

# **M - P a c t - P A C K / V S E**

**AUTOMATIC VSAM FILE COMPRESSION**

## **PROGRAM DESCRIPTION AND OPERATIONS MANUAL**

**Release 5.0**

(COPYRIGHT © 2003, FANTOM SYSTEMS, INC.)

M-Pact-PACK is a proprietary product of Fantom Systems, Inc. It cannot be reproduced, changed, copied, or stored in any form (including, but not limited to, copies on magnetic media) without the express prior written permission.

Original Printing .....09/25/2003  
Last Revised .....09/25/2003

# TABLE OF CONTENTS

---

---

<b>GETTING STARTED</b> .....	<b>1</b>
Introduction .....	3
Benefits .....	4
Compression Modes .....	5
Basic Compression (mode 1) .....	5
Basic+Numeric Compression (mode 3) .....	5
Basic+Alpha Compression (mode 5) .....	5
Full Compression (mode 7) .....	6
Compression Range .....	7
Component Description .....	8
Migration Requirements .....	10
Installation.....	11
GETVIS Usage .....	15
Removing the VSAM OPEN/CLOSE Router .....	16
<b>USING M-Pact-PACK</b> .....	<b>17</b>
File Analysis.....	19
VSAM File Considerations.....	22
VSAM KSDS File Definition Considerations .....	22
VSAM KSDS Files Accessed via ISAM Interface Program (IIP).....	24
VSAM ESDS File Definition Considerations .....	24
Defining a M-Pact-PACK File.....	25
CSIPACTB Compression Table Generation .....	26
User Compression and Expansion.....	28
User Compression Sample Program .....	30
Compressing DL/I Segments .....	32
Questions and Answers .....	33
<b>MESSAGES</b> .....	<b>37</b>



**GETTING STARTED**



## Introduction

M-Pact-PACK addresses one of the major problems of today's computer installations - disk storage. With the ever increasing demands to keep more and more data on-line, computer installations are faced with the need to continually install more disk storage devices with even greater storage density and capacity. This situation leads to the inevitable problems of the cost of more disk drives, floor space for the new disk drives and controllers, air-conditioning upgrades, and the scheduling of long running backup jobs with their ever increasing number of magnetic tape reels. If you use VSAM and are not using M-Pact-PACK you are probably wasting valuable disk space. Many installations have files that contain records where only a small amount of the record is used, giving rise to large areas of unused space, and records not being efficiently blocked into VSAM Control Intervals and accordingly Control Areas.

M-Pact-PACK addresses these problems for Entry- and Key-Sequenced VSAM files by allowing the user to compress the data on selected files, and thus reduce the amount of space required to hold such files.

The amount which can be saved by compression depends on the contents and layout of the user's records, and the degree of compression chosen by the user. Compression of more than 70% has been achieved by M-Pact-PACK's sophisticated algorithms.

M-Pact-PACK includes a file analysis program which can be run against selected files to establish the potential compression ratio, and thus the value of using M-Pact-PACK for the file, and the site generally. The compression ratio for each of the four compression modes mentioned below is calculated for each file being analyzed.

M-Pact-PACK is fully transparent to programmers, operators, and application programs. Selected files are re-DEFINED and REPRO'd, after which the compression process is automatic and invisible.

Please note that RRDS and VSE SAM-managed files are not able to be compressed.

Programs are included for compression of DL/I variable length segments, and for user application program special use, for example to compress sequential disk or tape files.

### Benefits

As a consequence of installing M-Pact-PACK, the user should experience:

- Dramatic reductions in disk space requirements.
- Reduced disk arm movement due to high data compression.
- Substantially faster retrieval of data records when reading files sequentially.
- For KSDS files a reduction in index levels and index records.
- Reduction of CI/CA splits given the same percentage of freespace per CI/CA.
- More productive use of data transfer time, and possible reductions in channel contention.
- Reduced backup times for VSAM files and catalogs with a corresponding reduction in the number of tapes held in each cycle.

ALL of the above give rise to QUICKER on-line response times, FASTER batch throughput and BETTER utilization of virtual storage.

## **Compression Modes**

Four degrees of compression are provided, to cater to the varying record contents of the user's files. Each file can be given a different compression mode depending on the contents of its records.

The M-Pact-PACK file analysis program CSIPACAN (see later) reports the differing percentage savings that may be achieved for each of the four modes, and can help the user to decide the optimum mode for each file, to achieve the best "mix" between disk saving and CPU overhead.

Very brief details of each of the four compression modes follow:

### **Basic Compression (mode 1)**

This mode performs, amongst other things, compression of Space and Null characters, and repetitive characters. This mode is suitable for most files, and has the least CPU overhead.

### **Basic+Numeric Compression (mode 3)**

This mode performs basic compression as described above and also applies compression algorithms to numeric data. The CPU overhead is greater than for mode 1, depending on the record content, and it may not be advisable to use this mode unless CSIPACAN shows an increased saving over mode 1 of over 5%.

### **Basic+Alpha Compression (mode 5)**

This mode performs basic compression as described above and also applies compression algorithms to alpha data. The CPU overhead is similar to mode 3, depending on the record content, and it may not be advisable to use this mode unless CSIPACAN shows an increased saving over mode 1 of over 5%.

**Full Compression (mode 7)**

This mode performs all the above compression's, and therefore the CPU overhead is the greatest. Depending on the usage of the file, it may not be advisable to use this mode unless CSIPACAN shows an increased saving over mode 1 of over 10%.

The CPU overhead in compression/expansion is virtually impossible to predict, since it depends solely upon the contents of the record. However, tests have shown that basic compression (mode 1) gives rise to only a negligible overhead. Full compression could give rise to an overhead of up to 15 percent. The file analysis program (see later) will tell you whether the extra compression achieved by choosing an option other than "basic" (mode 1) will be worthwhile. Also, bear in mind that the I/O time saved by having records in a compressed form may well counteract the CPU overhead.

The compression mode is established when the file is DEFINE'd to VSAM, and is expressed by the letter "n" throughout this manual

Refer to section "Defining a M-Pact-PACK File" on page 25 for information on how to activate M-Pact-PACK for a particular VSAM dataset.

