

##	Limit	Description	Resource/Table
1	2048	AIF user PORTs per system	AIF Ports ID Table
2	32768	AIF PORT open senders	AIF Ports Open Table
2	32768	AIF PORT open receivers	AIF Ports Open Table
3	8144	AIF PORT message length (user)	AIF Port Message Table
4	200	Installed Vendor ID's	AIF Known Vendor Table

(1) AIF ports id table

The Architected Interface Ports facility has it's own KSO_AIF_PORTS (\$123,#291) to keep track of user ports. The KSO_AIF_PORTS is created the first time AIFPORTOPEN is called to create a port. This KSO is also called GPD_TABLE (Global Port Data table), it contains several pointers to other AIF ports tables. The first of these is the PortId table. This one keeps track of all open AIF ports. A software constant limits the maximum number of open ports to 2048.

(2) AIF ports open table

The AIF Port Open table keeps track of open accessors. One entry is allocated for send access and one for receive access. The maximum number of concurrently open receivers is 32,768. The maximum number of concurrently open senders is 32,768. The maximum number of open entries per system is 65,536.

(3) AIF port message table

The maximum message size a user can specify is 8144 bytes. The AIF subsystem adds on envelope information to the user message of 48 bytes making the maximum message size #8192.

The AIF ports reside on top of the system ports functionality. When an AIF port is created this results in a call to the lower level ports subsystem. The ports subsystem will create the port out of the non-resident port descriptor table. Each AIF port is created in a single object. This object contains the following:

```

Port Record
Server Control Block
Pool Record
Table Management Header
Message Pool ( Table Management Body )

```

This object is restricted to a 4MB size and is allocated out of SR6/SR7 space. It is very important for the application developer to use this space wisely. Most system data structures are allocated out of SR6/SR7 and this space is limited to 2 gigabytes.

Example:

```

$188      port record
  2C      server control block
  40      ( for cache alignment )
  40      pool record
 24C      table management header
nnn      message pool size

```

where nnn = (message size + 30 + message_frame_rec) * number of messages
 (default \$100 + \$30 + \$10) * \$20 = \$2800

This creates a \$2c80 total size object which would use a 32K allocation unit. When a user specifies a #8144 message size at port creation it is possible to allow #510 messages. This will fit in the 4 MB object. However, it is not possible to create 2048 ports which use 4 MB objects.

(4) AIF known vendor table - KSO290

KSO #290 keeps track of all the installed Vendor Ids on the system. The number of Vendor Ids in this table has been set to 200.

CM object management:

##	Limit	Description	Resource/Table
1	2047	Code Segments (Loaded CM Libraries)	Code Segment Table
1	255	Code Segments (Physically Mapped)	Code Segment Table
2	255	Code Segments (Per Loaded CM Program)	Code Segment Table Ext
3	16383	Data Segments	Data Segment Table

(1) Code segment table - KSO253

The Code Segment Table (CST) is used to manage Compatibility Mode (CM) code segments which are a part of a Segmented Library (SL). There are 255 entries reserved for physically mapped segments. Physically mapped segments are reserved for use by the system. These typically reside in SL.PUB.SYS. The remaining entries are available for user library segments. These are referred to as logically mapped segments.

A CST entry is allocated whenever an SL segment is loaded for the first time. This occurs when:

- A CM program is executed via the :RUN command with the ;LIB= option.
- SL segments are :ALLOCATED.
- A procedure (in an SL) is loaded via the LOADPROC intrinsic.

CST entries are shared. Loading an SL segment after the first time will not result in the allocation of a CST entry. Share counts on segments are maintained in the Loader Segment Table (LST). The share count for a segment is decremented when the segment is unloaded. When the share count for a particular segment drops to zero, the CST entry for that segment is released.

The CST table limits the number of compatibility mode library segments.

(2) Code segment table ext

The Code Segment Table Extension (CSTX) is used to manage Compatibility Mode (CM) code segments which are a part of a CM program. There is one CSTX for each CM program that is loaded or :ALLOCATED. Each CM program can reference up to 255 logically mapped segments. Logically mapped segments include those that are a part of the program file and any user SL segments that are referenced. Each logically mapped segment that is a part of the program file will use an entry in the processes' CSTX. Logically mapped user SL segments will count toward the maximum number of 255 logically mapped segments, but will use CST entries instead of CSTX entries.

This table limits the number of logically mapped compatibility mode segments in any program file to 255.

(3) Data segment table - KSO192

The Data Segment Table (DST Table) is used to manage Compatibility Mode (CM) Data Segments (DSTs). There is a maximum of 16383 entries in the DST Table.

Several DSTs are used by the Operating System (and other system software, eg: Data Communications software) for global system data structures. It is difficult to estimate how many global DSTs will be required on a particular system. Below are some examples of system DSTs:

- 63 Reserved system DSTs (eg. FMAVT, LOG TABLE, etc)
- Up to 118 File System Control Block Tables
- Dynamically allocated system DSTs (eg. BIPC DST, LOG BUF DST, LSTX)
- CM Stacks for system processes (eg. SESSION, JOB, PROGEN)

Each Job uses 4 DSTs and each Session uses 5 DSTs to get to the Command Interpreter prompt. Running additional programs will require additional DSTs. The DSTs required for a Job or a Session are listed below:

- Job Information Table (JIT)
- Job Directory Table (JDT)
- CM Stack for JSMAIN process
- CM Stack for CI process
- [FOR SESSIONS ONLY] 1 PACB for the \$STDIN/\$STDLIST device

Each additional user process uses 1 or more DSTs. At a minimum each process will require 1 DST for a CM Stack. ALL processes need a CM stack (including those created from NM program files). Process DST usage is summarized below:

- CM Stack
- DSTs explicitly allocated using GETDSEG
- PACBs for CM files (RIO, Circular, Message, & Device Files) opened for Buffered Access

In order to estimate the number of DSTs used by the system we will use the following assumptions:

- The system has the number of sessions logged on is 1700
- 1700 of these sessions are "active"
See the Total Concurrent Logons - Concurrent Process Limit discussion in the JOB/SESSION section.
- 500 global system DSTs are used by the operating system.
(500 is NOT an exact number, but is an educated guess.)

For the model above the number of DSTs used by the system would be equal to the following formula:

$$\# \text{ Of System DSTs} = 500 + (1700 * 6) = 10700$$

NOTE: Each of the 1700 application programs executing in the above example may require additional DSTs. Since the maximum number of DSTs is 16383, this leaves approximately 5683 (16383-10700) DSTs for use by user applications. This is approximately 3 per application (5684/1700). This limitation of 3 DSTs available for the user application can be a problem on systems that run CM applications.

Here is another scenario with different assumptions.

- The system has 1000 sessions logged on via DTCs.
- The system has 700 sessions logged remotely via virtual terminals.
- The system has 50 jobs logged on.
- All jobs and sessions are active.
- 500 global system DSTs.

For the model above the number of DSTs used by the system would be equal to the following formula:

$$\begin{aligned} \# \text{ Of System DSTs} &= 500 + (1000 * 6) + (700 * 8) + (50 * 5) \\ &= 12350 \end{aligned}$$

Remember that your mileage may vary depending on: the number of processes per logon, the number of CM files open, and the number of extra data segments an application uses.

