When IBM Says You Can't ... Sometimes You Can: Part II

BY SAM GOLOB

I admit that I'm very spoiled by my systems programmer tool collection. With it, I can perform tasks that other people can't. But I'm willing to share my "secrets." That's what this column is all about. Any tool or trick that makes our working life easier, reduces system down-time, or helps the system to run more efficiently is fair game for discussion.

IBM supplies many tools with the MVS and OS/390 operating systems. However, some of these tools are very basic, functional and user-unfriendly, such as AMASPZAP. Others are very complex, quite friendly and have lots of features,such as ISPF and PDF. Still others are very complicated, not too friendly but do a good job, such as SMP/E. Just about everything that IBM supplies works. But no matter how many tools IBM specifically supplies, there's always room for improvement. That's because IBM (realistically) never made a goal for itself to supply every tool a user will ever need. Rather, IBM delivers very functional basic tools with robust processing structures that allow users and vendors to further build upon.

I'd say that MVS and OS/390 are quite "accessible systems." By this I mean users can write programs to run under MVS that perform functions IBM never intended to write. For example, I don't think IBM ever intended to release a program to dynamically load new versions of JES2 exits. Another example:Initially, the SPF and ISPF designers never designed one multi-function member list menu under which you could BROWSE, EDIT, RENAME, DELETE, add or change ISPF statistics, or examine many internal details of a pds member all from the same member list. A third example: Although IBM recently came out with an interactive disassembler (load module to assembler source code), they haven't yet developed a convenient interactive delinker (load module to 80-byte object decks). IBM hasn't done any of these things. But users have. And their tools are largely built on the building blocks that IBM itself has supplied.

I think that IBM deserves praise for building products that provide the foundation for adding more function. However, the function won't come unless the users or vendors take it upon themselves to do the development work.

I think that IBM deserves praise for building products that provide the foundation for adding more function. However, the function won't come unless the users or vendors take it upon themselves to do the development work. Fortunately, many users have already done a lot of work and their products are available to the public. The result is large collections of free tools for MVS that everyone can use if they'd take the time to obtain and install them.

One of the largest collections of tools is the CBT MVS Utilities Tape and its companion, the CBT Overflow Tape. These are independently produced collections of machine-readable software and other goodies, most of which are in source code form. The CBT Tapes contain documentation about other collections as well. Information about the CBT Tapes can be obtained at web site http://members.aol.com/cbttape. The tapes themselves can also be obtained from NaSPA. Using the COPYMODS program from File 229, you can make copies of the CBT Tape for others.

Let's examine a few of the tasks that IBM says you can't do. Just because IBM doesn't provide tools to perform the following tasks doesn't mean that these tasks can't be done. On the contrary:I'll show you how to use tools from the CBT MVS Tape to get the job done.

CLIPPING A PACK WHILE IT'S ONLINE

"Clipping" a disk pack is another name for changing its volume serial name. In other words, if the pack's id was PACK01, you might need to change it to PACK02. IBM's standard method for doing this is to take the pack offline and to run its low-level disk pack support program,ICKDSF, in a batch job to change the id. This works well, as do almost all of the IBM-supplied utilities. The only problem is that you have to take the pack offline first.

Normally, taking a disk pack offline is not a big problem. You go to the operator console and vary the unit offline. If the unit is currently being used, it won't come offline right away, but it will remain "pending offline" until all of the active tasks using that pack have terminated. However, let's say there's an emergency and you can't wait for the pack to go offline, or you can't get it to go offline at all. Then, you need to have another technique up your sleeve.

Enter the UCLA Fullscreen ZAP program. Originally written by a group of systems programmers at UCLA, Fullscreen ZAP is made to change bytes of data in any dataset, not just load modules. The ZAP program is a TSO Command Processor designed to work under TSO. ZAP shows you a full screen of data in a record (really a block) from the dataset and allows you to examine or change any byte. Additionally, Fullscreen ZAP has its own built-in "help" information

and it's a lot more user-friendly than IBM's AMASPZAP program. Fullscreen ZAP is powerful stuff, but it's not ready for our pack clipping needs, just yet.

Along came Greg Price who added a super-powerful feature to the Fullscreen ZAP program — the FULLVOL keyword. If you're running ZAP as an authorized TSO command (normally it doesn't have to run authorized), and you add the FULLVOL keyword after the dataset name, some high power "magic" happens. Before opening the dataset, ZAP (under FULLVOL) manipulates the DEB (Data Extent Block) for the dataset to say that the extents for the dataset encompass the entire pack. So instead of showing you the beginning of the dataset, ZAP, under FULLVOL, shows you the beginning of the pack — cylinder 0, track 0, record 1!