## Compression Range

It is not usually possible to compress the entire record. For instance, on a KSDS, the key must not be compressed, since VSAM must see it as it really is, and in its correct position. Also, for a file with an alternate index, that portion of the record up to and including the alternate index key must not be compressed for the same reason.

In the case of an ESDS, the entire record may be compressed UNLESS the record will be updated. This is because the update of a byte in the compressed portion of a record may cause a record length change. This is not permitted by VSAM for an ESDS.

For "straight" KSDS with no alternate index, M-Pact-PACK will automatically calculate the end position of the key, and compress the record from that point up to the end of the record. For KSDS with alternate indexes, and updatable ESDS files, M-Pact-PACK has no way of knowing where to start compressing, and must therefore be told. Refer to section "Defining a M-Pact-PACK File" on page 25 for more information.

## Component Description

The M-Pact-PACK system consists of the following components:

- CSIPACAN** This module is supplied to enable the user to analyze each problem VSAM file and to establish the compression ratios possible on the file. It reports whether compression was possible, and if so, reports the compression ratios achieved and the average compressed record length. For large files, CSIPACAN can be run against a user-specified number of records rather than the whole file. Just by running CSIPACAN against a few of your major files you will see at a glance the savings that can be made with M-Pact-PACK.
- CSIPACST** This module starts the M-Pact-PACK system. the user must ensure that CSIPACST is executed immediately after IPL. CSIPACST checks that the M-Pact-PACK components are present in the system, and triggers the VSAM intercept mechanism. After this phase has been executed, all VSAM calls are intercepted to see whether the call is related to a M-Pact-PACK file. If so, the appropriate compression or expansion is performed automatically.
- CSIPACCM** This module does the actual compression. It is loaded automatically by various M-Pact-PACK components. It is also used to perform user-controlled compression (see later).
- For VSE, this phase must be placed in the SVA.
- CSIPACEX** This module does the actual expansion, It is loaded automatically by various M-Pact-PACK components. It is also used to perform user-controlled expansion (see later).
- For VSE, this phase must be placed in the SVA.
- CSIPACCP** This module receives control for all calls to VSAM for record management from whatever source. If the file concerned is running under the control of M-Pact-PACK, the record is compressed on an output-type call, or expanded on an input-type call.

<b>CSIPACOP</b>	This module receives control for all VSAM open requests. It determines if the VSAM dataset is compressed, and if so, alters the VSAM ACB to call CSIPACCP for every VSAM I/O request from the application.
<b>CSIPACCL</b>	This module receives control for all VSAM close requests. It releases all storage and resources acquired by M-Pact-PACK for the VSAM dataset being closed..
<b>CSIPACK</b>	This relocatable module is supplied to allow user-controlled compression and expansion. It performs the same functions as the automatic compression, however it allows the user to compress and expand files not currently supported by M-Pact-PACK. See section "User Compression and Expansion" on page 28 for more information.
<b>CSIPACDL</b>	This VSE module is used for compression and expansion of variable length DL/I segments. See section "Compressing DL/I Segments" on page 32.

## Migration Requirements

The following sections cover the changes in the installation of M-Pact-PACK that may be required if you are migrating from an earlier release.

### Changes from Release 3.3

- A new method has been added for specifying which VSAM datasets are to be compressed and which programs are to be excluded from compression. This method uses a new compression control table (CSIPACTB). Refer to section "Defining a M-Pact-PACK File" on page 25 for more information.
- A new operand has been added to the calling sequence for subroutine CSIPACK. If you have any programs using this routine, review section "User Compression and Expansion" on page 28.
- Several new modules have been added to M-Pact-PACK/VSE. Review the Installation section for the new installation procedure.
- Before activating the new release of M-Pact-PACK/VSE on a running system, make sure all VSAM files are closed, and execute the following in BG:

```
SET SDL  
IKQVRM, SVA
```

This will deactivate your current release of M-Pact-PACK.

- If you activate the new M-Pact-PACK/VSE while any CICS systems are active, you may run short of partition GETVIS when attempting to re-open your VSAM files. This is because the storage areas used by the old M-Pact-PACK cannot be reused by the new M-Pact-PACK. You will need to cycle your CICS systems to reclaim the GETVIS storage.

### Installation

M-Pact-PACK/VSE is distributed on a LIBR BACKUP cartridge consisting a sublibrary, named CSIDIST.CSIPACK containing the following:

#### Load Modules

<b>CSIPACK</b>	User compression interface phase.
<b>CSIPACCM</b>	Compression program.
<b>CSIPACAN</b>	File analysis program.
<b>CSIPACDL</b>	DL/I compression program.
<b>CSIPACCP</b>	VSAM I/O interface program.
<b>CSIPACEX</b>	Expansion program.
<b>CSIPACOP</b>	VSAM open interface program.
<b>CSIPACCL</b>	VSAM close interface program.
<b>CSIPACSA</b>	Storage Anchor module.
<b>CSIPACST</b>	System startup program.
<b>CSIPACTB</b>	Default compression table.
<b>BIMVSRM</b>	Routing Facility command module.
<b>BIMVSROP</b>	Routing Facility Open SVC router.
<b>BIMVSRCL</b>	Routing Facility Close SVC router.
<b>BIMVSRPO</b>	Routing Facility Post Open handler.
<b>BIMVSRPC</b>	Routing Facility Post Close handler.
<b>BIMVSRTB</b>	Routing Facility product table.
<b>BIMVSRJE</b>	Routing Facility end-of-job handler.
<b>BIMVSRQE</b>	Routing Facility SVC 14 intercept.
<b>BIMVSRSC</b>	Routing Facility storage anchor for Close.
<b>BIMVSRSO</b>	Routing Facility storage anchor for Open.

