Unit OS5: Memory Management

5.5. Lab Manual

Windows Operating System Internals - by David A. Solomon and Mark E. Russinovich with Andreas Polze

Copyright Notice

Roadmap for Section 5.5.

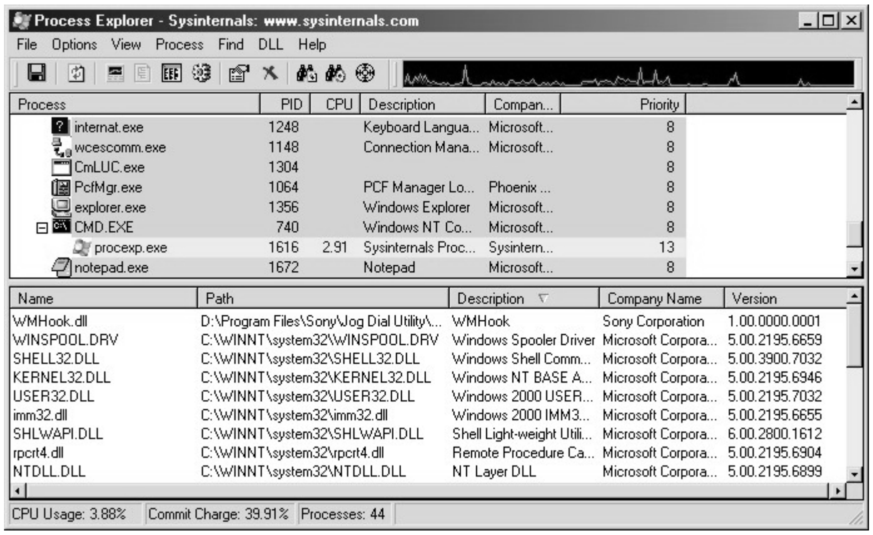
- Dynamic Link Library (DLL) Usage
- Viewing the Working Set
- Inspecting the Page Frame Number Database
- Perfmon and memory-related counters
- Monitoring page file consumption

3

This LabManual includes experiments investigating the algorithms for memory management implemented inside the Windows operating system. Students are expected to carry out Labs in addition to studying the learning materials in Unit OS5.

A thorough understanding of the concepts presented in Unit OS5: Memory Management is a prerequisite for these Labs.

Viewing DLLs and Memory Mapped Files



Process Explorer - Sysinternals: www.sysinternals.com

Process	PID	CPU	Description	Compan...	Priority
internat.exe	1248		Keyboard Langua...	Microsoft...	8
wcescomm.exe	1148		Connection Mana...	Microsoft...	8
CmLUC.exe	1304				8
PcfMgr.exe	1064		PCF Manager Lo...	Phoenix ...	8
explorer.exe	1356		Windows Explorer	Microsoft...	8
CMD.EXE	740		Windows NT Co...	Microsoft...	8
procexp.exe	1616	2.91	Sysinternals Proc...	Sysintern...	13
notepad.exe	1672		Notepad	Microsoft...	8

Name	Path	Description	Company Name	Version
WMHook.dll	D:\Program Files\Sony\Jog Dial Utility\...	WMHook	Sony Corporation	1.00.0000.0001
WINSPOOL.DRV	C:\WINNT\system32\WINSPOOL.DRV	Windows Spooler Driver	Microsoft Corpora...	5.00.2195.6659
SHELL32.DLL	C:\WINNT\system32\SHELL32.DLL	Windows Shell Comm...	Microsoft Corpora...	5.00.3900.7032
KERNEL32.DLL	C:\WINNT\system32\KERNEL32.DLL	Windows NT BASE A...	Microsoft Corpora...	5.00.2195.6946
USER32.DLL	C:\WINNT\system32\USER32.DLL	Windows 2000 USER...	Microsoft Corpora...	5.00.2195.7032
imm32.dll	C:\WINNT\system32\imm32.dll	Windows 2000 IMM3...	Microsoft Corpora...	5.00.2195.6655
SHLWAPI.DLL	C:\WINNT\system32\SHLWAPI.DLL	Shell Light-weight Utili...	Microsoft Corpora...	6.00.2800.1612
rpcrt4.dll	C:\WINNT\system32\rpcrt4.dll	Remote Procedure Ca...	Microsoft Corpora...	5.00.2195.6904
NTDLL.DLL	C:\WINNT\system32\NTDLL.DLL	NT Layer DLL	Microsoft Corpora...	5.00.2195.6899

