MemTool

Installation and Operation

THE NETWORK IS THE COMPUTER™

Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043 USA
415 960-1300 fax 415 969-9131

Part No.: 8xx-xxxx-xx
Revision X, July 1997

Please
Recycle

Adobe PostScript

Contents

Installation 1

The Memtool package may be obtained from your local Sun SE, and is at the time of writing available for Solaris 2.5, 2.5.1 and Solaris 2.6.

Installation requires loading a kernel module,and it is recommended that you excercise care when installing, as not to disrupt production workloads.

Adding the Memtool Package

The package is provided in compress tar format, and uses the standard Solaris pkgadd command.

```
# zcat RMCmem3.5b.tar.Z |tar xvf -
x RMCmem, 0 bytes, 0 tape blocks
x RMCnen/pkgmap, 2343 bytes, 5 tape blocks
.
.
# pkgadd -d .

The following packages are available:
  1  RMCmem     The MemTool Package
                (sparc) 3.5b

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1

Do you want the Memtool module loaded now? y
```

The Memtool package uses a loadable kernel module. If you would like this module loaded now so that you may use the tools before reboot, then answer yes to this question. To just install the software without impacting the system, answer no.

Please be aware that loading modules onto a running system should be done without production users on the system.

```
Do you want to install the module loader in /etc/rc2.d? y
```

The loadable kernel module can optionally be loaded at boot time. This is the normal method of installing Memtool. Answer yes to have a script placed in /etc/rc2.d to load the module at boot. If you don't want the module to be loaded at reboot you can answer no.

```
.Do you want the kernel module loaded now? y

Do you want to install the module loader in /etc/rc2.d? y

Using </opt> as the package base directory.
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of <RMCmem> [y,n,?] y

Installing The MemTool Package as <RMCmem>

## Executing preinstall script.
## Installing part 1 of 1.
/etc/init.d/memtool
/opt/RMCmem/README
.
## Executing postinstall script.

Installation of <RMCmem> was successful.
```

Manually Loading the kernel Module

If you answered no to loading the kernel loadable module, or have rebooted and the module was not loaded at boot then you will need to manually load the module.

There is a script to do this in the Memtool base directory, /opt/RMCmem/drv.

```
# cd /opt/RMCmem/drv

# ./bunyipload

# modinfo |grep bunyip

110 6167c000   5162   -   1  bunyipmod.56 (VFS and Proc Memory Stats Module)
```

1

MemTool Overview 2

The basis for the operation of MemTool is a loadable kernel module which uses the /proc interface to look at the memory allocation of processes and the UFS buffer cache.

Obtaining a summary of the systems memory

A summary of the systems memory can be reported with the prtmem command.

```
# prtmem
Total Physical Memory:    384 Megabytes
Buffer Cache Memory:      112 Megabytes
Kernel Memory:            63 Megabytes
Free Memory:              17 Megabytes
```

Process memory usage and the pmem command

Traditionally, the only information about process memory utilisation was the virtual memory size and RSS figure available from the *ps* command and *top*.

The virtual address size of a process often bares no resemblance to the amount of memory a process is using because it contains all of the unallocated memory, libraries, shared memory and sometimes hardware devices (in the case of Xsun).

The RSS figure is a measure of the amount of physical memory mapped into a process, but often there is more than one copy of the process running, and a large proportion of a process is shared with another.

The MemTool tools provide a mechanism for getting a detailed look at a processes memory utilisation. MemTool can show how much memory is in-core, how much of that is shared, and hence how much private memory a process has.