#### Object/Source Modules

<b>CSIPACK.OBJ</b>	User compression interface module.
<b>CSIPACTM.A</b>	Macro required to generate CSIPACTB
<b>CSIPACTB.A</b>	Source for dummy CSIPACTB
<b>CSIPACJ1.A</b>	Required SVA list

#### Installation Step 1

Use a job similar to the following one to restore the sublibrary from the installation tape:

```
* $$ JOB JNM=INSTALL
// JOB      INSTALL
// ASSGN   SYS006, CUU                INSTALL TAPE
// DLBL    USRLIB1  etc.
// EXTENT  etc.
// EXEC    PGM=LIBR, SIZE=256K
RESTORE SUBLIB=CSIDIST.CSIPACK:USRLIB1.CSIPACK TAPE=SYS006
/*****/
/* Optional step to move members to a permanent      */
/* sublibrary and delete sublibrary created above.    */
/*****/
CONNECT SUBLIB=USRLIB1.CSIPACK:USRLIB.PROD
MOVE *.* LIST=YES REPLACE=YES
DELETE SUBLIB=USRLIB1.CSIPACK
/*****/
/*
/&
* $$ EOJ
```

### Installation Step 2 - Amend the BG ASI procedure as follows:

Add the following to SET SDL step of your BG ASI procedure:

```
BIMVSROP, SVA
BIMVSRCL, SVA
BIMVSRPO, SVA
BIMVSRPC, SVA
BIMVSRTB, SVA
BIMVSRCM, SVA
BIMVSRJE, SVA
BIMVSR0E, SVA      ('0' IS A ZERO)
CSIPACOP, SVA
CSIPACCL, SVA
CSIPACCP, SVA
CSIPACTB, SVA
CSIPACEX, SVA
CSIPACCM, SVA
CSIPACK, SVA
```

These phases must reside in the SVA. These statements are provided in source member CSIPACJ1.A.

In addition to the SVA modules, the following modules must be in a VSE library that is accessible to all partitions via a permanent LIBDEF:

```
BIMVSRSC.PHASE  
BIMVSRSO.PHASE  
CSIPACSA.PHASE
```

This can be accomplished by copying these modules to a VSE library already in a permanent LIBDEF, or by adding the M-Pact-PACK sublibrary to your permanent LIBDEF.

After the SVA load insert the following step:

```
// EXEC CSIPACST,PARM='STRT'
```

### WARNING:

M-Pact-PACK should be started directly after IPL. This means BEFORE the start-up of POWER. The reason for this is that if M-Pact-PACK fails to startup for any reason, the operator can prevent any other partitions from starting, perhaps with dire consequences, in a M-Pact-PACK-less system, until the problem with the startup has been corrected (see also description of message CSIPACK-99).

### VSAM OPEN/CLOSE Routing Facility

M-Pact-PACK/VSE uses an SVC Routing Facility to intercept VSAM OPEN and CLOSE processing. This facility allows one or more CSI/BIM products to intercept OPEN and/or CLOSE processing in a prescribed order. The Routing Facility is implemented transparently in M-Pact-PACK by means of the STRT and STOP parm of program CSIPACST.

Installation Step 2 installed the SVC Routing modules into the SVA. In general, no problems will occur if the copy of the Routing Facility distributed with one CSI/BIM product is overlaid by another.

The following modules make up the Routing Facility:

<b>BIMVSRM</b>	Routing Facility command module.
<b>BIMVSRP</b>	Routing Facility Open SVC router.
<b>BIMVSRCL</b>	Routing Facility Close SVC router.
<b>BIMVSRPO</b>	Routing Facility Post Open handler.
<b>BIMVSRPC</b>	Routing Facility Post Close handler.
<b>BIMVSRTB</b>	Routing Facility product table.
<b>BIMVSRJE</b>	Routing Facility end-of-job handler.
<b>BIMVSR0E</b>	Routing Facility SVC 14 intercept.
<b>BIMVSRSC</b>	Routing Facility storage anchor for Close.
<b>BIMVSRSO</b>	Routing Facility storage anchor for Open.

## GETVIS Usage

M-Pact-PACK acquires partition/region GETVIS to use as a work area during the compression and expansion of records.

M-Pact-PACK attempts to acquire a large area that will be used for the duration of the job. M-Pact-PACK first tries to acquire 33K. If this amount is not available, M-Pact-PACK reduces its request by steps of 1K, down to 9K. If this amount is not available, then M-Pact-PACK issues an error message, and returns an error code to the program.

No matter what size is actually obtained by M-Pact-PACK, data integrity is always assured. Also, most applications will not need anything near the maximum size to run without performance problems, and extra GETVIS allocation will almost never be necessary.

### Removing the VSAM OPEN/CLOSE Router

The VSAM OPEN/CLOSE Router used by M-Pact-PACK will not have to be removed from your system under normal conditions. The individual products using the router, such as M-Pact-PACK, can be activated and de-activated as needed using provided interfaces for each product.

You will only need to remove the router if you are experiencing problems with your system that are not corrected by de-activating the individual products.

The following example contains the JCL required to remove the router from your system:

```
// JOB CSIVSR          remove VSAM router
// EXEC PGM=BIMVSRM, PARM='DELT01'
/*
/ &
```

The router command module (BIMVSRM) can perform several functions depending on the PARM value specified:

- |        |  |
|--------|--|
| SHUT   | Stops the router from dispatching any of the CSI/BIM products, but leaves the router in your system.   |
| INIT   | Starts the router, and resumes dispatching the CSI/BIM products. This function can be performed to reverse the 'SHUT' function.  |
| DELT01 | Removes the router from your system, and restores the original address pointers. <b>This process will 'unhook' any other vendors' products that were started <u>after</u> the first CSI/BIM product that uses this router.</b> |
| DELT02 | Removes the router from your system, and restores the original IBM address pointers. <b>This process will 'unhook' <u>all</u> other vendors' products that 'hook' the VSAM OPEN/CLOSE process.</b>                             |

**USING M-Pact-PACK**



## File Analysis

Run program CSIPACAN to establish the compression factor of your files:

### Sample JCL

```
// DLBL fname1,'dsn.fname.1',,VSAM,CAT=catname
// DLBL fname2,'dsn.fname.2',,VSAM,CAT=catname
.
.
// DLBL fnamen,'dsn.fname.n',,VSAM,CAT=catname
// EXEC CSIPACAN
FILE=fname1,COUNT=count,HIKEY=kkkk,CTYPE=n
FILE=fname2,COUNT=count,HIKEY=offset,KEYLEN=ll
.
.
FILE=fnamen
/*
```

### where:

fname	is the file name of the file you wish to analyze, and relates to the corresponding DLBL card.
count	is (optionally)the number of records you wish to analyze. If "count" is not specified, the entire file is analyzed.
CTYPE	is (optionally)the specific compression type to be analyzed. If omitted, all compression types will be analyzed, which will result in a longer run time.
kkkk	is the offset within the record that you want the compression routine to start compressing, as fully described above. If you don't specify the HIKEY=kkkk parameter, the default of zero is used, which is what you want for "straight" KSDS/ESDS.

You can optionally specify the offset of the highest alternate index key for the HIKEY operand, and specify the length of the alternate index key in the KEYLEN operand.

### Example control cards:

```
FILE=MASTER COUNT=1000
```

analyzes the first 1000 records of file "MASTER"

```
FILE=TRANS
```

analyzes the entire file "TRANS"

Control cards are completely free format. There is no limit to the number of files which can be analyzed in a single run.

It is important to analyze a representative sample of each file to accurately determine the compression factor.

The following information is reported for each file analyzed, and is repeated for each compression mode:

- Number of records analyzed, and number and percent which could be compressed
- Total byte count and average record length of the expanded (i.e., uncompressed) file
- Total byte count and average record length of the file should you decide to compress it
- Total, average, and percent saving that compressing the file would achieve

### Sample Report Output:

FILE NAME	COMP TYPE	RECORDS READ	RECORDS COMPRESSED	# OF BYTES BEFORE	# OF BYTES SAVED	# OF BYTES AFTER	AVG. LRECL BEFORE	AVG. LRECL AFTER	AVG. SAVINGS	PERCENTAGE SAVED
OUTDD1	1	5480	5358	3233946	1319702	1914244	590	349	240	40
OUTDD1	3	5480	5358	3233946	1335264	1898682	590	346	243	41
OUTDD1	5	5480	5360	3233946	1413781	1820165	590	332	257	43
OUTDD1	7	5480	5360	3233946	1410729	1823217	590	332	257	43

After a file is loaded under M-Pact-PACK, you may notice that the disk space used (High RBA) is more or less than CSIPACAN determined, even for the exact same data and a full file analysis. This is because the CISIZE may be a better or worse choice for the new average (compressed) record size. However, it should be fairly close to the projected savings, and the CISIZE can, of course, be adjusted.

## VSAM File Considerations

When planning to install M-Pact-PACK it is important that each VSAM file to be controlled by M-Pact-PACK is reviewed. For each different file type there is a separate section following. Read the section carefully before conversion.

### VSAM KSDS File Definition Considerations

VSAM KSDS files are fully supported by M-Pact-PACK. The section "Defining a M-Pact-PACK File" on page 25 details how to define the file, however, this section gives more details.

A VSAM KSDS is normally defined with only a DATA and INDEX component. In this case M-Pact-PACK will automatically compress the records from the end of the prime key to the end of the record. For example, if you specify that the key is 8 bytes long and start at position 1 of the record, the record length is 500, then the record will be compressed from position 9 up to position 500. The compression will be transparent to any user wishing to access the file at record level.

In the event of a program accessing the file by CI, there are three situations that can arise:

- a) The program is a utility (e.g. VSAM BACKUP/RESTORE) and it does not want to interrogate each individual record. The program will have each CI returned with COMPRESSED records in it, because M-Pact-PACK will not attempt to expand the records. It therefore follows that if the utility is backing a file up to tape that a corresponding reduction in tape usage will also be realized. When the file is restored again it will read COMPRESSED CI records from the tape and build the corresponding VSAM file in COMPRESSED format. This particular method gives maximum gain from M-Pact-PACK file compression.
- b) The program is a utility such as above, but the program backs up by CI, but restores by individual record. This presents a problem because if M-Pact-PACK attempts to compress an already compressed record, it will damage the record. The only known case of this is the VSE FAVOR product from Legent. To deal with this, M-Pact-PACK/VSE does not attempt to compress records if a VSE JCL statement:

```
// UPSI 10101010
```

is present in a job stream. Thus this statement should be present in all FAVOR restore (but not backup) jobs.

An alternative to this would be to add these programs to the CSIPACTB compression table, as an excluded program.

- c) The program is user written and for reasons of performance, reads the file by CI, decipheres each CI and looks at each individual record. M-Pact-PACK will again pass back a CI that contains COMPRESSED records, so in this case the user program needs to be amended to call M-Pact-PACK to expand the compressed records, and if it changes records call M-Pact-PACK again to compress the records. As M-Pact-PACK compression can lead to a record length change if the data is changed, the program, if it rewrites at CI level, must be able to handle change of record length situations and all the subsequent consequences, for example CI splits. This type of user program is very rare.

If a KSDS file has an alternate index, then the definition to M-Pact-PACK is slightly different and so is the resulting compression percentage.

### IMPORTANT:

It must be remembered that if you wish to add or change an alternate index to an existing file then you must backup and DEFINE the file again with the correct CSIPACTB table HIKEY and KEYLEN operands before attempting to build the new alternate index.

### **VSAM KSDS Files Accessed via ISAM Interface Program (IIP)**

When a KSDS is accessed via the IIP, the following considerations must be taken into account:

- a) If the file is accessed with DTF's specifying BLOCKED record format, then these accesses will be successful on a compressed file. This is by far the most frequent occurrence.
- b) If the file is accessed with UNBLOCKED format DTF's, then GET-type requests to a compressed file will still be successful, but PUT requests to a compressed file will fail. Message CSIPACK-215 will be displayed on the console, and the PUT request will be returned with an error code denoting "Physical Data-Write Error". The reason for this is that in such circumstances, VSAM translates the PUT request to a "PUT-LOCATE" request. This means that the updated record is presented to VSAM (and therefore to M-Pact-PACK) in the Control Interval itself, rather than in a work area as in all other cases. The newly updated record may compress to a longer length than the original compressed record. Obviously, M-Pact-PACK cannot carry out this request since in this case the rest of the CI would be corrupted.

### **VSAM ESDS File Definition Considerations**

VSAM ESDS files are supported by M-Pact-PACK. See section "Defining a M-Pact-PACK File" on page 25.

If the ESDS file is never updated in place then specify HIKEY=0 and KEYLEN=0 for the CSIPACTB table definition for the ESDS file.

