I've heard there is a problem passing pointers to C functions. What is it?

Use the -cp compiler option to enable full POINTER support, including USAGE POINTER, SET ADDRESS, pointer arithmetic and exchanging pointers with external routines.

The remainder of this article applies to programs compiled without the -cp option.

Without the -cp compile option, you can pass data-items by reference to external routines such as C functions, and those routines will receive a valid system memory address.

However, it is a violation of Java memory management rules for the external routine to save the pointer for later use. This is because the Java memory garbage collector requires the ability to move contents of memory around at will changing only the Java references to that memory.

From a COBOL perspective this means that the address of a particular data item is not guaranteed to stay the same throughout program execution. External references to memory (e.g. pointers saved by calls to C functions) are considered invalid after garbage collection.

Without the -cp compile option, USAGE POINTER items are actually handles to object references. So passing these by value to a C function will not give desired results. The C function expects a memory address, but the item actually passed is a handle value which has no exposed relation to the memory address.

Certain technologies such as Oracle Pro*COBOL rely on the ability to save a passed pointer for later use. The Java memory management constraint means that without the -cp compile option, isCOBOL will not work with Oracle Pro*COBOL. To use Oracle Pro*COBOL programs must be compiled with -cp.

A potentially better solution is to use isCOBOL built-in ESQL compiler technology. The isCOBOL ESQL compiler supports many of Pro*COBOL extensions to ease transition of applications that use Pro*COBOL to isCOBOL. The resulting applications are pure Java and use JDBC to communicate with the Oracle database.

Online URL: https://support.veryant.com/phpkb/article.php?id=81