Devices:

##	Limit	Description	Resource/Table
1	~ 4000	Device Classes	DCT DST
2	255	Disc Drives	Max Tested Configuration
3	4679	Logical Devices	3 Digit Ldev Numbers
2	127	Mirrored Disc Pairs	Max Tested Configuration
4	4649	Terminals Connected via DTCs	NMMGR Config Limit

(1) Dct DST - DST040

The Device Class Table (DCT) is used to map device classes to their respective logical devices. For devices configured in SYSGEN, there is a limit of 256 ldevs for each unique device class. In addition, SYSGEN limits the number of different device classes that can be assigned to a specific ldev to 8.

(2) Max tested configuration

The maximum number of disc drives (255) and the maximum number of mirrored disc pairs (127) which are supported is based on the maximum tested configurations. Software or hardware limits may be reached if these values are exceeded.

NOTE: Although it is possible to mount this number of disk drives on a system, putting them all in a single volume set with a heavy transaction load may cause problems. Please see your systems consultant.

(3) 3 digit ldev numbers

In MPE iX 5.5 the maximum number of devices is limited to 4679. This value will also be the maximum LDEV number tolerated by NMMGR and SYSGEN. The maximum number of terminal devices connected via DTCs is 4649, which leaves 30 empty device numbers for other peripherals.

(4) NMMGR config limit

The utility program NMMGR.PUB.SYS will not allow a user to configure more than 4649 terminals (connected via DTCs). This value was chosen to be 30 less than the maximum number of devices. Being able to configure 4649 terminals does not mean that MPE will support 4649 sessions. (See the JOB/SESSION section of this document for more information).

NOTE: Serial printers are also configured as terminals using NMMGR. The total number of serial printers and terminals cannot exceed 4649. The maximum number of serial printers that is supported varies, depending on the system model and the types of printers that are being connected. Refer to the Systems Configuration Guide for specific details.

File system:

##	Limit	Description	Resource/Table
1	16384 <input type="text"/>	Concurrent Writers of Msg File	NM Writer ID Table
2	16384	Files Open (G-Multi or Job-Multi Access)	FS GDPD Table
3	1024	Files Open (per Process)	PLFD
4	~8000000 <input type="text"/>	Open Message files (w/o timed reads)	NM IPC Table
4	~5714285 <input type="text"/>	Open Message Files (w/timed reads)	NM IPC Table
5	5460	Open Multi Access CM Files	FMAVT Dst
6	2330	Open Non Sharable Devices	Input/Output Device Directory
7	128	User Logging ID's	Log ID Table
8	128	User Logging Processes	Log Table
9	1140 <input type="text"/>	Processes Per User Logging ID	Log Buf DST

(1) NM writer id table

The new native mode Writer ID Table (KSO227), contains a maximum of 16,384 entries. Each concurrent writer of a message file requires one entry. As a result a maximum of 16384 processes can concurrently open an IPC file with write access.

(2) Fs gdpd table - KSO222

Every file that is opened gets a Global Data Pointer Descriptor (GDPD) entry. Most GDPD entries are allocated within the Process Local File Descriptor (PLFD) table. Each PLFD entry is paired with a GDPD entry within the PLFD table. However, if a native mode file is opened with Multi Access (intra-job) or G-Multi Access (inter-job), then the GDPD entry is allocated from the System GDPD table. Subsequent opens of the same Multi Access file will share the same GDPD entry. If the file is a CM type file (eg. Message file, RIO, Circular, or Device file), then the GDPD entry is allocated in the PLFD, and the FMAVT table (DST 44) manages the Multi Access.

The GDPD table limits the number of unique NM files that can be opened simultaneously with Multi Access or G-

Multi Access to 16,384.

(3) Plfd - kpo017

There used to be a Process Local File Descriptor (PLFD) table allocated for each process. Every file that is opened gets a PLFD entry. (For those of you familiar with, MPE VE, this is analogous to an AFT entry). In MPE iX 4.0 in order to save SR6/SR7 space, the Process Local File Descriptor was modified to allocate files in 64 file chunks. The KPO #17 in the PIBX will still point to the PLFD, but it will now point to a PLFD index. The PLFD index is a data structure that points to the plfdare 64es. The index value for a filenum will be the file number div 64 (there is 64 files per PLFD).

The maximum number of files that can be opened simultaneously by each process has been set to 1024.

(4) NM ipc table

The FSIPC Object (KSO225) is used as an internal data structure for message files (IPC) to pass data between processes. Within this data structure, one or more port entries are allocated for processes that open message files and to send WAIT or REPLY messages under EOF conditions. As a result, two processes accessing a single message file may use anywhere from 5 to 7 FSIPC entries in the FSIPC object.

The FSIPC object is created when the files system is initialized during system bootup. The FSIPC object can contain a maximum of 40,000,000 (40 million) entries. The number of entries required to open a file depends on several factors explained below.

There are four different types of entries in the FSIPC object: message queue entries (MQE), timer list entries (TLE), port entries, and free blocks. These entries are described in more detail below.

Port entries

The first time a particular message file is opened, three port entries will be allocated in this table. One of the port entries serves as the communication port for the process that opened the message file. The other two ports are global read and write queues for the file. Additional opens of the same message file will only require one port entry per open, which will be used as a communication port.

Message files are not particularly useful unless they are opened by at least two processes. As a result, at least four port entries will usually be required per message file in the simplest case. If more than two processes have the file open, more port records will be required.

Message queue entries

MQEs are allocated when an EOF condition is encountered during a read or write operation. For reads, this occurs when a process tries to read from an empty message file. For writes, this occurs when a process tries to write to a

message file which is full (ie. has hit the LIMIT for the file).

Usually, message files will have one or more MQE entries. The exact number of MQE entries will depend on the nature of the application. Applications which use message files tend to be structured as "producer/consumer" applications. One or more processes will "produce" records for the message file (eg. writers), and one or more processes will "consume" them (eg. readers).

If the writer processes are faster than the reader processes, then the message file may reach the EOF. If an EOF condition is encountered when a writer is attempting to write a new record to the message file, the process will block, and an MQE entry will be allocated. This will happen for each writer that blocks. The number of MQE entries required for any given message file will be less than or equal to the number of writer processes. One way to minimize the possibility that this situation will occur is to ensure that the file limit of each message file is large enough so that the writer processes will not have to block on an EOF condition.

If the reader processes are faster than the writer processes, then an MQE entry will be required as soon as a reader attempts to read the message file when it is empty. If the readers "lead" the writers (ie. always requesting data before it is actually written to the message file), then the number of MQE entries required will probably be equal to the number of reader processes for that message file.

In either case above, the number of MQE entries will vary according to how many processes are using the message file, and what they are doing.

Timer list entries

TLEs are allocated when timeouts are specified (via the FCONTROL intrinsic). TLEs are deallocated when the timed read or write completes (either normally or when the timer expires).

If an application does use timed reads and writes, the maximum number of TLE entries is equal to the number of processes which have the message file open minus one.