The *pmem* command (or */usr/proc/bin/pmap -x* in Solaris 2.6) can be used to show the memory utilisation of a single process.

```
# pmem 25888
 or
# /usr/proc/bin/pmap -x 25888

25888:  ksh

Addr       Size    Res Shared   Priv Prot               Segment-Name
-------- ------ ------  ----- ------ -----------------  ----------------------------
00010000   184K   184k   184k     0k read/exec          /bin/ksh
0004C000     8K     8k     8k     0k read/write/exec    /bin/ksh
0004E000    40K    40k     0k    40k read/write/exec    [ heap ]
EF5E0000    16K    16k     8k     8k read/exec          /usr/lib/locale/en_AU.so.1
EF5F2000     8K     8k     0k     8k read/write/exec    /usr/lib/locale/en_AU.so.1
EF600000   592K   568k   560k     8k read/exec          /usr/lib/libc.so.1
EF6A2000    24K    24k     8k    16k read/write/exec    /usr/lib/libc.so.1
EF6A8000     8K     8k     0k     8k read/write/exec
EF6B0000     8K     0k     0k     0k read/write/exec
EF6C0000    16K    16k    16k     0k read/exec          /usr/lib/libc_psr.so.1
EF6D0000    16K    16k    16k     0k read/exec          /usr/lib/libmp.so.2
EF6E2000     8K     8k     8k     0k read/write/exec    /usr/lib/libmp.so.2
EF700000   448K   400k   400k     0k read/exec          /usr/lib/libnsl.so.1
EF77E000    32K    32k     8k    24k read/write/exec    /usr/lib/libnsl.so.1
EF786000    24K     8k     0k     8k read/write/exec
EF790000    32K    32k    32k     0k read/exec          /usr/lib/libsocket.so.1
EF7A6000     8K     8k     8k     0k read/write/exec    /usr/lib/libsocket.so.1
EF7A8000     8K     0k     0k     0k read/write/exec
EF7B0000     8K     8k     8k     0k read/exec/shared   /usr/lib/libdl.so.1
EF7C0000   112K   112k   112k     0k read/exec          /usr/lib/ld.so.1
EF7EA000    16K    16k     8k     8k read/write/exec    /usr/lib/ld.so.1
EFFFC000    16K    16k     0k    16k read/write/exec
EFFFC000    16K                                         [ stack ]
-------- ------ ------  ----- ------
          1632K  1528k  1384k   144k
```

The example output from pmem shows the memory map of the /bin/ksh command. At the top of the output is the executable text and data segments. All of the executable binary is shared with other processes because it is mapped read only into each process. A small portion of the data segment is shared, whilst some is private because of copy-on-write operations (COW).

The next segment in the address space is the heap space, or user application data. This segment is typically 100% private to a process.

Following the heap space is the shared libraries. Each shared library has a text, and data segment, which are partially shared.

At the bottom of the process dump is the stack, which like the heap is 100% private.

A summary of the total Virtual size, resident portion and private memory are printed at the bottom.

Buffer cache memory

Traditionally there has been no method of showing where the pool of buffer cache memory has been allocated. MemTool makes this possible by providing a list of all of the VNODE's in the buffer cache.

The list summarises the size of each VNODE in the buffer cache, and where possible the real filename. If the real filename cannot be determined, then the device and inode number are printed for that VNODE.

The MemTool kernel module collects filenames as each file is opened or referenced. If the kernel module has recently been loaded, then not all of the filenames will be available. The best way to cure this is to use the /etc/rc2.d script to load the bunyip module at boot, which will capture the first 8192 filenames referenced.

If you have a system with many files, you might like to put the following statement into /etc/system. Note that this uses extra kernel memory.

```
set bunyipmod:vfsname_maxitems = 32768
```

The list of VNODE's in the UFS buffer cache can be displayed with the memps command, and with the MemTool GUI.

```
# memps -m |more

SunOS devnull 5.6 SunOS_Development sun4u    07/21/97

11:27:03
   Size  Filename
 12152k  /export/home/webarchives/mail/network-engrs
 10680k  /export/home/webarchives/mail/sun-managers
  8032k  /2b40001:  370743
  6576k  /15c0007:  709619
  5152k  /export/home/webarchives/mail/unigram
  5056k  /export/home/webarchives/index/work/JAVAINTERNAL.dct
  3744k  /15c0008:  166191
  3288k  /usr/dt/lib/libXm.so.3
  2456k  /15c0007:  709592
  2376k  /export/home/webarchives/mail/firewall
  2272k  /15c0007:  586146
  2264k  /15c0008:  196636
  2016k  /800078:    5970
  1912k  /usr/openwin/lib/libxview.so.3
  1744k  /export/home/webarchives/mail/javaace
  1720k  /15c0007:  709594
  1696k  /15c0007:  132642
  1504k  /2b40001: 1206281
  1504k  /800078:  106190
  1496k  /2b40001: 1204243
  1448k  /15c0007:  709611
  1392k  /export/home/webarchives/index/work/JAVAINTERNAL.inv
  1264k  /usr/lib/libc.so.1
  1256k  /80007b:  182313
  1200k  /15c0007:  132666
  1096k  /800078:  100213
  1096k  /usr/openwin/lib/libX11.so.4
  1088k  /15c0007:  586141
  1080k  /usr/openwin/lib/libtt.so.2
  1072k  /15c0007:  709632
  1056k  /15c0007:    8844
  1032k  /2b40001:  929861
  1000k  /800078:  200260
   952k  /export/local/bin/perl
   880k  /usr/dt/lib/libDtSvc.so.1
   880k  /15c0007:  709610
   856k  /6167c1ac:       0
   856k  /usr/openwin/lib/libXt.so.4
   800k  /15c0008:    7231
   752k  /80007b:  113922
   720k  /800078:   82526
   .
   .
   .
```