If any record on the file can potentially be updated by a program, then because of M-Pact-PACK's compression it could change the length of the record. A change of record length for this type of file is not allowed and would generate a VSAM error. To enable M-Pact-PACK to compress the file, specify the HIKEY and KEYLEN operands to define the portion of the record that can be updated. See earlier sections "Compression Modes" on page 5 and "Compression Range" on page 7.

## Defining a M-Pact-PACK File

Once the analysis program CSIPACAN has been run and you are satisfied that M-Pact-PACK will reduce your disk space requirements you must convert the file to a M-Pact-PACK compressed dataset.

The following procedure describes both methods:

### STEP 1

Backup or copy your selected VSAM file either to tape or to another disk area. You may not at this time use a backup program that reads VSAM files by Control Intervals, such as IBM's BACKUP and RESTORE. The suggested method would be to use VSAM's IDCAMS to REPRO the file to tape so that the file gets reorganized on the subsequent reloading. However you backup the file, it must be in a form that will allow for a restore of the file by logical record, not by CI.

### STEP 2

- a) DELETE/DEFINE your file on the designated VSAM catalog.
- b) Add an entry to your sites M-Pact-PACK compression table CSIPACTB as described in the next section.
- c) Refresh your CSIPACTB table as described in the next section.

### STEP 3

Reload your file from your VSAM backup. Remember, you must restore the file by logical record, not by CI. As the records are written to the newly defined file they will be compressed.

All access to the file is now being intercepted by M-Pact-PACK and records will be expanded automatically before control is returned to the user program.

## CSIPACTB Compression Table Generation

M-Pact-PACK decides whether to compress a VSAM file or not depending on whether or not the cluster has been defined in the compression table CSIPACTB.

### Sample CSIPACTB

```

1...5...10...15...20...25...30                                ...72
      PRINT OFF                                               (note 1)
      COPY CSIPACTM                                           (note 1)
      PRINT ON                                                 (note 1)
CSIPACTB CSECT
      CSIPACTM DSNM=VSAM.NAME1
      CSIPACTM DSNM=VSAM.CLUSTER.NAME,                        X
              DSNML=44,CTYPE=1,                               X
              HIKEY=4,KEYLEN=8
      CSIPACTM PGM=GVRESTOR
BIMEOT  DC  X'FF'      MARKS THE END OF THE TABLE
      END
    
```

Note 1: These lines are only required for VSE users running a release of VSE prior to VSE/ESA 2.1.

### Explanation

DSNM	is a dsname mask. This can be a complete dataset name, or the leading characters of a group of dataset names.
DSNML	is the length of the dsname mask specified in DSNM. If omitted, the actual length of DSNM is used. To ensure that a complete dataset name is not used as a mask, specify DSNML=44.
CTYPE	is the compression type to be used for the matching dataset(s). If omitted, CTYPE=1 is assumed.
HIKEY	is the highest relative key position of any alternate index associated with this base cluster. If omitted, HIKEY=0 is assumed.
KEYLEN	is the length of the alternate index key specified in HIKEY, or for an updatable ESDS, the last updatable byte position.

PGM is the name of a PGM on an EXEC statement, that will have compression/decompression automatically suppressed by M-Pact-PACK.

### Activating a new table

After you have modified your CSIPACTB table, you must inform M-Pact-PACK that a new version is to be used. The following JCL can be used:

The table is contained in the SVA. You will need to reload module CSIPACTB in the SVA before running the refresh. This must be run in partition BG:

```
SET SDL
CSIPACTB, SVA
/*
```

## User Compression and Expansion

It is also possible to compress and expand records under user application program control, using the supplied interface. This facility may be particularly useful when creating non-VSAM files, such as sequential tape and disk.

The calling sequence is as follows:

**BAL:**

```
CALL CSIPACK, (function, exp-record, comp-record, length, max-  
length, offset), VL
```

**COBOL:**

```
CALL 'CSIPACK' USING function, exp-record,  
                      comp-record, length,  
                      max-length, offset.
```

where:

- |             |  |
|-------------|--|
| function    | is "E" for Expand, "A" "B" "C" or "D" for Compress with compression mode 1, 3, 5, or 7 respectively.   |
| exp-record  | is the expanded record.  |
| comp-record | is the compressed record.  |
| length      | is the length of the input (i.e. length of the compressed record for the Expand function, length of the expanded record for a Compress function). This field MUST be a two-byte binary field. The resulting length is returned in this field. For example, before a Compress operation, set this field to the actual length of the record. After the Compress, this field is set to the compressed length.                                       |
| max-length  | is the maximum length of this record when NOT compressed. This value must be specified identically for all functions, that is, the same number must be used for Compress and Expand calls. This field MUST be a two-byte binary field. Normally, the maximum length would be a constant value for all compress and expand calls for a file, but technically it is only imperative that the same value be used for all calls for the same record. |

`offset` is the offset, relative to zero, into the record that the compression/decompression is to start. This field MUST be a two-byte binary field.

The 'offset' operand is optional. If omitted, zero is assumed. If you are calling CSIPACK from a BAL program, and you omit the 'offset' operand, you must specify `x'80'` in the upper byte of the address pointer to the 'max-length' operand. The VL operand of the CALL macro automatically sets the `x'80'` bit in the last operand address pointer.

On return from the CSIPACK module, the function byte will be set as follows:

"0"	Call successful
"1"	Invalid function
"2"	GETVIS macro failed
"3"	CDLOAD macro (of either CSIPACCM or CSIPACCM) failed
"4"	FREEVIS macro failed
"5"	'Length' greater than 'Max-Length'

Also, the length of the record after expansion or compression will be returned in the 'length' parameter.

For VSE, the user interface is supplied as both a relocatable module and a phase, both called CSIPACK. The phase is primarily supplied to enable access from software products that require called programs to be stored in a Core Image (phase) format.