Estimating maximum number of message files

The maximum number of open message files on a system is limited by the equation below:

$$\# \text{ OPEN MESSAGE FILES} = \# \text{ FSIPC ENTRIES} \quad / \quad (\# \text{ FSIPC ENTRIES PER FILE})$$

The number of FSIPC ENTRIES is a constant for all systems (40,000,000). The number of FSIPC ENTRIES PER FILE will depend on the application(s) used on the system. For our calculation we will assume the following model:

- Each application using message files has N processes accessing the files.
- The number of port entries required is N+2 (One per process plus the two global read/write queues).
- The number of MQE entries is N-1 (All processes should not be blocked

on an EOF condition).

- Each application either uses timed reads and writes uniformly, or it does not use them at all. If timed reads and writes are used then the number of TLE entries will be N .

For applications which do not use timed reads and writes, the number of FSIPC ENTRIES PER FILE will be $(N+2) + (N-1)$ or $2N+1$. For applications which do use timed reads and writes, the number of FSIPC ENTRIES PER FILE will be $(N+2) + (N-1) + N$ or $3N+1$.

Let's assume that ALL the applications are simple applications with two processes accessing each message file, ($N = 2$), and that the applications are not using timed reads and writes. Then, the maximum number of OPEN MESSAGE FILES would be $40,000,000 / 5 = 8,000,000$.

If the applications are using timed reads, (and $N = 2$), then the maximum number of OPEN MESSAGE FILES would be $40,000,000 / 7 = 5,714,285$.

(5) Fmavt DST - DST044

One FMAVT entry is allocated for each unique Compatibility Mode (CM) file that is opened for Multi Access. Currently, this means the following file types: Message files, Circular files, RIO files, and Device files.

Note that \$STDIN and \$STDLIST are opened for Multi Access. Since terminals are still considered to be CM Device files, each session will require one entry in the FMAVT, assuming that \$STDIN and \$STDLIST are the same terminal. (The FMAVT entry is shared between them in this case).

This table limits the number of unique CM files that can be opened simultaneously for Multi Access to 5460. Files that are opened more than once will share an FMAVT entry.

(6) Input/output device directory

The Input Device Directory (IDD) and Output Device Directory (ODD) data structures are kept in CM DSTs. The IDD is DST 45, and the ODD is DST 46. The format/layout of the IDD and the ODD are identical. For this reason they are often referred to using the generic acronym XDD.

The maximum number of entries in the XDD is limited by the maximum size to which the table can grow. Since the XDD is a CM table, this section will use CM words (16-bits) for all size calculations. The table is managed in "sectors" (128 CM word chunks). The maximum size of the table is 255 "sectors" times 128 CM words per "sector", or 32640 CM words. This is only slightly less than the maximum size of a CM DST (32764 CM words) so it does not significantly reduce the maximum number of entries. The XDD now supports 4679 devices. The entry size is 5 CM words. Subentries do not exist since 4.7.

The XDD has one 5 CM word header for the table itself. This contains various linkage information, as well as information about the current number of "sectors" allocated. There is an entry for each LDEV on the system. Non-sharable devices which include all terminals (both nailed and non-nailed), virtual terminals (VT), non-spooled printers and tape drives can have a reallocation entry.

Summary of values

- Size of header for XDD table is 5 CM words
- Size of entry for each device configured is 5 CM words
- Size of reallocation entry for each device configured is 5 CM words

The maximum number of XDD entries is the maximum number of LDEVS which is 4679. The maximum number of reallocation entries is 1847.

(7) Log id table - DST033

When a user creates a new logid through the :GETLOG command, an entry is added to the LOG ID table. This table is created large enough to contain 128 entries. This limits the system to a total of 128 user logging IDs.

(8) Log table - DST027

When a User Logging process is started through the :LOG command (with either the START or RESTART option) an entry is added to the LOG Table. The log table is created large enough to contain 128 entries. This limits the system to a maximum of 128 User Logging IDs in use concurrently.

NOTE: A single User Logging ID can be used by multiple processes/programs concurrently. See discussion of Processes Per User Logging ID - LOG BUF DST - in this section.

(9) Log buf DST

A unique User Logging process is created for each active Log ID. Each User Logging process has a LOG BUF DST associated with it. User Logging processes are started using the :LOG command with the START or RESTART option (see the discussion of User Logging Processes - LOG TABLE - in this section).

The maximum number of processes logging to a single User Logging ID is configurable in SYSGEN (log section, ulog command, NLOGPROCS parameter). The maximum value for NLOGPROCS is 1140. This limit is based on the maximum possible size of the LOG BUF DST.

Job/session:

##	Limit	Description	Resource/Table
1	500	Console Requests	Reply Info Table
2	1365	RINs, Global	RIN Dst

2	5459	RINs, Total	RIN Dst
3	2700	Total Concurrent Logons (Local Only)	Concurrent Process Limit
4	~ 2700 <input type="text"/>	Total Concurrent Logons (Remote & Local)	NS/Concurrent Processes

(1) Reply info table - KSO166

The Reply Information Table (RIT) contains one entry for each console request. There is a limit of 500 console requests at any one time.

(2) Rin DST - DST022

Entries in the RIN table are allocated whenever a Global RIN or Local RIN is acquired using the :GETRIN command or GETLOCIN intrinsic. Global RINs are deallocated with the :FREERIN command, and Local RINs are deallocated by the FREELOCIN intrinsic. The CM file system also uses RINs (called File RINs) whenever a CM type file is opened for FLOCK access.

The total number of RINs that are available on the system is 5,459, of which a maximum of 1365 can be used as Global RINs.

(3) Concurrent process limit

Release 5.5 supports a maximum of 2700 active logons. The term "active" implies that the job or session is running a program. Additionally, the process limit has increased to 8190. For planning purposes, you should allow 100 processes for the operating system's use (this is not an exact number, but should be enough). This leaves 8090 processes for jobs and sessions.

Each direct logon requires a minimum of 2 processes; a JSMAIN and a CI. If the job or session runs a simple program (which does no process handling), then it requires a total of 3 processes per logon. This allows for a maximum of approximately 2700 active logons ($3 \times 2700 = 8100$) with processes to spare.

It is very important to note that many applications and third party products use process handling. You must determine how many processes will be used by each session or job in order to accurately predict the number of jobs or sessions that the system will support.

Below is a worksheet which can be used to estimate the concurrent process requirements for your system. If your active users are running programs which do process handling, then you will need to change the multipliers used for active jobs and sessions.

Active Jobs _____ x 3 = _____

```

# Inactive Sessions      _____  (<= 2500)                x 2 = _____
# Active Sessions       _____  (<= 2500)                x 3 = _____
+                               system processes + 100
=====
Total Connections      _____  Total Processes _____ (<= 8190)

```

Although it is possible to configure more than 2500 terminal I/O devices, neither the inactive sessions nor the active sessions should exceed 2500. An HP support representative should be contacted before exceeding these limits.

(4) NS/concurrent processes

The exact number of remote sessions which can be supported on a given system will depend on the exact mix of jobs and sessions (remote and local, active and inactive) on that system.

