgi/fonts/1jn.3
./usr/lib/cgi/fonts/thinkjet.bld
./usr/lib/cgi/fonts/thinkjet.std

./usr/lib/cgi/gstdd

./usr/lib/cgi/sample/Fcgifctns.INC
./usr/lib/cgi/sample/Fcgitest.ftn
./usr/lib/cgi/sample/Fsignalc.c
./usr/lib/cgi/sample/Pcgitest.pas
./usr/lib/cgi/sample/Psignalc.c
./usr/lib/cgi/sample/cexit.c
./usr/lib/cgi/sample/cgifctns.h
./usr/lib/cgi/sample/cgitest
./usr/lib/cgi/sample/cgitest.c
./usr/lib/cgi/sample/cgitypes.h
./usr/lib/cgi/sample/c1sprintf.c
./usr/lib/cgi/sample/globcom.INC
./usr/lib/cgi/sample/makefile
./usr/lib/cgi/sample/pasfctns.INC

./usr/lib/cgi/sample/ctest/Makefile
./usr/lib/cgi/sample/ctest/README
./usr/lib/cgi/sample/ctest/cgifctns.h
./usr/lib/cgi/sample/ctest/cgistart.c
./usr/lib/cgi/sample/ctest/cgistuff.h
./usr/lib/cgi/sample/ctest/cgitypes.h
./usr/lib/cgi/sample/ctest/colortests.c
./usr/lib/cgi/sample/ctest/ctest
./usr/lib/cgi/sample/ctest/ctest.c
./usr/lib/cgi/sample/ctest/ctest.h
./usr/lib/cgi/sample/ctest/cteststuff.c
./usr/lib/cgi/sample/ctest/filltests.c
./usr/lib/cgi/sample/ctest/gintests.c
./usr/lib/cgi/sample/ctest/gtexttests.c
./usr/lib/cgi/sample/ctest/linetests.c
./usr/lib/cgi/sample/ctest/marktests.c
./usr/lib/cgi/sample/ctest/pixeltests.c
./usr/lib/cgi/sample/ctest/scrnsaver.c
./usr/lib/cgi/sample/ctest/timer.c
./usr/lib/cgi/sample/ctest/wkoutparms.c

./usr/sys/io/cn286.o
./usr/sys/io/cn386.o
./usr/sys/mdep/machdep286.o
./usr/sys/mdep/machdep386.o

3.3 The SCO CGI Drivers

The following drivers are distributed with SCO CGI:

./usr/lib/cgi/cgabw
./usr/lib/cgi/cgaco
./usr/lib/cgi/ega
./usr/lib/cgi/epson100
./usr/lib/cgi/epson80
./usr/lib/cgi/hercules
./usr/lib/cgi/hpplot
./usr/lib/cgi/laserjet
./usr/lib/cgi/postscript
./usr/lib/cgi/paintjet
./usr/lib/cgi/thinkjet
./usr/lib/cgi/vgabw
./usr/lib/cgi/vga16
./usr/lib/cgi/vga256

4. Features of SCO CGI

The following section describes new features of SCO CGI 1.1 that are not covered in the standard SCO CGI documentation.

4.1 Signals

SCO CGI no longer interferes with the way an application uses signals. This section explains how this change reduces the possibility of uncaught signals. There are also step-by-step instructions on recovering from the effects of uncaught signals, for people still using certain agents from SCO CGI 1.0.

The SCO CGI binding creates one agent process for each call to Open Workstation. This agent is a separate process that performs all CGI operations and communicates with the application via shared memory and semaphores. An uncaught signal can kill either the application-binding process or the agent process, leaving an orphan process waiting on semaphore signals that might never come. The uncaught signal also leaves lock files, shared memory, and semaphore resources outstanding after the death of one of the processes. The resulting situation is difficult to clear up, especially when the console is left in graphics mode and the user can see no command prompt.

4.1.1 Preventing Uncaught Signals

To reduce the frequency of uncaught signals, the following procedures are included in this release of SCO CGI:

1. On the first call to Open Workstation, and only on the first call, the binding sets any signals that are set to SIG_DFL to be caught by the binding's stray-signal catcher. (Signals set to SIG_DFL are those signals not yet set by the application.) This stray-signal catcher shuts down the SCO CGI system gracefully if it catches any of these signals.

2. In addition, the agent process spawned during any Open Workstation call sets all signals to be caught by the agent's own signal catcher. Except for the signal SIGTERM, which is sent to the agent by the binding to shut the agent down, all other signals to the agent process are ignored.

If the agent receives a fatal signal (SIGILL, SIGTRAP, SIGBUS, SIGSEGV, or SIGPWR), it attempts to pass the signal to the application and waits for the binding to return SIGTERM. If you need to have the application catch signals and exit upon catching them, you must accommodate the above safety features by calling Close Workstation for each SCO CGI device still open.

It is very important that you do not set any signals to be SIG_DFL after the first call to Open Workstation. If you do, the signal set at SIG_DFL can leave outstanding agents hanging, killing the application-binding process. The application can set any signals before the first call to Open Workstation to SIG_IGN, or to any function address the application chooses. However, if the application is going to handle the fatal types of signals and shut down the SCO CGI system gracefully from within the application's signal handler, that signal handler must call Close Workstation for each open CGI device. After that is done, the application can exit without problems.

Any signals the application has not set to SIG_IGN before the first call to Open Workstation cause the function to return the error indication *-1* during the following operations:

- Request/Sample Choice
- Read Cursor Keys
- Request/Sample Locator
- Request/Sample String
- Request/Sample Valuator

If you respond to the *-1* error indication with **Inquire CGI error**, the system returns *-3000*, signifying a device driver error. No SCO CGI functions other than those listed above are interrupted. If the application is expecting to catch the signal, it sees the signal after the CGI function has completed. CGI functions cannot remain partially done; they will be completed whether or not error status is returned.

4.1.2 Recovering From an SCO CGI Crash

If an SCO CGI function has been abnormally terminated for any reason, the following steps must be undertaken to restore order to the console:

1. If the console is stuck in graphics mode and the application process is hung up, you see no shell prompt. Go to a terminal on a serial port on the same system, and determine the process IDs by typing:

   ps -t tty*00*

   where *00* is the tty number of the console that is stuck. Then kill those processes by sending either **kill -15** or **kill -9** to them.

2. Once the hung processes are killed, the stuck console can be restored to alpha text mode with the **vidi** command described in VIDI(C) and SCREEN(HW). For example, for a color display 80 columns by 25 rows, type:

   vidi C80x25

   For XENIX systems earlier than 2.3, you must use the **stty** command described in STTY(C). For a color display 80 columns by 25 rows, type:

   stty C80x25

   If the **vidi** or **stty** command does not restore the console to alpha text mode when you terminate it with <Return>, it means the tty port is not in sane mode. Enter the command again, and terminate it with <CTL>**j**. Then you must also enter the following command to restore the tty port to sane mode:

   stty sane <CTL>**j**

3. Your command line prompt should now be restored. Next you must delete any outstanding semaphores and shared memory. First enter:

   ipcs

   There are two possible responses to this command. If there are no outstanding semaphores or shared memory, your screen displays something similar to the following:

```
IPC status from /dev/kmem as of Fri Aug 12 15:59:12 1988
T         ID          KEY         MODE        OWNER       GROUP
Message Queues:
Shared Memory (5.0):
Shared Memory (3.0):
Semaphores (5.0):
Semaphores (3.0):
```

   In this case, go on to Step 5.

