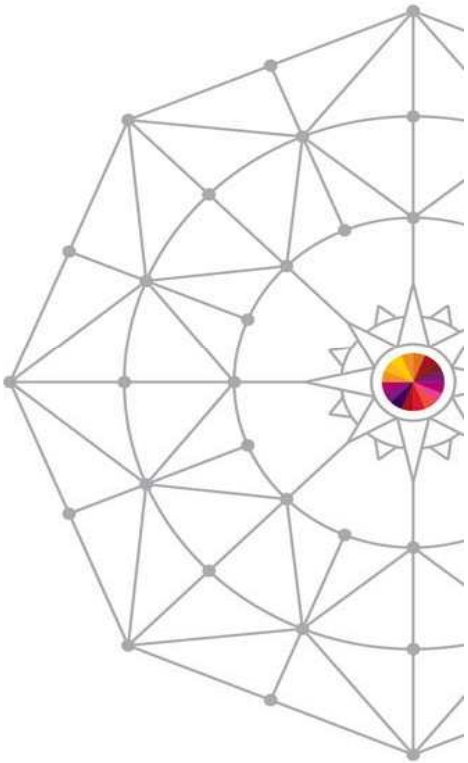
# How to take advantage of the new COBOL V5 compiler: Migration



Tom Ross

Aug 6, 2014

#SHAREorg



SHARE in Pittsburgh 2014

# Enterprise COBOL V5.1 Migration

## Standard Legal Disclaimer

# Enterprise COBOL V5.1 Migration

- Agenda
  - Introduction to COBOL V5.1
  - Are you ready to migrate to COBOL V5?
  - Pre-requisite software
  - Differences in using the new compiler
  - Binding COBOL V5 programs
  - Restrictions for running COBOL V5 programs
  - Differences in running the new programs
  - How to migrate!

# Intro to Enterprise COBOL for V5.1

- Enterprise COBOL for z/OS, Version 5 Release 1 (V5.1)
  - Announced April 23, GA June 21
- Introduces advanced optimization technology
  - Removed code generator/optimizer from V4 and put in newer one from Java JIT
  - Designed to optimize applications for current and future System z hardware
  - Initiate delivery of performance improvements seen in C/C++ and Java compilers on System z
- Provides compatibility with previous COBOL releases
  - No need to recompile entire applications to take advantage of new V5 features
- Support modern development tools
  - IBM & ISVs
- Continue to deliver new features
  - to simplify programming and debugging to increase productivity
  - to modernize existing business critical applications

# Intro to Enterprise COBOL for V5.1

- New compiler, new Web Sites!
- COBOL for z/OS Home page:
  - http://www-03.ibm.com/software/products/us/en/entecoboforzos
- Includes links to new publications page:
  - Enterprise COBOL for z/OS library
  - http://www-01.ibm.com/support/docview.wss?uid=swg27036733
- Links to new resources (news) page
  - Enterprise COBOL for z/OS resources
  - http://www-01.ibm.com/support/docview.wss?uid=swg21634215

# Intro to Enterprise COBOL for V5.1

- New compiler, new Migration Guide!
- Run-time migration information no longer included
  - The Enterprise COBOL V4.2 Compiler and Runtime Migration Guide should be used
- Enterprise COBOL V5.1 Migration Guide
  - Streamlined!  Compiler migration information only
  - Specific chapters for each compiler ‘starting point’
    - EG: Version 3 to Version 5
  - Easy to find the small amount of information that you need

# Ready to migrate to COBOL V5?

- Are you still using PDS datasets for your COBOL Load Libraries?
- COBOL V5 executables are Program Objects that can only resided in PDSE
- If using PDS load libraries, start moving to PDSE load libraries ASAP!
- If you try to bind COBOL V5 into a PDS you get this message:

IEW2606S 4B39 MODULE INCORPORATES VERSION 3 PROGRAM OBJECT FEATURES AND CANNOT BE SAVED IN LOAD MODULE FORMAT.