CPU Usage: 3.88% Commit Charge: 39.91% Processes: 44

4

Experiment: Viewing DLLs and Memory Mapped Files

Click on View->DLL View

- Shows more than just loaded DLLs
- Includes .EXE and any “memory mapped files”

Uses:

- Detect DLL versioning problems
  - Compare the output from a working process with that of a failing one (use File->Save As)
- Find which processes are using a specific DLL (search for it)

Show Relocated DLLs option

- Highlights relocated DLLs in yellow

Process Explorer DLL Lab1: Run Word and Excel

1. In ProcExp, switch to DLL view
2. Look at the DLL list for both Word and Excel and find a common Office DLL loaded in both processes
   - Hint: sort by path
3. Try and delete that DLL with Explorer
   - Should get access denied error (not file locked)
4. In ProcExp, use search to confirm who has this DLL loaded
   - Should show up in both processes

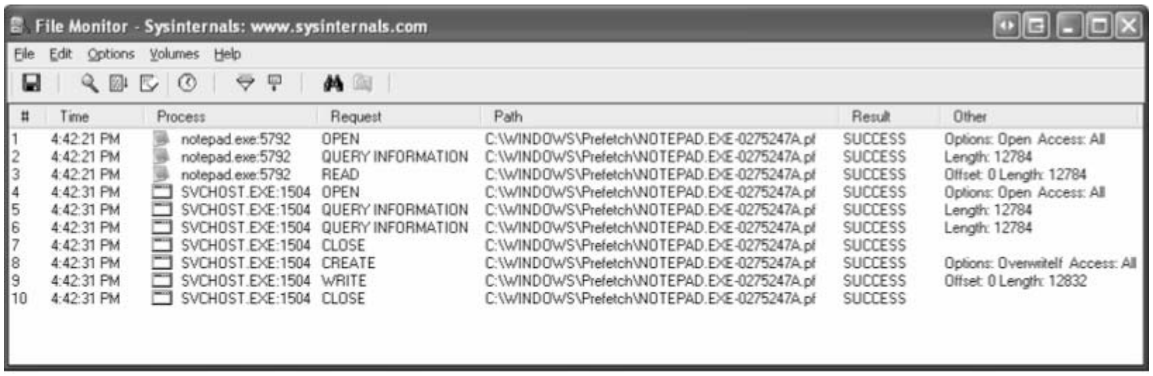
Prefetch Lab

- Lab
  - Run Filemon – set filter as Notepad.exe
  - Make a temporary directory somewhere (e.g. \temp)
  - Run “Notepad \temp\x.y”
  - Exit Notepad
  - Run Notepad again
  - In Filemon log, find creation of .PF file after first run, then use of new .PF in 2nd run

5

Lab objective: Watching Prefetch File Reads and Writes

If you capture a trace of application startup with Filemon from www.sysinternals.com in Windows XP, you can see the prefetcher check for and read the application’s prefetch file (if it exists), and roughly ten seconds after the application started, see the prefetcher write out a new copy of the file. Below is a capture of Notepad startup with an Include filter set to “prefetch” so that Filemon shows only accesses to the \Windows\Prefetch directory.