4. The other possible response to the **ipcs** command is the following type of screen display:

```
IPC status from /dev/kmem as of Fri Aug 12 16:02:09 1988
T     ID          KEY         MODE        OWNER       GROUP
Message Queues:
Shared Memory (5.0):
m     1           0x00000001  --rw-------  hiramc      drifters
Shared Memory (3.0):
Semaphores (5.0):
s     430         0x00000001  --ra-------  hiramc      drifters
Semaphores (3.0):
```

   This indicates the login name hiramc has a shared memory and a semaphore. To remove them, note their ID numbers, and then enter the following **ipcrm** command for each outstanding shared memory or semaphore resource:

   ipcrm -m *memory_ID* **-s** *semaphore_ID*

Note that the removal does not take place if there are any processes still attached to those resources. That is why all processes must be killed in Step 1.

5. Finally, there might be lock files in */tmp* that must be removed. To discover whether there are any, type:

   ls -l /tmp

   Lock files show up in a line similar to the following:

```
-rw-rw-rw-  1 hiramc  drifters  22 Aug 12 16:08 LCK..tty08
```

   Remove any such files with the following command:

   rm /tmp/LCK..tty08

   Sometimes a lock file left in */tmp* has unprintable characters in its name. Such a file cannot be removed with the above **rm** command. To find a filename with unprintable characters, type:

   ls -i /tmp | cat -v

   The file is displayed with its "inum" (in this case, 3430) as follows:

```
3430 LCK..tty08
```

   Now remove the offending file with the following command, using the file's inum:

   find /tmp -inum *inum* **-exec rm -f {} \;**

4.2 Mouse Support

The SCO CGI 1.1 screen drivers allow you to take advantage of the mouse support offered by SCO XENIX System V Release 2.3. Mouse movement when in Request Locator is tracked by the cursor, and any mouse key click terminates the call as if a return key was pressed. Sample Locator also tracks any mouse movements. In addition, the state of the mouse button(s) is

returned in the keystate, pressed, and released output fields of the Sample Locator call. Each of these fields has the value 0x80 in the upper byte if Sample Locator is returning the state of the mouse buttons. The upper byte has the value 0x00 if Sample Locator is returning the state of the keyboard. In both locator calls, keypad input is still accepted. Note that this is for SCO CGI 1.1 screen drivers only. Release 1.0 screen drivers do not have this capability. For a list of supported mouse hardware, please consult your SCO XENIX 2.3 *Release Notes*.

4.3 Hercules Graphics Cards

SCO CGI 1.1 supports the Hercules Graphics Card and the Hercules Graphics Card Plus high-resolution monochrome displays. This release does not contain raster fonts specifically designed for the Hercules Graphics Card, other than its one native font. However, any of the 11 fonts provided for IBM CGA, IBM EGA, and HP ThinkJet can be used with the Hercules driver. In general, text displayed in one of these fonts appears to be slightly larger or slightly smaller than text in the native font.

To make use of these fonts, you must set up the FONTS environment variable as described in the section "Font Utility Programs" later in these *Release and Installation Notes*. You also need to make sure that the fonts are not rejected as unsuitable. To do this from the Bourne shell, type:

```
VERIFYFONT=OFF
export VERIFYFONT
```

From the C shell, the command is:

```
setenv VERIFYFONT OFF
```

These commands make more fonts available to other device drivers as well. If this presents a problem, you should set VERIFYFONT=ON when using other drivers. Note that these fonts can be used only in graphics mode, never in cursor-text mode.

The Hercules display driver is supported under SCO XENIX System V Release 2.2.2 and later. However, setting cursor-text mode and restoring cursor-text mode on exit cause SCO XENIX 2.2.2 through 2.2.5 to return a 25x80 screen. Kernel patches to solve this problem are detailed below. The characters the user enters (in **bold type**) and the characters the system returns (in roman type) are both shown.

On a 386 machine, enter the following while logged in as *root*:

```
adb -w /xenix -
cnioctl+0x522?x
_cnioctl+0x522:  0x50
cnioctl+0x522?w 0x19
_cnioctl+0x522:  0x50+   0x19
cnioctl+52c?x
_cnioctl+0x52c:  0x19
cnioctl+0x52c?w 0x50
_cnioctl+0x52c:  0x19+   0x50
$q
cd /
reboot
```

For a 286 machine, use the following patch:

```
adb -w /xenix -
cnioctl+0x426?x
_cnioctl+0x426:  0x50
cnioctl+0x426?w 0x19
_cnioctl+0x426:  0x50+   0x19
cnioctl+0x42c?x
_cnioctl+0x42c:  0x19
cnioctl+0x42c?w 0x50
_cnioctl+0x42c:  0x19+   0x50
$q
cd /
reboot
```

5. Upgrade Notes

This section describes important differences between this release and the previous release of SCO CGI.

- Use of mouse and bit pad is now supported. See the section "Mouse Support" earlier in these *Release and Installation Notes* for further details.

- SCO CGI 1.1 supports Video Graphics Array (VGA), the HP PaintJet printer, the Hercules Graphics Card, and the Hercules Graphics Card Plus. The Apple® LaserWriter® support has been expanded to include all PostScript printers.

- Under SCO XENIX 2.2, accessing the high-resolution text font of certain non-IBM EGA cards filled the screen with random, meaningless characters. To eliminate this problem under SCO XENIX 2.3, the screen agent now enters cursor-addressable text mode using whatever text mode is present when the SCO CGI program is executed.

- With SCO CGI 1.0, the final element in the device driver logical name stored in *work_in*[11] to *work_in*[18] could not be a null (0). This limitation was contrary to XENIX/UNIX convention, and has been corrected. Now both the blank space and the null are legal final elements for the device driver logical name.

- The VDIPATH environment variable in SCO CGI 1.0 is now the CGIPATH environment variable. See the section "Environment Variables" later in these *Release and Installation Notes* for more information.

- The operation of the HP plotter has changed for this release of SCO CGI. Under release 1.0, the driver queried the plotter for the plotter's identification number and the paper sizes available. By eliminating interaction between the driver and the plotter, release 1.1 facilitates the use of SCO CGI output devices in a networking environment and allows spooling of SCO CGI output. Therefore, the user must specify the plotter's identification number in the

environment variable PLOTID. The paper size should be set in the environment variable PAPER. The resulting output should be directed into a file, or piped through **lp**(C) to a spooled output device.

- For SCO CGI release 1.1, Apple LaserWriter support has been expanded to include PostScript printers in general. In addition, the printer is no longer required to be connected directly to the system running the driver, as was the case with release 1.0. Output from the SCO CGI 1.1 PostScript driver can be spooled or redirected to a temporary file for later printing. Finally, you can change the output resolution, in dots-per-inch, using the RESOLUTION environment variable. The default resolution is 300dpi.

  At this time, the PostScript driver supports only the following font types:

SCO CGI Font Number	PostScript Font
0	Courier
1	Courier-Bold
2	Courier-Oblique
3	Times-Roman
4	Times-Bold
5	Times-Italic
6	Helvetica
7	Helvetica-Bold
8	Helvetica-Oblique
9	Symbol
10	Times-BoldItalic
11	Helvetica-BoldOblique

  The environment variable CGIPREP specifies the name of the PostScript initialization file for the driver. This file must reside in the directory specified by the environment variable CGIPATH. If it is not found, the driver does not execute successfully.