## User Compression Sample Program

The COBOL source coding below contains a part of a program which builds a (blocked) variable-length sequential file of records in M-Pact-PACK compressed format, using the CSIPACK called subroutine. Following that is replacement logic which could be used to expand the file for processing.

```
FILE SECTION.
FD SQFILE
    LABEL RECORDS ARE STANDARD
    RECORDING MODE IS V
    BLOCK CONTAINS 16 RECORDS.
01 SQFILE-RECORD.
    05 SQFILE-RECLEN                PIC S9(3) COMP.
    05 SQFILE-RECDATA.
        10 FILLER                OCCURS 1 TO 400 TIMES
            DEPENDENT ON SQFILE-RECLEN PIC X.
WORKING-STORAGE SECTION.
77 PACK-FUNCTION                    PIC X.
77 PACK-LENGTH                    PIC S9(3) COMP.
77 PACK-MAXLENGTH                  PIC S9(3) COMP.
77 PACK-OFFSET                    PIC S9(3) COMP.
01 PACK-RECORD                    PIC X(400).
PROCEDURE DIVISION.
OPEN-FILES.
    OPEN OUTPUT SQFILE.
MAIN-LOOP.
    MOVE 'A' TO PACK-FUNCTION.
    MOVE 400 TO PACK-MAXLENGTH.
    MOVE 400 TO PACK-LENGTH.
    MOVE 0 TO PACK-OFFSET.
    * COMPRESSION MODE 1 OF A 400-BYTE RECORD.
    *
    * 'PACK-RECORD' WOULD BE LOADED HERE WITH 400 BYTE
    * COMPRESSED DATA RECORD.
    * LOOP UNTIL INPUT SOURCE EXHAUSTED.
    *
    CALL 'CSIPACK' USING PACK-FUNCTION, PACK-RECORD,
        SQFILE-RECDATA, PACK-LENGTH,
        PACK-MAXLENGTH, PACK-OFFSET.
    IF PACK-FUNCTION NOT EQUAL '0',
        DISPLAY 'PACK-ERROR, FUNCTION = ', PACK-FUNCTION,
        GO TO END-RUN.
    MOVE PACK-LENGTH TO SQFILE-RECLEN.
    WRITE SQFILE-RECORD.
    GO TO MAIN-LOOP.
```

The statements below could be used to expand the file compressed above:

```
OPEN INPUT SQFILE.
MAIN-LOOP.
  READ SQFILE AT END GO TO END-RUN.
  MOVE SQFILE-RECLEN TO PACK-LENGTH.
  MOVE 'E' TO PACK-FUNCTION.
  MOVE 400 TO PACK-MAXLENGTH.
  MOVE 0 TO PACK-OFFSET.
* EXPAND PREVIOUSLY COMPRESSED FILE OF 400-BYTE RECORDS.
* EXPANDED RECORD PUT INTO 'PACK-RECORD' AREA.
  CALL 'CSIPACK' USING PACK-FUNCTION, PACK-RECORD,
    SQFILE-RECDATA, PACK-LENGTH,
    PACK-MAXLENGTH, PACK-OFFSET.
  IF PACK-FUNCTION NOT EQUAL '0',
    DISPLAY 'PACK-ERROR, FUNCTION = ', PACK-FUNCTION,
    GO TO END-RUN.
  GO TO MAIN-LOOP.
```

## Compressing DL/I Segments

It is possible to compress and expand DL/I segments using M-Pact-PACK/VSE, but this is only possible for VARIABLE LENGTH segments. To assist the user in this respect, we provide an interface module "CSIPACDL". The steps to be taken are as follows:

- Unload the database using the standard IBM utility.
- Set up the DBD with M-Pact-PACK support. This involves adding the parameter "COMPRTN=CSIPACDL" to the SEGM statement for each segment you wish to be compressed. "COMPRTN" is a standard exit which IBM makes available for the purpose of compression and expansion. Each time a write or read of the segment is required, the routine CSIPACDL will be entered to provide the necessary compression and expansion.
- Generate the ACB with the standard IBM utility.
- Reload the database with the standard IBM utility.

The database will be reloaded with the segments compressed. Thereafter compression and expansion of the requested segments will happen automatically.

## Questions and Answers

A selection of common questions and answers posed by "M-Pact-PACK" users.

### **How will I see my records in a dump?**

The data in your program's I/O areas will never be in COMPRESSED format. Compression and expansion takes place in areas acquired by M-Pact-PACK. However, if you delve into the depths of VSAM, you may see a compressed control interval.

### **How can I send data to another installation?**

To de-compress a file to tape, simply REPRO it. The de-compression will take place automatically. Similarly, should you wish to de-compress a file onto disk for any reason, again use REPRO. If the output file has not been defined with an EEXT in M-Pact-PACK format, de-compression will take place.

### **How can I tell if a file is compressed?**

This is a tricky one! The system was designed to be COMPLETELY transparent to the user. One way is to LISTCAT the file to see whether it has an EEXT of M-Pact-PACK format. You could ALTER the file to have an EEXT of non-M-Pact-PACK format, and then use a VSAM PRINT command. The file will then be printed in its "raw" state.

### **I read a file in a SORT exit. Will it work on a file compressed by M-Pact-PACK?**

Yes. You will be returned an expanded record.

**How can I gauge the anticipated CPU overhead of M-Pact-PACK?**

You can't. The overhead is dependent on the compression mode chosen, and the contents of the record. In our experience, Basic compression will involve the least overhead. Full compression could involve an overhead of as much as 15% under extreme circumstances. However, the expansion process is much, much, faster than compression, so if a file is read substantially more than it is written, the overhead will be diminished. In many circumstances, the CPU overhead involved will be more than compensated for by the decrease in I/O time.

**Do I need to allocate extra storage to run M-Pact-PACK?**

As previously stated, M-Pact-PACK expands and compresses records into its dynamic work areas. These work areas are acquired by "GETVIS" as necessary. The program attempts first to acquire an area of length 33K. This is more than adequate in normal circumstances for all M-Pact-PACK requirements. If this amount of (contiguous) storage is not available in a partition, M-Pact-PACK dynamically lowers its request, until storage CAN be obtained. This lower allocation may be at the expense of performance. Should M-Pact-PACK not be able to acquire any GETVIS, a message is displayed on the console, and the VSAM return code indicating "Insufficient GETVIS" is returned to the user, exactly as if VSAM itself had detected the error. Data integrity is assured.

**How do I add an alternate index to a M-Pact-PACK file?**

If the new alternate index key finishes at a lower position than the prime key, and any existing alternate indexes, just build the new index as normal. If the new index is outside this range, REPRO the file to another disk or tape, re-DEFINE the file with a different EEXT parameter based on the new AIX position, REPRO the file back in, and build the new index. Read section "Compression Range" on page 7 for full explanation.