The maximum number of concurrent processes may limit the number of remote logons before the maximum number of data communications servers does. Likewise, the server limit may be reached before the concurrent process limit. This section will address the question of total concurrent jobs and sessions on a system with a mixture of local and remote logons.

Concurrent process limit

A remote logon to another MPE/iX system, (logon via a Virtual Terminal), requires 3 processes on the host system and 3 on the remote system. There will be a JSMAIN, a CI, and a VTSERVER process on both sides. If the user runs a simple program (which does no process handling), then there will be a total of 4 processes per logon.

Release 5.5 supports 2000 Virtual Terminal (VT) sessions. With 1250 active VT logons, it is estimated that 8100 processes are used ((2000*4)+100). With a maximum of 8190 processes, it is possible to achieve a maximum of 2280 active logons if 2000 of these active logons are VT logons.

$$8190 - 100 \text{ system processes} - (2000 * 4) = 90 \text{ processes leftover.}$$

With 2000 active VT logons it is estimated that a maximum of 30 additional active logons is possible (90 processes/3 processes per non-local connection).

It is very important to note that many applications and third party products use process handling. You must determine how many processes will be used by each session or job in order to accurately predict the number of users that the system will support.

Below is a worksheet which can be used to estimate the concurrent process requirements for a system with networked connections. If the active users are running programs which do process handling, then the multipliers used for active jobs and sessions (local or remote) will need to be modified.

```

# Active Jobs      _____  x 3 = _____
# Inactive Local Sessions _____ (<- ~2500) x 2 = _____

```

```

# Active Local Sessions      _____ (<- ~2500)                x 3 = _____
# Inactive Remote Sessions  _____ (<= 2000)                x 3 = _____
# Active Remote Sessions    _____ (<= 2000)                x 4 = _____
                               +
                               system processes + 100
=====
Total Connections _____ Total Processes _____ (<= 8190)
    
```

Although it is possible to configure more than 2500 terminal I/O devices, neither the inactive local sessions nor the active local sessions should exceed 2500. An HP support representative should be contacted before exceeding these limits. Also, the sum of all remote sessions cannot exceed 2000 due to the limit on Network Servers imposed by the NS transport layer.

Data communication servers

In addition to application programs, data communication servers also require processes. The table above takes into account VTSERVER processes created for the purpose of initiating a remote logon. Using additional network services may require additional server processes as shown in the table below.

For example, if the user is using Network File Transfer (DSCOPY), there will be an NFT server created on both sides. Also, a VT session must be established or the auto logon feature must be used. Unless a VT session is required, the auto logon feature is preferable since it will reduce the number of servers required.

Service Used	SERVERS REQUIRED		Function
	Local Side	Remote Side	
NFT	NFT	NFT	
RFA	-	RASERVER	
NSSTAT	-	NSSTATUS	
LOOPBACK	LOOPINIT	LOOPBACK	
RPM	-	DSSERVER RPMDDAD	
PTOP	-	DSSERVER	
VTR	VTSERVER	VTSERVER	
VT	VTSERVER	VTSERVER	
Access local ALLBASE data	ALLBASE		
Access remote MPE ALLBASE Data	ALLBASE, NS	ALLBASE, NS	ALLBASE/NET ISQL, PPs

Access remote Unix ALLBASE Data	ALLBASE, NS	ALLBASE, NS	ALLBASE/NET ISQL, PPs
Access local Turbo data via ALLBASE	IMAGE/SQL		
Access remote Turbo data via ALLBASE	ALLBASE	IMAGE/SQL	ALLBASE/NET
Access ALLBASE or IMAGE/SQL from PC	DOS, NS, ARPA Svs, or IPX Windows 3.1 ALLBASE/SQL PC API Gupta or ODBC	ALLBASE (HPIP DVR) Netware/iX	ALLBASE/NET
Access ALLBASE or IMAGE/SQL from PC	DOS, NS or ARPA Svs Information Access Windows 3.1	ALLBASE (IASQL)	

Loader:

##	Limit	Description	Resource/Table
1	2047	CM Code Segments (Loaded CM Libraries)	Code Segment Table
1	255	CM Code Segments (Physically Mapped)	Code Segment Table
2	255	CM Code Segments Per Loaded Program File	Code Segment Table Ext
3	519	Loaded Unique CM Program Files	CST Block DST
4	See text	Loaded CM Programs (& LOADPROCs)	LST DST
5	~ 22771	Loaded Unique NM Program Files (No XLs)	Loaded File Table
5	~ 90	Loaded Unique NM Pgms (1 Unique XL/ Pgm)	Loaded File Table
6	88	NM User Libraries per Process	PFL
6	~ 10000	SG Data Exports/Process (75 avg imp/exp)	PFL

(1) Code segment table - KSO253

The Code Segment Table (CST) is used to manage Compatibility Mode (CM) code segments which are a part of a Segmented Library (SL). There are 255 entries reserved for physically mapped segments. Physically mapped segments are reserved for use by the system. These typically reside in SL.PUB.SYS. The remaining entries are available for user library segments. These are referred to as logically mapped segments.

A CST entry is allocated whenever an SL segment is loaded for the first time. This occurs when:

- A CM program is executed via the :RUN command with the ;LIB= option.
- SL segments are :ALLOCATED.
- A procedure (in an SL) is loaded via the LOADPROC intrinsic.

CST entries are shared. Loading an SL segment after the first time will not result in the allocation of a CST entry. Share counts on segments are maintained in the Loader Segment Table (LST). The share count for a segment is decremented when the segment is unloaded. When the share count for a particular segment drops to zero, the CST entry for that segment is released.

The CST table limits the number of compatibility mode library segments.

(2) Code segment table ext

The Code Segment Table Extension (CSTX) is used to manage Compatibility Mode (CM) code segments which are a part of a CM program. There is one CSTX for each CM program that is loaded or :ALLOCATED. Each CM program can reference up to 255 logically mapped segments. Logically mapped segments include those that are a part of the program file and any user SL segments that are referenced. Each logically mapped segment that is a part of the program file will use an entry in the processes' CSTX. Logically mapped user SL segments will count toward the maximum number of 255 logically mapped segments, but will use CST entries instead of CSTX entries.

This table limits the number of logically mapped compatibility mode segments in any program file to 255.

(3) CST block DST - DST035

The CST Block table contains a pointer to the CSTX object for each CM program that is loaded or allocated. An entry is used whenever a program is loaded or :ALLOCATED for the first time. The entry is deallocated when the program is terminated for the last time or when the program is :DEALLOCATED.

This table limits the number of concurrently running CM programs to 519.

(4) Lst DST - DST018

There is an SL File entry in the LST for every loaded SL (including SL.PUB.SYS). These entries are deleted when all SL segments are unloaded. An SL File entry may expand if it was previously loaded and an unloaded segment is loaded. There is a Program File entry for every loaded/allocated program file. These entries are deleted when all processes running the file terminate and it is no longer allocated. There is a Sharer entry for every process running a

CM program file. These entries are deleted when the process terminates. Loading, Waiter, and Loaded entries are created and deleted during the load/allocate of a program file. Extension and Loadproc Master entries are normally stored in the LSTX DSTs. However, during a LOADPROC operation, they are created in the LST before being transferred to the LSTX. (Switching to CM by name causes a LOADPROC operation to occur.) The maximum size of the LST is #65528 bytes. As the entry size is variable, here is the LST internal description you may use to evaluate the number of entries used. The LST is made up of two parts: a fixed-sized area for overhead information and a variable-sized area for directory entries. The overhead area is #197 CM words in size at the beginning of the LST. The LST directory entries are variable in size (size in CM words, includes 3 word overhead):

```

Garbage           :   size of free area
System SL File    :   #41+3*(number of loaded segments+#10)
Other SL File     :   #25+3*(number of loaded segments+#10)
Program File     :   #13+#35+#19*(number of group/pub SLs loaded)+1+
                   :   (number of loaded program/SL segments)

Loading          :   6
Waiter           :   8
Sharer           :   7
Extension        :   8+(procedure name length)+1+#35+
[in LSTX]        :   #19*(number of group/pub SLs loaded)+1+
                   :   (number of referenced program/SL segments, min #32)
LOADPROC Master:  :   #38+2+2*(number of group/pub SLs loaded)+2+
[in LSTX]        :   2*(number of loaded program/SL segments)

```

Also, the LOAD process permanently allocates two entries in the LST directory that are NOT linked into the entry type headers: a maximum-sized LOADPROC Master entry (#563 CM words, incl. overhead) and a maximum-sized Extension entry (#349 CM words, incl. overhead).

(5) Loaded file table - KSO145

There is one entry in the Loaded File Table (LFT) for every NM program file and NM library file that are loaded. Each entry consists of a primary LFT entry (40 bytes) and an LFT Extension entry (44 bytes * maximum number of SOMs for that program or library). NOTE: For a program file, the maximum number of SOMs is one. For library files, the maximum number of SOMs can be specified when the library is created (the default value is 500). Therefore, a typical LFT entry for a program file will be 84 bytes. A typical entry for a library file (maxsoms = 500) will be 22,040 bytes.

SUMMARY OF VALUES:

```

- Size of an LFT entry           =           40 bytes
- Size of an LFT Extension entry =           44 bytes
- Size of LFT entry for a program file = (40+ 44)           84 bytes
- Size of typical LFT entry for a library file = (40+(44*500)) 22,040 bytes
- Size of LFT table available for user libraries =           2,003,928 bytes
  (LFT entries for NL.PUB.SYS and XL.PUB.SYS assumed to be allocated)

```

Each symbol can have any value, subject to the following limit:

LFT FORMULA

$$(P \times (40 + 44)) + (L \times (40 + (44 \times N))) \leq 2003928$$

SYMBOLS

P = The number of unique program files

L = The number of unique library files

N = Maximum number of SOMs in the library. The default is 500.

(6) Pfl - kpo018

The semantic of the Process File List (PFL) has been redefined in the following ways: First, the term is used for an object that holds the original PFL and a Process Data Dictionary (PDD). Secondly, the term is used for the chain of PFL entries that record the load status of each executable file for the process.

For each process, an PFL object is created with a size of 20,971,520 bytes. From this object, a PFL chain and a PDD are built.

An PFL chain is composed of a PFL header, the main PFL entries, and an PFL extension for each SOM in the file.

The PDD is composed of a PDD header, a PDD hash table, main PDD entries for each data export, and attached to each data export is its list of data imports.

The executable files for a process are generally the program file and the Native Mode (NM) libraries required by the program, including XL.PUB.SYS and NL.PUB.SYS.

SUMMARY OF VALUES FOR THE PFL ENTRIES:

- Size of the PFL header = 100 bytes
- Size of an PFL entry = 184 bytes
- Size of an PFL extension = 32 bytes

The size requirement for the PFL chain is given as follows:

PFL CHAIN FORMULA

$$\text{PFL chain length} = \text{PFL header} + F * (\text{pfl_entry size} + \text{pfl_extension size} * N)$$

SYMBOLS

F = The number of executable files

N = The average number of SOMs per Library. The default set by the MPE/iX Link Editor is 500.

SUMMARY OF VALUES FOR THE PDD:

- Size of the PDD header = 96 bytes
- Size of the PDD hash table = 16,000 bytes
- Size of an PDD export data entry = 84 bytes
- Size of an PDD import data entry = 24 bytes

The size requirement for PDD is given as follows:

PDD FORMULA

$$\text{PDD size} = \text{PDD header} + \text{PDD hash table} + E * (\text{PDD export data entry size} + \text{PDD import data entry size} * I)$$

SYMBOLS

E = The number of shared global data exports

I = The average number of data imports per data export

Considering both the PFL chain and the PDD, each symbol can be of any value--subject to the limits above.

PFL LIMIT FORMULA

$$\text{PFL chain length} + \text{PDD size} \leq 20,971,520$$

The primary limit on the number of executable files that can be opened for a process is the system limit imposed by the Loaded File Table (LFT). (See that section in this document.)

We can use the limits imposed by the LFT to calculate the number of libraries a process can have opened.

LFT LIMIT FORMULA

$$(P * (40 + 44) + (L * (40 + (44 * N))) \leq 2,003,928$$

$$84 + L * (40 + 22,000) \leq 2,003,928$$

$$L \leq 90$$

L = The number of unique library files

N = Maximum number of SOMs in the library. The default is 500.

Using 90 libraries plus the program file, or 91 executable files, and assuming 10,000 shared global data exports, we can calculate the average number of data imports per data export before exhausting all the space available in the PFL object.

$$\text{PFL chain length} = 100 + 91 * (184 + 32 * 500)$$

$$= 1,472,844 \text{ bytes}$$

$$\begin{aligned} \text{PDD size} &= 96 + 16,000 + 10,000 * (84 + 24 * L) \\ &= 856,096 + 240,000 * L \end{aligned}$$

$$1,472,844 + (856,096 + 240,000 * L) \leq 20,971,520$$

$$L \leq 75 \text{ average data imports per data export}$$

From this calculation, the PFL contains a maximum of 91 executable files with 10,000 shared global data exports with an average of 75 data imports per data export. But, any combination of values for the listed items may be used as long as they satisfy the limits outlined above.

Label management:

##	Limit	Description	Resource/ Table
1	262144	entries in Label Table	Label Table

(1) Label table

There is one label table for each volume on the system. The Label table exists on disc and it is mapped into a virtual space when the volume is mounted. The label table contains for each file on a volume the LABEL itself and the extent descriptors of the file (each extent block describes up to 20 extents of the file and there is no theoretical limit for the number of extent descriptors of a file.)

The label table is managed by the Table Management. Each table entry is three sectors. With the introduction of Cascade Disk Array, the capacity of a disk volume was increased. That's why the maximum number of entries per label table was increased from 32,768 in MPE/iX 3.0 to 262,144 in MPE/iX 4.0. The table's maximum size is 201,326,592 bytes. The initial allocation of the table is 790,528 bytes and the increment size is 786,432 bytes.