Now, you can begin to see what I'm getting at. Where is a disk pack's id record? It's on cylinder 0, track 0, record 3. After specifying a dataset that's on the pack you want to clip (you can use 'FORMAT4.DSCB' as a dataset name, with the VOL( ) keyword to specify the particular pack), you add the FULLVOL keyword and Fullscreen ZAP will place you just two records before the record you want to see. You can advance two records by entering the "R" subcommand twice and there you are! The volume id field can be reached at 8 bytes off the beginning of the third record, so you enter the subcommand "+8" to get there. The next 6 bytes are the volume id. You can enter an "S" subcommand to replace the id with another name and you use the "ZAP" subcommand of ZAP to write the changes to disk. The pack is now clipped. On newer MVS systems and on OS/390 systems the new pack id will go into effect when the pack goes offline and comes back online again. On older systems such as MVS/XA, you vary the pack offline and you have to do a MOUNT command, specifying the new volume id, after the pack is varied online again.

Here's one of my own war stories. I once had a problem where there were two identically-named packs on the same "running MVS system." The first one, with the lower unit name, had incorrect data on it. The second one, which had been restored from a backup, had the correct data. For some reason, we had much difficulty getting the first pack off-line. It was during production time, and we had to re-IPL very soon. I'm sure you know what would happen after the IPL. The wrong pack, which had the lower unit address, would come on-line again, unless you entered the proper operator response to the "duplicate id" message. Our operators (on a different floor) weren't trained correctly for the reply. Fortunately, I had a quick solution. I placed a dummy "uniquely named" dataset on the "wrong" volume and invoked ZAP for that dataset using the FULLVOL keyword. Then I changed that pack's volume serial name as described above. After the IPL, the correct pack came up under its own volser name and the old pack came up under its new name. See how useful such a technique is?

PRESERVING ISPF STATS AFTER IEBUPDTE SEQUENTIALIZATION

I don't know how important ISPF statistics are to you, but they often are invaluable to me. ISPF statistics show how large a pds member is. They show who updated the member last and when. When you're looking through a library of source code, the ISPF statistics will provide you with tell-tale information and necessary history for each member. If you're moving a "source-type" pds from one location to another, I'd bet you'd really prefer to preserve the ISPF statistics of the members.

IEBUPDTE is an IBM utility that is most often used for updating a card-image source code member with an incremental update using its ./ CHANGE NAME=memname control cards. IEBUPDTE can also be used to "string out" all (or several) members of a partitioned dataset into a single sequential dataset with the member data strung out, one member after another. Data from one member is distinguished from another by an IEBUPDTE header card in the form:

```
./ ADD NAME=memname
```

This format separates each member from the next. You can reconstitute the partitioned dataset from the sequentialized IEBUPDTE-format dataset by another invocation of IEBUPDTE, with PARM=NEW. Each sequentialized member will be stowed as a separate new member by the IEBUPDTE program, which reads the ./ ADD cards, to act as separators.

IEBUPDTE has one disadvantage in this process. IBM invented IEBUPDTE before it invented ISPF statistics. The IEBUPDTE utility has a provision to preserve SSI information, which was present in "ancient times", but it simply has no provision to include control information about ISPF statistics in its ./ ADD cards. So the stowed members, after an IEBUPDTE sequentialization and reload of a pds, do not have the ISPF statistics preserved. They simply get lost.

Enter some clever users who created and standardized their own system of preserving ISPF statistics in ./ ADD cards. This system has become so popular that we don't need IBM. Several programs on the CBT Tape will perform the sequentialization and create ./ ADD cards with the ISPF stats in this format, and several other programs will reconstitute pds members from this format with the ISPF statistics preserved. See Figure 1 for the format of the ./ ADD cards that have the ISPF stats.

Programs that sequentialize a pds and preserve ISPF stats are OFFLOADW from File 093 of the CBT Tape, REVIEW from File 134, and LISTPDS from File 316 of the CBT Tape, which uses the following parms:

```
//    PARM='UPDTE(><),SPF,NOLIST,DECK,NOSEL' ,
```

To use REVIEW under TSO for this purpose, allocate FILE(SYSUT2) to an already created sequential output dataset,REVIEW a pds member, use the DIR subcommand while under REVIEW to get a directory listing, and from the directory listing, invoke the =OFFLOAD subcommand. In addition, the vendor product STARTOOL (from SERENA International) with its COMBINE subcommand, adheres to these same conventions.