- Pascal and FORTRAN languages are now supported, and their use with SCO CGI is fully documented in the *Pascal Reference Manual* and the *FORTRAN Reference Manual* distributed with this release.

- Signal-handling under SCO CGI 1.1 has been improved. Problems SCO CGI 1.0 had with certain signals (for example, the <DEL> key) have been solved.

6. Environment Variables

To use SCO CGI, you must export certain environment variables to the XENIX Operating System. This can be done from the operating system, in either the Bourne shell or the C shell. The environment variables can also be set in *.profile* (Bourne shell) or *.login* (C shell) files which are executed each time a user logs on.

6.1 Binding (SCO CGI libraries)

CGIPATH	Full pathname of the location of the SCO CGI drivers (usually */usr/lib/cgi*).
FONTS	Allows you to use raster fonts. For further information, see "Font Utility Programs" and "The instfont Program" in these *Release and Installation Notes*.
SHMMAX	Size of shared memory segment for SCO CGI applications and drivers.
VDIPATH	Former name of CGIPATH, included for 1.0 compatibility.

6.2 SCO CGI Drivers

CARTRIDGE (HP LaserJet driver only)	Specifies which font cartridge(s) are installed. Valid values are A-Z and a-z.
CGIPREP (PostScript driver only)	Initialization filename. Default is *$CGIPATH/cgiprep*.
EGA (EGA driver only)	Specifies the EGA resolution mode to use.

EGAMEM (EGA driver only)	Amount of memory available on EGA card. Valid values are 64, 128, 192, and 256; default is 256.
EGASWITCH (EGA driver only)	Specifies the numeric mode for the EGA card when the driver is running.
ORIENTATION (Printer drivers only)	Sets orientation of output. Valid values are landscape and portrait; default is portrait.
PAPER (Epson 100 driver)	Determines paper width. Default is narrow; W specifies wide-form paper.
(HP plotter driver)	Specifies paper type. Valid values are A, B, A4, and A3; default is A.
PLISTSIZE (Bit-mapped printer drivers only)	Specifies the size, in Kbytes, of the buffer used to create a bit-mapped image of the output frame. Valid values range from 8 to the maximum amount of memory that can be allocated.
PLOTID (HP plotter driver only)	Specifies the model number of the HP plotter used with the driver. Valid values are 7090, 7440, 7475, 7510, 7550, 7580, 7585, and 7595; there is no default.
RESOLUTION (HP LaserJet, HP PaintJet, and PostScript drivers only)	Specifies output resolution in dots-per-inch (dpi). Valid values for the HP LaserJet are 75, 100, 150, and 300; default is 150. Valid values for the HP PaintJet are 90 and 180; default is 90. Any value greater than 1 is acceptable for the PostScript driver; default is 300.
ROTATION	Identical to ORIENTATION.

SWITCHBW (HP PaintJet driver only)	Determines background color. Valid values are OFF for white and ON for black; default is OFF.
TEMPDIR (Bit-mapped printer drivers only)	Specifies where to put buffers of portions of the output frame before output occurs. Default is */tmp*.
VERIFYFONT (All drivers with bit-mapped fonts)	Determines whether to validate bit-mapped fonts for use with the driver. Valid values are ON and OFF; default is ON.

6.3 Setting SCO CGI Environment Variables

1. Set the CGIPATH parameter. (In SCO CGI 1.0, this was the VDIPATH parameter.) This parameter provides a path to the directory in which the device driver files reside. If placed by the installation procedure, this directory is */usr/lib/cgi*.

   Set CGIPATH from the Bourne shell by typing:

   ```
   CGIPATH=/usr/lib/cgi
   export CGIPATH
   ```

   From the C shell, type:

   ```
   setenv CGIPATH /usr/lib/cgi
   ```

2. Any device logical name that is referenced by the workstation identifier, *work_in* [11] to *work_in* [18], of the Open Workstation routine must be assigned to the appropriate device driver filename. These logical names are user-selectable; SCO CGI has no pre-assigned logical names. The device driver files must be located in the directory specified by CGIPATH.

If you pass the array elements 'C','G','I','D','I','S','P',' ', in *work_in* [11] to *work_in* [18], assign the logical device name from the Bourne shell by typing:

CGIDISP=*driver_name*
export CGIDISP

From the C shell, type:

setenv CGIDISP *driver_name*

3. The device driver filenames referenced in step 2 must be assigned to the system's physical devices. (If a device is not assigned, STDIO is assumed. See step 5.) There are two methods of assigning the filenames.

   The first method is to add the assignment of the physical device to the workstation identifier assignment shown in the previous step. For example, from the Bourne shell, type:

   CGIDISP='*driver_name* **/dev/ttynn'**
   export CGIDISP

   From the C shell, type:

   setenv CGIDISP '*driver_name* **/dev/ttynn'**

   In general, this is the preferred method, as it reduces the size of the environment.

   The second method is to assign the physical device via the driver name. For example, from the Bourne shell, type:

   CGIDISP=*driver_name*
   driver_name=**/dev/ttynn**
   export CGIDISP *driver_name*

   From the C shell, type:

   setenv CGIDISP *driver_name*
   setenv *driver_name* **/dev/ttynn**

4. If you select a printer, you might also want to pipe the output through the system's spooler.

   As an example of this, from the Bourne shell, enter the following in place of step 3:

   ```
   printer_driver='|spooler_program'
   export printer_driver
   ```

   From the C shell, enter:

   ```
   setenv printer_driver '|spooler_program'
   ```

   where *printer_driver* is the printer device driver name, and *spooler_program* is the name of the program for sending data to the print spooler.

5. To send output to the console from a serial tty, you can use the redirection symbol ">" when assigning physical devices.

   As an example of this, enter the following from the Bourne shell in place of step 3:

   ```
   CGIDISP='driver_name>/dev/ttym'
   export CGIDISP
   ```

   From the C shell, enter:

   ```
   setenv CGIDISP 'driver_name>/dev/ttym'
   ```

Note

Input redirection is not possible with the device drivers.

---

6. If you want to access the device driver's standard I/O streams instead of a physical device, you can assign the keywords **STDOUT** and **STDERR**.

As an example of this, from the Bourne shell, enter the following in place of step 3:

CGIDISP='*driver_name*>STDERR'
export CGIDISP

From the C shell, enter:

setenv CGIDISP '*driver_name*>STDERR'

7. Using redirection as described in step 5, you can assign a physical device along with a standard I/O stream.

   As an example of this, enter the following from the Bourne shell in place of step 3:

   CGIDISP='*driver_name*>STDOUT'
   export CGIDISP

   From the C shell, enter:

   setenv CGIDISP '*driver_name*>STDOUT'

8. If you want to access a regular file instead of a physical device, you can assign the filename. Filenames assigned without a full directory path specification are accessed or created in the device driver process' current working directory. The redirection symbol "**>**" can be used to truncate the assigned file before sending output. The redirection symbol "**>>**" can be used to append output to the end of the assigned file.

   As an example of this, in place of step 3, enter the following from the Bourne shell:

   CGIDISP='*driver_name*>>*file_name*'
   export CGIDISP

   From the C shell, enter:

   setenv CGIDISP '*driver_name*>>*file_name*'

9. If you send output to a metafile (device driver file *cgmdd*), you might want to assign the output metafile filename to be used instead of the default *metafile.dat*.

