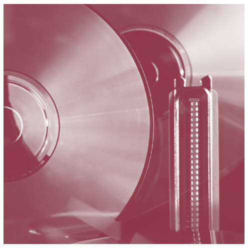
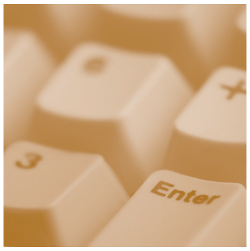
ASG-Manager Products™
Performance Tuning

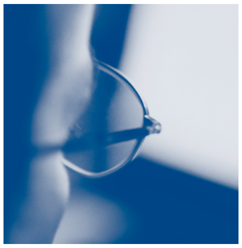
Version: 2.5

Publication Number: MPR1400-25-TUNE

Publication Date: August 1991







ASG Worldwide Headquarters Naples, Florida USA | asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 941.435.2200 Fax: 941.263.3692 Toll Free: 1.800.932.5536

ASG Documentation/Product Enhancement Fax Form

Please FAX comments regarding ASG products and/or documentation to (941) 263-3692.

Company Name	Telephone Number	Site ID	Contact name

Product Name/Publication	Version #	Publication Date
Product:		
Publication:		
Tape VOLSER:		

Enhancement Request:

ASG Support Numbers

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours. To expedite response time, please follow these procedures.

Please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely or displayed
- A description of the specific steps that immediately preceded the problem
- The severity code (ASG Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)

If You Receive a Voice Mail Message:

1. Follow the instructions to report a production-down or critical problem.
2. Leave a detailed message including your name and phone number. A Support representative will be paged and will return your call as soon as possible.
3. Please have the information described above ready for when you are contacted by the Support representative.

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Business Hours Support

Your Location	Phone	Fax	E-mail
United States and Canada	800.354.3578 1.941.435.2201 **Secondary Numbers:** 800.227.7774 800.525.7775	941.263.2883	support@asg.com
Australia	61.2.9460.0411	61.2.9460.0280	support.au@asg.com
England	44.1727.736305	44.1727.812018	support.uk@asg.com
France	33.141.028590	33.141.028589	support.fr@asg.com
Germany	49.89.45716.300	49.89.45716.400	support.de@asg.com
Singapore	65.224.3080	65.224.8516	support.sg@asg.com
All other countries:	1.941.435.2201		support@asg.com

Non-Business Hours - Emergency Support

Your Location	Phone	Your Location	Phone
United States and Canada	800.354.3578 1.941.435.2201 **Secondary Numbers:** 800.227.7774 800.525.7775 **Fax:** 941.263.2883		
Asia	011.65.224.3080	**Japan/Telecom**	0041.800.9932.5536
Australia	0011.800.9932.5536	**New Zealand**	00.800.9932.5536
Denmark	00.800.9932.5536	**South Korea**	001.800.9932.5536
France	00.800.9932.5536	**Sweden/Telia**	009.800.9932.5536
Germany	00.800.9932.5536	**Switzerland**	00.800.9932.5536
Hong Kong	001.800.9932.5536	**Thailand**	001.800.9932.5536
Ireland	00.800.9932.5536	**United Kingdom**	00.800.9932.5536
Israel/Bezeq	014.800.9932.5536		
Japan/IDC	0061.800.9932.5536	**All other countries**	1.941.435.2201

ASG Web Site

Visit http://www.asg.com, ASG's World Wide Web site.

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/products/suggestions.asp

If you do not have access to the web, FAX your suggestions to product management at (941) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

Contents

Preface

This *ASG-Manager Products Performance Tuning* guide provides suggestions for tuning ASG-Manager Products (herein called Manager Products).

Allen Systems Group, Inc. (ASG) provides professional support to resolve any questions or concerns regarding the installation or use of any ASG product. Telephone technical support is available around the world, 24 hours a day, 7 days a week.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on any ASG product.

About this Publication

This publication consists of these chapters:

- Chapter 1, "Introduction," gives an overview of performance tuning.
- Chapter 2, "Optimizing the Structure of Manager Products Datasets," gives guidance on choosing physical and logical blocksizes.
- Chapter 3, "Reducing Dataset I/O Activity," describes how to reduce dataset I/O activity.
- Chapter 4, "Reducing Resource Contention," describes how to reduce resource contention.
- Chapter 5, "Reducing Real/Virtual Storage Needs," describes how to reduce real/virtual storage requirements.
- Chapter 6, "Reducing CPU Usage," describes how to reduce CPU usage.
- Chapter 7, "Mechanisms for Limiting Resource Usage," describes how to limit use of resources.
- Chapter 8, "Facilities to Use Selectively," lists facilities that you should use selectively.

Publication Conventions

Allen Systems Group, Inc. uses these conventions in technical publications:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
`lowercase italic monospace`	Information that you provide according to your particular situation. For example, you would replace `filename` with the actual name of the file.
`Monospace`	Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.

The following conventions apply to syntax diagrams that appear in this publication.

Diagrams are read from left to right along a continuous line (the "main path"). Keywords and variables appear on, above, or below the main path.

Convention	Represents
➤➤	at the beginning of a line indicates the start of a statement.
➤◄	at the end of a line indicates the end of a statement.
————➤	at the end of a line indicates that the statement continues on the line below.
➤————	at the beginning of a line indicates that the statement continues from the line above.

Keywords are in upper-case characters. Keywords and any required punctuation characters or symbols are highlighted. Permitted truncations are not indicated.

Variables are in lower-case characters.

Statement identifiers appear on the main path of the diagram:

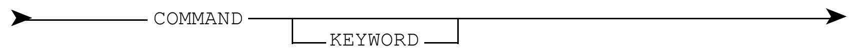
```
➤────────COMMAND──────────────────────────────────────────────────➤
```

A required keyword appears on the main path:

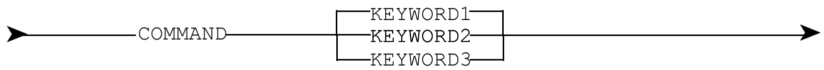
```
➤────────── COMMAND ──── KEYWORD ─────────────────────────────────➤
```

An optional keyword appears below the main path:

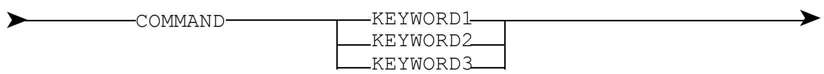
Convention	Represents



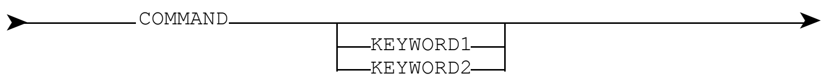
Where there is a choice of required keywords, the keywords appear in a vertical list; one of them is on the main path:



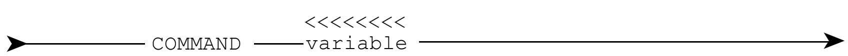
or



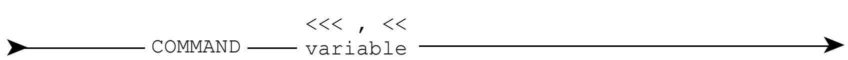
Where there is a choice of optional keywords, the keywords appear in a vertical list, below the main path:



The repeat symbol, <<<<<<, above a keyword or variable, or above a whole clause, indicates that the keyword, variable, or clause may be specified more than once:

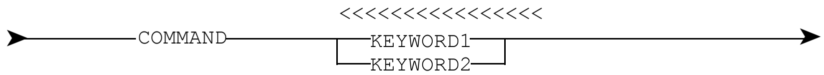


A repeat symbol broken by a comma indicates that if the keyword, variable, or clause is specified more than once, a comma must separate each instance of the keyword, variable, or clause:

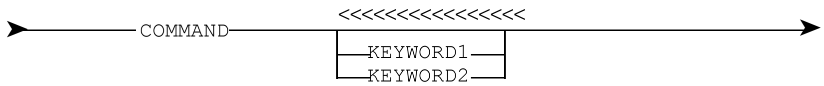


The repeat symbol above a list of keywords (one of which appears on the main path) indicates that any one or more of the keywords may be specified; at least one must be specified:

Convention	Represents



The repeat symbol above a list of keywords (all of which are below the main path) indicates that any one or more of the keywords maybe specified, but they are all optional:



1 Introduction

This manual assists systems administrators and controllers to improve the efficiency of their Manager Products environment. This process of improving efficiency is known as tuning. In this context, efficiency means obtaining results by the use of fewer computer resources.