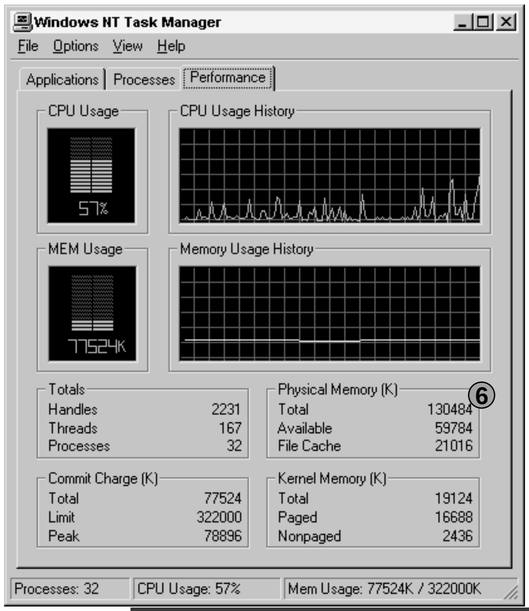

File Monitor - Sysinternals: www.sysinternals.com
File Edit Options Volumes Help

```
#   Time        Process              Request             Path                                         Result   Other
1   4:42:21 PM  notepad.exe:5792     OPEN                C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Options: Open Access: All
2   4:42:21 PM  notepad.exe:5792     QUERY INFORMATION   C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Length: 12784
3   4:42:21 PM  notepad.exe:5792     READ                C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Offset: 0 Length: 12784
4   4:42:31 PM  SVCHOST.EXE:1504     OPEN                C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Options: Open Access: All
5   4:42:31 PM  SVCHOST.EXE:1504     QUERY INFORMATION   C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Length: 12784
6   4:42:31 PM  SVCHOST.EXE:1504     QUERY INFORMATION   C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Length: 12784
7   4:42:31 PM  SVCHOST.EXE:1504     CLOSE               C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS
8   4:42:31 PM  SVCHOST.EXE:1504     CREATE              C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Options: OverwriteIf Access: All
9   4:42:31 PM  SVCHOST.EXE:1504     WRITE               C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS  Offset: 0 Length: 12832
10  4:42:31 PM  SVCHOST.EXE:1504     CLOSE               C:\WINDOWS\Prefetch\NOTEPAD.EXE-0275247A.pf  SUCCESS
```

Memory Management Information

Task Manager Performance tab

- ⑥ “Available” = sum of free, standby, and zero page lists (physical)
- Majority are likely standby pages
- Windows 2000/XP/Server 2003: count of shareable pages on standby, modified, and modified nowrite list are included in what was “File Cache” in NT4
  - New name is “System Cache”



Totals
Handles 2231
Threads 167
Processes 32

Physical Memory (K)
Total 130484
Available 59784
File Cache 21016

Commit Charge (K)
Total 77524
Limit 322000
Peak 78896

Kernel Memory (K)
Total 19124
Paged 16688
Nonpaged 2436

Processes: 32 | CPU Usage: 57% | Mem Usage: 77524K / 322000K

Screen snapshot from:
Task Manager | Performance tab

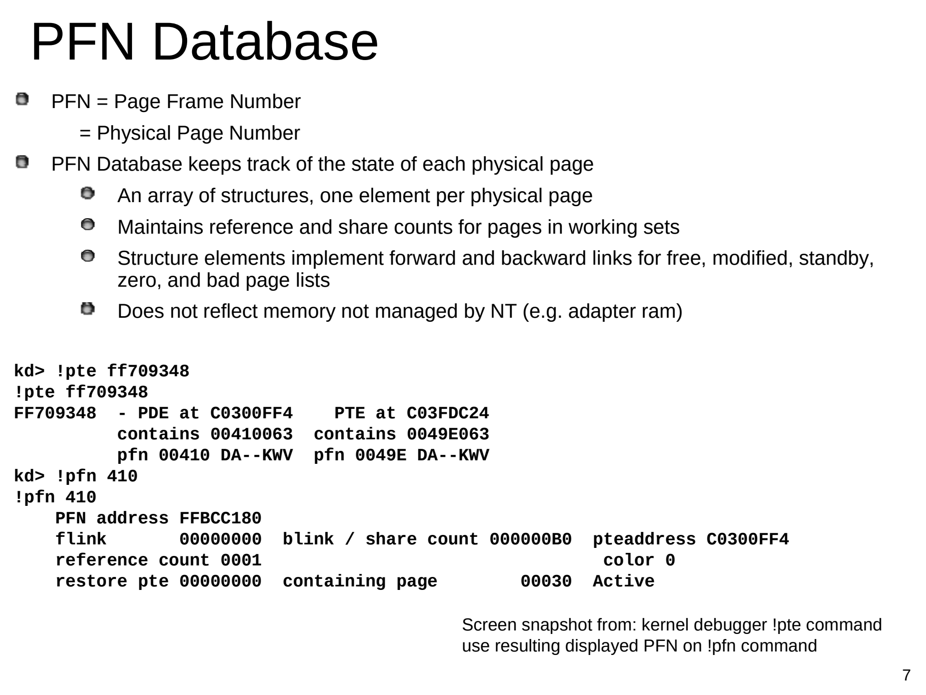
Lab objective: Viewing System Memory Information

The Performance tab in the Windows Task Manager displays basic system memory information. This information is a subset of the detailed memory information available through the performance counters.