The MemTool GUI 3

The MemTool GUI interface provides an easy method of invoking most of the functionallity of the kernel interfaces.

Invoke the GUI as the root user to see all of the process and file information.

```
# /opt/RMCmem/bin/memtool &
```

There are three basic modes on the MemTool GUI, Buffer cache memory, Process memory, and a Process/Buffer cache mapping matrix.

Buffer Cache Memory

The initial screen shows the contents of the Buffer Cache memory.

The Buffer Cache Memory display shows each entry in the UFS Buffer cache. The fields shows are as follows:-

Table 3-1 Buffer Cache Fields

Field	Description
Resident	The amount of physical memory that this file has associated with it.
Used	The amount of physical memory that this file has mapped into a process segment or SEGMAP. Generally the difference between this and the resident figure is what is on the free list associated with this file.
Shared	The amount of memory that this file has in memory that is shared with more than one process
Pageins	The amount of minor and major pagein's for this file
Pageouts	The amount of pageouts for this file
Filename	The filename of the VNODE or if not known the device and inode number in the format 0x0000123:456

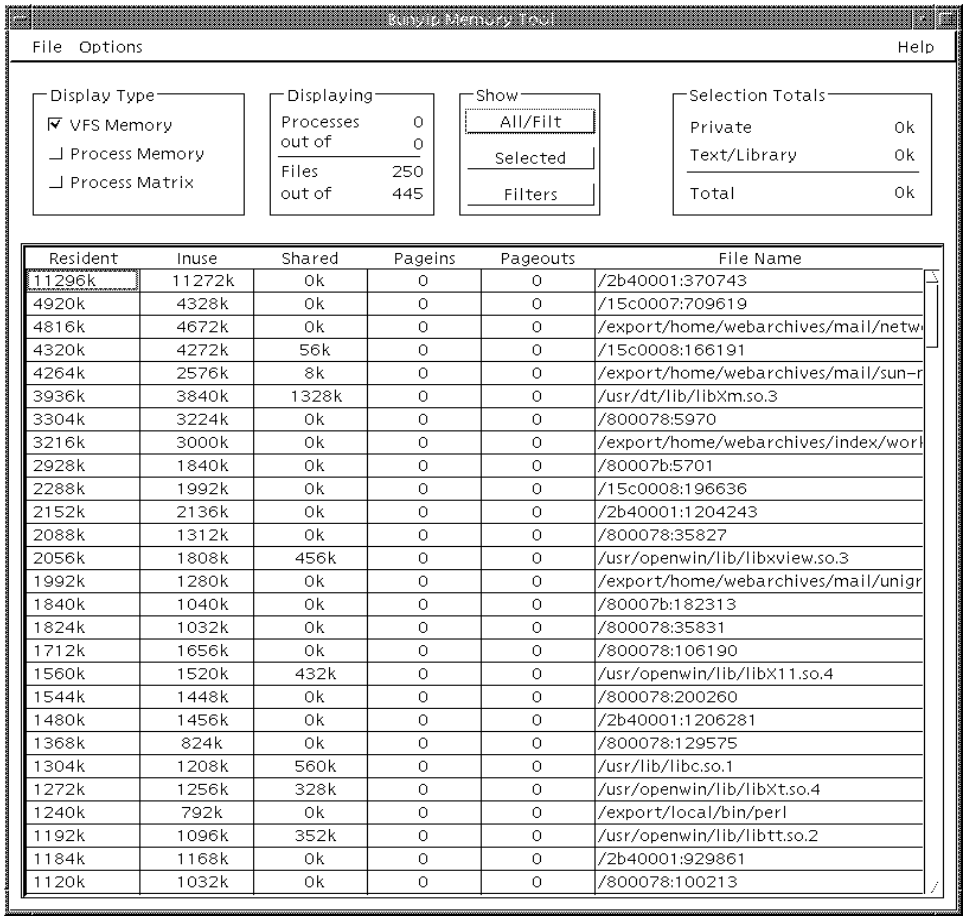


Figure 3-2 MemTool GUI - Buffer Cache Memory

The GUI will only display the largest 250 files. A status panel at the top of the display shows the total amount of files and the number that have been displayed.

3

Process Memory

The second mode of the MemTool GUI is the process memory display. Click on the “Process Memory” checkbox at the left of the GUI to select this mode.

The process memory display shows the process table with a memory summary for each process. Each line of the process table is the same as the per-process summary from the *pmem* command.