You can reduce the consumption of computer resources by:

- Optimizing the logical and physical blocksizes of the Manager Products datasets
- Minimizing the number of input/output (I/O) operations
- Eliminating or reducing contention for Manager Products resources
- Reducing each user's real and virtual storage requirements
- Reducing each user's CPU usage
- Imposing limits on consumption of resources
- Avoiding excessive use of facilities intended primarily for diagnostic or monitoring purposes.

MP-AID and repository physical dataset I/O has a major impact on the performance of your Manager Products environment This is because, for every physical I/O performed:

- Around 6,000 operating system instructions are executed
- The Manager Products task has to wait for the I/O to finish.

Thus physical I/O:

- Consumes CPU time
- Increases elapsed time.

You should therefore reduce physical I/Os as much as possible. The main ways of doing this are by:

- Optimizing dataset blocksizes
- Allocating large buffer pools
- Regularly reorganizing repositories and the MP-AID.

Reducing physical I/O is by far the most important area in tuning Manager Products. You should fully exploit this before looking at other areas of tuning.

The other areas of tuning are of less general use. They will produce significant benefits only for certain installations, repositories or tasks. You can experiment and see which are of benefit to you.

Tuning Manager Products is all about trade-offs; you optimize use of one resource and so usually use another resource less efficiently. For example, consider choice of logical blocksizes. You can choose to do any one of the following:

- Calculate optimum logical blocksizes, which balances used disk space against performance
- Set logical blocksizes to the minimum permitted values, which minimizes used disk space, but may increase processing time
- Set larger logical blocksizes to optimal performance, which may result in inefficient use of disk space.

You must decide which of these alternatives best suits your installation.

The benefits of tuning can be substantial. For example, if at present you use the default buffer pool allocations, you can reduce I/Os for many tasks by at least 75%, by allocating large buffer pools.

We divide the facilities described in this manual into seven groups, according to the main benefit that each facility gives. A facility may give more than one benefit. For example, logical units of work (LUWs) are in Chapter 6, "Reducing CPU Usage," on page 31 since that is their main benefit. However, when used with large buffer pools, LUWs can also greatly reduce physical I/O.

In the benchmarks we normally give three measures of performance:

- CPU time
- Elapsed time
- Total physical I/O.

Where we give elapsed time in a benchmark, the jobs were run on an empty machine. So it is meaningful to compare the elapsed times.

2 Optimizing the Structure of Manager Products Datasets

Repository and MP-AID Datasets

Introduction

Manager Products repositories and MP-AIDs consist of datasets held on disk. An MP-AID consists of a single dataset. A repository consists of five datasets (four if logging is not implemented).

Each dataset consists of a number of fixed length physical blocks. Each physical block consists of a fixed number of logical blocks. In the Index, Source, and Data Entries repository datasets and the MP-AID dataset each record occupies one or more logical blocks. In the Recovery and Log datasets the logical blocksize is the same as the physical blocksize.

You can optimize the logical blocksizes of the following datasets:

- Source
- Data Entries
- MP-AID.

By defining small logical blocksizes that best fit the majority of your data, you leave less space unused in each physical block and the dataset space allocation can probably be reduced. This reduction in unused space means that extra, potentially useful, data is available when a record is read from disk.

You can use the ANALYZE and MP-AID ANALYZE commands to review your logical block utilization. You can assess the block usage of the relevant records and, thereby, the optimum logical blocksize.

One use of the MP-AID ANALYZE command is to examine how suitable a proposed blocksize is.

In some datasets it will be easy to identify all optimum record size where it is not easy to do this, we recommend that you take the most frequently occurring smallest record size as the optimum record size.

If one or more of the following applies:

- You do not wish to calculate optimum logical blocksizes
- You are setting up a repository or MP-AID
- You are more concerned about disk space allocations than about performance

then we recommend that you set your logical blocksizes to the minimum permitted values. Sometimes, however, this will not be possible. For example, if you create a repository with a very large number of statuses, the required Internal Pointer Table (IPT) size will be larger than the minimum permitted Data Entries logical blocksize.

Physical Blocksizes

Optimal physical blocksizes depend on your underlying disk architecture, particularly on the track capacity of the storage device in use. Performance improves as you choose larger optimal physical blocksizes. A physical blocksize is optimal when it maximizes track utilization.

MVS and VSE Environments

We recommend the following:

- A 3380 or 3390 half-track, or equivalent, blocksize for the Index, Source, Data Entries, and MP-AID datasets
- A blocksize of 4K for the Recovery and Log datasets.

A half-track physical blocksize for the MP-AID dataset is particularly important if InfoBank is frequently used, or if the MP-AID will contain large amounts of frequently accessed user data.

Selecting the Access Method

You can create and access a repository or MP-AID using either VSAM or BDAM (DAM in VSE) as the access method. Some installations may have a standard preferred access method, in which case you cannot choose.

However, if you do have the freedom to choose, note the following points:

- Opening a VSAM dataset tends to be slower than opening a BDAM dataset
- For normal Manager Products processing VSAM uses about 3% less CPU time.

Part of the explanation for the difference in CPU time is as follows. In MVS/XA and ESA environments, where buffer pools are allocated from storage above the 16MB line, Manager Products can interface more efficiently with the VSAM code by passing 31-bit parameters. However, for BDAM access only 24-bit parameters can be passed. This means that for BDAM the following additional processing takes place:

- After a physical read the record must be copied to the buffer pool from the I/O buffer
- Before a physical write the record must be copied from the buffer pool to the I/O buffer.

We recommend that you use VSAM datasets where possible.

VSAM Datasets

For a VSAM dataset, the RECORDSIZE (as specified in DEFINE CLUSTER statements) is the equivalent of a BDAM physical blocksize. DEFINE CLUSTER statements are used when allocating the dataset with the VSAM utility IDCAMS. The specified RECORDSIZE is always seven bytes less than the specified CONTROLINTERVALSIZE. The optimal RECORDSIZE and CONTROLINTERVALSIZE are as follows.

	3380 Storage Device		3390 Storage Device	
Blocking	**REC-SIZE**	**CI-SIZE**	**REC-SIZE**	**CI-SIZE**
Half-track	22521	22528	26617	26624
Third-track	14329	14336	18425	18432
Quarter-track	10233	10240	12281	12288
Fifth-track	8185	8192	10233	10240

BDAM Datasets

The optimal physical blocksizes for 3380 and 3390 storage devices are as follows.

Blocking	3380 Storage Device	3390 Storage Device
Half-track	23476	27998
Third-track	15476	18452
Quarter-track	11476	13682
Fifth-track	9076	10796

CMS Environments

The minidisk that will contain the repository and/or MP-AID datasets should be formatted with a CMS blocksize of 4K. Physical blocksizes should be equal to, or a multiple of, 4K. We recommend the following:

- A blocksize of 20K for the Index, Source, Data Entries, and MP-AID datasets
- A blocksize of 4K for the Recovery and Log datasets.