The commands to do this from the Bourne shell are:

METAOUTPUT=*file_name*
export METAOUTPUT

From the C shell, the command is:

setenv METAOUTPUT *file_name*

10. It is possible to tune the performance of the drivers by changing the size of the shared memory buffer used by the SCO CGI library to communicate to the drivers. This is done by setting the value of the SHMMAX environment variable. The maximum size allowable is 32 Kbytes; this is the default. The value can be set to anything between 2 and 32 Kbytes.

    From the Bourne shell, the commands to do this (for a 10 Kbyte buffer) are:

    SHMMAX=10
    export SHMMAX

    From the C shell, the command is:

    setenv SHMMAX 10

7. Font Utility Programs

To use the raster fonts provided in this release of SCO CGI, it is necessary to set up your FONTS environment variable. From the Bourne shell, the commands to set FONTS are:

FONTS=*directory_path*
export FONTS

From the C shell, the command is:

setenv FONTS *directory_path*

If you are creating your own fonts or are using fonts not in this distribution, or if you have moved fonts out of */usr/lib/cgi/fonts*, then you must run the **instfont** program described below (*/usr/lib/cgi/fonts/instfont*) to set up the file *fontlist.dat*. The **instfont** utility requires the FONTS environment variable to be set to the directory containing the fonts.

If any new fonts are placed into */usr/lib/cgi/fonts*, **instfont** should be executed.

7.1 The instfont Program

Before an application uses font files, the utility program **instfont** must be run. **instfont** installs specific font-file information into a file named *fontlist.dat*. This file is subsequently used by the device drivers when an application uses fonts.

To execute this utility, use the following command:

instfont

If you add more font files at a later date, re-execute **instfont** to update *fontlist.dat* with information about the new font files.

The environment parameter FONTS tells **instfont** the location of your font files, and then creates the *fontlist.dat* file in the FONTS directory. If FONTS is not set, this utility assumes that the current working directory contains the font files and creates *fontlist.dat* in the current working directory.

Device drivers use the FONTS environment parameter to access the *fontlist.dat* file. If FONTS is not set, the drivers assume that the current working directory contains the *fontlist.dat* file.

8. The ctest Program

The following section describes the **ctest** program included with the SCO CGI 1.1 distribution, and is based on information in *./usr/lib/sample/ctest/README*. For more information on the program, consult the *./usr/lib/sample/ctest* directory.

The **ctest** program provides a demonstration of certain SCO CGI functions, tests the SCO CGI agents, and offers examples useful to developers writing SCO CGI applications. Its files reside in *./usr/lib/sample/ctest*, and are listed in the section "The SCO CGI Files" earlier in these *Release and Installation Notes*. Each file is described in the *ctest/README* file. The following files contain the tests, and are listed here in the order in which the program calls them:

Filename	Test function
wkoutparms.c	Displays parameters returned from the Open Workstation function, parameters obtained by inquiry functions, and some useful calculated information.
linetests.c	Demonstrates some capabilities of the line-drawing functions.
pixeltests.c	Demonstrates byte pixel array output and the use of an offscreen bitmap.
marktests.c	Demonstrates polymarker functions.
filltests.c	Demonstrates fill area functions.
gtexttests.c	Demonstrates the graphics text capabilities of SCO CGI.
colortests.c	Demonstrates the color capabilities of an SCO CGI agent.
gintests.c	Demonstrates the Graphic Input (GIN) capabilities of SCO CGI.

8.1 ctest Environment Variables

Before you execute **ctest**, you must define the following environment variables:

CGIPATH — This variable must name the location of the SCO CGI agents (usually */usr/lib/cgi*).

CGIDISP — This specifies the agent to be used, such as *ega*. If a printer agent is chosen, **ctest** produces output via that agent.

CGIPRNT — This environment variable can designate an alternate display agent or a printer agent. If you run the **ctest** program with the **-p** flag, **ctest** uses this environment variable in place of CGIDISP. (The **-p** option is explained below.)

8.2 ctest Options

The following flags specify which tests to run:

Flag	Capabilities tested
-c	Color
-f	Fill area
-g	Graphics text
-i	GIN
-k	Marker
-l	Line
-r	Report of Open Workstation parameters
-x	Pixel operations

If none of the options listed above are specified, **ctest** runs the color, fill area, graphics text, marker, line, and pixel operations tests. If one or more of the flags are used, **ctest** runs only the tests specified.

The following flags determine the conditions under which the tests run:

Flag	Option description
-a	Runs **ctest** in ASTI defer mode.
-b	Runs **ctest** in BNI defer mode.
-d	Use when running **ctest** under a debugger. For further information, see the *README* file and other files in */usr/lib/sample/ctest* directory.
-h	Displays a help screen.
-m	Puts **ctest** into manual mode, causing it to wait for prompts between each screen before it continues.
-p	Governs printer selection, where the environment variable CGIPRNT specifies the output device. See the section "**ctest** Environment Variables" earlier in these *Release and Installation Notes* for further information.
-t	Runs **ctest** without pausing between screen displays and sends timing statistics to **stderr**. Save the timing statistics with the redirection command **ctest -t 2>** *filename*.
-0	Specifies full-screen coordinate transformation mode for v_opnwk.
-1	Specifies preserve aspect ratio coordinate transformation mode for v_opnwk.
-2	Specifies device units coordinate transformation mode for v_opnwk.
-3	Specifies short-axis coordinate transformation mode for v_opnwk.

Keep in mind the following defaults when considering the operational conditions options:

1. The default defer mode is ASAP.

2. The default mode for v_opnwk is full-screen coordinate transformation mode.

3. By default, **ctest** displays each screen, pauses a few seconds, then begins the next screen. The **-m** and **-t** flags offer alternatives.

9. The CGITEST Program

A test program **CGITEST** has been included with SCO CGI. The purpose of this section is to provide operational information about **CGITEST**.

CGITEST is a program that tests some of the CGI functions. The test produces a maximum of twenty separate displays, called "frames." Frames for which the functions are not supported in a particular device are not generated. For more information, refer to the *SCO CGI Programmer's Guide* and the *SCO CGI Device Driver Supplement.*

9.1 Compiling CGITEST

1. Verify that the CGIPATH parameter is set correctly. Refer to step 1 in the section "Setting SCO CGI Environmental Variables" earlier in these *Release and Installation Notes.*

2. Select a graphics output device. The demonstration program uses the logical device name CGIDISP. Verify that the proper environment variables for this device have been set. Refer to step 2 in the section "Setting SCO CGI Environmental Variables" earlier in these *Release and Installation Notes.*

3. Copy the demonstration files to your work directory using the following commands:

   cp /usr/lib/cgi/sample/cgitest.c .
   cp /usr/lib/cgi/sample/makefile .

4. To compile the demonstration program, enter:

   make

5. To execute the demonstration program, enter:

   cgitest

If you receive any error messages or your device does not display any graphics, then:

a. Check to see that all environment variables have been properly set.

b. Check to see that filenames are correct.

c. Review the capabilities of your device, listed in the *SCO CGI Device Driver Supplement.*

d. Compare any error codes with those listed in Appendix A of the *SCO CGI Programmer's Guide.*

9.2 Test Program Frame Descriptions

Frame 1

Frame 1 tests MARKERS, POLYLINES, FILLED AREA and GRAPHIC TEXT:

- First a box, enclosing the entire display surface, is drawn using the default color of 1 and the default line style of 1.

- Above a grid and an arrow inside the box are two centered graphic text strings using bottom alignment that names the test and company.

- Inside the box on the left side is drawn a column of six defined markers of standard (default) size, in one to six colors. The number of colors is limited by the device's maximum number of colors.