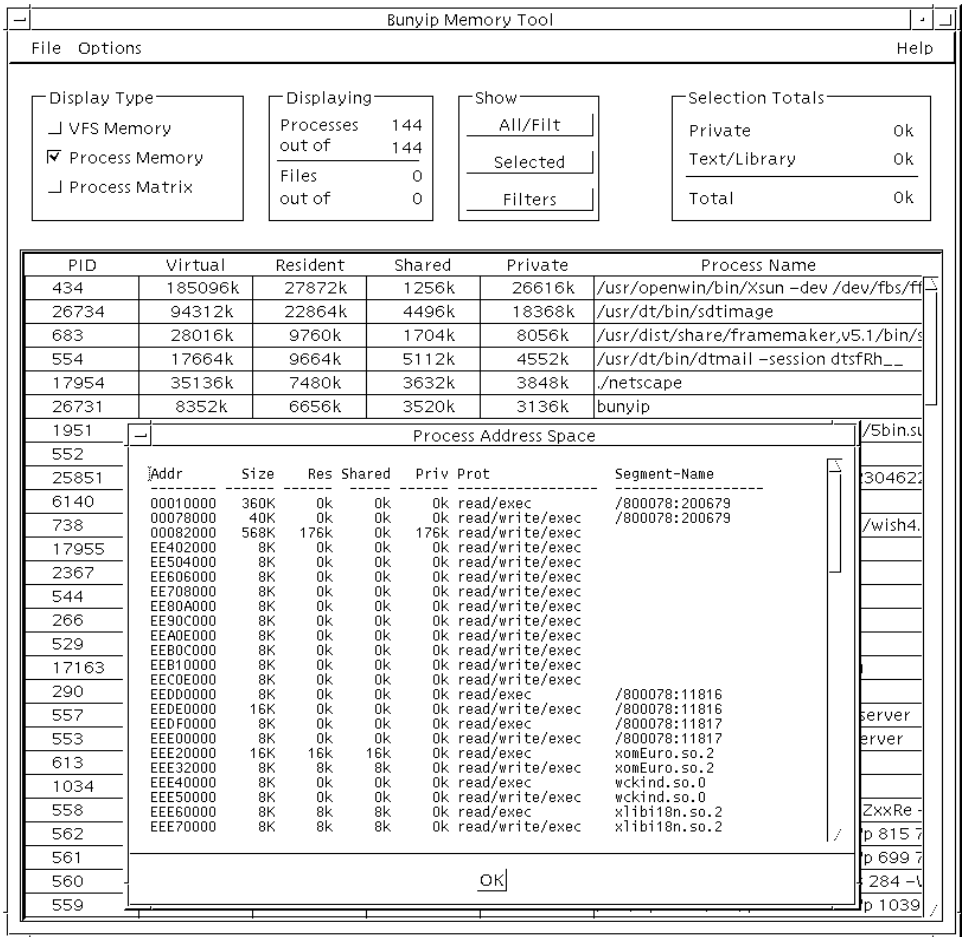


Figure 3-3 MemTool GUI - Process Memory

The fields for the Process Memory display are as follows:-

Table 3-4 Process Memory Fields

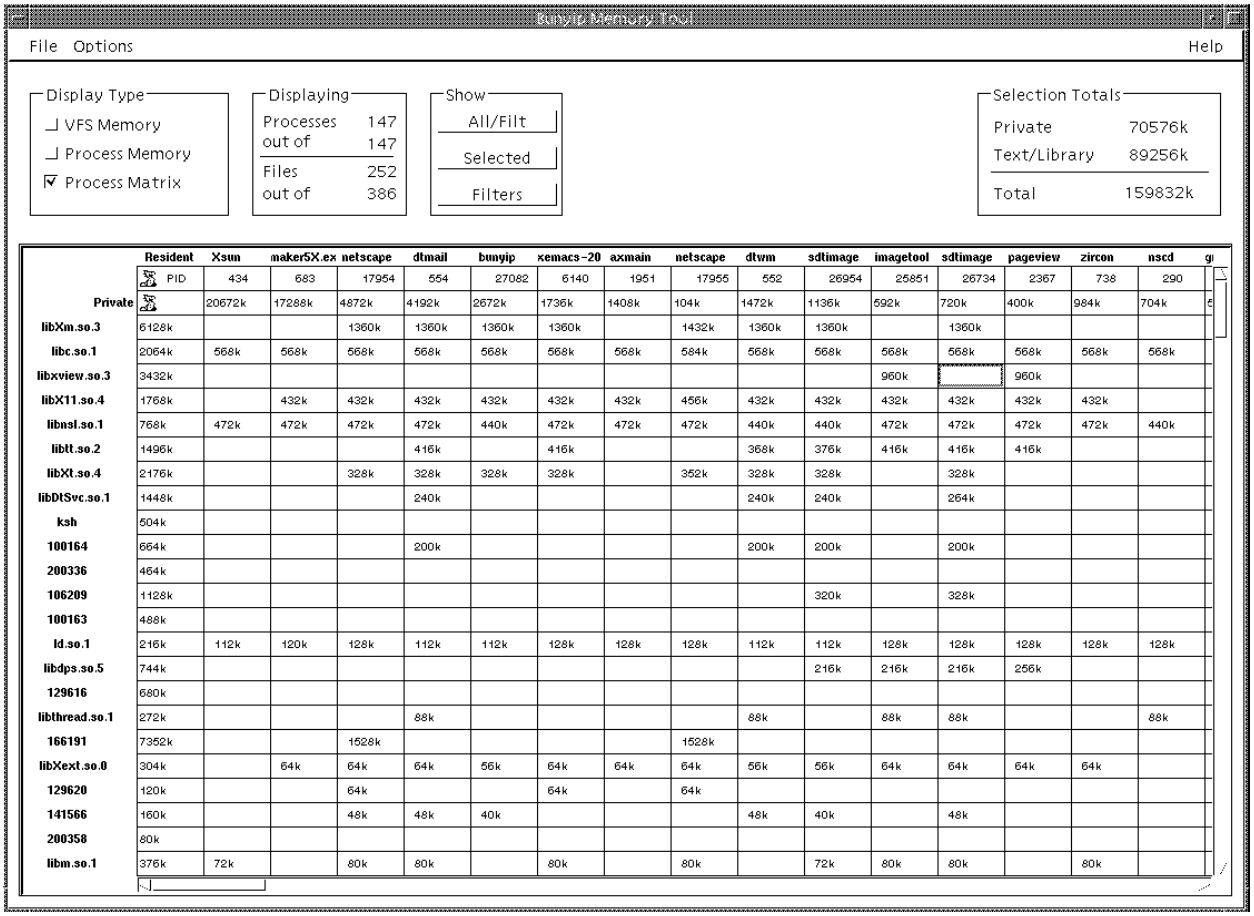
Field	Description
PID	Process ID of process
Virtual	The virtual size of the process, including swapped out and unallocated memory
Resident	The amount of physical memory that this process has, including shared binaries, libraries etc
Shared	The amount of memory that this process is sharing with another process, ie shared libraries, shared memory etc.
Private	The amount of resident memory that this process has which is not shared with other processes. This figure is essentially Resident - Shared and does not include the application binaries.
Process	The full process name and arguments

The individual process map for a process can be selected by clicking on one of the process entries.

Process Matrix

The process matrix shows the relationship between processes and mapped files. Across the top of the display is the list of processes that we viewed in the process memory display, and down the side is a list of the files which are mapped into these processes.

Each column of the matrix shows the amount of memory mapped into that process for each file, with an extra row for the private memory associated with that process.