Siemens Environments

Physical blocksizes should be a multiple of 2K. We recommend the following:

- A blocksize of 20K for the Index, Source, Data Entries, and MP-AID datasets
- A blocksize of 4K for the Recovery and Log datasets.

Logical Blocksizes

Index Dataset

We recommend that you choose a logical blocksize that is:

- Approximately 2K
- Leaves the minimum of space unused per physical block.

For example, assuming a 3380 storage device and half-track (23476 bytes) physical blocksize, a suitable logical blocksize would be 2134 bytes. (2134 x 11 =23474).

Source Dataset

Source records occupy one or more logical blocks. Calculate the optimum logical blocksize as follows:

1. Assess the optimum record size. Identify a size that will allow the majority of records, or those most frequently used, to be held in one logical block with the minimum of unused space. This size can be assessed using the output from the ANALYZE command.

2. Calculate the optimum logical blocksize. The optimum logical blocksize is the same as the optimum record size.

3. Round up the optimum logical blocksize. Round up the optimum logical blocksize so that the minimum of space is unused per physical block.

Data Entries Dataset

It is particularly important that you calculate the optimum logical blocksize for the Data Entries dataset, as it is the most frequently accessed dataset. The Data Entries dataset contains:

- An Internal Pointer Table (IPT) for every member on the repository
- Encoded records, which occupy one or more logical blocks.

IPTs contain:

- Index pointers
- Source pointers
- Encoded record pointers
- Member type details and other control information.

A member’s IPT is accessed every time the member itself is accessed. So you want access to IPTs to be as efficient as possible.

Calculate the optimum logical blocksize as follows:

1. Assess the optimum record size. Identify a size that will allow the majority of encoded records, or those most frequently used, to be held in one logical block with the minimum of unused space. This size can be assessed using the output from the ANALYZE command.

   For example, if the current logical blocksize is 803 bytes and the most frequently occurring smallest record size is 1 block + 13% of a second block, then the optimum record size is:

   113% x 803 = 907 bytes

2. Calculate the size of Internal Pointer Tables (IPTs).

   IPT size = 4 + (16 x number of statuses)

   For example, if there are 10 statuses, then:

   IPT size = 4 + (16 x 10) = 164

3. Calculate the optimum logical blocksize.

   Choose the lowest integer *n* such that:

   (*n* x IPT size) + 25 >= optimum record size

   Then, the optimum logical blocksize is:

   (*n* x IPT size) + 25

   In our example:

   (6 x 164) + 25 = 1009 >= 907

   So the optimum logical blocksize is 1009 bytes.

4. Round up the optimum logical blocksize. Round up the optimum logical blocksize so that the minimum of space is unused per physical block.

   For example, using a 3380 storage device and an optimum logical blocksize of 1009 bytes, you can store 23 logical blocks in one physical block using half-track (23476 bytes) physical blocking. However, that would leave 269 bytes unused per physical block. So, round up the logical blocksize to 1020 bytes, leaving only 16 bytes unused per physical block.

MP-AID Dataset

We recommend a logical blocksize of 1024 bytes when you create an MP-AID and load the ASG-supplied members.

The MP-AID contains:

- A directory for each MP-AID member type
- Records, occupying one or more logical blocks, for each member.

Directory blocks consist of a number of MP-AID Directory Entries (MPDEs). MPDEs contain the member name, a pointer to the member’s data, and other information related to the member.

Calculate the optimum logical blocksize as follows:

1. Assess the optimum record size. Identify a size that will allow the majority of records, or those most frequently used, to be held in one logical block with the minimum of unused space. This size can be assessed using the output from the MP-AID ANALYZE command.

   See "Example: MP-AID ANALYZE Command" on page 9 for an example of how to interpret the output from the MP-AID ANALYZE command.

2. MP-AID Directory Entries (MPDEs). Each MPDE occupies 46 bytes.

3. Calculate the optimum logical blocksize. Choose the lowest integer *n* such that:

   (*n* x MPDE size) + 16 >= optimum record size

   Then the optimum logical blocksize is:

   (*n* x MPDE size) + 16

   For example, if the optimum record size is 600, then the optimum logical blocksize is:

   (13 x 46) + 16 = 614

4. Round up the optimum logical blocksize. Round up the optimum logical blocksize so that the minimum of space is unused per physical block.

   For example, using a 3380 storage device and an optimum logical blocksize of 614 bytes, you can store 38 logical blocks in one physical block using half-track (23476 bytes) physical blocking. However, that would leave 144 bytes unused per physical block. So, round up the logical blocksize to 617 bytes, leaving only 30 bytes unused per physical block.

Recovery Dataset

In the Recovery dataset physical and logical blocksizes are equal. The minimum permitted size is the maximum of the logical blocksizes of the Index, Source and Data Entries datasets (this value is the default if no physical blocksize is specified). We recommend an initial blocksize of 4K.

Log Dataset

In the log dataset physical and logical blocksizes are equal. We recommend an initial blocksize of 4K.

Benchmark Logical Blocksizes

Case 1. We restored the User Interface repository (dataset MP.UIDICT on the ASG Installation Tape) using Source and Data Entries logical blocksizes of 1067 bytes. (In the past a user would typically guess a logical blocksize of about 1K.) The following is an extract from the output of a QUERY DICTIONARY command issued after the RESTORE command completed.

Dataset Identification	Index	Source	Data	Rcvry	Log
Logical Blocksize	2134	1067	1067	4096	4096
Physical Blocksize	23476	23476	23476	4096	4096
Used Logical Blocks	91	7971	10993	300	1500
Free Logical Blocks	1229	11829	8807	0	0

Case 2. We calculated:

- The Source dataset optimum blocksize to be 300 ("Source Dataset" on page 6)
- The Data Entries dataset optimum blocksize to be 426 ("Data Entries Dataset" on page 6).

Then using the original space allocations, we recreated the repository using the optimal blocksizes and repeated the RESTORE command. The following is an extract from a QUERY DICTIONARY command issued after the RESTORE command completed.

Dataset Identification	Source	Data
Logical Blocksize	300	426
Physical Blocksize	23476	23476
Used Logical Blocks	9506	16366
Free Logical Blocks	60694	33134

Summary. The used logical blocks in the Data Entries dataset now occupy 6809K instead of 11455K, a reduction of 41 percent.

The used logical blocks in the Source dataset now occupy 2784K instead of 8306K, a reduction of 66 percent.

Example: MP-AID ANALYZE Command

This is an example of how to assess the optimum record size of the MP-AID dataset. Figure 1 on page 10 is an extract from the output of an MP-AID ANALYZE command.

```
DATA BLOCK USAGE ANALYSIS
-----------------------------------------------------------------------------------
                            MP-AID MEMBER LENGTH DISTRIBUTION (PERCENTAGE)
MEMBER TYPE             0-600         601-841        842-1082       1083-1323
------------------------------------------------------------------------------------
UDS TABLES             1   (1)         8   (7)        11   (9)          5 (4)
INFOBANK PANELS     1250  (17)      1711  (24)       917  (13)        606 (8)
PROFILE MEMBERS      123  (98)         3   C2)         0   (0)          0 (0)
USER MEMBERS        1985  (68)       182   (6)       140   (5)         93 (3)
UDO FORMATS           55  (87)         1   (2)         3   (5)          1 (2)
EXECUTIVE MEMBERS    503  (25)       262  (13)       336  (17)        164 (8)
TRANSLATION RULES     78  (58)        12   (9)         8   (6)          4 (3)
KEPT-DATA LISTS       45  (90)         3   (6)         0   (0)          0 (0)
COMMAND MEMBERS      342  (27)       123  (10)       103   (8)         67 (5)
WORKBENCH MEMBERS      0   (0)         0   (0)         0   (0)          0 (0)
TRANSIENT MEMBERS      1  (50)         0   (0)         0   (0)          0 (0)
------------------------------------------------------------------------------------
ALL MEMBERS         4383  (31)      2305  (17)      1518  (11)        940 (7)
```

Figure 1. Example output from ANALYZE command

First consider all members 59% (31% + 17% + 11%) of members have record lengths of less than 1082 bytes. So you may choose an optimum record length of 1082.