- The center contains a 6-by-6 grid that is a series of polylines with each row-column combination in a different color (up to the device maximum). Marker number 6 (diamond) is drawn at each row-column intersection on a diagonal of the intersections from lower-left to upper-right, except at the lower-left corner.

- Across the bottom is a row of six VDC (Normalized Device Coordinates) specific size markers centered on a horizontal line. The line and the six markers are drawn in color 1. Actual device limits might produce fewer sizes but the sizes should increase from left to right.

- Above the test name is a horizontal row of six different line-style segments drawn with increasing color index values (1-6). Each segment's length is approximately 1/8 of the screen width.

- To the right of the grid is a solid filled area (arrow) in color 1.

- Below the arrow and to the right of the lower-right corner of the grid is the version number of the device driver being used with the test.

- Above the line-styles test is a prompt message in graphic text telling the user to press <Return> to continue.

Frame 2

Note

Frames 2-5 are cursor mode tests. If your device does not support cursor-addressable mode, press <Return> for each frame.

Frame 2 tests CURSOR ADDRESSING using the device's escape functions:

- The display surface is cleared and cursor text mode is enabled.
- The test outputs a series of uppercase "A" characters diagonally downward across the screen from the upper-left corner of the display surface.
- After each "A" is written, commands are issued to move the cursor down and then right one character position. One "A" is written for each row on the display, so the number of columns used is determined by the number of rows available. In other words, the "A" in the lower-right corner of the display appears in column *x*, where *x* is the maximum number of rows in the display.
- Every other column is skipped because the cursor automatically advances one column with a move-right command after writing each character.
- The user is then prompted to continue.

Frame 3

Frame 3 tests REVERSE VIDEO and ERASE TO END OF LINE.

- The display is not cleared from Frame 2 and a series of uppercase "B" characters is written in reverse video, beginning in the lower-right corner of the display surface.
- Each "B" written is followed by a move-left and a move-up command, which result in a vertical column of "B" characters at the right edge of the display.
- The cursor is then moved to the center of the display and an erase-to-end-of-line command is issued. A single "B" is erased on the center row of the display.
- The user is prompted to continue.

Frame 4

Frame 4 tests ERASE TO END OF SCREEN:

- Again the previous frame is not cleared. The cursor is moved down four lines and an erase-to-end-of-screen command is issued.
- An inquiry is made to the device, and the returned cursor position is saved.
- The user is prompted to continue.

Frame 5

Frame 5 tests HOME CURSOR and ERASE TO END OF SCREEN.

- The previous screen is not cleared until a home-cursor command is issued, followed by an erase-to-end-of-screen command.
- An uppercase "C" character is printed in the home position, the cursor is moved to the previously saved cursor position (center

screen plus four lines down), and another uppercase "C" is written.

- The user is then prompted to continue.

Frame 6

Frame 6 tests WRITING MODE:

- The cursor text mode is exited and the display surface cleared. Four columns of output are generated by this test frame.
- The left column is the number of the writing mode being tested (1-16, bottom to top).
- The second column is the number of the writing mode written in the selected mode on a normal background.
- The third column is first written with a solid bar in index 1 color, and then the writing mode number selected is written over the bar using the selected mode.
- The rightmost column is the graphic string "ABCabc" written in REPLACE MODE.
- A horizontal line of fixed VDC length (3200) and position then overwrites a portion of the string. The number of characters overwritten with the line is dependent on the device's character size.
- The user is prompted to continue.

Frame 7

Frame 7 tests GENERALIZED DRAWING PRIMITIVES (GDPs).

- The display is cleared and sixteen bars are drawn in two rows across the bottom of the screen.

- The bottom row of bars is filled with the fill area color index incremented from 0 to 7, and the fill area style incremented from 0 to 7 (or to the device maximum). The fill area interior style index runs from 0 to the maximum number of styles available on the device.

Note

The lower-left bar is not visible as it is drawn in background color.

- The second row of bars is filled with the fill area color index incremented from 1 to 9, and the fill area interior style index at 2 (pattern). The fill area style is incremented to the maximum number of patterns available according to the following pattern: 6, 9, 12, 16, 19, 22, 26, 29. (Pattern styles 1 to 6 are the same as hatch styles 1 to 6.) The fill area color index is set to 2 for the remainder of this frame's tests.

- A 90-degree arc GDP is then drawn on the left side of the display above the bars.

- A 90-degree pie slice with pattern as the fill interior style and a fill style index of narrow-spaced 45-degree lines is drawn to the right of the arc.

- Two circles of equal radii and different interiors are drawn to the right of the pie slice.

  The first circle is drawn with an interior style set to PATTERN and a fill style index of 2. The second circle (right-most) is drawn with an interior style set to HATCH and a fill area hatch index of 1.

- A 90-degree elliptical arc GDP is then drawn on the left side of the display above the circular arc.

- A 90-degree elliptical pie slice with pattern as the fill interior style and a fill style index of narrow-spaced 45-degree lines is drawn to the right of the arc.

- Two ellipses of equal radii and different interiors are drawn to the right of the pie slice.

  The first ellipse is drawn with an interior style set to PATTERN and a fill style index of 2. The second ellipse (right-most) is drawn with an interior style set to HATCH and a fill area hatch index of 1.

- The user is prompted to continue.

Frame 8

Frame 8 tests GRAPHIC TEXT ROTATION:

- The display is cleared and the string "ABCabc" is written, then rotated and written again in 45-degree increments.

Note

On devices that do not support character rotation or support only 90-degree rotations, the string is rotated to the devices' ability. With most devices, the rotated string appears longer than the horizontal string.

- The user is prompted to continue.

Frame 9

Frame 9 tests GRAPHIC TEXT SIZE and POSITION:

- The display is cleared and eleven "A" characters are drawn on a horizontal line in increasing sizes and color indices from left to right across the center of the screen.

Note

The first "A" character is not visible because it is drawn in background color.

The number of different sizes and colors shown is limited by the device's capabilities.

- A single "A" character is then drawn on a type-5 polymarker below the line of "A" characters. The lower-left baseline of the character should be at the center of the polymarker. How much of the polymarker is obscured by the "A" depends on the device font.

- The user is then prompted to continue.

Frame 10

Frame 10 tests OUTPUT CELL ARRAY:

- The display is cleared and two cell arrays with the same parameters are drawn, one on top of the other. Each cell array is filled with up to six different colors depending on the device's maximum.

Note

On devices that do not support pixel operation capabilities, the area of the cell array is only outlined.

- The top cell array is then inquired on and, after saving the results, the user is prompted to continue.

Frame 11

Frame 11 tests INQUIRE CELL ARRAY.

- The display is cleared and, if the results of the inquiry of frame 10 resulted in no error and the device has pixel capabilities, the top cell array is redrawn.
- If an error status was returned or the device has no pixel capabilities, nothing is drawn.
- The user is then prompted to continue.

Frame 12

Frame 12 tests LINE WIDTH:

- Horizontal lines of increasing width (to the device's maximum) are drawn, starting from the bottom of the display. The lines increase slightly in length from bottom to top. The narrowest lines is at the bottom and the widest at the top.

Note

On a device that supports only one line width, only one line appears.

- The user is then prompted to continue.

Frame 13

Frame 13 tests GRAPHIC TEXT FONTS.

- For each font supported by the device hardware, a string of 45 characters is written, beginning with a 'space' and ending with a 'tilde' character.