Both Pmon.exe and Pstat.exe (in the Windows Support Tools) display system and process memory information.

Finally, the !vm command in the kernel debugger shows the basic memory management information available through the memory-related performance counters. This command can be useful if you’re looking at a crash dump or hung system. Here’s an example of its output:

```
kd> !vm
    *** VirtualMemory Usage ***
            PhysicalMemory: 32620 ( 130480Kb)
            PageFile: \??\C:\pagefile.sys
                Current: 204800Kb Free Space: 101052Kb
                Minimum: 204800Kb Maximum: 204800Kb
            Available Pages: 3604 ( 14416Kb)
            ResAvailPages: 24004 ( 96016Kb)
            ModifiedPages: 768 ( 3072Kb)
            NonPagedPoolUsage: 1436 ( 5744Kb)
            NonPagedPoolMax: 12940 ( 51760Kb)
            PagedPool 0Usage: 6817 ( 27268Kb)
            PagedPool 1Usage: 982 ( 3928Kb)
            PagedPool 2Usage: 984 ( 3936Kb)
            PagedPool Usage: 8783 ( 35132Kb)
            PagedPool Maximum: 26624 ( 106496Kb)
            …..
```



PFN Database

- PFN = Page Frame Number
  = Physical Page Number
- PFN Database keeps track of the state of each physical page
  - An array of structures, one element per physical page
  - Maintains reference and share counts for pages in working sets
  - Structure elements implement forward and backward links for free, modified, standby, zero, and bad page lists
  - Does not reflect memory not managed by NT (e.g. adapter ram)

```
kd> !pte ff709348
!pte ff709348
FF709348  - PDE at C0300FF4    PTE at C03FDC24
          contains 00410063  contains 0049E063
          pfn 00410 DA--KWV  pfn 0049E DA--KWV
kd> !pfn 410
!pfn 410
    PFN address FFBCC180
    flink       00000000  blink / share count 000000B0  pteaddress C0300FF4
    reference count 0001                                 color 0
    restore pte 00000000  containing page        00030  Active
```

Screen snapshot from: kernel debugger !pte command
use resulting displayed PFN on !pfn command

7

Lab objective: Viewing PFN Entries

You can examine individual PFN entries with the kernel debugger !pfn command. You first need to supply the PFN as an argument. (For example, !pfn 0 shows the first entry, !pfn 1 shows the second, and so on.) In the following example, the PTE for virtual address 0x50000 is displayed, followed by the PFN that contains the page directory, and then the actual page:

```
kd> !pte 50000
00050000 - PDE at C0300000          PTE at C0000140
            contains 00700067       contains 00DAA047
            pfn00700 --DA--UWV      pfn00DAA--D---UWV

kd> !pfn700
    PFN00000700 at address 827CD800
    flink 00000004 blink/ share count00000010 pteaddress C0300000
    reference count 0001                        color 0
    restore pte 00000080 containing page 00030 Active M
    Modified

kd> !pfn daa
    PFN00000DAA at address 827D77F0
```

PFN Database

- Only way to get actual size of physical memory lists is to use !memusage in Kernel Debugger

```
lkd> !memusage
 loading PFN database
                 Zeroed:     0 (     0 kb)
              Free:      0 (     0 kb)
           Standby: 139069 (556276 kb)
          Modified:    161 (   644 kb)
   ModifiedNoWrite:      0 (     0 kb)
      Active/Valid: 122759 (491036 kb)
        Transition:      8 (    32 kb)
           Unknown:      0 (     0 kb)
             TOTAL: 261997 (1047988 kb)
```

Screen snapshot from:kernel debugger
!memusage command

8

EXPERIMENT: Viewing the PFN Database

Using the kernel debugger !memusage command, you can dump the size of the various

paging lists. The following is example output from this command:

lkd> !memusage

loading PFN database

loading (100% complete)

Compiling memory usage data (99% Complete).