Alternatively, you might be interested only in certain frequently-used member types. For example, if USER MEMBERS were used much more than any other MP-AID member type, you might choose an optimum record length of 60 bytes, since 68% of USER MEMBERS have record lengths of 600 bytes or less.

Allocation of Statuses

When you create a repository, consider how many statuses you need. Create the repository with the number of statuses that you need for your immediate and short/medium term usage. Only name statuses when you start to use them, since all named statuses are examined as part of repository processing, even if they are empty.

If any statuses contain small amounts of data, you should consider merging them with another status. Reduce the number of statuses to those that are essential.

If you still have a number of unused or unnamed statuses, you should consider recreating the repository with fewer statuses. If you do this, you should calculate and use the Data Entries optimum logical blocksize (see "Data Entries Dataset" on page 6).

Refer to the publication *ASG-Manager Products Status Concepts* for further details on statuses.

3 Reducing Dataset I/O Activity

Allocation of Large Buffer Pools

Introduction

A buffer pool is allocated for each of the following datasets:

Index
Source
Data Entries
MP-AID.

Each buffer pool consists of two or more buffers. Each buffer provides an I/O area into which physical records are read and possibly updated.

When a particular record, requested by a Manager Products component, is not available from the appropriate buffer pool, a physical I/O is performed to read a physical record from disk.

During the execution of a Manager Products command, a particular record may be referenced or updated several times. If the required record is available from the buffer pool, no physical I/O is necessary. Allocation of a large buffer pool significantly increases the likelihood of a record remaining available in the buffer pool for the rest of the current command.

The Manager Products buffer pool manager uses a least recently used (LRU) buffer replacement strategy, ensuring that the most frequently referenced records remain available.

When a user's repository or MP-AID command starts, a check is made to see if the repository or MP-AID has been updated by another user since the end of the user's previous command. If the repository or MP-AID has been updated, the appropriate buffer pool(s) are invalidated. Buffer pools are not invalidated for commands within logical units of work (LUWs).

You can monitor the effectiveness of the currently allocated buffer pools by using the QUERY DICTIONARY BUFFERS command or the MP-AID BUFFERS command. The output of the MP-AID BUFFERS command is similar to that of the QUERY DICTIONARY BUFFERS command.

There is a permanent record of I/Os for the Index, Source, Data Entries and Recovery datasets in the log (if there is one), which may be examined using the AUDIT command.

Refer to the *ASG-Manager Products Controller's Manual* for further details of the AUDIT command.

Normal Use

For normal use, you should specify global buffer pool sizes via the DCUST macro. Specify sizes on the high side rather than the low, since any additional paging performed by the operating system is more efficient than physical I/O performed by Manager Products.

If you are using half-track physical blocksizes, we recommend the following minimum buffer pool sizes:

MP-AID	20
Index	20
Data Entries	50
Source	10

In MVS/XA and MVS/ESA environments, where the buffer pools are allocated in 31-bit storage, the allocation of very large buffer pools is possible, and you should consider taking advantage of this.

By comparison with the Index and Data Entries datasets, the Source dataset is not frequently accessed. So it is not normally worth allocating a large Source dataset buffer pool.

Refer to your installation manual for further details of the DCUST macro.

Special Use

For commands that cause very high I/O activity, for example a RESTORE ALL or MP-AID LOAD command, additional buffers can be specified using the DICTIONARY command or the MP-AID BUFFERS command. This will often save a large number of physical I/Os.

However, use minimum buffer pool allocations when performing a repository UNLOAD, since access to the repository is strictly sequential, and blocks retained in the buffer pools are never accessed a second time.

CICS Environments

When you execute Manager Products under the CICS Teleprocessing monitor, you are likely, if you allocate large buffers, to degrade performance for all users within the CICS region/partition. We recommend that you use the ASG default buffer allocations.

Benchmark 2: Effects of Increased Buffer Allocations

We ran the following commands on the User Interface repository with three different buffer allocations.

```
KEEP WHAT FORMS 'PRODUCE';
LIST KEPT;
LIST CATALOG;
LIST ALIAS;
LIST GROUPS;
LIST HISTORY KEPT;
KEEP WHAT CONSTITUTES DPI;
LIST KEPT;
BULK REPORT KEPT;
PROD COBDL FROM DPI USE HELD-AS NOGEN;
PROD RECORD FROM DPR USE HELD-AS NOGEN;
WHICH KEPT HAVE NOTE LINE 1 INCLUDE 'HEADING';
ANALYSE;
SAVE ALL NOPRINT;
```

Case 1. CPU time for default buffering was 1 minute 14.91 seconds. The buffer usage statistics were as follows:

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	4	2	4	10
Maximum Buffers Used	4	2	4	10
Buffer Pool Size (K)	92	46	92	230
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	95%	97%	90%	92%
Dataset Logical Reads	2896	21130	77204	101230
Dataset Physical Read	150	440	7741	8331
Dataset Logical Writes	0	0	0	0
Dataset Physical Writes	0	0	0	0

Case 2. CPU time for moderate buffering was 1 minute 10.17 seconds. The buffer usage statistics were as follows:

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	8	4	8	20
Maximum Buffers Used	8	4	8	20
Buffer Pool Size (K)	184	92	184	460

Dataset Identification	Index	Source	Data	Totals
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	98%	98%	95%	95%
Dataset Logical Reads	2987	21146	80849	104982
Dataset Physical Read	59	424	4096	4579
Dataset Logical Writes	0	0	0	0
Dataset Physical Writes	0	0	0	0

Case 3. CPU time for high buffering was 1 minute 11.46 seconds. The buffer usage statistics were as follows:

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	50	10	150	210
Maximum Buffers Used	9	10	150	169
Buffer Pool Size (K)	1147	230	3439	4816
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	99%	98%	99%	98%
Dataset Logical Reads	3020	21196	84140	108356
Dataset Physical Read	26	374	805	1205
Dataset Logical Writes	0	0	0	0
Dataset Physical Writes	0	0	0	0

Overall, CPU time hardly varies. Total I/Os fall from 8331 to 1205, a reduction of 86%. Even with a small increase in buffering you get a 45% reduction in I/O.

Repository Reorganization

Introduction

You should regularly save and restore the repository because:

- The RESTORE command optimizes access to the most frequently used internal tables and pointers.
- The RESTORE command reorganizes the placement of repository records on disk. This improves the performance of some commands.
- A saved copy of the repository provides additional security if UNLOAD tapes become corrupted, lost or overwritten.

If you are adding lots of new members to a repository, we recommend that you save and restore it every month. The cost of the save and restore is likely to be outweighed by improved performance.

Refer to the *ASG-Manager Products Controllers Manual* for further details of the SAVE and RESTORE commands.