The string is shown below enclosed in quotes:

" !#$%&'()*+,-./0123456789:;<=>?ABC[]^_abc{|}~"

Note

Devices with only one font show only one row of characters.

A device with large characters can cause the string to run off the right edge of the display surface. If this happens, the displayed line is truncated.

- The user is then prompted to continue.

Frames 14-16

Frames 14 through 16 test COLOR REPRESENTATION. These three frames display two colored horizontal lines with the color index written to the left of each line:

- Initially, the screen is cleared and (if the device has more than two indices) the color indices 1 and 2 are redefined.
- The two lines and the associated color index numbers are then drawn.
- In all except the last frame, the realized values are then inquired on and the values returned are used to set the color representation for the next frame.

  All three frames should therefore appear to be the same color.
- The user is then prompted to continue after each of the frames.

Frame 17

Frame 17 tests REQUEST LOCATOR.

- This frame clears the display and requests the user to input six points. The points are selected by moving a graphic cursor around on the display surface with the graphics input device and then making each selection by terminating the input. Input is terminated by pressing an alpha key if the GIN device is a keyboard or by pressing one of the mouse's buttons if the device is a mouse. For every input function, one of the six available cursors is used.

- Inking is turned on for the first point and alternately turned off and on for each of the remaining five points.

- Rubberbanding is cycled for the six points as follows:

  Off for the first and fourth points.
  Rubberband line for the second and fifth points.
  Rubberband rectangle for the third and sixth points.

- After the input of the sixth point, the polygon generated by the first five points is filled.

- The user is then prompted to continue.

Frame 18

Frame 18 tests REQUEST CHOICE.

- The display is cleared and the user is requested to press a function key.

- An inquiry is made to the device and a message is displayed echoing the choice (for example, "key=4" for function key 4).

- The user is then prompted to continue.

Frame 19

Frame 19 tests REQUEST STRING with echo off:

- The display is cleared and the user is requested to enter a string of characters. No characters are displayed as they are typed but the entire string is displayed when the <Return> key is pressed.

Note

The default line edit characters "backspace" (<CTL>**H**) and "kill line" (<CTL>**U**) are valid inputs.

- The user is then prompted to continue.

Frame 20

Frame 20 tests REQUEST STRING with echo on.

- The display is cleared and the user is requested to enter a string of characters. The characters are echoed to the display as the keys are pressed.

  The entire string is echoed again when the <Return> key is pressed.

- The user is then prompted to continue.

The test clears the screen and returns the user to the operating system.

10. Known Bugs and Inconveniences in This Release

The following section lists any known bugs in this release of SCO CGI, along with suggested workarounds. It also describes unsupported features and other inconveniences you might encounter while using SCO CGI.

- Due to SCO CGI's multiple-process nature, running SCO CGI applications on machines with less than 1 Mbyte of memory can provoke system thrashing.

- Command-line input redirection does not work with SCO CGI applications. For example, the following redirection cannot be used:

  cgitest < cgi.automatic

  Developers should make sure their applications documentation explains this limitation.

- Using the SCO CGI CGA medium-resolution color agent on SCO XENIX 386 Operating System release 2.3.1 or 2.3.2 has been observed to cause system reboots on some types of 80386-based machines when using a CGA graphics display card. The problem originates with the operating system kernel, and can be corrected by applying the following **adb** patch to your kernel:

  1. Bring up your system in System Maintenance mode.

  2. Make a backup copy of your kernel by entering the following command:

     cp /xenix /xenix.bak

  3. Invoke **adb**(CP) on the kernel as follows:

     adb -w /xenix

4. **adb** responds with a "*" prompt. Enter the following command:

   vidioctl+0x17e?w 0xf075

   adb responds with a line similar to this:

   vidioctl+0x17e: 0x1075= 0xf075

5. End the **adb** session by entering the command:

   $q

6. Shut down and reboot your system.

This patch must be reapplied to your kernel whenever any change to the kernel is made: for example, installing or removing a device driver; changing a tunable kernel parameter; installing a new link kit; reinstalling your operating system.

Warning

Do not apply this patch to any SCO XENIX Operating System other than release 2.3.1 or 2.3.2. If you are unsure about the release of your operating system, you can use the **uname**(C) command to find out what release your system is running.

- If an SCO CGI application is set-UID, it functions incorrectly when a user other than its owner tries to run it. To prevent this, the application should execute the following command before calling the v_opnwk function:

  setuid(getuid()) ;

- SCO CGI currently does not support switching console multi-screens.

- Abnormal exits from SCO CGI can leave lockfiles in */tmp*. In addition, semaphores and shared memory segments might not be removed when an abnormal exit occurs. See the section "Signals"

earlier in these *Release and Installation Notes*, and Appendix A of the *SCO CGI Programmer's Guide* for more information.

- The device drivers only support choice keys 1-10 (<F1> to <F10>).

- The SCO CGI VGA device driver requires an SCO XENIX Console Driver with VGA support in order to work.

- The VGA agent, *vgabw*, will not function with a monochrome monitor attached to the VGA card.

- The EGA agent, *ega*, will not function with a monochrome monitor attached to the EGA card.

- The *ega* driver is only supported in 256-Kbyte configuration and is not supported on monochrome monitors.

- When running the *ega* driver with a normal color display, the EGA environment variable must either not be set at all, or must be set to HR3 or MR3.

- The *cgaco* driver does not allow you to change colors with VGA and EGA hardware. You must use the *ega* driver.

- The *cgabw* agent produces output only on an EGA or VGA video card, not on a CGA video card.

- With the Hercules driver, setting cursor-text mode or restoring cursor-text mode on exit causes SCO XENIX 2.2.2 through 2.2.5 to return a 25x80 screen. See the section "Hercules Graphics Cards" in these *Release and Installation Notes* for the workaround to this bug.

- The XENIX-Net remote printing facility does not reliably transfer the binary files created by the LaserJet and other drivers. To print graphics output on a remote printer, you must transfer the data by hand to the machine associated with the printer.

- Although multiple SCO CGI applications can be run on multiple monitors on the same system, it is presently impractical to access

more than one pointing device (mouse or bit pad) per XENIX system.

- If a second process controls the CPU on a system with multiple users, the pointing device (mouse or bit pad) does not respond immediately. The more heavily the CPU is loaded, the greater the delay. In some situations this results in unacceptable mouse or bit pad response.

- This release of SCO CGI is not supported on the HP Vectra.

- The cell array functions cannot handle more than 16 Kbytes of pixels.

- The amount of information the byte pixel functions can handle depends on the memory model you use. For the large model 286 program, the functions can handle up to 64 Kbytes of pixels. For the medium model 286 programs, they must share a total of 64 Kbytes with the rest of the program data. For the small model 286 programs, the byte pixel functions must share a total of 64 Kbytes with the rest of the program data and with the program itself.

  In each case the integer pixel functions can handle one-half the amount of information the byte pixel functions can.

- The 286 architecture makes it impossible to save an EGA screen in a single array. For a workaround to this limitation, see the file */usr/lib/cgi/sample/ctest/scrnsaver.c*.

