PDS/E: The Extended PDS

BY JIM MOORE

For many years, MVS programmers of all stripes have had to deal with the peculiarities of the partitioned access method (or PAM, DSORG=PO). Older PDS datasets are still widely used. However, now, there is an alternative: The PDS/E or extended PDS. PDS/E datasets go a long way toward solving most of the problems associated with the older PAM access method. There is no limit to the number of directory blocks; they reuse space internally and never need to be compressed or "de-gassed."

This month, I'll examine PDS/E datasets and point out what you need to know to use them in your day-to-day work. As always, please verify your site's release levels and other site-specific factors when considering the information contained here.

PDS/E DATASETS MUST BE SMS-MANAGED

The first requirement for allocating a PDS/E dataset is SMS. OS/390 will not allow a non-SMS managed PDS/E. If you are using ISPF option 3.2 to allocate a new PDS/E, this means that you must use the M (Managed) option. At ISPF 3.2, M option, you indicate that you want a PDS/E by using the LIBRARY parameter in the Data Set Name Type field. All other fields would be entered as if you were allocating an older style PDS. Note that directory blocks are not really required when the LIBRARY type is used. Something else that I have noticed at two separate sites is an apparently less-than-ideal system-determined BLKSIZE (SDB) for PDS/E datasets. Normally, when allocating an older style PDS, if you omit the BLKSIZE value at allocate time, a half-track block will be assigned by the SDB routines. At both of the sites where I have allocated PDS/E datasets, I have gotten a 32K block size when relying on SDB. I would appreciate any feedback on this. I prefer to have consistent blocking factors for all of my own PDS datasets. So I force the PDS/E datasets that I allocate to half-track optimal block sizes by actually coding the BLKSIZE.

The first requirement for allocating a PDS/E dataset is SMS. OS/390 will not allow a non-SMS managed PDS/E.

BATCH ALLOCATION OF PDS/E DATASETS

To allocate a PDS/E in batch, you need to use the new DSNTYPE keyword. There is nothing in the DCB parameters that indicates the extended PDS attribute. Again, the LIBRARY parameter is used:

```
//PDSE     DD DSN=A.PDS.E,DSNTYPE=LIBRARY.....
```

All other JCL parameters would look just like an older PDS allocation including DSORG=PO and SPACE=(units,(prime,dir,sec)). The same SDB problem (32K vs. half-track) occurs in batch PDS/E allocations. So, you may want to code BLKSIZE on a batch PDS/E allocation.

WHY PDS/E DATASETS ARE SUCH AN IMPROVEMENT

One of the nastiest things about the older style PDS datasets is that you are forced to make a directory block "guesstimate" up front at PDS allocation time. Most people have no idea what to code for number of directory blocks. So typically, directory blocks are either vastly over or under allocated. When directory block space is all used up, a STOW failure with an x37 ABEND will occur. This is frequently seen in ISPF profile datasets. **Tip:** Convert all of your ISPF profile PDSes to PDS/E datasets and eliminate this problem forever!

The internal architecture of traditional PDS datasets places all of the 256-byte directory blocks at the beginning of the allocated primary extent. When their space is exhausted, the PDS must be re-allocated with additional directory blocks, followed by a copy of all members to the new file. Or, if you have software such as SERENA International's StarTool or its cousin, PDS, you can allocate new directory blocks to the existing PDS.

With PDS/E datasets, when the "front of extent" directory blocks are exhausted, new ones are allocated in much the same manner as normal secondary data extents. That is, the directory blocks are treated like data extents. With a PDS/E, it is not a requirement that all of the directory blocks be at the front of the file.

Call for Reader Experiences

If you have any information or experience with PDS/E datasets, good, bad or indifferent, please send me an email at conlogco@ix.netcom.com. In a future column, I will present a compilation of PDS/E experiences from the trenches. Please indicate if I can use your name and/or company name in published form.

Another "feature" of older PDS datasets is that data is never written back to the same TTR that it was read from, thus, causing them to eventually become full. Again, unless you have some specialized software, you can't access these prior copies that reside at different TTRs within the file's extents. When all 16 extents are exhausted and a member needs to be written back to the PDS, a compress will have to be done to reclaim wasted internal space. IEBCOPY is the most frequently used utility when performing a PDS compress. The in-place compress "de-gasses" the entire PDS and places only the most current data member's TTR in the directory.

PDS/E datasets do rewrite to the same TTR that they read from. If the member being written has increased in size, new space is allocated from unused portions of the dataset and chained to the original TTR. If the member has decreased in size, the unused space is again made available. Isn't this the way that PDS datasets should have always worked?

USES FOR PDS/E DATASETS

Basically, a PDS/E can be used any place that an older PDS can be used. However, you must be careful with concatenations. I will explain why shortly. I use concatenations for object code libraries, source code libraries, load libraries and ISPF profile datasets. They are especially good for heavily updated PDS datasets such as object and load libraries. When PDS/E datasets were first introduced they could not be used as load libraries. This was because the binder (linkage editor) could not properly update their relocatable directories. However, at current OS/390 levels, Version 2 and higher, they can be used as load libraries.

To convert all PDS datasets to PDS/E datasets site-wide certainly would be an ideal goal. However, it may not be as easy as it seems. It seems worthy of an impact analysis.

Most IBM-supplied software will support a mixture on PDS/E and traditional PDS concatenations. However, some third-party software might not. I have mixed PDS and PDS/E libraries at the following places where concatenations are frequently found:

- **SYSLIB** - COBOL for MVS, High-Level Assembler and IEWL (Binder)
- **JOBLIB** - Standard JCL
- **JCLLIB** - Standard JCL

Really, it probably depends upon what type of BLDL macro was used when the software build was done for a product. Some older versions of software might be using old versions of BLDL.

CONCLUSION

To convert all PDS datasets to PDS/E datasets site-wide certainly would be an ideal goal. However, it may not be as easy as it seems. It seems worthy of an impact analysis.

Even if a global conversion to PDS/E cannot be done at your site, that doesn't mean that individual users can't start exploiting PDS/E datasets. You will notice no differences when using them in everyday things such as ISPF and link-editing. One little thing: You may forget how to do an in-place compress! ts

NaSPA member Jim Moore is the president of Concentrated Logic Corporation, a Glendale Heights, Ill.-based software development firm specializing in TSO/ISPF/PDF and database design. He can be reached at conlogco@ix.netcom.com.