CL Compiler Enhancements

Guy Vig
Senior Software Engineer
(gwvig@us.ibm.com)
“General” CL Commands

- There have been new and changed IBM CL commands in EVERY release
- For V5R3:
  57 new CL commands
  247 changed CL commands
Binary (Integer) Variables

• New TYPE values on DCL statement
• Values *INT and *UINT chosen for consistency with PARM TYPE values
• LEN(2) and LEN(4) supported
• Much "cleaner" than using %BIN
• Passing parameters to APIs
• Passing parameters to other HLL programs
Control Flow Enhancements

• Three “flavors” of DO loop commands
  – DOWHILE, DOUNTIL, and DOFOR
  – Up to 25 levels of DOxxx nesting supported

• Loop control flow commands
  – LEAVE and ITERATE

• Case/subcase commands
  – SELECT, WHEN, OTHERWISE, and ENDSELECT
  – Up to 25 levels of SELECT nesting supported
DOWHILE Loop

- Same COND support as IF statement in CL
- Evaluates COND at "top" of loop
- Old-style coding example:

        DCL   VAR(&LGL)  TYPE(*LGL)
        :
        CHECK:  IF COND(*NOT &LGL) THEN(GOTO DONE)
        :     (group of CL commands)
        GOTO CHECK
        DONE:

- New-style coding example:

        DOWHILE   COND(&LGL)
        :     (group of CL commands) ← body will be run zero or more times
        ENDDO
DOUNTIL Loop

- Similar COND support as IF statement in CL
- Evaluates COND at "bottom" of loop
- Old style coding example:

  DCL   VAR(&LGL)  TYPE(*LGL)  
    :  
    LOOP:  
      :    (group of CL commands)  
      IF COND(*NOT &LGL) THEN(GOTO LOOP)  

- New style coding example:

  DOUNTIL   COND(&LGL)  
            :    (group of CL commands)  ← body will be run one or more times  
ENDDO
DOFOR Loop

- Syntax:
  
  DOFOR VAR( ) FROM( ) TO( ) BY( )

- BY defaults to '1', other parameters are required
- VAR must be *INT or *UINT variable
- FROM and TO can be integer constants, expressions, or variables
- BY must be an integer constant (can be negative)
- FROM/TO expressions are evaluated at loop initiation; TO evaluated after increment
- Checks for loop exit at "top" of loop (like DOWHILE)
DOFOR Loop  (continued)

• Old-style coding example:

```cl
DCL &LOOPCTL  TYPE(*DEC) LEN(5 0)
DCL &LOOPLMT  TYPE(*DEC) LEN(5 0)

:  
CHGVAR   &LOOPCTL  VALUE(1)
CHECK: IF COND(&LOOPCTL *GT &LOOPLMT)  THEN(GOTO  DONE)
:     (group of CL commands)
CHGVAR  &LOOPCTL  VALUE(&LOOPCTL+1)
GOTO  CHECK
DONE:
```

• New-style coding example:

```cl
DCL &LOOPCTL  TYPE(*INT) LEN(4)
DCL &LOOPLMT  TYPE(*INT) LEN(4)

:  
DOFOR  VAR(&LOOPCTL)  FROM(1)  TO(&LOOPLMT)  BY(1)
:     (group of CL commands) ← body will be run zero or more times
ENDDO
```
LEAVE and ITERATE

- Allowed only within a DOWHILE, DOUNTIL or DOFOR group
- LEAVE passes control to next CL statement following loop ENDDO
- ITERATE passes control to end of loop and tests loop exit condition
- Both support CMDLBL (Command label) parameter to allow jump out of multiple (nested) loops
  - Both default to *CURRENT loop

Example:

```cl
:   (group of CL commands)
IF (%SST(&*NAME 1 10) *EQ ' *NONE') THEN(LEAVE LOOP1)
ELSE   (DO)
    IF (%SST(&*NAME 11 10) *EQ ' *LIBL') THEN(ITERATE)
ENDDO
:   (group of CL commands)
```
SELECT Group

- SELECT starts a group; this command has no parameters
- ENDSELECT ends group; this command has no parameters
- Group must have at least one WHEN
  - WHEN command has COND and THEN parameters (like IF)
- OTHERWISE (optional) run if no WHEN COND = True
  - OTHERWISE command has only CMD parameter (like ELSE)
- Example:

```
SELECT
  WHEN COND(&TYPE *EQ *CMD) THEN (DO)
    : (group of CL commands)
  ENDDO
WHEN COND(&TYPE *EQ *PGM) THEN (DO)
    : (group of CL commands)
  ENDDO
OTHERWISE CMD(CHGVAR &BADTYPE '1')
ENDSELECT
```
Enhanced File Support

• Will support up to 5 file "instances" using DCLF
• Instances can be for the same or different files
• New OPNID (Open identifier) parameter added to DCLF statement
• Default for OPNID is *NONE
  – Only one DCLF allowed with OPNID(*NONE)
• OPNID accepts simple name, up to 10 characters
File Support (continued) ...

