V5R3 CL Enhancements

Larry Bolhuis
Arbor Solutions, Inc.
lbolhuis@arbsol.com

CL Command Enhancements

- There have been new and changed IBM CL commands in EVERY release
- For V5R3:
  - 57 new CL commands
  - 247 changed CL commands
- A small number of CL commands compromise the CL HLL

Support for Integer Variables

- New TYPE values on DCL statement
- Values
  - *INT – Integer
  - *UINT Unsigned Integer
    - chosen for consistency with PARM TYPE values
- LEN(2) and LEN(4) supported
- OPM does not fully support 8-byte integers
  - Use CLLE

Support for Integer Variables

- Much "cleaner" than using %BIN
  - Use the value natively
- Useful for
  - passing parameters to OS/400 APIs
  - passing parameters to other HLL programs
- Command PARM statement will allows RTNVAL(*YES) for integer parameters

Control Flow Enhancements

Additional ‘standard’ control flow commands:

- DOWHILE, DOUNTIL, DOFOR

  Each support

  – LEAVE

  – ITERATE

- CASE

  SELECT, WHEN, OETHERWISE, ENDSELECT

  25 level nesting

Common DOxxx Support

- Loop starts with the DOxxx statement
  - The DOxxx statement supports a label (note this)
- ENDDO marks end of loop
  - All types of DO loop use ENDDO
- ITERATE – Discontinue processing remainder of code before ENDDO and transfer to label on DOxxx
  - Can be the label on the current DOxxx or loops external to this loop
  - If no label given the current DOxxx loop is assumed

Common DOxxx Support

- LEAVE – Discontinue processing remainder of loop and jump to statement following the matching ENDDO
  - Can be the label on the DOxxx or the DOxxx loops external to this loop
  - If no label given the current DOxxx loop is assumed
- Can be nested (up to 25 levels)
  - i.e. you could have a DOWHILE loop within a DOFOR loop
  - or a DOWHILE inside a DOWHILE etc.

DOWHILE Loop

- Same COND support as IF statement in CL
- