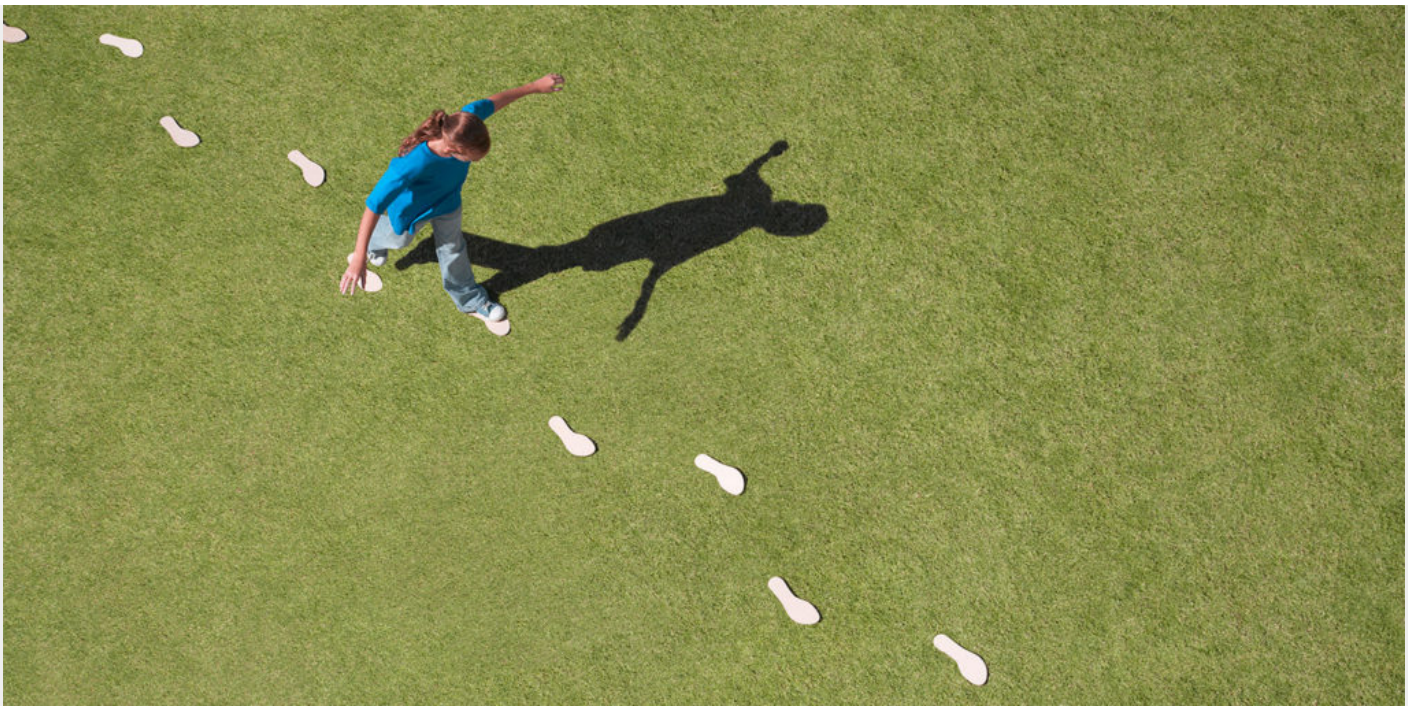
FIRST STEPS WHEN MIGRATING TO THE LATEST VERSION OF COBOL



The best way to optimize your COBOL programs is to migrate them to the latest version of COBOL. The first part of that strategy involves roadmap planning.

Today, digital business is driving more mainframe activity, according to a survey from BMC. Instead of sacrificing application performance to save a little cash, you should optimize your mainframe programs so they perform well against the modern demands they face from new and growing digital engagement.

The best way to optimize your COBOL programs is to migrate them to the latest version of COBOL, spanning COBOL Version 5 and 6. The first part of that strategy involves roadmap planning.

First Steps Migrating to the Latest Version of COBOL

There are a few first steps you need to take before beginning a migration to the latest version of COBOL:

1. Read the Migration Guide for the COBOL release you will be upgrading to.

The correct guide can be found via the IBM Enterprise COBOL for z/OS documentation library. It will provide the cautionary points pertinent to your migration, but up front, there are a few to consider, including:

- Source language incompatibilities when migrating from OS/VS COBOL and VS COBOL II to any newer COBOL version

- Data problems undetected by the compiler must be found by repetitive testing (some helpful compiler options: SSRANGE, RULES and DIAGTRUNC).
- JCL, REGION size and more work datasets are required (there are major changes to REGION size)
- Some compiler options have been removed, such as NUMPROC(MIG) (refer to your Migration Guide for a complete list)
- Executables must be in PDSE libraries
- OS/VS COBOL or VS COBOL II programs should be upgraded to COBOL 4.2 before recompiling with COBOL Version 5 or Version 6

2. Determine a business application to serve as the template for upgrading others.

A suggested approach is to use a business application that has encountered elongated run times to see the potential impact of recompiling problematic code under COBOL Version 5 or 6. You don't have to recompile every program, just those you think might benefit from the new compiler and advanced optimization. Use mainframe profiling tools to set a performance baseline before you start. For example, Compuware Strobe, our application performance monitoring and analysis tool, can capture the information for later use.