Benchmark 3: Repository Reorganization

We chose a repository that frequently has new members added to it, and which had not been restored for over a year. We ran the following commands before and after the restore:

```
LIST MODULES;
LIST ALIASES;
LIST CATALOGS;
KEEP WHAT FORMS 'MODE-31';
BULK REPORT KEPT;
WHICH KEPT HAVE DESCRIPTION LINE 1 INCL 'CONTROL';
KEEP WHAT DIRECTLY USES DCADMZ01;
LIST KEPT;
```

Case 1. CPU time before the restore was 1.36 minutes. The buffer usage statistics were as follows:

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	20	10	50	80
Maximum Buffers Used	20	10	50	80
Buffer Pool Size (K)	400	200	1000	1600
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	78%	74%	84%	83%
Dataset Logical Reads	12504	2045	57687	72236

Dataset Identification	Index	Source	Data	Totals
Dataset Physical Read	3471	704	10600	14775
Dataset Logical Writes	0	0	0	0
Dataset Physical Writes	0	0	0	0

Case 2. CPU time after the restore was 1.02 minutes. The buffer usage statistics were as follows:

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	20	10	50	80
Maximum Buffers Used	20	10	50	80
Buffer Pool Size (K)	400	200	1000	1600
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	97%	86%	95%	95%
Dataset Logical Reads	14111	2385	65397	81893
Dataset Physical Read	419	364	2879	3662
Dataset Logical Writes	0	0	0	0
Dataset Physical Writes	0	0	0	0

Summary

Case	CPU (Mins)	Elapsed (Mins)	Physical I/Os
Pre RESTORE	1.36	10.20	14775
Post RESTORE	1.02	3.38	3662

Total physical I/O fell from 14775 to 3662, a reduction of 75%.

Recovery Dataset

The Recovery dataset uses a single I/O buffer, equal in size to the physical blocksize of the dataset.

The number of physical I/Os performed on the Recovery dataset is determined by two factors:

- The number of records on the Index, Source and Data Entries datasets actually updated. (The pre-update image of every updated record is written to the Recovery dataset, so that if a failure occurs, the repository can be recovered to a consistent state. When the Recovery dataset buffer is full, a physical I/O is performed.)
- The frequency with which updated records in the Index, Source and Data Entries buffer pools are written back to disk. Updated records are only written to disk, during a repository update, when a modified record is displaced by a new record. In this situation, unwritten data in the Recovery dataset buffer (often not a full buffer's worth) is first written to disk. Increasing the buffer pool sizes thus reduces and possibly eliminates these Recovery dataset 'synchronize' writes.

Recovery dataset I/Os are recorded in the log (if there is one), which can be examined using the AUDIT command. The minimum number of Recovery dataset I/Os per repository update command is five. If significantly more than five Recovery I/Os occur per command, consider increasing buffer pool sizes. (This should be done anyway as part of tuning.) If the number of I/Os is still high, consider increasing the blocksize.

Refer to the *ASG-ControlManager User's Guide* for details of how to monitor Recovery dataset I/Os.

Log Dataset

The Log dataset uses a single I/O buffer, equal in size to the physical blocksize of the dataset. The minimum number of Log dataset I/Os per logged command is four.

If you frequently put large member definitions into your repository, you should consider increasing the blocksize of the Log dataset.

Reorganizing an MP-AID

The MP-AID RELOAD command reorganizes fragmented members and their associated directories, which dramatically improves performance. So, for example, you should unload and reload your MP-AID after the installation of a new release, when multiple MP-AID LOAD commands will have fragmented MP-AID directories.

In addition, monitor the condition of your MP-AID using the MP-AID ANALYZE command. Unload and reload your MP-AID whenever fragmentation of any active directory or free chain exceeds 20 percent.

See "Introduction" on page 11 for the syntax of the MP-AID ANALYZE command.

Refer to the *ASG-Manager Products Systems Administrator's Manual* for further details of the MP-AID UNLOAD and MP-AID RELOAD command.

Turning Off the Check for a Disabled MP-AID

Whenever a Manager Products command is entered, the control record of the MP-AID is read in order to determine if the MP-AID has been disabled by the systems administrator since the execution of the previous command.

If you have some alternative way of communicating with users, you can disable this facility by using the command:

```
SET DISABLED-MPAID-CHECK OFF;
```

This will save you one physical I/O per command.To check the current setting enter:

```
QUERY DISABLED-MPAID-CHECK;
```

Refer to the *ASG-Manager Products Systems Administrator's Manual* for full details.

Re-blocking the Manager Products Load Library

This section is only applicable to MVS users.

The ASG-supplied load library is blocked at 6K (6144 bytes). This blocksize is suitable for all device types in current use. However, performance can be enhanced and program fetch time reduced considerably by re-blocking the load library to a blocksize specifically geared towards the device type in use at your installation. For example, for half track blocking on a 3380 device, you should use a blocksize of 23476 bytes.

The member MPOJCL06 in dataset MPJCL contains sample JCL which can be used as the basis for a re-blocking operation. A new library must be allocated with the appropriate blocksize to receive the output from the link-edit steps. SYSLIN input to the linkage-editor comes from two members, MPRLINK1 and MPRLINK2, supplied as members of dataset MFSOURCE.

Alternatively, you can use the IBM utility IEBCOPY in conjunction with the COPYMOD control statement.

When the re-blocking process is completed successfully, the original Manager Products load library can be deleted. You will probably then need to rename the new load library to that of the original.

If you choose to re-block the load library and wish to generate an overlaid version of the DesignManager executable module DSR00, you must re-block the load library before generating that overlaid version. If you do not, the overlay structure of the module DSR00 will be lost since the re-blocking process assumes a non-overlay version is required.

Creation of Repository Catalogs

Extensive usage of a catalog entry, for example INVOICE, will create a correspondingly large number of entries in the Data Entries dataset, spanning several logical, or even physical, blocks. So a command based on catalog may involve a large number of I/Os.

To improve response time, consider subdividing the catalog. For example, instead of one catalog INVOICE use INVOICE-NUMBER, INVOICE-ADDRESS, INVOICE-DETAIL, and so on.

In some cases, it may be worth using a catalog entry to identify a particular set of members, instead of repeatedly using an expensive interrogation command.

Refer to the *ASG-Manager Products Dictionary/Repository User's Guide* for further details of catalogs.

Creation of Repository Filler Names

Fillers should be individually named in order to avoid extensive searching of the Data Entries dataset to establish the many references, uses and connections of FILLER. For example, use FILLERA, FILLERB, FILLER1 and so on.

Refer to the *ASG-Manager Products Dictionary/Repository Users' Guide* for further details of filler names.

Repository Backup

Always use the UNLOAD and RELOAD commands (which work at the physical block level) for backup purposes as they are significantly faster than the SAVE and RESTORE commands (which work at the logical block level).

CPU time is reduced if you use minimum buffers when using the UNLOAD command.

Refer to the *ASG-Manager Products Controller's Manual* for further details of the UNLOAD and RELOAD commands.

Using the Repository in Read-only Mode

Opening the repository in read-only mode causes the Recovery dataset to remain closed. This saves two I/Os per command issued. Note, however, that if an abend occurs during an update by another user, the repository may be in an inconsistent state, and will only be returned to a consistent state when a user accessing the repository in update mode, enters a new command.

If there are times during the day when no updates are done, all users at these times should open the repository in read-only mode. For example, all your updates may be done overnight in batch.

If you decide that the repository should be opened in read-only mode, you can use the UPDICT operand of the DCUST macro. Alternatively you can use the SET DICTIONARY-UPDATES OFF command.

Refer to your installation manual for further details of the DCUST macro.

MP-AID In-core Indexes

When an INFOBANK-PANEL, USER-MEMBER or CORPORATE-EXECUTIVE member is located in the MP-AID directory, an entry is added to an in-core index maintained for each of these member types. Subsequent retrieval of these members is then faster, and physical I/Os are eliminated, because a search of the MP-AID directory is not required.

Each index has a size between 1K and 255K (the maximum under CICS is 8K). The default sizes are:

INFOBANK- PANEL	1K
USER-MEMBER	2K
CORPORATE-EXECUTIVE	2K

Approximately 25 entries can be accommodated for each kilobyte allocated in the index.

The index sizes can be set using the commands:

```
SET INFOBANK-PANEL-INDEX n;
SET USER-MEMBER-INDEX n;
SET CORPORATE-EXECUTIVE-INDEX n;
```

where *n* is an integer. These commands have the short forms:

```
SET IPI n;
SET UMI n;
SET CEI n;
```

Monitor the usage of these indexes using the following commands:

```
QUERY INFOBANK-PANEL-INDEX;
QUERY USER-MEMBER-INDEX;
QUERY CORPORATE-EXECUTIVE-INDEX;
```

which have the short forms:

```
QUERY IPI;
QUERY UMI;
QUERY CEI;
```

If the number of free entries falls to zero, you should consider increasing the index size. When an index is full, the least recently used (LRU) entry is deleted to make room for a new entry.

Reducing MP-AID Directory Search Time

When the Manager Products software searches for an Executive Routine, it by default first searches the User Member directory, then the Corporate Executive directory.

A single User Member directory is maintained for all users' USER-MEMBERs. This directory may become large and take a considerable time (with a large number of physical I/Os) to search.

The SET USER-DIRECTORY-SEARCH command allows you to modify the directory search order, or to exclude the User Member directory completely.

For the User Member directory to be searched first, enter:

```
SET USER-DIRECTORY-SEARCH FIRST;
```

For the User Member directory to be searched last, enter:

```
SET USER-DIRECTORY-SEARCH LAST;
```

For the User Member directory not to be searched at all, enter:

```
SET USER-DIRECTORY-SEARCH OFF;
```

To check the current setting enter:

```
QUERY USER-DIRECTORY-SEARCH;
```

The syntax of the SET USER-DIRECTORY-SEARCH Command is:

```
>>-- SET USER-DIRECTORY-SEARCH --+-- ON ----+--+-- ; --+------------> <
                                 +-- OFF ---+  +-- . --+
                                 +-- FIRST -+
                                 +-- LAST --+
```

4 Reducing Resource Contention

Maintaining Unique Repository and MP-AID Enqueue Names

The repository enqueue system is based upon the dataset name of the Index dataset. If repository datasets are copied for any reason, the internally held dataset name of the Index dataset, used for enqueue, is not amended by the copy operation. This means that two or more repositories may be using the same enqueue, which needlessly degrades performance. Use the CONTROL ENQ-NAME command to prevent this.

Similarly, use the MP-AID CONTROL ENQ-NAME command when the MP-AID dataset has been copied.

In a VM environment, the repository and MP-AID control disks should be amended via the CONTROL CMS and MP-AID CONTROL CMS commands respectively.

Refer to the *ASG-Manager Products Systems Administrator's Manual* for details of the MP-AID CONTROL ENQ-NAME and MP-AID CONTROL CMS commands.

Refer to the *ASG-Manager Products Controller's Manual* for details of the CONTROL ENQ-NAME and CONTROL CMS commands.

Maximizing Command Throughput

There are two enqueueing methods for Manager Products repositories:

- The standard method
- The alternative method.

Refer to your installation manual for full details of these methods.

When INTERROGATE-ENQUEUE is set ON, which is the default, commands use the standard method. When INTERROGATE-ENQUEUE is set OFF, commands use the alternative method.

The standard method is suitable for most purposes. However, if both the following are true of a repository:

- Long-running interrogation commands are common
- Update commands can occur at any time

then you should consider using the alternative method. This allows one update to access the repository at the same time as interrogation commands. This approach maximizes command throughput, and therefore improves response time for all users logged onto the repository.

With the alternative method, I/O flushes may occur. (Refer to your installation manual for full details of I/O flushes.) You can use the SET IO-FLUSH-LIMIT command to limit the number of buffer flushes allowed during an interrogation command. A suitable maximum depends on the size and usage of the repository. For example, it may be one if update commands are infrequent, otherwise higher.

To check the current enqueueing method enter:

```
QUERY INTERROGATE-ENQUEUE;
```

To check the current IO-FLUSH-LIMIT enter:

```
QUERY IO-FLUSH-LIMIT;
```

Refer to the *ASG-Manager Products Systems Administrator's Manual* for full details.

Creation of Multiple Repositories

Appropriate repositories should be created and maintained for specific purposes. An installation should have at least two repositories. For example, DATA and ADMIN, where DATA contains the member definitions for the organization's data and ADMIN contains the member definitions for Executive Routines, InfoBank, UDS structures and so on. Other repositories may exist for test, demonstration and user interface data.

Prudent and sensible separation of specific data in this manner reduces the amount of data not relevant to the task being carried out, thus enhancing performance and reducing resource contention between users.

Refer to your installation manual for further details of the Administration repository.

5 Reducing Real/Virtual Storage Needs

Sharing Executable Code

In most environments, each user accesses a separate copy of the Manager Products executable code. However, if there are several concurrent users, this unnecessarily increases Manager Products use of real and virtual storage. Exceptions to this are in teleprocessing environments such as CICS, where the program code is automatically shared by all users.

The Manager Products executable code is split into a number of separately loadable program segments. In most environments, these segments are reentrant and can therefore be shared by all users executing Manager Products.

You can choose to share all (or the most frequently used) code segments. This will:

- Reduce real and virtual storage needs
- Improve overall response times.

Manager Products code segments are identified as MPR*xx*, where *xx* is a number between 01 and 80. The most frequently used code segments are the nucleus segments: MPR00, MPR01 and MPR02. To find out which other segments are used often, use the QUERY SEGMENTS command.

Refer to your installation manual for further details of Manager Products executable code.

Refer to "Set Segments" on page 27 for details of the QUERY SEGMENTS command.

To share code, the selected components must be placed into the appropriate shared storage area (ask your installation's systems programmers about this). The components eligible for inclusion vary depending on your operating system.

The following table shows the environments in which executable code can be shared, how they are shared, and what segments/load modules can be shared:

Environment	Code-sharing Mechanism	What Can Be Shared
MVS	(Extended) Pageable Link Pack Area (EPLPA/PLPA)	All code segments. Also, any load module link-edited with the RENT attribute. To find out which modules are link-edited, use the ISPF option 3.4, or refer to your *ASG-Manager Products Installation in OS Environments Manual*.
VM	Creation and execution of a discontiguous shared segment (DCSS)	You can create a single DCSS equivalent of all Manager Products code segments.
VSE	Shared Virtual Area (SVA)	Most code segments are reentrant, and can therefore be shared, except for the nucleus segment MPR00.
Siemens	Automatic	Any shareable code is automatically made available.

Dynamic Code Loading

When executing Manager Products, the executable code segments are dynamically loaded into virtual storage as and when needed. Once loaded, segments are retained. This loading process continues until all required segments have been loaded into virtual storage. This enhances performance, as the overhead of repeated segment loading is avoided.

However, if you have limited virtual storage, you can load segments into virtual storage when needed, and delete them when not needed. Code segments that are deleted when not needed must be defined as *purgeable*. By default, all segments are defined as *resident*.

You can make any or all of the Manager Products segments purgeable or resident, except for the nucleus segments MFR00, MPR01 and MPR02, which are permanently resident.

You can define one or more code segments to be purgeable or resident using the SET SEGMENTS command. You can obtain information about segment usage via the QUERY SEGMENTS command.

Refer to "Set Segments" on page 27 for details of the SET SEGMENTS command.