# Ready to migrate to COBOL V5?

- Complete your migration to LE
  - The Language Environment dataset SCEERUN is installed in LNKLST or LPALST
  - There are no instances of COBLIB, VSCLLIB or COB2LIB in LNKLST or LPALST
  - There are no instances of COBLIB, VSCLLIB or COB2LIB in JCL STEPLIB or JOBLIB statements in batch jobs or in CICS startup JCL
  - All statically bound runtime library routines (for programs compiled with NORES) have been REPLACEd with routines from Language Environment
  - All VS COBOL II bootstrap modules IGZEBST must have PN74000 applied or be replaced with LE version

# Ready to migrate to COBOL V5?

- Identify any old COBOL in load libraries
  - Use Debug Tool LMA, Edge Portfolio Analyzer, COBANAL, AMBLIST, or similar
  - OS/VS COBOL
  - VS COBOL II compiled with NORES
- Identify OS/VS COBOL still being used
  - z/OS APAR PM86742 adds warning messages when OS/VS COBOL programs are run or called
    - IGZ0268W An invocation was made of OS/VS COBOL program "program-name".
    - IGZ0269W "program-lang" version "program-version" program "program-name" made a call to OS/VS COBOL program "program-name".
  - Can be suppressed or severity increased using CEEWUCHA

# Pre-requisite software

- z/OS V1R13 or later
- CICS Transaction Server for z/OS, V3 or later
- DB2 V9 or later
- IMS V11 or later
- PD tools V12 or later
  - Debug Tool
  - Fault Analyzer
  - Application Performance Analyzer **(V13)**
- Rational Developer for System z V9

# Pre-requisite software

- COBOL is pushing the limits!
  - Requires updates to many other products
  - As of Enterprise COBOL for z/OS V5.1, you must use SMP/E FIXCAT to identify the required PTFs on other products to work with COBOL for z/OS V5.1.
  - The required service PTFs for COBOL for z/OS V5.1 are not documented in the COBOL for z/OS V5.1 Migration Guide, are not included in PSP buckets, and are not included in any handouts for conferences.
  - SMP/E FIXCATs allow you to have the most up to date and correct information about Enterprise COBOL for z/OS V5.1, required service, and is the easiest way to quickly determine if you have all the necessary required service PTFs installed.

# Pre-requisite software

- How to use FIXCAT
  - For Enterprise COBOL for z/OS V5.1, you should use the SMP/E V3R6 or SMP/E V3R5 product support for FIXCAT HOLDDATA to do programmatic target system PTFs verification.
  - These PTFs are identified with a FIXCAT called IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R1 in Enhanced HOLDDATA.
  - A HOLDDATA type FIXCAT (fix category) is used to associate an APAR to a particular category of fix for necessary target system PTFs.
  - To help identify those PTFs on your current system that would be needed for your upgrade to Enterprise COBOL for z/OS, V5R1 and are not yet installed, you can use the SMP/E REPORT MISSINGFIX command.
  - Here is a sample command used to run against your z/OS CSI:

    ```
    SET BDY(GLOBAL)
    REPORT MISSINGFIX ZONES(ZOS13T,ZOS13P)
    FIXCAT(IBM.TargetSystem-RequiredService.Enterprise-COBOL.V5R1)
    ```

  - For complete information about the REPORT MISSINGFIX command, see SMP/E Commands.

# Pre-requisite software

- COBOL is pushing the limits!
  - Requires updates to many other products
- LE APARs for z/OS V1R13
  - PM88047 - ADD SUPPORT FOR ENTERPRISE COBOL V5
    - COBOL V5 runtime compid 568819812
  - PM88048 - ENTERPRISE COBOL V4/V5 INTEROPERABILITY
    - 'Old' COBOL runtime compid 568819802
  - PM91332 - EXTERNAL FILE WITH PRIOR VERSION OF COBOL ABEND AT TERMINATION
    - 'Old' COBOL runtime compid 568819802
  - PM87183 - NEW FUNCTION SUPPORT COBOL V5
    - CEL core compid 568819801
- Binder APARs for z/OS V1R13
  - OA41268 - for problems related to WSA size.
  - OA40593 - for problems with Debug Tool in CICS
  - OA42047 - for problems with double relocation

# Pre-requisite software

- LE APARs – z/OS V2R1
  - PM91795 - ADD SUPPORT FOR ENTERPRISE COBOL V5
    - COBOL V5 runtime compid 568819812
  - PM91150 - ENTERPRISE COBOL V4/V5 INTEROPERABILITY
    - ‘Old’ COBOL runtime compid 568819802
  - PM91332 - EXTERNAL FILE WITH PRIOR VERSION OF COBOL ABEND AT TERMINATION
    - ‘Old’ COBOL runtime compid 568819802
  - PM87183 - NEW FUNCTION SUPPORT COBOL V5
    - CEL core compid 568819801
- Binder APARs – for z/OS V2R1
  - OA42047 - for problems with double relocation

