CHOOSING YOUR COBOL MIGRATION PATH: LEADING WITH VERSION 5.2 OR 6.1

Deciding which COBOL release you want to migrate to first and then following through with a stable migration requires a well-researched and carefully considered plan. Here we present the similarities and differences between the two latest versions of COBOL.

Converting COBOL programs to Java can create a multitude of problems, including performance issues. Migrating to a newer version of COBOL can yield desired reductions in CPU costs without the headaches caused by migration to Java. In this post we'll discuss the difference between COBOL Version 5.2 and 6.1 to help you determine which release you should migrate to first.

To start, let's look at some fundamental differences between COBOL Version 5.2 and 6.1. Per IBM's announcement of COBOL Version 5.2, the upgrade included new and improved features such as:

- Support for the new IBM z13 hardware
- New, restored and enhanced compiler options for ease of migration and programmer productivity
- New features added from the ISO 2002 COBOL Standard
- New IBM extensions to COBOL
- Product-related enhancements

However, more improvements and features were added in the release of COBOL Version 6.1, including:

- **Increased compiler capacity**
  This provides a new "back end" that converts the intermediate form of a program to machine

code. It resolves problems in previous COBOL versions that failed to compile very large COBOL programs. Large programs are typically those that used “code generators.” These code generators spew out a lot of code and use a lot of CPU and memory resources. The IBM “back end” for COBOL Version 6.1 uses 64-bit storage to overcome this problem. You may use more CPU to compile a program with 6.1 than with 4.2, but if you run the program several times a day, you will end up using far less CPU during run-time.

- **New COBOL language features**
  More new language features were added from the ISO 2002 COBOL Standard, introducing the use of ALLOCATE, FREE and INITIALIZE statements.

- **New and enhanced compiler options**
  These new options ease migration and improve programmer productivity, but to implement some of them, you may have to consider how they’re to be used and when to use them properly. For example:

  - VSAMOPENFS allows compatibility with "file status 97" in pre-COBOL Version 6 programs
  - SUPPRESS/NOSUPPRESS allow debuggers to use the copybook information in the listing
  - SSRANGE sub option enhances range checking for subscripts, reference modifications group ranges and indexes at run time
  - Diagnostic message for ZONECHECK(MSG) compiler option assists the detection of invalid zoned decimal data items
  - LVLOPTION was replaced by a seven-character build level identifier in the format PYYMMDD to the compiler listing header

- **Runtime and product-related enhancements**
  For RENT compiled programs, WORKING-STORAGE will be acquired from HEAP storage and will be initialized at the COBOL program call and not when the program object was loaded. This can help applications by:

  - Freeing storage below the 16MB line and using storage in the HEAP
  - Reducing storage usage and performance tuning in Table SORT
  - Improving performance for INSPECT, UNSTRING, SEARCH ALL (exploiting z13 features such as SIMD)

Which Version Is Better?

To reiterate the final thought from our last post, migration to both COBOL Version 5.2 and 6.1 is a smart idea. It’s true that starting at COBOL Version 6.1 eliminates duplicate compiler upgrades twice, once to 5.2 and again to 6.1. However, by migrating to COBOL Version 5.2 first, you can save costs by leveraging the IBM Enterprise COBOL trials for both 5.2 and 6.1 to gain maximum experience and application comfort before making a formal decision to upgrade. There will not be any Single Version Charging (SVC) changes during these trial periods. Consult the IBM announcement letter Enterprise COBOL Developer Trial for z/OS, V6.1 for more details.

Keep in mind COBOL Version 5 can be ordered until September 2017, so you have a limited time to take advantage of the benefits from migrating to this version. Also, consider that COBOL Version 5.1 doesn’t support the z13 and 5.2 is the recommended first choice for the first migration phase

because it provides XML compatibility with the Enterprise COBOL Version 3 parser.

Be sure to use ZONECHECK and ZONEDATA to identify "improperly formatted data fields." It has been said that many migration problems are data related. Data that was accepted as valid (or not detected as invalid) by Enterprise COBOL Version 4.2 and earlier could now be invalid with COBOL Version 5.2 and 6.1.

And, of course, spend some time reading through the migration guides for each version to get a better sense of where you're going:

- Enterprise COBOL V6.1 for z/OS Migration Guide
- How to take advantage of the new Enterprise COBOL V5/V6 compilers - Migration!
- Migration recommendations to Enterprise COBOL V5 and V6

Performing a Migration

The following steps and, admittedly very technical, information should help guide you on your migration journey.

1. **Prepare to Migrate**
   To prepare for a migration to the latest version of COBOL, consider whether you have programs that were last compiled under OS/VS COBOL, or COBOL V2. You need to upgrade the code to Enterprise COBOL 4.2 before migrating to anything higher. Add FLAGMIG4 and FLAG(I,I) to identify necessary coding changes. Next, determine what programs you want to upgrade. Focus on applications that consume CPU resources as candidates to convert first. Note that migration to PDSEs is required. Try to do that when you are at Enterprise COBOL V4.2. Also, note there will be more work datasets required during compile time.

1. **Brace for Compile Time "Shocks"**
   There are a few "shocks" to brace for. Compile time CPU usage will rise five to 15 times its current volume, depending on the optimization level. Meanwhile, compile time memory usage will rise upwards of 20 times its current level. IBM requires the region size of the compile step to be a minimum of 200MB. Many shops are having to utilize a default size of 500MB or 0M because of the amount of storage required for optimization.

1. **Find Invalid COBOL Data**
   Invalid COBOL data must be found using repetitive testing during the migration process. This data generally cannot be found using a compiler feature or option, but programmatically can be found using COBOL statements such as IF NUMERIC. You can use compiler options such as SSRANGE to flush out the problems during testing.

1. **Stay Up to Date**
   Keep the service updates for the compiler and LE up to date. Several PTFs are resolved nearly every month. Refer to the IBM COBOL Fix list web page for this information.

1. **Learn and Watch for Pitfalls**
   As usual, there are a few pitfalls to watch for. It's important to learn from what others have gone through. Refer to literature and bug lists on the above-mentioned fix list web page. Also, make sure you repetitively resolve problems in each of your applications. It will get easier for later migrations as you build a comprehensive list of common errors in your codebase.

1. **Understand the Impact of New Compile Options**
   When compiling, adding new compile time options will help quantify the impact. IBM recommends compiling with SSRANGE, ZONECHECK and OPT(0) to flush out all table misuse and invalid data, and recompiling with NOSSRANGE, NOZONECHECK and OPT(2) for QA and production.

   - **SSRANGE** can average as much as 18 percent more CPU.
   - **ZONECHECK** will add an **IF NUMERIC** before any numeric statements, which also will increase the CPU time of the run.
     - **SSRANGE, ZONECHECKS** and other options will definitely speed up the ability to clean up bad zone bits.
     - Compile with **NUMPROC(PFD)**, which is more efficient.
   - **ZONEDATA** will confirm the zoned part of decimal data is valid and provide compatibility with prior COBOL Version 6.1 compiler behavior.
   - **RULES** will cause the compiler to issue warning messages when detecting non-standard or poor coding practices that were tolerated prior to COBOL Version 6.1. Systems personnel can change the warning to an error to force changes.

Deciding which COBOL release you want to migrate to first and then following through with a stable migration requires a well-researched and carefully considered plan. As you begin migrating your COBOL applications, testing and properly debugging early in the process are also vital to avoiding future problems.