Refer to "Query Segments" on page 28 for details of the QUERY SEGMENTS command.

Set Segments

Purpose

To make a selection of Manager Products code segments purgeable or resident.

Use

You can use the SET SEGMENTS command to specify the residency attributes of one or more Manager Products code segments. Which segments you wish to affect depends on your virtual storage availability, and how frequently used each code segment is. You can find out the frequency of use for code segments using the QUERY SEGMENTS command.

By default, all code segments are defined as resident, and (except for the three nucleus segments MPR00, MPR01 and MFR02, which are permanently resident) may be redefined as purgeable.

To make all code segments purgeable, enter:

```
SET SEGMENTS PURGEABLE;
```

To make all code segments resident, enter:

```
SET SEGMENTS RESIDENT;
```

To make one or more selected segments purgeable or resident, enter:

```
SET SEGMENTS segment-list RESIDENT;
```

or

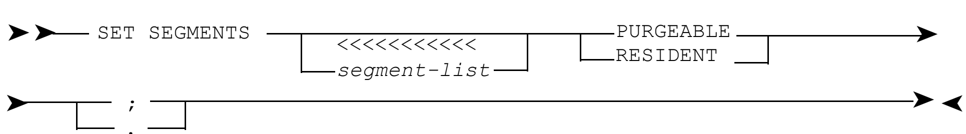
```
SET SEGMENTS segment-list PURGEABLE;
```

where *segment-list* is a list of one or more segments, separated by spaces. Each segment is given as a two-digit integer between 03 and 80.

For example, to make the segments MPR05, MPR12, and MPR34 purgeable, enter:

```
SET SEGMENTS 05 12 34 PURGEABLE;
```

Syntax



where *segment-list* is one or more integers between 3 and 80, separated by spaces.

Query Segments

Purpose

To display information about usage of Manager Products code segments.

Use

To display information about usage of currently-loaded Manager Products executable code segments, enter:

```
QUERY SEGMENTS;
```

A table is displayed, giving relevant information about the code segments' usage, status, size and functionality. You can use this information to decide whether to make any code segments purgeable or resident (using the SET SEGMENTS command).

To get significant information from the QUERY SEGMENTS command, you must first use the SET SEGMENTS command to make all segments purgeable. You should then run a session for a typical amount of time, using a typical amount of functionality. You can then use the QUERY SEGMENTS command to find out what code segments are most frequently used in your environment.

To display information about all Manager Products code segments, enter:

```
QUERY SEGMENTS ALL;
```

The information provided in tile table is given as follows:

Column in Table	Explanation
SEG-ID	Segment identity: MPR*xx*, where *xx* is 00 to 80.
LOAD-ADDR	Segment load address: the starting virtual storage address of the segment. If no address is shown, the segment is currently purged, but has been previously loaded. An asterisk (*) after the address denotes that this segment has been loaded into extended virtual storage (that is, above the 16 MB line). Therefore, you probably would not wish to make this segment purgeable.
LD-CT	Segment load count: shows the number of times that this segment has been loaded. For a nucleus or resident segment, this value will normally be set to one.
SIZE(K)	Segment size: given in kilobytes.
ATTR	Segment attribute: the status of the segment: NUC for a nucleus segment, RES for a resident segment, or PUR for a purgeable segment.
DESCRIPTION	Segment description: a brief description of the functionality provided by that segment.

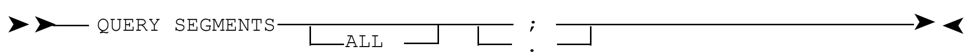
The current and maximum virtual storage used for segment loading (in K) are also displayed.

Figure 2 on page 29 is an example of the output from a QUERY SEGMENTS command, where the nucleus segments and the segments MPR72, MPR73, and MPR77 have been loaded:

```
----------------------------------------------------------------------------------------
SEG-ID    LOAD-ADDR     LD-CT    SIZE(K)    ATTR    DESCRIPTION
----------------------------------------------------------------------------------------
MPR00     00040590          1         48     NUC    MPR NUCLEUS
MPR01     00068120          1        135     NUC    REPOSITORY NUCLEUS
MPR02     022B9000*         1        232     NUC    MPR EXTENDED NUCLEUS
MPR72     02345000*         1          6     RES    SWITCH PROCESSOR
MPR73     022AF000*         1         38     RES    SET/QUERY PROCESSOR
MPR77     02310000*         1         11     RES    MPR SIGN-ON SCREEN
----------------------------------------------------------------------------------------
CURRENT STORAGE USED BY SEGMENTS   470K
MAXIMUM STORAGE USED BY SEGMENTS   470K
----------------------------------------------------------------------------------------
```

Figure 2. Example output from QUERY SEGMENTS command

Syntax



6 Reducing CPU Usage

Expanding the Manager Products Free Storage Pool

Introduction

The Manager Products software requests virtual storage as required from the operating system. Subject to an upper limit (by default 128K), virtual storage released by various Manager Products components is not returned directly to the operating system, but is retained in the Manager Products free storage pool. Subsequent requests for virtual storage can then be satisfied from this pool, rather than making a request to the operating system.

Savings Using the Free Storage Pool

In situations where large numbers of Manager Products commands are issued, increasing the size of the storage pool can reduce CPU usage by up to 10 percent.

Use the command:

```
SET FREE-POOL n;
```

to set the maximum amount of storage (in kilobytes) that can be retained in the freepool.

To check the current setting enter:

```
QUERY FREE-POOL;
```

Refer to the *ASG-Manager Products Systems Administrator's Manual* for full details.

Benchmark 4: Free Storage Pool

We ran our standard set of commands (see "Allocation of Large Buffer Pools" on page 11) with two different freepool settings. With FREE-POOL set to 0 the CPU time was 2 minutes 48.23 seconds. With FREE-POOL set to 128 (the default) the CPU time was 2 minutes 37.83 seconds, a reduction of 6 percent.

Logical Units of Work

Introduction

You can define a logical unit of work (LUW) as exclusive or shared, for either a repository or an MP-AID.

The commands defined in an LUW are, in effect, treated as a single command. Therefore savings can be made in CPU usage by eliminating inter-command overheads, such as the enqueue and dequeue processing associated with each repository or MP-AID command.

Note that some of the reduction in CPU time in the benchmarks is due to the reduction in physical I/O.

Additional Savings Using LUWs

When you use LUWs with large buffer pool allocations, you can also realize significant savings in dataset I/Os, which further reduces CPU usage.

You can save further dataset I/Os by using the ROLLBACK option available with repository LUWs. The repository updates are not committed until completion of the LUW. However, note that the size of the Recovery dataset must be sufficient to accept the pre-update images of all members updated in the LUW.

You can achieve maximum savings, both in CPU usage and dataset I/Os, by using the SPEED option of a repository LUW. Note, however, that if an abend occurs, the repository is likely to be in an unusable condition, and will need to be restored. An appropriate use of the SPEED option is with the RESTORE ALL command, where if an abend occurs, you need only recreate the repository.

The main benefit of LUWs is for the repository, but LUWs defined for the MP-AID can also produce savings, particularly when you construct many members onto the MP-AID.

Benchmark 5: Repository LUW —Use of SPEED Option

Case 1. A RESTORE ALL of the User Interface repository took 7 minutes 0.96 seconds of CPU time. The buffer usage statistics were as follows.