• If OPNID specified, declared CL variables are prefixed by this name and an underscore (e.g. &MYTESTFILE_CUSTNAME )

• OPNID added to existing file I/O CL commands
  – RCVF
  – ENDRCV
  – SNDF
  – SNDRCVF
  – WAIT
Increased max size for *CHAR

• Old limit was 9999 bytes for TYPE(*CHAR)
• New limit is 32767 bytes for TYPE(*CHAR)
• DCLF will (still) not generate CL variables for character fields longer than 9999 bytes in a record format; same compile-time error
• Limit for TYPE(*CHAR) and TYPE(*PNAME) on PARM, ELEM, and QUAL command definition statements stays at 5000 bytes
• VALUE (on DCL) limited to first 5000 bytes
Increased number parameters

- Previous limit was 40 for PGM and TFRCTL, and 99 for CALL command
- New limit is 255 parameters for PGM, CALL, and TFRCTL
- Limit for CALLPRC (only allowed in ILE CL procedures) will stay at 300
- Number of PARM statements in a CL command will increase from 75 to 99
Passing parameters "by value"

- CALLPRC (Call Procedure) command supports calls from ILE CL procedures to other ILE procedures
- In prior releases, CALLPRC only supported passing parameters "by reference"
- Can specify *BYREF or *BYVAL special value for each parameter being passed
- Enables ILE CL to call many MI and C functions and other procedure APIs
- Maximum numbers of parameters still 300
Run New Support on V5R2

- Normally, new CL function can only be used in a CL procedure (i.e. CL program or CL module) if *CURRENT is specified for TGTRLS parameter
- This isn’t a “normal” release for the CL compiler!
- CL team wants to remove roadblocks to having CL programmers use new CL compiler function
- There will be V5R2 and V5R3 PTFs to allow developers to use new CL compiler function (except multiple DCLF) on **V5R3** and compile specifying TGTRLS(V5R2M0)
PTF numbers

• V5R3 (base OS and option 9)
  – SI13505
  – SI13508
  – SI13509

• V5R2 (CL runtime)
  – SI13416
  – SI13417
Follow-on CL Improvements

• V5R3 is the biggest release for CL compiler enhancements since ILE CL compiler in V3R1
  – Most new CL compiler function since System/38
• But we’re not done yet!
• Currently working on next set of enhancements
• Your opportunity to provide early feedback/input
Subroutines

• Simple code block between SUBR and ENDSUBR statements
• Invoked by new CALLSUBR statement
  – No argument/parameter passing
  – Optional RTNVAL can specify 4-byte *INT variable
  – No local scoping of subroutine variables
  – No nesting allowed (subroutines in subroutines)
• Return to caller via RTNSUBR or ENDSUBR
• Would not allow GOTO to enter or leave the body of a subroutine
Pointer CL variables

- Add TYPE(*PTR) on DCL statement
- New %ADDRESS built-in to set pointer
- New %OFFSET built-in to store pointer offset
- Add STG(*BASED) attribute on DCL statement
- Makes many functions available to ILE CL
  - Full record-level file I/O
  - String functions
Defined-on CL variables

• Add STG(*DEFINED) attribute on DCL statement
• Must give name of defined-over CL variable (new DEFVAR attribute on DCL statement)
• Can optionally provide starting position (default = 1) from beginning of the defined-over CL variable
• Useful for varying-character fields and providing simple structure capability
Other possible improvements

• Longer CL variable names (& + 30 characters)
• Arrays (single-dimension)
• Structures (one-level deep nesting)
• CL source includes in QSYSINC library
• Compile from stream file
• INCLUDE of source member or stream file or record format
• Allow MAX > 300 for a PARM/ELEM on *CMD
• Soft remove of obsolete command parameters
Continuing CL improvements

• Intention is to keep adding improvements
• We want to deliver enhancements that will delight iSeries customers, including business partners
  – If we're hitting the mark, tell an IBM exec
  – If we missed, tell me
• Funding at risk if little or no positive customer feedback
Summary

• Recognition by IBM that having “healthy” CL is important to existing and future iSeries customers
• Continued investment in providing CL commands
• Biggest changes to CL compiler since System/38
Thank YOU