Misc:

##	Limit	Description	Resource/ Table
1	16384	Timer Entries	Timer Table

(1) Timer table

Timer entries are used throughout the system to manage events. Jobs, sessions, file system, i/o and networking subsystems all generate timer requests. Additionally, users can utilize intrinsics such as FREAD for timed reads or PAUSE to request a timer entry.

The Timer Globals (KSO 80) contains a pointer to the timer table. The maximum number of timer entries is 16384.

Memory management:

##	Limit	Description	Resource/Table
1	13104	MIB entries in the MIB table	MIB
2	32768	IO notification entries	MM IO Notify Object
3	12288	IO request entries	MM IO Request Object
4	55000	Entries in process Locality List	MM LL Object
5	523600	Entries in VS Locality List table	MM VS LL Headers

(1) Mib - KSO204

The MIB table is a MEMORY Management data structure that contains one entry for each Memory Information Block on the system. A MIB is a global data structure used by the Memory Manager, the Virtual Space Management, and the I/O subsystem to handle requests for I/O. MIBs are allocated by VSM in response to Memory Manager requests to execute an I/O. Each of the three subsystems contains their own areas of the MIB for which they are responsible. MIB entries will be created at I/O request time and will be released when the I/O has completed. The MIB table contains 13104 MIB entries.

(2) MM IO notify object - KSO008

The MM IO Notification Table contains one entry per page for each io request on that page (a page can be requested by several processes at a time) The entry is added when a process needs to be notified of an I/O completion. It is added before the I/O is sent to HLIO. It is deleted when the process has been notified or when we determine that a process need not to be notified for a particular page.

(3) MM IO request object - KSO009

The I/O request table contains one entry for each I/O request on the system, The entry keeps track of the status of the request. The entries are added at the time the I/O request is sent out to HLIO. This table contains in general one entry per MIB chain. The entries are deleted when the processes have been notified. This table is different from the IO Notification table as far as the IO notification table has one entry per process that has requested an I/O (the same

page can be requested several times by different processes) whereas the I/O request table holds one entry per MIB chain (I/O request).

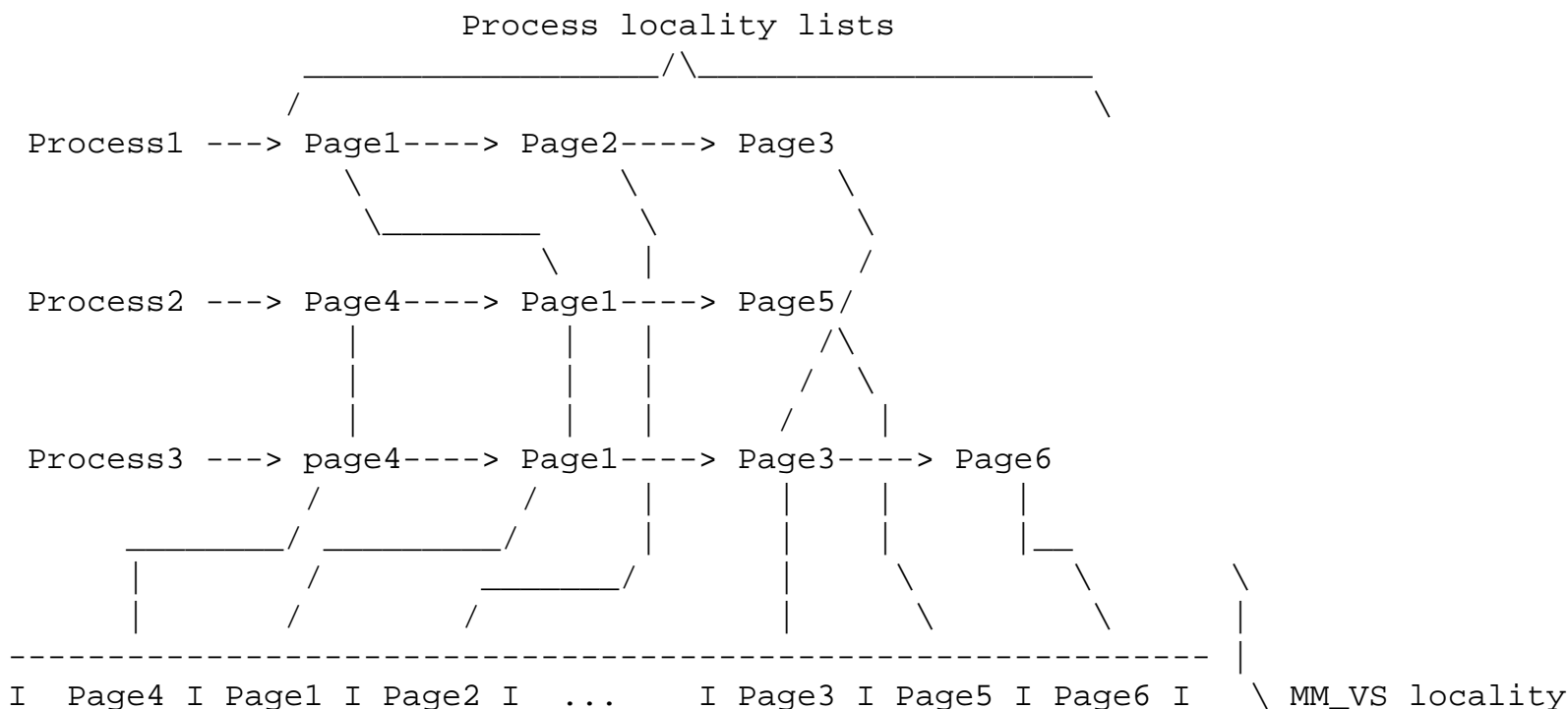
(4) MM II object - KSO010

When a process needs to have pages brought in in order to continue execution, these pages will be added to its locality list. Before a process is launched, MPE/iX scans the locality list, bring the pages in to memory and then allow the process to run. This will avoid the process to faulting on page it already faulted on recently. The entries in this table are added each time the process page faults on a virtual address or when a process prefetches an object. They are deleted when the pages have been brought into memory.

(5) MM VS II headers - KSO026

MPE/iX maintains the process locality lists in the MM locality table. These lists are deleted when a process is launched. But when a process releases a sharable object that is also linked into other processes locality lists, MPE/iX needs to remove from all these locality lists the entries that correspond to this object. This case is possible since an object may be in a process locality list as part of a prefetch request, but the pages have not yet been brought into memory, and the process does not need the object anymore or the process has died. In that case removing this object from all the process locality lists would mean scanning its locality list for each process on the system and removing this entry if it's there. As this would be very expensive, another table was created for Memory Management and Virtual Space Management. This table is the MM_VS_LL_HEADERS table. It consists of headers that point to a linked list of entries into the MM_LL_TABLE. As this table exists, releasing an object consists of just scanning the corresponding list of entries pointed to by the header entry and removing each entry from the locality list.

In this table at least one entry is needed per swappable object (including files), and up to three entries for transient objects with shared pages.