**What if M-Pact-PACK won't start up?**

This is a MOST unlikely event. In a testing environment, or one that has few, unimportant, compressed files, this may not be a problem. However, in a production environment whose files are heavily compressed, it could be pretty disastrous to let the system continue, since records read will not be expanded etc. In the former case, the operator should reply "YES", to allow the system to continue. In the latter case, however, he or she should ascertain the correct course of action from their technical support people.

## **What if I need to DITTO print a M-Pact-PACK file?**

No problem. Use DITTO functions VDP and VPR as normal and DITTO will print the record in expanded format. If however, you use DITTO to print from a physical disk location using functions DP, DD, DPD, or DDD then the resulting print will show the records in compressed format.

## **Why did I get less (more) compression than CSIPACAN projected?**

This is because the CISIZE may be a better or worse choice for the new average (compressed) record size. However, it should be fairly close to the projected savings, and the CISIZE can, of course, be adjusted.



**MESSAGES**



<b>CSIPACK-1</b>	<b>RELEASE x.xx ACTIVATED [EXPIRES IN nn DAYS]</b> Program CSIPACST has successfully started the M-Pact-PACK system. The expiration note will appear if the product is within 30 days of expiration. Contact support representative.
<b>CSIPACK-2</b>	<b>CSIPACCM PHASE NOT IN SVA</b> Program CSIPACST has failed to locate phase CSIPACCM, or it is not located in the SVA. Ensure correct libraries have been accessed, then rerun CSIPACST.
<b>CSIPACK-3</b>	<b>CSIPACEX PHASE NOT IN SVA</b> Program CSIPACST has failed to locate phase CSIPACEX, or it is not located in the SVA. Ensure correct libraries have been accessed, then rerun CSIPACST.
<b>CSIPACK-4</b>	<b>IKQVRM PHASE NOT FOUND</b> Program CSIPACST has failed to locate phase IKQVRM. Ensure correct libraries have been accessed, then rerun CSIPACST.
<b>CSIPACK-5</b>	<b>IKQVRM PHASE NOT IN SVA</b> M-Pact-PACK requires the VSAM phase IKQVRM to be in the SVA. Reload the SVA and then rerun CSIPACST.
<b>CSIPACK-6</b>	<b>CSIPACSV PHASE NOT FOUND</b> Program CSIPACST has failed to locate phase CSIPACSV. Ensure correct libraries have been accessed, then rerun CSIPACST.
<b>CSIPACK-7</b>	<b>CSIPACSV PHASE NOT IN SVA</b> Program CSIPACST requires phase CSIPACSV to be in the SVA, reload the SVA and then rerun CSIPACST.
<b>CSIPACK-8</b>	<b>SYSTEM ALREADY STARTED</b> Program CSIPACST has already been run during this VSE session and the M-Pact-PACK system is already active.
<b>CSIPACK-9</b>	<b>SDL ENTRY NOT FOUND</b> Program CSIPACST can not find the SDL entry for either IKQVRM or CSIPACSV. DUMP the SVA and check for the above entries in the SDL. Contact your support representative if problem cannot be resolved.
<b>CSIPACK-10</b>	<b>SDL FORMAT INCORRECT</b> Program CSIPACST has detected an invalid SDL format. DUMP SVA and report problem to your support representative.

- CSIPACK-98**            **M-Pact-PACK HAS EXPIRED**  
Program CSIPACST cannot startup due to the product having expired. (Verify system date is correct.) Contact support representative immediately.
- CSIPACK-99**            **RUN TERMINATING - ENTER "YES" TO CONTINUE**  
Program CSIPACST has detected an error condition and is now terminating. See previous message for reason for termination. In a testing environment, or one that has few, unimportant, compressed files, this may not be a problem. However, in a production environment whose files are heavily compressed, it could be pretty disastrous to let the system continue, since records read will not be expanded etc. In the former case, the operator should reply "YES", to allow the system to continue. In the latter case, however, he or she should ascertain the correct course of action from their technical support people.
- CSIPACK-101**          **ANALYSING FILE xxxxxxxx, RECORDS READ nnnnnn**  
Program CSIPACAN has issued this message in response to an interrupt to the partition (e.g., MSG pp). xxxxxxxx is the name of the file currently being processed and nnnnnn is the number of records read so far.
- CSIPACK-102**          **ENTER OPTION :- C=CONTINUE,S=SKIP TO NEXT FILE,E=END**  
Program CSIPACAN has issued this message in response to an interrupt to the partition (MSG pp). To continue processing enter "C", if you wish to go on to the next file for analysis enter "S". To stop the run enter "E". In either of the last two cases the analysis so far will be printed on SYSLST.
- CSIPACK-103**          **CDLOAD CSIPACEX FAILED**  
Program CSIPACAN cannot load phase CSIPACEX, either because it is not in an available library, or there is insufficient GETVIS available to process the CDLOAD request. Correct the problem and rerun.
- CSIPACK-104**          **CDLOAD CSIPACCM FAILED**  
Program CSIPACAN cannot load phase CSIPACCM, either because it is not in an available library, or there is insufficient GETVIS available to process the CDLOAD request. Correct the problem and rerun.
- CSIPACK-110**          **MODULE xxxxxxxx NOT FOUND**  
Program CSIPACAN cannot load phase xxxxxxxx, because it is not in an available library. Correct the problem and rerun.
- CSIPACK-111**          **VSAM ERROR, PREVIOUS LINE EXPLAINS THE ERROR**  
A VSAM error such during a GENCB, SHOWCB, or OPEN has occurred. The exact error is displayed in the previous text line. Request technical assistance for this error.