- Information sent by SCO XENIX to a Hewlett-Packard plotter is stored in the plotter's input buffer until it can be processed. If the input buffer fills up, the plotter sends flow control information back to SCO XENIX. Under SCO XENIX 2.3, uninitialized serial ports do not check for flow control, causing input buffer overflow errors to occur in the plotter. This situation can be remedied in one of the following ways:

  Send output from the SCO CGI hpplot agent to the XENIX print spooler **lp**(C). The spooler will handle setting up the serial port for flow control. This requires that your System Administrator set up an **lp**(C) interface script for the plotter.

  or:

  Explicitly turn on flow control when sending output to the plotter. This can be accomplished by using the following **sh**(C) command line, where "file" contains the plotter commands to be sent and */dev/ttyxx* is the name of the serial port to which the plotter is connected:

  $ (stty ixon ixoff; cat file) > /dev/ttyxx 0<&1

  We recommend that the first approach be used.

11. Documentation Errata

This section of the *Release and Installation Notes* contains information that was unavailable when the SCO CGI documentation was completed.

- There is incorrect information in Chapter 3, "Display Devices," in the SCO CGI Device Driver Supplement. In the "Monochrome (MONO)" portion of the section "IBM Enhanced Graphics," it is stated that the IBM Enhanced Graphics Adapter is supported for use with a monochrome monitor, which is incorrect. The EGA agent, *ega*, will not function with a monochrome monitor attached to the EGA card.

- Page 2-1 of the "FORTRAN Reference Manual" shows a command for linking object files. Although this command is shown on two lines, it must be entered on one line on your terminal, or on two lines with the first ending with a backslash (\), as in the following:

  lpild *<fname.o>* *<name.o>* **/usr/lib/386/Pbindcgi.a**
  /usr/lib/cgi/386/Slibccgi.a -o *<exfile>*

- Page 2-1 of the "Pascal Reference Manual" shows a command for linking object files. Although this command is shown on two lines, it must be entered on one line on your terminal, or on two lines with the first ending with a backslash (\), as in the following:

  ldfortran *< fname.o >* *< fname.o >* **/usr/lib/386/Fbindcgi**
  /usr/lib/cgi/386/Slibccgi.a -o *< exfile >*

12. Questions and Answers

Here are some questions and answers about SCO CGI and its documentation.

The answers here are all based on documentation in the *SCO CGI Programming Guide, Device Driver Supplement,* and *Language Guides.* Refer to these documents for complete details on all topics.

Q: What can I use SCO CGI for?

A: SCO CGI is ideal for software developers who want to create original, high-performance XENIX graphics applications, or port existing DOS CGI packages to XENIX. SCO CGI provides the graphics functions, device drivers and language interface that programmers need to create graphics-based software such as CAD, business graphics, and page-composition applications.

Q: What video display adaptors are supported by SCO CGI?

A: SCO CGI release 1.1 supports EGA, CGA (Color and High Resolution Black and White), VGA, and Hercules graphic adaptors.

Q: Why can't I switch multi-screens when I run an SCO CGI application?

A: SCO CGI currently does not support switching console multi-screens.

Q: How does an SCO CGI driver differ from a regular XENIX device driver?

A: XENIX drivers are linked into the kernel and are designed to service multiple users. They usually service hardware which uses interrupts to request service from the CPU. SCO CGI drivers (agents) service only one application at a time, communicating with the hardware through regular I/O channels (in the case of hard-copy drivers), or through the use of service routines provided by the kernel ioctl's for video graphics adaptors.

Q: How can my customer run an SCO CGI application we have developed? Will they need the XENIX Development System?

A: For customers that do not have or do not need the SCO XENIX Development System on their computer there is the SCO CGI Graphics RunTime System for running graphic applications developed with SCO CGI.

SCO also has plans to offer a version of the RunTime System for SCO CGI for software developers who want to package the RunTime directly with their SCO CGI application. Please contact your SCO Sales Representative for more information.

13. Documentation Replacement Pages

The documentation replacement pages listed below are included with these *Release and Installation Notes*. These pages add to or clarify information in your SCO CGI documentation.

To update your SCO CGI documentation, replace the pages in your *SCO CGI Device Driver Supplement* with the corresponding pages supplied with these *Release and Installation Notes*.

The pages included here are:

3-19, 3-20
3-21, 3-22
3-23, 3-24
3-25, 3-26
3-27, 3-28
3-29, 3-30
3-31, (blank)

Request Locator

When locator is invoked, a graphics input cursor appears on the screen at the initial locator position. The cursor can be moved by pressing one of the following keys on the numeric keypad: **1** (down & left), **2** (down), **3** (down & right), **4** (left), **6** (right), **7** (up & left), **8** (up), and **9** (up & right). The Numeric Lock function must be off for the cursor to be moved. Initially, the cursor moves in large increments. Pressing the **5** (INS) key toggles the distance between large movements and small movements. When the cross is at the desired location, the point can be selected by pressing any alpha key on the keyboard.

Hard Copy

Hard Copy is not supported under XENIX.

Request Choice

This driver supports the following function key values:

Function	Value
F1-F10	1-10

Cursor—Addressable Text

In addition to the common features, this device supports the following attributes:

Color = Expanded palette

Color Index	Standard	Bold
0	Black	Dark Grey
1	Light Grey	White
2	Red	Light Red
3	Green	Light Green
4	Blue	Light Blue
5	Brown	Yellow
6	Cyan	Light Cyan
7	Magenta	Light Magenta

3.5 IBM Video Graphics Array — 2 Color Mode

Filename

vgabw

Device Logical Name

Display

Default Resolution and Aspect Ratio

The Horizontal and Vertical dpi are used for selection of raster fonts.

Resolution	Aspect Ratio	Hor. dpi	Vert. dpi
640x480	1:1	75	75

Communications

not applicable

Features Supported

Environmental Settings

None available.

Color

There are two colors available. Color index 0 is the background color, and color index 1 is the foreground color. Either can be chosen from the palette, which has a total of 262,144 colors. This represents 64 distinct values for each of the RGB levels.

Request Locator

When locator is invoked, a graphics input cursor appears on the screen at the initial locator position. The cursor can be moved by pressing any of the following keys on the numeric keypad: **1** (down & left), **2** (down), **3** (down & right), **4** (left), **6** (right), **7** (up & left), **8** (up), **9** (up & right). The Numeric Lock function must be off for the cursor to be moved. Initially, the cursor moves in large increments. The **5** (INS) key toggles between large and small

movements. When the cross is at the desired location, the point can be selected by pressing any alpha key on the keyboard.

Hard Copy

Hard Copy is not supported under XENIX.

Request Choice

This driver supports the following function key values:

Function	Value
F1-F10	1-10

Cursor-Addressable Text

In addition to the common features, this device supports the following attributes:

Reverse Video
Blink
Bold Intensity
Color: the available colors are shown below

Color Index	Standard	Bold
0	Black	Dark Grey
1	Light	White
2	Red	Light Red
3	Green	Light Green
4	Blue	Light Blue
5	Brown	Yellow
6	Cyan	Light Cyan
7	Magenta	Light Magenta

Hardware Text

The VGA device driver supports 3 hardware text sizes: 24 lines, 34 lines, and 43 lines. Graphics text font 1 is 24 lines, graphics text font 2 is 34 lines, and graphics text font 3 is 43 lines.

In Alpha Text, the 34 line font is Alpha Text font 7, and the 43 line font is Alpha Text font 8.

3.6 IBM Video Graphics Array — 16 Color Mode

Filename

vga16

Device Logical Name

Display

Default Resolution and Aspect Ratio

The Horizontal and Vertical dpi are used for selection of raster fonts.

Resolution	Aspect Ratio	Hor. dpi	Vert. dpi
640x480	1:1	75	75

Communications

not applicable

Specific Features Supported

Environmental Settings

None available.