Zeroed: 8474 (33896 kb)

Free: 256 (1024 kb)

Standby: 50790 (203160 kb)

Modified: 496 (1984 kb)

ModifiedNoWrite: 0 (0 kb)

Active/Valid: 201980 (807920 kb)

Transition: 1 (4 kb)

Unknown: 0 (0 kb)

TOTAL: 261997 (1047988 kb)

Lab: Memory Leaks

- Run Leakyapp.exe (Resource Kit)
- In Task Manager Process tab, watch Mem Usage & VM Size grow (also look at Performance tab Commit limit/peak)
  - Mem Usage will eventually reach an upper limit
  - VM Size will grow until no more page file space

9

Experiment: Creating a Memory Leak

To observe the impact of a memory leak, run the Leakyapp.exe tool and press Start Leaking (run several copies to fill virtual memory more quickly). Observe the commit charge total rising (either using Task Manager or Performance Monitor). Notice the error displayed when the system runs out of virtual memory.

Finally, kill the Leakapp process(es) and notice the commit charge total return to the previous size.

Detailed instructions:

Click on Start, Run and type Leakyapp

Click the “Start Leaking” button.

Bring up Task Manager and click on the Processes tab.

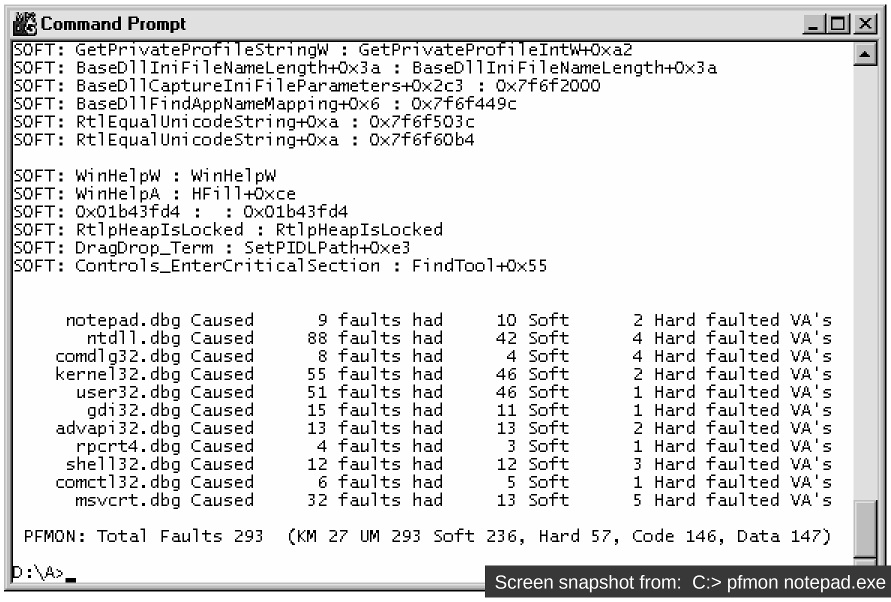
Make sure you have the Mem Usage and VM Size columns configured to be displayed.

Sort by the VM Size column from highest to lowest – if not right away, shortly, you should see Leakyapp.exe as the process with the largest VM Size.

Notice that the Mem Usage column and the VM Size column are growing together. This is because the operating system is permitting the process working set size to grow physically as the process virtual size grows. At some point, the working set size will be capped, but the virtual size will continue to grow until there is no more system virtual memory.