<b>CSIPACK-112</b>	<b>SYNVSAM ERROR</b> A VSAM physical error has occurred. Request technical assistance for this error.
<b>CSIPACK-113</b>	<b>LERVSAM ERROR</b> A VSAM logical error has occurred. Request technical assistance for this error.
<b>CSIPACK-114</b>	<b>SYSIN ERROR, xxxxxxxxxxxxxxxxxxxxxxxxx</b> A parameter error was detected in the input. The text of the message will describe the exact error detected. Correct the problem and rerun.
<b>CSIPACK-201</b>	<b>PHASE CSIPACEX NOT IN SVA</b> Program CSIPACSV has failed to load CSIPACEX. This message should not occur as CSIPACST will ensure that this phase is in the SVA.
<b>CSIPACK-202</b>	<b>PHASE CSIPACCM NOT IN SVA</b> Program CSIPACSV has failed to load CSIPACCM. This message should not occur as CSIPACST will ensure that this phase is in the SVA.
<b>CSIPACK-203</b>	<b>NOT ENOUGH GETVIS FOR CONTROL AREA</b>
<b>CSIPACK-204</b>	<b>NOT ENOUGH GETVIS FOR WORK AREA</b>
<b>CSIPACK-205</b>	<b>NOT ENOUGH GETVIS FOR ACB AREA</b>
<b>CSIPACK-206</b>	<b>NOT ENOUGH GETVIS FOR BUFFERS</b>
<b>CSIPACK-207</b>	<b>NOT ENOUGH GETVIS FOR WORK AREA</b> Program CSIPACSV cannot get enough GETVIS, to process the VSAM file request. Increase the size of your partition GETVIS.
<b>CSIPACK-208</b>	<b>VSAM CONTROL BLOCKS NOT SETUP</b> Program CSIPACSV has found an error in the VSAM control blocks. Ensure the file was opened correctly, before a VSAM request was issued. If problem persists dump the problem program after the file has been opened and contact your support representative.
<b>CSIPACK-209</b>	<b>NOT ENOUGH GETVIS FOR I/O AREAs</b> Program CSIPACSV can not get enough GETVIS to process this VSAM request. Increase the size of your partition GETVIS.
<b>CSIPACK-210</b>	<b>FREEVIS WRONG AREA</b> Program CSIPACSV has tried to FREEVIS storage, and has found that the area has been overwritten. Ensure problem program is correctly using GETVIS. If problem persists dump partition and contact your support representative.

<b>CSIPACK-211</b>	<b>UNABLE TO FREEVIS I/O AREAS</b> Program CSIPACSV has received an unexpected return code when trying to release its I/O areas. If problem persists dump partition and contact your support representative.
<b>CSIPACK-212</b>	<b>UNABLE TO FREE WORK AREA</b>
<b>CSIPACK-213</b>	<b>UNABLE TO FREE BUFFER AREA</b>
<b>CSIPACK-214</b>	<b>UNABLE TO FREE WORK AREA</b> Program CSIPACSV has received an unexpected return code when trying to release area one of its control areas. If problem persists dump partition and contact your support representative.
<b>CSIPACK-215</b>	<b>UNSUPPORTED CALL VIA IIP</b> Program CSIPACSV has received a "PUT LOCATE" type call from the user program. These types of calls are generated by the ISAM Interface Program (IIP), and are not supported by M-Pact-PACK. The request is aborted, and a physical data-write error code is passed back to the caller.
<b>CSIPACK-300</b>	<b>CSIPACTB TABLE NOT FOUND</b> Table CSIPACTB cannot be loaded, because it is not in an available library. Correct the problem and rerun.
<b>CSIPACK-301</b>	<b>TABLE NOT FOUND ON JPQ</b> Request technical assistance for this message.
<b>CSIPACK-302</b>	<b>FREEMAIN ERROR, (description of error)</b> Request technical assistance for this message.
<b>CSIPACK-303</b>	<b>GETMAIN ERROR, (description of error)</b> Request technical assistance for this message.
<b>CSIPACK-304</b>	<b>NO BUFFER ASSOCIATED WITH RPL</b> Request technical assistance for this message.
<b>CSIPACK-305</b>	<b>COMPRESS TYPE NOT VALID</b> Valid compression types are:1, 3, 5, and 7. Correct the problem and rerun.
<b>CSIPACK-307</b>	<b>MODULE xxxxxxx NOT FOUND</b> Module xxxxxxx cannot be loaded because it is not in an available library. Correct the problem and rerun.
<b>CSIPACK-308</b>	<b>SHOWCB ERROR, (description of error)</b> Request technical assistance for this message.

CSIPACK-309	<b>NO PARM SPECIFIED ON EXECUTE CARD</b> CSIPACST requires a PARM to be specified. Correct the problem and rerun.			
CSIPACK-310	<b>INVALID PARM SPECIFIED</b> CSIPACST requires a PARM to be specified. Correct the problem and rerun.			
CSIPACK-311	<b>TABLE CANNOT REFRESH, NOT LOADED</b> Attempting to use PARM='REFR' with CSIPACST while M-Pact-PACK is not started. Use PARM='STRT' to start M-Pact-PACK instead.			
CSIPACK-312	<b>TABLE ALREADY LOADED</b> Attempting to use PARM='STRT' with CSIPACST while M-Pact-PACK is already started. Use PARM='REFR' to refresh the CSIPACTB table.			
CSIPACK-313	<b>CDE ERROR, (description of error)</b> Request technical assistance for this message.			
CSIPACK-314	<b>ERROR RETURNED FROM ROUTER, (description of error)</b> Request technical assistance for this message.			
CSIPACK-315	<b>SUBPOOL NOT FOUND</b> Request technical assistance for this message.			
CSIPACK-316	<b>M-Pact-PACK RELEASE x.xx HAS BEEN</b> <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td><b>STARTED</b></td></tr><tr><td><b>STOPED</b></td></tr><tr><td><b>REFRESHED</b></td></tr></table> This is an informational message in response to a CSIPACST execution to confirm that the requested function has been performed.	<b>STARTED</b>	<b>STOPED</b>	<b>REFRESHED</b>
<b>STARTED</b>				
<b>STOPED</b>				
<b>REFRESHED</b>				
CSIPACK-400	<b>INVALID FUNCTION SPECIFIED</b> An invalid function has been provided to the CSIPACK user program interface. Correct the problem and rerun.			
CSIPACK-401	<b>LENGTH GREATER THAN MAX LENGTH</b> A record length greater than the maximum define to VSAM for the dataset has been provided to the CSIPACK user program interface. Correct the problem and rerun.			
CSIPACK-402	<b>MODULE xxxxxxx NOT FOUND</b> Module xxxxxxx cannot be loaded because it is not in an available library. Correct the problem and rerun.			