# Pre-requisite software

- DB2 APAR – V9 & V10 (included in V11)
  - PM77300 - DB2 Stored Procedures for COBOL V5
- CDA
  - PM88903 – COBOL V5 support
- Debug Tool – V12
  - PM85967 – Add support for COBOL V5.1 programs
- CCCA
  - PM86253 – add support for changing the latest new reserved word
    - XML-INFORMATION
  - PM89219 – add support to correct programs that use COBOL V4.2 language that has been removed from COBOL

# Pre-requisite software

- Enterprise COBOL V4 Migration Aids
  - PM85873 for new compiler option FLAGMIG4
    - This one will help customers identify if they have COBOL statements that are unsupported in V5
  - PM85035 for XML-INFORMATION (new function)
    - This one will help customers on V4 migrate to XMLPARSE(XMLSS) and therefore to V5 and later
  - PM87347(LE APAR) for XML-INFORMATION run-time support

# Pre-requisite hardware

- Enterprise COBOL for z/OS, V5.1
  - Has minimum hardware requirement of z990/z890
  - z/990 and z/880 shipped in June 2003
  - Hopefully old enough not to affect you!
  - Older compilers would compile and programs would run almost anywhere
  - With new ARCH option we can now compile programs that will not run at all on some machines…progress? ☺

# Differences in using the new compiler

- **Compiler resides in PDSE**
  - Can no longer be added to LPA via LPALST
  - Can add to LPA dynamically after system IPL
- **Compiler uses LE at compile time**
  - Must be the latest, with all V5 PTFs for installed
  - SCEERUN and SCEERUN2
- **Compiler EXITs (EXIT compiler option) changes**
  - Cannot use RTEREUS, IGZERRE or CEEPIPI
  - Can still use non-LE-conforming assembler
  - Can use LE-conforming assembler
  - Can use old or new COBOL
    - Cannot use STOP RUN in compiler exit programs
    - STOP RUN still supported everywhere else

# Differences in using the new compiler


- **More datasets used by COBOL V5**
  - More working datasets
    - SYSUT8-SYSUT15 are now required
    - SYSMDECK is now required
- **More memory required at compile time**
  - Recommend 200M region size
    - Old compiler could compile most programs with 10M
    - You may have to change system user id limits
- **More time required to compile**
  - 5 to 15 times as much time, depending on OPT level and source program size

# Differences in using the new compiler

- **If your COBOL V5 compile has strange ABENDs:**
  - First make sure you made the changes from previous slide
    - Add region and time before you call IBM ☺
  - With less than 80M region, you get one ABEND
    IEA995I SYMPTOM DUMP OUTPUT 869
    USER COMPLETION CODE=4093 REASON CODE=0000001C
  - 80M to 120M another ABEND, at least with message
    IEW4000I FETCH FOR MODULE IGYCBE FROM DDNAME STEPLIB FAILED BECAUSE INSUFFICIENT STORAGE WAS AVAILABLE.
  - Not enough time may get yet another strange ABEND
    I could not reproduce, but it hit me during development

# New compiler options for performance:


- **New ARCH compiler option requires system knowledge**
  - ARCH allows you to fully exploit your hardware
  - What is your hardware?
  - New challenge for COBOL, not new for PL/I or C
- **What hardware do you run COBOL on?**
  - What hardware is your disaster recovery system?
    - Compile for lowest level of hardware
- **A supported system with z/OS V1R13 could be on old hardware that does not support COBOL V5!**
  - Operation Exception (0C1) should trigger checking of ARCH setting and hardware level

# New compiler options for performance:


- ARCH(6)
  - 2084-xxx models (z990)
  - 2086-xxx models (z890)
- ARCH(7)
  - 2094-xxx models (IBM System z9 EC)
  - 2096-xxx models (IBM System z9® BC)
- ARCH(8)
  - 2097-xxx models (IBM System z10 EC)
  - 2098-xxx models (IBM System z10 BC)
- ARCH(9)
  - 2817-xxx models (IBM zEnterprise z196 EC)
  - 2818-xxx models (IBM zEnterprise z114 BC)
- ARCH(10)
  - 2827-xxx models (IBM zEnterprise EC12)
  - 2828-xxx models (IBM zEnterprise BC12)