While Leakyapp is running, click on the Performance tab and notice the page file

Page Fault Monitor (pfmon)



```
SOFT: GetPrivateProfileStringW : GetPrivateProfileIntW+0xa2
SOFT: BaseDllIniFileNameLength+0x3a : BaseDllIniFileNameLength+0x3a
SOFT: BaseDllCaptureIniFileParameters+0x2c3 : 0x7f6f2000
SOFT: BaseDllFindAppNameMapping+0x6 : 0x7f6f449c
SOFT: RtlEqualUnicodeString+0xa : 0x7f6f503c
SOFT: RtlEqualUnicodeString+0xa : 0x7f6f60b4

SOFT: WinHelpW : WinHelpW
SOFT: WinHelpA : HFill+0xce
SOFT: 0x01b43fd4 :  : 0x01b43fd4
SOFT: RtlpHeapIsLocked : RtlpHeapIsLocked
SOFT: DragDrop_Term : SetPIDLPath+0xe3
SOFT: Controls_EnterCriticalSection : FindTool+0x55


     notepad.dbg Caused      9 faults had     10 Soft      2 Hard faulted VA's
       ntdll.dbg Caused     88 faults had     42 Soft      4 Hard faulted VA's
    comdlg32.dbg Caused      8 faults had      4 Soft      4 Hard faulted VA's
    kernel32.dbg Caused     55 faults had     46 Soft      2 Hard faulted VA's
      user32.dbg Caused     51 faults had     46 Soft      1 Hard faulted VA's
       gdi32.dbg Caused     15 faults had     11 Soft      1 Hard faulted VA's
    advapi32.dbg Caused     13 faults had     13 Soft      2 Hard faulted VA's
      rpcrt4.dbg Caused      4 faults had      3 Soft      1 Hard faulted VA's
     shell32.dbg Caused     12 faults had     12 Soft      3 Hard faulted VA's
    comctl32.dbg Caused      6 faults had      5 Soft      1 Hard faulted VA's
      msvcrt.dbg Caused     32 faults had     13 Soft      5 Hard faulted VA's

 PFMON: Total Faults 293  (KM 27 UM 293 Soft 236, Hard 57, Code 146, Data 147)

D:\A>_
```

Screen snapshot from: C:> pfmon notepad.exe

Lab objective: Viewing Page Fault Behavior

With the Pfmon tool (in the Windows 2000 and 2003 resource kits, as well as in the Windows XP Support Tools), you can watch page fault behavior as it occurs. A soft fault refers to a page fault satisfied from one of the transition lists. Hard faults refer to a diskread. The following example is a portion of output you'll see if you start Notepad with Pfmon and then exit. Be sure to notice the summary of page fault activity at the end.

```
C:\>pfmon notepad
SOFT:KiUserApcDispatcher :KiUserApcDispatcher
SOFT:LdrInitializeThunk :LdrInitializeThunk  SOFT:0x77f61016: : 0x77f61016
SOFT:0x77f6105b: : fltused+0xe00  HARD:0x77f6105b: : fltused+0xe00
SOFT:LdrQueryImageFileExecutionOptions :  LdrQueryImageFileExecutionOptions
SOFT:RtlAppendUnicodeToString: RtlAppendUnicodeToString
SOFT:RtlInitUnicodeString: RtlInitUnicodeString

notepad     Caused 8faultshad 9 Soft 5 Hardfaulted VA’s
ntdll       Caused 94faultshad 42 Soft 8 Hardfaulted VA’s
comdlg32    Caused 3faultshad 0 Soft 3 Hardfaulted VA’s
shlwapi     Caused 2faultshad 2 Soft 2 Hardfaulted VA’s
gdi32       Caused 18faultshad 10 Soft 2 Hardfaulted VA’s
kernel32    Caused 48faultshad 36 Soft 3 Hardfaulted VA’s
user32      Caused 38faultshad 26 Soft 6 Hardfaulted VA’s
advapi32    Caused 7faultshad 6 Soft 3 Hardfaulted VA’s
rpcrt4      Caused 6faultshad 4 Soft 2 Hardfaulted VA’s
comctl32    Caused 6faultshad 5 Soft 2 Hardfaulted VA’s
shell32     Caused 6faultshad 5 Soft 2 Hardfaulted VA’s
            Caused 10faultshad 9 Soft 5 Hardfaulted VA’s
winspool    Caused 4faultshad 2 Soft 2 Hardfaulted VA’s
PFMON: Total Faults250   (KM 74 UM250Soft 204,Hard 46, Code121, Data129)
```