3. Analyze the load library modules to determine the release of COBOL the business application was last compiled under.

When going about this, refer to your Migration Guide to learn about changes that may be necessary to make within the code for those modules compiled in earlier versions of COBOL. There may also be compile options that have been deleted, have changed in function or offer new options. Portfolio Analyzer—accessed via the Library Utility (3.1) of Compuware File-AID file and data management tool—allows you to view and export COBOL and PL/I compiler options for all releases to a .CSV file for later analysis.

In addition to these steps, there are other preliminaries that will ease the migration execution, including learning from your own and others' mistakes and making sure you understand your hardware machines.

Learning from Mistakes

Although migrating older COBOL programs to the latest version of COBOL presents fewer pitfalls than converting them to Java, you should still consider potential hazards that may crop up along the journey.

For one, there's a chance you could take on too much at one time. It's best to stick to a small application or set of applications to start, as mentioned in point two above. You can learn from that experience, and apply your new understanding to other application migrations to prevent recurring missteps.

Also, avoid making assumptions about programs before migrating them. Do some analysis up front. You need facts about the programs that help you understand their composition and behavior to make good determinations that help you prevent problems and avoid creating them.

Know Your Hardware Machines: ARCH and OPT

It's important to understand some background information on COBOL and how various versions affect performance. The type of COBOL version has a significant effect on how you migrate the application.

From the language perspective, COBOL has been upgraded many times—OS/VS COBOL with features of the ANSI 74 standard, VS COBOL II with features of the COBOL 85 standard, COBOL Version 5 had elements of ISO standard 2002 and a future COBOL version may have elements of a newer standard.

But from the performance perspective, one of the past knocks against COBOL was its failure to take advantage of the newest processor hardware machines. IBM mainframe customers bought huge new machines and COBOL didn't use any of the newest features.

This meant all COBOL versions, from VS COBOL II (1985) through Enterprise COBOL Version 4.2 (2009), used the same back end and generated the same code. Hence, over the course of 25 years, the same older and less efficient code continued to be generated for COBOL and didn't use any of the new features introduced on the z Systems machines, starting with the z900/800 (CY2000).

Fortunately, this changed with COBOL Version 5.1. All levels of COBOL starting with Version 5.1 are designed to take advantage of the newest machine hardware. Newer hardware machines have features that, if used, tend to offload work to the hardware, reducing CPU consumption and enabling them to run more efficiently.

The COBOL ARCH compile time option controls these features. ARCH(6) is for the z990/890 systems all the way up until z13, which uses ARCH(11). Likewise, the COBOL OPT compile time option controls how complex, and potentially more efficient, of an optimization to generate. OPT(0) is the minimum optimization level and OPT(2) level is the maximum (OPT(0) does not mean "none," rather "some minimal amount"). Higher ARCH levels tend to generate more use of the newest machine features, while higher OPT levels tend to create more efficient optimizations.

All COBOL versions default to OPT(0). COBOL Version 5.1 defaults to ARCH(6), a z990/z890 code base. (Note: COBOL Version 5.1 doesn't support ARCH(11), a z13, and programs compiled with ARCH(10) will run on a z13, but will not generate code that exists only on a z13.) COBOL Version 5.2 and 6.1 default to ARCH(7), a z9 code base. However, all COBOL versions allow customers to change the defaults.

Still, a critical concern is the need to set a standard: the highest ARCH level of any COBOL compile should be set to the lowest ARCH level of any machine in your enterprise. For example, if your enterprise consists of a z10, a z11 and a z12, setting the default to ARCH(8) will allow that program to run on the z11 and z12 without problems. However, setting the default to ARCH(10) will cause problems if a program is subsequently run on a z10.

Disaster site hardware is also of special concern, as disaster sites may be at a different ARCH machine level than your production machines. In the previous example, if the disaster site has a z9, you will have problems at the site, as there is the potential for hardware instructions to be generated for an ARCH type higher than your disaster recovery machine.

Determine the Latest Version of COBOL You're Migrating to

After these initial preparatory steps, you should begin thinking about where you want to go: COBOL Version 5.2 or 6.1. In our opinion, you should first try migrating to 5.2 to maximize experience and eliminate bugs through a couple of test sets.

Although there is no technical reason to start a migration at COBOL Version 5.2 and then jump to 6.1, customers need to determine what's right for their environment. Consider this:

- Starting at COBOL Version 6.1 eliminates duplicate compiler upgrades twice, once to 5.2 and again to 6.1. However...
- You can save costs by leveraging the IBM Enterprise COBOL trials for both COBOL Version 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.