# New compiler options for performance:

- ***OPTIMIZE(0 | 1 | 2)***
  - Levels of optimization
    - Higher levels improve run time performance
      - But also increase compile time
    - Highest level has slightly reduced “debuggability”
  - Applications that have user-written condition handlers are restricted, must use OPT(0) w/NOTEST
    - With TEST compiler option can use any OPT level
- ***MAXPCF(nnnnn)***
  - *Some programs can take too much time or memory to compile*
  - *Sets OPT level to 0 if program is too large or complex*
  - *Automatic control of OPT levels for large or complex programs*
  - *Default = 60,000  (MAXPCF(0) means always optimize)*

# New compiler options for performance:

- ***HGPR (PRESERVE | NOPRESERVE)***
  - Use high word of registers (upper 32 bits of 64-bit registers)
  - Effectively adds 16 more registers to improve optimization
- ***AFP( VOLATILE | NOVOLATILE)***
  - Use full complement of floating point registers.
- ***STGOPT / NOSTGOPT***
  - Allows compiler to delete unreferenced data items
  - Analogous to FULL suboption of OPTIMIZE in V4
  - Do not use if you have ‘eye-catchers’ in WORKING-STORAGE
    - *Or use a fake reference:*
    - *IF X NOT = X THEN Display Eye-Catcher*
- **Use IBM defaults as your installation defaults for all of these**
  - Change only for special cases, individual applications

# Changed compiler options:

- *NOTEST suboptions for debugging information*
  - *NOTEST(DWARF)*
    - *Full optimization plus variables in CEEDUMP*
  - *NOTEST(NODWARF)*
- *TEST suboptions*
  - *HOOK removed (always no compiled-in hooks)*
  - *SEPARATE removed (debug information always in object)*
    - *NOLOAD class segments mean no increase in LOAD size*
  - *EJPD still supported*
    - *Debug Tool now allows GOTO/JUMPTO with NOEJPD*
      - *“Dirty GOTO”, much like competitors debuggers*
  - *SOURCE new suboption*
    - *To control whether source information in Debug information*

# Changed compiler options:

- *SIZE compiler option*
  - The SIZE option value is no longer an upper-limit for the total storage used by a COBOL compilation. Storage used by the code-generation and optimization phases of the compiler is not restricted by the SIZE option value.
  - SIZE(MAX) is no longer supported
    - *SIZE(MAX) converted into SIZE(5000K) if specified*
  - *Code generation and optimization can require a variable amount of compile-time storage, depending on the size and complexity of the COBOL source program being compiled, and is limited only by the MVS region size.*
  - *A minimum region size of 200M is recommended*

# Changed compiler options:

Removed compiler options:

- *NUMPROC(MIG)*
- *NOLIB*
  - *LIB behavior always in effect*
- *DATEPROC/YEARWINDOW*
- *XMLPARSE*
  - *XMLPARSE(XMLSS) behavior always in effect*
- *SSRANGE range checks cannot be disabled*
  - *Runtime options CHECK(OFF) and NOSSRANGE have no effect*
  - *Compiler option NOSSRANGE is still available*
- *NORENT + RMODE(ANY)*
  - *Now illegal:  NORENT programs must have RMODE 24*

Compiler options restrictions removed:

- *MDECK does not require LIB*
- *EXIT does not conflict with DUMP*

# COBOL language removed

- *Millennium Language Extensions*
- **The removed elements are:**
  - **`DATE FORMAT` clause on data description entries**
  - **`DATEVAL` intrinsic function**
  - **`UNDATE` intrinsic function**
  - **`YEARWINDOW` intrinsic function**
  - **`DATEPROC` compiler option**
  - **`YEARWINDOW` compiler option**

# COBOL language removed

- *LABEL DECLARATIVES*

  **Format 2 declarative syntax:**

  ```
  USE ... AFTER ... LABEL PROCEDURE ...
  ```

  **and the syntax:**

  ```
  GO TO MORE-LABELS
  ```

  **are no longer supported**

  - **Note: GO TO is still supported.**
- **Old XML PARSER**
  - **The original COBOL Version 3 Parser**
  - **Avail in V4 via XMLPARSE(COMPAT) option**

# Binding COBOL V5 programs

- No more load modules
  - Compiler output is GOFF format object, input to binder
  - Binder output is Program Object
  - Program Object cannot reside in PDS, PDSE only
- NOLOAD segments
  - With TEST, debugging information is in program object
  - Always matches executable
  - Never needs to be searched for
  - Does not take memory when program is loaded!
  - The perfect solution for debug information

# Setting up for COBOL V5 programs

- CICS CSD
  - Use the latest updated CSD from LE
    - Add new V5 runtime library modules
  - APAR PM99301
  - http://www-01.ibm.com/support/docview.wss?uid=swg1PM99301
- CICS and Debug Tool and LE CEEDUMP
  - If you are running COBOL V5.1 (or later) programs compiled with the TEST compiler option on CICS, you must also add system libraries MIGLIB and SIEAMIGE in the DFHRPL DD concatenation. The DFHRPL concatenation is in the CICS region start-up JCL.
  - APAR PM96501
  - http://www-01.ibm.com/support/docview.wss?uid=swg1PM96501

# Setting up for COBOL V5 programs


- IMS exits must be in PDS, such as
  - DFSME127  "Input Message Segment Edit " user exit
  - DFSME000.  "Input Message Field Edit" user exit
- IMS documents not to use LE languages for exits
  - Many have been doing it with COBOL anyway
  - COBOL V5 programs cannot be in PDS datasets so…
- For COBOL programs used as these exits you have choices:
  - If the exit routine is COBOL, do not recompile with COBOL V5, keep using older COBOL in PDS
  - If the exit routine is COBOL, change to use an assembler program that LOADs your COBOL V5 program for exit logic
  - If the exit routine is assembler that then loads COBOL, you can recompile the COBOL with V5, bind into a PDSE dataset, and add that new dataset to the IMS concatenation

# Setting up for COBOL V5 programs

- IMS is in the process of enabling user exits for enhanced services, which allows them to be run out of PDSE datasets
- User exits that were enabled for enhanced services in IMS V11:
  - ICQSEVNT (new) IMS CQS Event user exit
  - ICQSSEVT (new) IMS CQS Structure Event user exit
  - INITTERM (new) Initialization / Termination user exit
  - RESTART (new in IMS 10) Restart user exit
  - PPUE (DSFSPPUE0) Partner Product user exit
- No additional exits were enabled in IMS 12.
- The user exit types enabled in IMS 13 were:
  - BSEX (DFSBSEX0) Build Security Environment user exit
  - LOGEDIT (DFSFLGE0) Log Edit user exit
  - LOGWRT (DFSFLGX0) Logger user exit
  - NDMX (DFSNDMX0) Non-Discardable Message user exit
  - OTMAIOED (DFSYIOE0) OTMA Input/Output Exit user exit
  - OTMARTUX (DFSYRTUX) OTMA Resume TPIPE Security user exit
  - OTMAYPRX (DFSYPRX0) OTMA Destination Resolution user exit
  - RASE (DFSRAS00) Resource Access Security user exit

# Restrictions running COBOL V5 programs


- Interoperability with old COBOL programs
  - OS/VS COBOL: not supported
    - IGZ0264S There was an attempt to run both OS/VS COBOL and Enterprise COBOL V5 programs in the same enclave.
  - VS COBOL II NORES: not supported
  - Recompile both types with COBOL V5
- Interoperability with AMODE 24 programs
  - With latest PTFs, completely compatible with COBOL V4 and V4

# Differences in running the new programs

- More memory required  (about 1.2 times)
  - Largely a result of more constants for better performance
- Applications with user-written condition handlers
  - Cannot use higher OPT compiler option levels with NOTEST
  - This will be fixed in a future release of Enterprise COBOL

# Differences in running the new programs


- Not all incorrect programs are diagnosed as incorrect (by either Version 4 or 5)
  - And the results may differ between the two
- Over-populated numeric values
  - E.g., S99 PACKED-DECIMAL set to value 123 (X'123C')
  - E.g., S9(6) USAGE BINARY set to +123456789
  - E.g., PACKED or ZONED with invalid sign code
  - MOVE of overpopulated data items to data items defined like this
- These issues are not common. We found them writing unit tests developing the new compiler
  - None in our vast test suite or in the beta program
- We recommend more vigorous testing then previously
  - Migrating from V3 to V4, almost no testing was necessary
  - Try invalid data, bad data, extremes, as well as normal testing

# Differences in running the new programs

- Not all incorrect programs are diagnosed as incorrect
- Programs that set the value of an ODO object to outside of the legal range:

```
77      VAR1 COMP-3 PIC 9(3).
01 X.
   02 VAR2 PIC X OCCURS 0 to 1 depending on VAR1.

MOVE 128 to VAR1
MOVE ALL 'C' to X                *>  This is illegal!
```

Results:

  - For V2, V3, V4: 128 bytes of 'C' were moved
  - For V5R1: 1 byte of 'C' and 127 bytes of junk was moved