	Resident	Xsun	maker5X.ex	netscape	dtmail	bunyip	xemacs-20	axmain	netscape	dtwm	sdtimage	imagetool	sdtimage	pageview	zircon	nscd	g
	PID	434	683	17954	554	27082	6140	1951	17955	552	26954	25851	26734	2367	738	290	
Private		20672k	17288k	4872k	4192k	2672k	1736k	1408k	104k	1472k	1136k	592k	720k	400k	984k	704k	5
libXm.so.3	6128k			1360k	1360k	1360k	1360k		1432k	1360k	1360k		1360k				
libc.so.1	2064k	568k	568k	568k	568k	568k	568k	568k	584k	568k	568k	568k	568k	568k	568k	568k	
libxview.so.3	3432k											960k		960k			
libX11.so.4	1768k		432k	432k	432k	432k	432k	432k	456k	432k	432k	432k	432k	432k	432k		
libnsl.so.1	768k	472k	472k	472k	472k	440k	472k	472k	472k	440k	440k	472k	472k	472k	472k	440k	
libtt.so.2	1496k				416k		416k			368k	376k	416k	416k	416k			
libXt.so.4	2176k			328k	328k	328k	328k		352k	328k	328k		328k				
libDtSvc.so.1	1448k				240k					240k	240k		264k				
ksh	504k																
100164	664k				200k					200k	200k		200k				
200336	464k																
106209	1128k										320k		328k				
100163	488k																
ld.so.1	216k	112k	120k	128k	112k	112k	128k	128k	128k	112k	112k	128k	128k	128k	128k	128k	
libdps.so.5	744k										216k	216k	216k	256k			
129616	680k																
libthread.so.1	272k				88k					88k		88k	88k			88k	
166191	7352k			1528k					1528k								
libXext.so.0	304k		64k	64k	64k	56k	64k	64k	64k	56k	56k	64k	64k	64k	64k		
129620	120k			64k			64k		64k								
141566	160k			48k	48k	40k				48k	40k		48k				
200358	80k																
libm.so.1	376k	72k		80k	80k		80k		80k		72k	80k	80k		80k		

Figure 3-5 MemTool GUI - Process / File Matrix

The matrix can be used to show the total memory usage of a group of processes. By default, the summary box at the top right hand corner shows the memory used by all of the processses displayed.

A group of processes can be selected with the left mouse button, and then summarised by hitting the *selection* button at the top-middle of the display. The full display can be returned by selecting the *all/filt* button.

GUI Options

There are also some options to configure the order of the rows of files or processes displayed. By default, they are sorted in reverse memory size order. The Options menu can be used to select the sort options dialog.

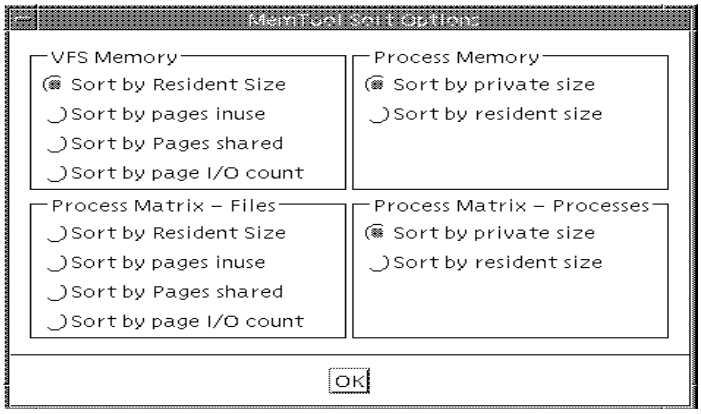


Figure 3-6 MemTool GUI - Sort Options

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
FAX 415 969-9131

For U.S. Sales Office locations, call:
800 821-4643
In California:
800 821-4642

Australia: (02) 413 2666
Belgium: 32 2 716 7911
Canada: 416 477-6745
Finland: 358-0-502 27 00
France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0
Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 221-7021
Korea: 822-563-8700
Latin America: 415 688-9464
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-831-5568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (1) 825 71 11
Taiwan: 2-514-0567
UK: 0276 20444

Elsewhere in the world,
call Corporate Headquarters:
415 960-1300
Intercontinental Sales: 415 688-9000