2	~ 9900	Spool Files, Linked Output	Output Spool File Dir
3	4679	Spooler Processes	Spooler Info Table

(1) Input spool file dir - KSO287

There is an entry in the Input SPFDIR table for each input spoolfile on the system. All input spoolfiles reside in the group IN.HPSPOOL, and have an entry in the Input SPFDIR.

Entries are added to the Input SPFDIR whenever an input spoolfile is created. This may be due to accessing an input spooled device or typing in the STREAM command.

Entries are deleted whenever access to the input spoolfile is complete or when the input spoolfile is deleted.

The Input SPFDIR limits the system to just over 9900 input spoolfiles.

NOTE: the actual number of input spoolfiles may be less than 9900 due to other system limitations. One of these is disc space. It is difficult to estimate the amount of disc space input spoolfiles will require. If the IN.HPSPOOL group or the HPSPOOL account has a filespace limit specified (via the :ALTGROUP or :ALTACCT command) then the number of input spoolfiles may be less than 9900. In addition the total amount of disc space available on your system may limit the number of input spoolfiles.

Another limitation is the number of jobs allowed on the system. There is one input spoolfile per job. There is a current limit of approximately 3500 JMAT entries, so this is a de facto upper limit to the number of input spoolfiles (if we ignore :DATA files). Note that although the JMAT is limited to 3500 entries, not all of these can be logged on concurrently, and some entries are probably devoted to sessions. This further limits the number of input spoolfiles on the system.

(2) Output spool file dir - KSO286

There is an entry in the Output SPFDIR table for each 'linked' output spoolfile on the system. You can think of a 'linked' spoolfile as one that will print automatically if the outfence is low enough and the printer is online and ready. A spoolfile will be linked automatically when a user accesses a spooled output device or issues the SPOOLF;PRINT command on an existing output spoolfile. Now that spoolfiles are just ordinary disc files (although they do have a special variable length record structure), they may reside in any group and account. However, to be 'linked', the spoolfile must reside in OUT.HPSPOOL and have an entry in the Output SPFDIR.

Entries are deleted from the Output SPFDIR when a user enters the SPOOLF; DELETE command or because all copies of the spoolfile were printed and the spoolfile was not flagged to be saved after printing.

The Output SPFDIR limits the system to just over 9900 'linked' output spoolfiles.

NOTE: the actual number of output spoolfiles may be less than 9900 due to disc space limitations. It is difficult to

estimate the amount of disc space output spoolfiles will require. If the OUT.HPSPOOL group or the HPSPOOL account has a limit specified (via the :ALTGROUP or :ALTACCT command) then the number of output spoolfiles may be less than 9900. In addition the total amount of disc space available on your system may limit the number of output spoolfiles.

(3) Spooler info table - KSO285

There is one Spooler Information Table (SPIT) entry allocated for each active spooler process (input or output).

The SPIT limits the number of spoolable devices to the maximum number of logical devices allowed on the system. For release 5.5, this is 4679. Note that the SPIT should never grow this large, as this number includes disks, terminals, and tape drives, none of which are spoolable (except tapes, which can be spooled for input).

The number was increased from 5.5 (fixed) so that large configurations on consolidated systems do not encounter it as a limit, nor does it require periodic maintenance to keep increasing this number as systems grow larger.

Volume management:

##	Limit	Description	Resource/Table
1	255	Mounted Disc Ldevs	Mounted Vol Table
2	255	Mounted Volume sets	Mounted Vol Set Table
3	127	Mirrored pair	Mirrored Volumes
1	127	Mirrored volume sets	Mounted Vol Table

(1) Mounted vol table - KSO223

The MVT is a permanent data structure in the Volume Management. It is created INSTALL time, and will be mapped in at START time. The MVT will be re-created each time the system is booted. There is one entry in the MVT per mounted volume on the system. A new entry is allocated in the MVT when a volume is initialized or mounted on the system. By default, the first entry is for system master volume. The max number of entries is 255, which means that the system can theoreticaly support 255 volumes.

(2) Mounted vol set table - KSO206

The MVST table is created at system boot time during the volume management initiation process. A new entry is added to the MVST when a master volume is initialized or mounted on the system. The entry will be added to the first free entry in the table, and all the fields in entry will be initialized with information about the mounted volume set on the system. The first entry is MPEXL_SYSTEM_VOLUME_SET. The max number of entries is 255, which means that the system can theoreticaly support 255 different volume sets.

(3) Mirrored volumes

The maximum number of DISC LDEVs supported is 255, that means that the maximum number of pairs of mirror disks will be 127 (LDEV 1 can't be mirrored). As a consequence the maximum disc space available on a system will be half of what is available without mirror disks. This because a pair of mirror disks uses two ldevs and has the storage of one.

NOTE: 127 is the absolute maximum number of mirrored pairs, but this leaves only ldev 1 as the entire system volume set. In most cases, this is not practical. All the necessary transient space for the system must come from the system volume set.

Virtual space management:

##	Limit	Description	Resource/Table
1	~1973784	Extents for Objects (Fixed Access Rights)	Extent B-Tree Table
2	64520	Extents for Objects (Var Access Rights)	Extent AR B-Tree Table
3	91180	File Objects	VSOD GUFD Table
4	490637	Objects Allowed (files + transient objs)	VSOD + VSOD GUFD Tables
5	310689	Pages Mapped in Memory (global A/R)	VPN Cache Table
6	310689	Pages Mapped in Memory (Variable A/R)	VPN AR Cache Table
1	246723	Swappable Objects (Fixed Access Rights)	Extent B-Tree Table
2	8065	Swappable Objects (Var Access Rights)	Extent AR B-Tree Table
7	399457	Transient Objects Allowed	VSOD Table
8	30840	VS B-tree Entries	VS B-Tree Table
9	80659	Virtual Space Domains	VS Domain Desc Table

(1) Extent b-tree table - KSO055

Each entry in the Extent B-tree is a node, which can describe up to eight extents. If an object contains more than eight extents, it will use additional entries (or nodes). Every non-resident object (with fixed access rights), will obtain one or more entries in the extent b-tree when it is mapped in (ie. opened). See the NOTE: below for a description of access rights.

This table limits the maximum number of open non-resident objects with fixed access rights to 246723. The total number may be less if one or more objects require more than one entry (or node) due to the fact that they have more than eight extents.

This table also limits the maximum number of extents for open non-resident objects with fixed access rights to 1973784. It is unlikely that this maximum number could be obtained since it would imply that each object has a multiple of eight extents and the tree is completely balanced and full. This number was obtained by multiplying the maximum number of entries (246723) by the maximum number of extents per entry (8).

NOTE: Objects with fixed access rights are defined as those for which ALL pages have identical read and/or write access as well as the same privilege level. Extent information about objects with fixed access rights is kept in the Extent B-tree (KSO 55). All but two types of objects have fixed access rights. However, NM program files and NM library files have variable access rights. Page 0 has write access in addition to the read/execute access that is given to the other pages of the file. Extent information about objects with variable access rights is kept in the Extent AR B-Tree (KSO 57).

