#SHAREorg

COBOL for z/OS V5 and PDSE load libraries

Tom Ross

March 2014

SHARE in Anaheim

PDSE requirement for COBOL V5 executables

- COBOL V5 executables are not “load modules”. They are “program objects”. Load modules reside in a PDS dataset. Program objects can only reside in a PDSE dataset (or z/OS UNIX file).
- Therefore, customers using PDS load libraries for COBOL executables must migrate to PDSE load libraries prior to creating COBOL V5 executables. There is no alternative to converting.
- If interested in COBOL V5, start migrating COBOL load libraries to PDSE datasets ASAP!
- Now, why PDSE datasets and why are PDSE datasets better than PDS datasets?

First some history about PDS datasets

- When using PDS datasets for load libraries, customers had problems with :
  - The need for frequent compressions,
  - Loss of data due to the directory being overwritten
  - Performance impact due to a sequential directory search
  - Performance delay if member added to beginning of directory
  - Problems when PDS went into multiple extents

First some history about PDS datasets

- More problems with PDS dataset load libraries:
  - PDS datasets could not share update access to members without an enqueue on the entire data set.
  - The biggest drawback to PDS load libraries was that they had to be taken offline from time to time for:
    - A compression to reclaim member space or
    - Directory reallocation to reclaim directory gas
  - Because of this, applications could not have 24/7/365 access

Introducing PDSE datasets for load libraries!

- PDSEs, which were introduced in 1990, were designed to eliminate or at least reduce these problems
- They have! It's unfortunate that the rollout of PDSEs was so painful (lots and lots of APARs) that many sites have steered clear of them
- OTOH, many sites HAVE moved their COBOL load libraries to PDSEs, it is fairly mechanical

How to migrate from PDS load libraries to PDSE load libraries:

- Assuming the conversion of an entire PDS to a PDSE, the general steps are as follows:
  - Allocate a new PDSE dataset, such as &pds.PDSE, where "&pds" is the PDS dataset name.
  - Use IEBCOPY (or ISPF) to copy the load modules from the PDS into the PDSE.
    - This will automatically convert the load modules to program objects in the PDSE.
  - Rename the PDS. Example: &pds.BACKUP. Retain this dataset (short term) for recovery purposes.
  - Rename the PDSE to &pds, where "&pds" is the original PDS dataset name.

How to migrate from PDS load libraries to PDSE load libraries, some notes:

- Any Load Module in a PDS can be copied into a PDSE
  - It then becomes a Program Object
  - Program Management Binder is called by IEBCOPY or ISPF to do the conversion for you
- Not all Program Objects in PDSEs can be copied back to PDS and Load Module form
- This means that if a Program Object member in a PDSE on a test system is then shipped to production, and the receiving dataset on the production system is a PDS, then there could be a copy problem.
- Convert the downstream library first, i.e. convert the production PDS to a PDSE. Then convert the test system PDS to a PDSE.

Why are PDSE load libraries required with COBOL Version 5?

- First some history about Load Modules
  - z/OS has been moving to solve problems due to limitations of Load Modules for years
  - Program Management BINDER has made many changes to solve these problems
  - Many of these solutions required a new format of executable
  - Program Objects was the answer
  - Program Objects have features that cannot be supported by PDS datasets, so they require PDSE datasets

Load Modules versus Program Objects

- Program Management Binder solves existing problems with Load Modules using new features of Program Objects
  - Example: when customers reached 16M text size limit of load module, our answer was always: “Re-engineer programs to be smaller, re-design” …expensive and not well received!
  - A program object can have a text size of up to 1 gigabyte
  - COBOL can take advantage of this by having more constants for improved MOVE and INITIALIZE performance
    - Makes object size bigger

Why are PDSE load libraries required with COBOL Version 5?

- COBOL V4 required Program Objects and thus PDSE for executable for certain features since 2001:
  - Long program names
  - Object-Oriented COBOL
  - DLLs using the Binder instead of prelinker
- COBOL V5 requires Program Objects and thus PDSE load libraries for all executables
- How about some examples of specific features that COBOL V5 has that can only be supported by Program Objects (PO) and PDSE Load libraries?

Why PDSE for COBOL V5 executables?

- COBOL improving performance using new features that are only available in Program Objects (PO)
  - Improved init/term scheme relies on user-defined classes in object, requiring PO
  - QY-con requires PO
    - That's a performance improvement for RXY (long displacement) instructions.
  - Condition-sequential RLD support requires PO
    - Performance improvement for bootstrap invocation
  - PO can get page mapped 4K at a time for better performance

Why PDSE for COBOL V5 executables?

- Other features requiring Program Objects
  - NOLOAD class DWARF debugging data requires PO
  - Common reentrancy model with C/C++ requires PO
  - XPLINK requires PO and will be used for AMODE 64

What about sharing COBOL load libraries across SYSPLEX systems?

- PDSE datasets cannot be shared across SYSPLEX boundaries
- If PDS load libraries are shared across SYSPLEX boundaries today, in order to move to PDSE load libraries, customers can use a master-copy approach
  - One SYSPLEX can be the writer/owner of master PDSE load library (development SYSPLEX)
  - When PDSE load library is updated, push the new copy out to production SYSPLEX systems with XMIT or FTP
  - The other SYSPLEX systems would then RECEIVE the updated PDSE load library

Can I mix PDS and PDSE load libraries?

- If you convert all load libraries to PDSE first, no worries
  - IE: You will no longer have any PDS load libraries
- If you create a new PDSE dataset and put new code there while keeping existing load modules in PDS load library, you could end up using both PDS and PDSE load libraries in a single application:
  - COBOL V5 in PDSE load library can call COBOL V4 in PDS load library without problems (and vice-versa)
  - DYNAMIC CALL only of course
- If you start with COBOL V4 (or V3, V2) code in a PDS load library and recompile one program of a load module with COBOL V5, and then re-BIND, the result will be a Program Object, and will go into a PDSE
  - STATIC CALL in this case