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	20	10	150	180
Maximum Buffers Used	9	10	150	169
Buffer Pool Size (K)	459	230	3439	4128
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	99%	98%	99%	99%
Dataset Logical Reads	21934	24529	276870	323333

Dataset Identification	Index	Source	Data	Totals
Dataset Physical Read	13	328	2451	2792
Dataset Logical Writes	206	19013	134273	153492
Dataset Physical Writes	63	125	27593	27781

Case 2. After defining an LUW using the command:

```
RESERVE DICTIONARY UPDATE SPEED;
```

a RESTORE ALL of the User Interface repository took 5 minutes 38.5 seconds of CPU time. The buffer usage statistics were as follows.

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	20	10	150	180
Maximum Buffers Used	9	10	150	169
Buffer Pool Size (K)	459	230	3439	4128
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	99%	98%	99%	99%
Dataset Logical Reads	21880	24529	280664	327073
Dataset Physical Read	13	328	588	929
Dataset Logical Writes	155	19013	136204	155372
Dataset Physical Writes	9	124	586	719

Summary

CASE	CPU (MINS)	ELAPSED (MINS)	PHYSICAL I/Os
No LUW	7.02	20.01	30510
Speed LUW	5.60	6.85	1648

Total physical I/O went down from 30510 to 1648, a reduction of 95 percent.

Benchmark 6: Repository LUW—Use of ROLLBACK Option

Case 1. A BULK ENCODE of 369 members, without an LUW, took 27.15 CPU seconds. The buffer usage statistics were as follows:

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	20	10	100	130
Maximum Buffers Used	9	10	76	95
Buffer Pool Size (K)	459	230	2293	2982
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	78%	97%	99%	96%
Dataset Logical Reads	1370	821	10513	12704
Dataset Physical Read	385	22	76	483
Dataset Logical Writes	370	370	4150	4890
Dataset Physical Writes	370	370	1385	2125

Case 2. A BULK ENCODE of the same members using a Rollback LUW took 21.77 CPU seconds. The Rollback Recovery Usage was 24% of a 15004K block Recovery dataset. The buffer usage statistics were as follows:

Dataset Identification	Index	Source	Data	Totals
Buffers Allocated	20	10	100	130
Maximum Buffers Used	9	10	76	95
Buffer Pool Size (K)	459	230	2293	2982
Read-only Buffer Steals	—	—	—	—
Modified Buffer Steals	—	—	—	—
Buffer Pool Hit Rate	99%	97%	99%	99%
Dataset Logical Reads	1370	821	10500	12691
Dataset Physical Read	13	22	76	111
Dataset Logical Writes	369	369	4146	4884
Dataset Physical Writes	1	1	57	59

Summary

CASE	CPU (MINS)	ELAPSED (MINS)	PHYSICAL I/Os
No LUW	0.45	2.90	2608
Rollback LUW	0.36	1.22	170

Total physical I/O was reduced from 2608 to 170, a reduction of 93 percent.

Benchmark 7: MP-AID LUW

Case 1. A CONSTRUCT of 489 INFOBANK-PANELs onto the MP-AID without an LUW took 31.92 CPU seconds. The MP-AID buffer usage statistics were as follows:

Buffers Allocated	100
Maximum Buffers Used	100
Buffer Pool Size (K)	2293
Read-only Buffer Steals	—
Modified Buffer Steals	—
Buffer Pool Hit Rate	94%
Dataset Logical Reads	18107
Dataset Physical Read	1113
Dataset Logical Writes	4889
Dataset Physical Writes	2020

Case 2. The same CONSTRUCT with an MP-AID LUW took 29.78 CPU seconds. The MP-AID buffer usage statistics were as follows:

Buffers Allocated	100
Maximum Buffers Used	100
Buffer Pool Size (K)	2293
Read-only Buffer Steals	—
Modified Buffer Steals	—
Buffer Pool Hit Rate	99%
Dataset Logical Reads	18109
Dataset Physical Read	137
Dataset Logical Writes	4889
Dataset Physical Writes	2004

Summary

CASE	CPU (MINS)	ELAPSED (MINS)	PHYSICAL I/Os
No LUW	0.53	3.58	3133
MP-AID LUW	0.50	2.70	2141

Total physical I/O went down from 3133 to 2141, a reduction of 32 percent.

Retaining Executive Routines in Virtual Storage

The SET EXECUTIVE-RETENTION ON command causes Executive Routines that execute the RETAIN directive to be held in virtual storage following their initial execution.

Thus, frequently used Executive Routines should execute the RETAIN directive. This reduces CPU time, and slightly reduces I/Os, whenever the Executive Routine is called.

This facility is not available in CICS environments.

To check the current setting enter:

```
QUERY EXECUTIVE-RETENTION;
```

To obtain information about retained Executive Routines enter:

```
QUERY RETAINED-EXECUTIVES;
```

Refer to the *ASG-Manager Products Procedures Language* manual for further details of the RETAIN directive.

Refer to the *ASG-ControlManager User's Guide* for details of the SET and QUERY EXECUTIVE-RETENTION commands.

Terminating Nested Executives

The TRANSFER directive allows you to terminate all active Executive Routines and call a new Executive Routine. It is particularly useful in error situations, where otherwise time is wasted returning from each active Executive Routine individually.

Refer to the *ASG-Manager Products Procedures Language* manual for further details of the TRANSFER directive.

7 Mechanisms for Limiting Resource Usage

This chapter applies to online users only.

Controlling Dataset I/Os

Use the SET EXCP-LIMIT command to impose an upper limit on the number of repository or MP-AID physical I/Os permitted in processing a command. This command gives you control over execution of long-running commands that consume machine resources. Such commands might be better done in batch as an overnight job.

The limit does not apply to the following commands:

DIAGNOSE
MP-AID DIAGNOSE
UNLOAD
RELOAD
STATUS
CREATE
RESTORE
SAVE.

The minimum limit that can be set is 50. There is no default limit. When the limit is reached, the executing command is terminated and message DM00008E is output.

To check the current setting enter:

```
QUERY EXCP-LIMIT;
```

Refer to the *ASG-Manager Products Systems Administrator's Manual* for further details of the SET EXCP-LIMIT command.

Controlling Command Output

Use the SET OUTPUT-LINE-LIMIT command to impose a limit on the number of output lines generated by a command when the limit is reached, the executing command is terminated and a message is output. The default limit is 1,000 lines.

This command gives you control over execution of long-running commands that consume machine resources. Such commands might be better done in batch as an overnight job.

The limit does not apply to the following commands:

DIAGNOSE
MP-AID DIAGNOSE
UNLOAD
RELOAD
STATUS
CREATE
RESTORE
SAVE.

When the limit is reached, the executing command is terminated and message DM00007E is output.

To check the current setting enter:

```
QUERY OUTPUT-LINE-LIMIT;
```

Refer to the *ASG-Manager Products Systems Administrator's Manual* for full details.

Restricting Output Buffer Creation

Use the SET BUFFER-LIMIT command to limit the number of output buffers that can be open at any time. The command prevents general users from holding storage unnecessarily. The default limit is five.

To check the current setting enter:

```
QUERY BUFFER-LIMIT;
```

Refer to the *ASG-Manager Products Systems Administrator's Manual* for full details.

8 Facilities to Use Selectively

You should avoid excessive use of facilities intended primarily for diagnostic purposes, since they can consume significant amounts of machine resources. Such facilities are mentioned below.

Use of the following commands can greatly increase terminal I/O, and therefore degrade response time:

SET SCREEN-REFRESH
SET EXCP-MONITOR
SET COUNTER
SET SCROLL-LIMIT

We recommend that you do not set EXCP-MONITOR to a value less than about 200, and that you leave COUNTER at the default (25).

Avoid prolonged use of the MONITOR keyword on the DICTIONARY MP-AID BUFFERS commands. Use of this monitoring facility can significantly increase use of virtual storage and CPU time.

Index