(2) Extent ar b-tree table - KSO057

Each entry in the Extent AR b-tree is a node, which can describe up to 8 extents. If an object contains more than eight extents, it will use additional entries (or nodes). Every non-resident object (with variable access rights), will obtain one or more entries in the Extent AR B-tree when it is mapped in (ie. opened). See the NOTE: below which describes access rights.

This table limits the maximum number of open non-resident objects with variable access rights to 8065. The total number may be less if one or more objects require more than one entry (or node) due to the fact that they have more than eight extents. Since there are only two types of objects that have variable access rights, (NM program files and NM library files), it is very unlikely that this limit can be reached.

This table also limits the maximum number of extents for open non-resident objects with variable access rights to 64520. It is unlikely that this maximum number could be obtained since it would imply that each object has a multiple of eight extents and the tree is completely balanced and full. This number was obtained by multiplying the maximum number of entries (8065) by the maximum number of extents per entry (8).

NOTE: Objects with fixed access rights are defined as those for which ALL pages have identical read and/or write access as well as the same privilege level. Extent information about objects with fixed access rights is kept in the Extent B-tree (KSO 55). All but two types of objects have fixed access rights. However, NM program files and NM library files have variable access rights. Page 0 has write access in addition to the read/execute access that is given to the other pages of the file. Extent information about objects with variable access rights is kept in the Extent AR B-Tree (KSO 57).

(3) VSOD GUFDF table - KSO201

Every object in the system requires an entry in either the VSOD or VSOD GUFDF table (with the exception of certain resident objects that reside in the Disabled Expandable VSOD table, kso #255). All file objects obtain an entry in the VSOD GUFDF table when they are mapped in (or opened). When the file is mapped out, the entry is linked onto a list

known as the 'LRU' (or Least Recently Used) list. If the same file is opened again, the LRU list is searched to see if an entry already exists for that file. If so, we re-use the entry and save the overhead of mapping the file in again. If the entry is not on the LRU list, a new entry must be obtained. If there are no free entries in the table, an entry is obtained from the LRU list. The object previously associated with that LRU list is now considered to be mapped out.

This table limits the number of open file objects to 91180. Each entry in the VSOD GUFDF table will have a pointer to one node in the Extent B-tree or Extent AR B-tree table. As discussed in the Extent B-tree documentation, one or more nodes of the B-tree may be used for any given object, depending on the number of extents that have been allocated for it.

(4) VSOD + VSOD GUFDF tables

The VSOD contains one entry for every transient object on the system. Transient objects are defined as objects that are required during the life of a process, that have disc space allocated on their behalf, and are released when the process terminates. The VSOD/GUFDF table contains one entry for every permanent object (ie. file) that is mapped in (opened) or that has recently been mapped in (on the LRU list). Refer to the discussions on the VSOD and VSOD/GUFDF tables for more details.

The total number of objects that can be mapped in at any given time is 490637, which is the sum of the maximum number of objects that can be defined in the VSOD and VSOD/GUFDF.

(5) VPN cache table - KSO049

Each VPN cache entry corresponds to pages of an extent that are currently in memory (or ready to be mapped in/out of memory), hence the VPN cache table indirectly related to the memory management's PDIR table and the VSM extend B tree table. Entries are "added" (locked in) when pages of an extent are mapped in memory and they are "deleted" (unlocked) when pages of an extent are mapped out of memory. The number of entries per open file depend on the pattern of access to the file. The VPN cache table size does not directly limit the number of open files, but does limit the total number of pages from all these file that can be simultaneously present in memory.

(6) VPN ar cache table - KSO051

As VPN CACHE table keeps track of pages that are currently in memory or in motion in for the pages with fixed access rights. The VPN AR CACHE TABLE provides the same function but for pages with variable access rights pages. The two tables have the same size and are managed the same way.

(7) VSOD table - KSO053

Every object in the system requires an entry in either the VSOD or VSOD GUFDF table (with the exception of certain resident objects that reside in the Disabled Expandable VSOD table, kso #255). All non file transient objects obtain an entry in the VSOD table when they are mapped in (or created). When the object is mapped out, the entry is released. There is no LRU list for the VSOD, as there is for the the VSOD GUFDF.

This table limits the number of transient objects to 399457. Each entry in the VSOD table, representing a non-resident object, will have a pointer to one node in the Extent B-tree or Extent AR B-tree table. As discussed in the Extent B-tree documentation, one or more nodes of the B-tree may be used for any given object, depending on the number of extents that have been allocated for it.

(8) VS b-tree table - KSO047

Every object in the system requires an entry in VS B-tree. Each entry (or node) of this b-tree can describe up to 32 different objects. Each object described by this b-tree will have a corresponding entry in the VSOD, VSOD GUF, or Disabled Expandable VSOD table. When the object is mapped out, the entry is released.

This table contains 30840 entries, therefore, if it was completely full, it would limit the total number of open objects on a system to 986880. This limit is obtained by multiplying the the number of entries (30840) by the maximum number of entries per node (32).

(9) VS domain desc table - KSO061

VS domain descriptor table contains one entry per process or job on the system. Entries are "added" when a new virtual space domain is needed, such as when a job or process is created. Theoreticaly, this table will never contain more entries that MAX number of process on system (5460) + Max number of job or session on system (1700) + 1 (system domain) = 7161 entries.

Transaction management:

##	Limit	Description	Resource/Table
1	10	XM control block locks per transaction	XM Locks
2	4096	Open files during recovery	Open Files During Recovery
3	4096	Open transactions during recovery	Open Transactions During Recovery
4	~48000	Copy forward user transaction per log	XM Checkpoint Mechanism
5	255	System logs	XM Logfile/Volset Restrictions

(1) XM locks

When a XM transaction is started a control block lock is done on the XM protected object. Currently the number of XM cb_locks per transaction is limited to 10.

(2) Open files during recovery

At the recovery time, the maximum number of files that can be opened is set to 4096.

(3) Open transactions during recovery

The maximum number of transactions that can be open at any one point during recovery is 4096.

(4) XM checkpoint mechanism

A transaction is considered to be long if it spans two checkpoints (how long it takes depends on how fast the log fills up). If we do not do something about long transactions, we will run out of log file space. Because of recycling of log space may overwrite log records for a long running transaction. It will cause system abort 2200 for system log. For user log, the solution is to "copy forward" log records of the long running transactions at wrap log time.

At checkpoint time, all active transactions are examined. Transactions spanning one checkpoint will be included in the checkpoint record; transactions spanning two checkpoints will be copied forward. If a long transaction exists, a temporary object will be created to hold the physical address of all chained log records for long transactions. The physical log address list will be sorted and then be moved to the next half of log file in ascending order. This algorithm reduces the size of these running transactions, consequently reduces the stalled transaction risk. The number of copy forward user transaction records per log file has been estimated to 48000.

The user process will be aborted if one of the three limits is reached: 1. the maximum number of time a transaction was copied forward gets bigger than 100. 2. the maximum size of a transaction gets bigger than 4 Mbytes. 3. the size of all the user transactions gets bigger than $\log_file_size/2$.

(5) XM logfile/Volset restrictions

As there is one XM system log file per volumeset, and the number of volumesets per system is limited to 255, the maximum number of XM system logfile is 255.