# Differences in running the new programs


- Parameter length mismatch:

```
WORKING-STORAGE SECTION.
77   GRP1 PIC X(100).
01   FILE-STATUS-DATA-ITEMS.
   02   OTHER-SENSITIVE-DATA-ITEMS …
. . .
    Call ‘SUBP’ Using GRP1.

Program-Id. SUBP.
Linkage Section.
 01  GRP2 PIC X(500).
Procedure Division Using GRP2
    MOVE ‘stuff’ To GRP2(300:320)     *>  This is illegal!
```

Results:

  - For V2, V3, V4: Illegal program did not fail
  - For V5R1: File-status in CALLER changed, flow changed

# How to migrate - Check list

- First, things to find out about your COBOL:
  - Type of datasets used for load libraries
  - Hardware levels where COBOL applications are run (Include disaster recovery site hardware)
  - Region size available for compiles
  - Compilations that use the EXIT compiler option?
  - OS/VS COBOL or VS COBOL II NORES?
  - User-written condition handlers?
  - IBM COBOL programs compiled with CMPR2
  - Enterprise COBOL Programs that use XML PARSE
  - Any programs that use LABEL DECLARATIVES or MLE
  - Default compiler options in current compilers

# How to migrate – TO Do Check list

- **Type of datasets used for load libraries**
  - If PDS, start moving to PDSE ASAP
- **Hardware levels used by COBOL applications**
  - Choose ARCH level appropriately
- **Region size available for compiles**
  - Increase to 200M if not there already
- **OS/VS COBOL or VS COBOL II NORES?**
  - Convert to COBOL V5.1 along with other programs in the applications that interact with these or
  - Convert to Enterprise COBOL V4 in preparation for V5
- **Compilations that use the EXIT compiler option?**
  - Modify EXIT programs as necessary

# How to migrate – To Do Check list

- **User-written condition handlers?**
  - Compile with TEST or with OPT(0)
- **IBM COBOL programs compiled with CMPR2**
  - VS COBOL II, COBOL/370 V1.1 thru COBOL OS/390 V2.2
  - Convert source code to NOCMPR2
  - CCCA can do it automatically
  - Use FLAGMIG option and make changes by hand
  - Once converted to NOCMPR2, can compile with COBOL V5.1
- **Any programs that use LABEL DECLARATIVES or MLE (very rare)**
  - Remove any such code

# How to migrate – To Do Check list

- **Enterprise COBOL Programs that use XML PARSE**
  - If V3 or V4 w/XMLPARSE(COMPAT) must change code or wait for September PTF to support COMPAT parser
    - Convert to new parser on Ent COBOL V4
  - If V4 w/XMLPARSE(XMLSS)
    - Skip this topic, code is ready for COBOL V5.1
  - Once converted to XMLPARSE(XMLSS) on COBOL V4, you can compile with COBOL V5.1 with no other changes!
  - New tool to help is available in Enterprise COBOL V4
    - XML-INFORMATION special register added by APARs (PM85035 and PM87347)

# How to migrate – To Do Check list

- **Enterprise COBOL Programs that use XML PARSE**
  - One big difference between old parser and new parser is new parser can receive multiple events for the same content (ATTRIBUTE-CHARACTERS and CONTENT-CHARACTERS)
    - Use XML-INFORMATION to tell you when you are ‘done’ collecting data for an element or attribute
  - Before XML-INFORMATION it was difficult to collect the data
    - For *-CHARACTERS events, you had to save data always ‘just in case’ there was more
    - When one of several new different events is encountered, then close out the old one

# How to migrate – To Do Check list

- **Install Enterprise COBOL V5.1**
  - Set up default compiler options
  - The same settings will give you the same results
  - ADV, ARITH, AWO, NUMPROC, TRUNC, ZWB, etc
  - **Special Cases**
    - NUMPROC
      - If currently use NUMPROC(MIG) you will have to migrate
      - COBOL applications always result in correct signs
      - If you have non-COBOL applications or sources of data
        - » Use NUMPROC(NOPFD)
    - If currently use OPT(FULL), use OPT(1 or 2) and STGOPT

# How to migrate!

- **Which programs to compile first**
  - PERFORMANCE
    - Programs with lots of arithmetic will get most performance benefit from new code generator
    - The code generator is all new for COBOL V5.  It is not the same code generator with improvements. So, some code will run faster than V4, some may run slower
  - Need for XML GENERATE and PARSE features
  - Need for Large data items

# How to migrate!

- Questions?
- Good luck!

# 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