Programs that reload a partitioned dataset from a sequential version and preserve ISPF stats are: PDSLOADW from File 093 of the CBT Tape, and the SEPARATE function of the vendor product STARTOOL. In addition, OFFLOADW and PDSLOADW aren't restricted to LRECL=80. They both can handle fixed blocked datasets with record lengths up to 256.

OLD LINKAGE EDITOR RESTRICTIONS

You may not know it, but you can still execute the old linkage editor (as opposed to the DFSMS binder) by executing PGM=HEWLKED instead of PGM=HEWL. The old linkage editor, which is still distributed with DFSMS on OS/390, currently restricts the block size of card-image object decks to 3200. This is a silly restriction from OS/MFT days that was finally eliminated only with the introduction of the binder.

Figure 1: PDS Members With ISPF Statistics and ./ADD NAME = Cards

This figure illustrates pds members with ISPF Statistics (listed first) and the corresponding ./ ADD NAME= cards that preserve these ISPF statistics in the "standard"LISTPDS format.

```
   NAME      TTR    VV.MM   CREATED     LAST MODIFIED   SIZE   INIT   MOD     ID
$$$COPYF   013901   01.03   90/11/28    96/05/06  19:20   32     23     0    SBGCSC
$$$SAM1    013903   01.00   95/03/16    95/03/16  10:52   48     48     0    SYSPAJA
$$$SAM2    013905   01.00   95/10/26    95/10/26  12:10   31     31     0    SYSPAJA
$$$SAM3    013907   01.00   96/03/15    96/03/15  11:45   30     30     0    SYSPAJA
$$JCL003   019901   01.99   92/02/05    98/06/28  11:45  1614   1433    0    SAGOLOB

1          10         20         30         40         50         60         70
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
./ ADD NAME=$$$COPYF   0103-90332-96127-1920-00032-00023-00000-SBGCSC
./ ADD NAME=$$$SAM1    0100-95075-95075-1052-00048-00048-00000-SYSPAJA
./ ADD NAME=$$$SAM2    0100-95299-95299-1210-00031-00031-00000-SYSPAJA
./ ADD NAME=$$$SAM3    0100-96075-96075-1145-00030-00030-00000-SYSPAJA
./ ADD NAME=$$JCL003   0199-92036-98179-1145-01614-01433-00000-SAGOLOB
```

Figure 2: Fix to the DFSMS Version of the Old Linkage Editor PGM=HEWLKED

This fix allows object decks to be blocked in larger than 3200-byte blocks. Notice how simple this fix is. It can be fitted to any version of the linkage editor.

```
++ USERMOD(MDZ11CA)  /* ALLOW LKED TO READ OBJ BLKSIZE UP TO 32720) */.
++ VER(Z038) FMID(HDZ11C0).
++ ZAP(HEWLFINT) DISTLIB(AOS04).
  NAME HEWLFINT
*                          DC     H'3200'
VER 000AE0 0C80
*                          DC     H'40'    ( = 3200/80 = NO OF RECDS/BLK)
VER 000F42 0028
*                          DC     H'32720'
REP 000AE0 7FD0
*                          DC     H'409'   ( = 32720/80 = NO OF RECDS/BLK)
REP 000F42 0199
  IDRDATA MDZ11CA
++ ZAP(HEWLFAPT) DISTLIB(AOS04).
*                          DC     H'40'    ( = 3200/80 = NO OF RECDS/BLK)
VER 000216 0028
*                          DC     H'409'   ( = 32720/80 = NO OF RECDS/BLK)
REP 000216 0199
  IDRDATA MDZ11CA
```

IBM didn't remove this restriction of the old linkage editor for all these years. What's so surprising is that the restriction is artificial and is hard-coded into the linkage editor code, even in the DFSMS version of HEWLKED. You can re-code the maximum block sizes for object decks to a block size of 32720 and a blocking factor of 409 by zapping a few places in the linkage editor. Sample zaps to do this are available on File 257 of the CBT Tape, and I've shown one of them in Figure 2. This illustrates how easily an age-old IBM restriction can sometimes be removed.

In summary, every time IBM mentions a restriction that bothers you, take it with a grain of salt. Sometimes, you can't get around it. However, more often than not, you can. If you have a question about some IBM restriction, post a query on the IBM-MAIN Internet forum (see my April 1998 column). Many of the people who monitor that forum are knowledgeable in free system tools. There may very well be a way around the restriction after all. *ts*

Documentation for the CBT MVS Tapes can be found on the web at http://members.aol.com/cbttape.

NaSPA member Sam Golob is a senior systems pro - grammer. He also participates in library tours and book signings with his wife, author Courtney Taylor. Sam can be contacted at sbgolob@aol.com or sbgolob@ibm.net.