Color

This device supports sixteen color indexes, each of which may be defined to any color displayable on the attached monitor. Any of the sixteen color indexes can be chosen from the palette, which has a total of 262,144 colors. This represents 64 distinct values for each of the RGB levels.

The RGB values for the default colors follow:

Color Display

R	G	B	Color	Index
0	0	0	Black	0
1000	1000	1000	White	1
777	0	0	Red	2
0	777	0	Green	3
0	0	777	Blue	4
1000	1000	0	Yellow	5
0	603	603	Cyan	6
603	0	603	Magenta	7
301	301	301	Dark Grey	8
603	603	603	Light Grey	9
1000	0	0	Light Red	10
0	1000	0	Light Green	11
0	0	1000	Light Blue	12
777	603	0	Brown	13
0	1000	1000	Light Cyan	14
1000	0	1000	Light Magenta	15

These colors can be redefined to any RGB value from 0 to 1000. The SET COLOR REPRESENTATION function will set only an individual color index. The entire color table may be set with the SET COLOR TABLE function.

Request Locator

When locator is invoked, a graphics input cursor appears on the screen at the initial locator position. The cursor can be moved by pressing any of the following keys on the numeric keypad: **1** (down & left), **2** (down), **3** (down & right), **4** (left), **6** (right), **7** (up & left), **8** (up), **9** (up & right). The Numeric Lock function must be off for the cursor to be moved. Initially, the cursor moves in large increments. The **5** (INS) key toggles between large and small movements. When the cursor is at the desired location, the point can be selected by pressing any alpha key on the keyboard.

Hard Copy

Hard Copy is not supported under XENIX.

Request Choice

This driver supports the following function key values:

Function	Value
F1-F10	1-10

Cursor-Addressable Text

In addition to the common features, this device supports the following attributes:

Reverse Video
Blink
Bold Intensity
Color: the available colors are shown below

Color Index	Standard	Bold
0	Black	Dark Grey
1	Light Grey	White
2	Red	Light Red
3	Green	Light Green
4	Blue	Light Blue
5	Brown	Yellow
6	Cyan	Light Cyan
7	Magenta	Light Magenta

Hardware Text

The VGA device driver supports 3 hardware text sizes: 24 lines, 34 lines, and 43 lines. Graphics text font 1 is 24 lines, graphics text font 2 is 34 lines, and graphics text font 3 is 43 lines.

In Alpha Text, the 34 line font is Alpha Text font 7, and the 43 line font is Alpha Text font 8.

3.7 IBM Video Graphics Array — 256 Color Mode

Filename

vga256

Device Logical Name

Display

Default Resolution and Aspect Ratio

The Horizontal and Vertical dpi are used for selection of raster fonts.

Resolution	Aspect Ratio	Hor. dpi	Vert. dpi
320x200	1.18:1	49	42

Communications

not applicable

Specific Features Supported

Environmental Settings

None available.

Color

This device supports 256 color indexes, each of which may be defined to any color displayable on the attached monitor. Any of the 256 color indexes can be chosen from the palette, which has a total of 262,144 colors. This represents 64 distinct values for each of the RGB levels.

The RGB values for the first sixteen default colors follow:

Color Display

R	G	B	Color	Index
0	0	0	Black	0
1000	1000	1000	White	1
777	0	0	Red	2
0	777	0	Green	3
0	0	777	Blue	4
1000	1000	0	Yellow	5
0	603	603	Cyan	6
603	0	603	Magenta	7
301	301	301	Dark Grey	8
603	603	603	Light Grey	9
1000	0	0	Light Red	10
0	1000	0	Light Green	11
0	0	1000	Light Blue	12
777	603	0	Brown	13
0	1000	1000	Light Cyan	14
1000	0	1000	Light Magenta	15

These colors can be redefined to any RGB value from 0 to 1000. The SET COLOR REPRESENTATION function will set only an individual color index. The entire color table may be set with the SET COLOR TABLE function.

Request Locator

When request locator is invoked, a graphics input cursor appears on the screen at the initial locator position. The cursor can be moved by pressing any of the following keys on the numeric keypad: **1** (down & left), **2** (down), **3** (down & right), **4** (left), **6** (right), **7** (up & left), **8** (up), **9** (up & right). The Numeric Lock function must be off for the cursor to be moved. Initially, the cursor moves in large increments. The **5** (INS) key toggles between large and small movements. When the cursor is at the desired location, the point can be selected by pressing any alpha key on the keyboard.

Hard Copy

Hard Copy is not supported under XENIX.

Request Choice

This driver supports the following function key values:

Function	Value
F1-F10	1-10

Cursor-Addressable Text

In addition to the common features, this device supports the following attributes:

Reverse Video
Blink
Bold Intensity
Color: the available colors are shown below

Color Index	Standard	Bold
0	Black	Dark Grey
1	Light Grey	White
2	Red	Light Red
3	Green	Light Green
4	Blue	Light Blue
5	Brown	Yellow
6	Cyan	Light Cyan
7	Magenta	Light Magenta

3.8 Hercules Graphics Card — High Resolution Monochrome
Hercules Graphics Card Plus — High Resolution Monochrome

Filename

hercules

Device Logical Name

Display

Communications

not applicable

Features Supported

Color

The Hercules supports two colors. Index 1 is displayed in the foreground color (white), and index 0 is displayed in the background color (black). These colors cannot be redefined.

Request Locator

When locator is invoked, a tracking cross appears on the screen at the initial locator position. The cross can be moved by pressing one of the following eight keys on the numeric keypad: **1** (down & left), **2** (down), **3** (down & right), **4** (left), **6** (right), **7** (up & left), **8** (up), and **9** (up & right). The Numeric Lock function must be off for the cross to be moved. Initially, the cross moves in large increments. Pressing the **5** (INS) key toggles the distance between large movements and small movements. When the cross is at the desired location, the point can be selected by pressing any alpha key on the keyboard. This causes the coordinates of the point to be transmitted back to the user program. If desired, the device will perform an inking function. When the locator is terminated, a line from the initial position to the desired position is drawn, honoring the current line attributes such as color and line style.

The device also performs rubberbanding if desired. There are two types of rubberbanding supported, lines and boxes. If rubberbanding lines are desired, then a line will be drawn from the initial locator position to the current position of the graphics cursor. The line changes dynamically as the cursor is moved. When the locator is terminated, the line is removed. If rubberband rectangle is specified, a rectangle is displayed with one corner at the initial locator position and the opposite corner at the current position of the graphics cursor. The rectangle changes dynamically as the cursor is moved. When the locator is terminated, the rectangle is removed from the display.

Hard Copy

Hard Copy is not supported under XENIX.

Request Choice

The function keys **F1** to **F10** are used to enter choice input.

Request String

The keyboard is used to enter strings. A string is terminated by the ENTER key.

Cursor—Addressable Text

Cursor-addressable text is supported. The device must be in Cursor-Addressing Mode before it can perform any cursor control functions. The following attributes are supported:

Reverse Video
Underline
Blink
Bold Intensity

To display graphics primitives, the device must be removed from Cursor-Addressing Mode.

Graphics Markers

Markers have five sizes as listed below in VDC coordinates:

Preserve Aspect Ratio Mode

1	458
2	851
3	1243
4	1636
5	2028

Non-Preserve Aspect Ratio Mode

1	660
2	1225
3	1790
4	2355
5	2919

Additional Information

Upon termination, the driver returns the display to the initial operating mode.

3-17-89
009-030-000