HOW TO IMPROVE COBOL MIGRATION STRATEGIES WITH MODERN TOOLS

Despite the availability, many mainframe shops are unaware modern tools contain improved functionality for aiding COBOL migration strategies and achieving improved performance.

There are a number of COBOL migration strategies an organization can choose from when contemplating the best avenue for upgrading their mainframe environment to the next generation version of COBOL, which spans COBOL 5.1 to 6.1.

But irrespective of which migration strategy an organization selects, it must keep in mind the array of new features, new functionality and new constraints associated with the next-generation version of COBOL.

Beginning in 2013, IBM introduced radical changes to COBOL with the Version 5.1 upgrade. Subsequently, there are a number of options the next-generation version of COBOL has deleted, changed or added, and release-by-release there are additional constraints.

Mainframe shops can improve COBOL migration strategies amidst these changes using modern mainframe tools like application failure resolution and fault management, as well as file and data management, that provide utilities to:

- Analyze your program load libraries to generate a report citing the language release and compiler options
- Provide an enhanced compiler listing showing the offset of every variable in working storage from the start of the static map, not just the offset within each group level
- Analyze your program load libraries to identify applications that would benefit the most by recompiling to the next-generation version of COBOL

According to IBM, performance improvements with the next-generation version of COBOL can be between 10-20 percent. But despite the availability, many mainframe shops are unaware modern tools contain improved functionality for aiding COBOL migration strategies and achieving improved performance.

Modernized mainframe tools providing this functionality can improve COBOL migration strategies organizations use by elucidating the hidden compile track records of older programs, enabling organizations to understand at a deeper level how to migrate programs to the next-generation version of COBOL.

How Modern Tools Improve COBOL Migration Strategies for Older Programs

When moving to the next-generation version of COBOL, organizations need to take into account older programs first. Older programs have the greatest possibility for requiring changes to align with the next-generation version of COBOL, usually with regard to recompilation or coding changes. But adjusting these programs is not a straightforward process. For example:

- **Load libraries are no longer used.** Programs are bound into PDSE objects. IBM recommends converting to PDSEs as soon as possible.
- **Converting applications to use PDSE object libraries is tough.** IBM recommends moving to PDSEs prior to migrating to a the next-generation version of COBOL. At sites, where CICS environments are up 24/7, this can be a daunting task. (There are ways to work around this, utilizing both PDS load libraries and PDSE object libraries, until you are able to do the conversion from PDS to PDSE.)
- **You can't use PDS load libraries for programs compiled with COBOL 5 or 6**. Program objects are generated by the binder into PDSE object libraries. If you try to bind a next-generation version COBOL program into a PDS, the binder will generate an error.

One of the most pressing issues mainframe shops share is the absence of a utility for reading a PDS and a PDSE, and for showing the release of COBOL and compile options. But a utility specifically designed for this is important for organizations performing COBOL migration strategies because it enables organizations to have to keep track of when a program was compiled.

To improve the efficiency and ease of COBOL migration strategies, organizations require—online or through batch—the ability to generate a report where it would read an entire PDS or PDSE and generate a report on every member in the load libraries or the program object library that would contain the release of COBOL, Assembler or PL/1.

In the case of COBOL, a tool with functionality like this not only should include all of the compile options for the mainline module, but also if there are any statically linked CSECTS that those would be the compile options and the release of COBOL would be listed there as well. This is especially important in CICS, as IBM states in their Migration guide that programs compiled under OS/VS COBOL will not function properly under COBOL 5 or 6.

Neglecting Modernized Mainframe Tools for Improving COBOL Migration Strategies

Due to the complications involved, IBM gives clients one year free of the next-generation version of COBOL so they have time to convert their site to the latest release; however, many organizations are discovering the migration takes longer than the grace period lasts. Consequently, organizations are paying IBM for multiple versions of COBOL. (Cheryl Watson, of Watson and Walker, has recommended several times in her presentations at SHARE that if you feel you need more time, try to negotiate with IBM. Negotiating has worked with a number of clients.)

In view of radical changes being made to COBOL and the time limit pressing down on organizations, they must be shrewd in their COBOL migration strategies, making sure to cover all of their bases by selecting the appropriate strategy and improving that process through the adoption of specialized modern tools. Learn more about those tools here.

Flickr: WhaleRiot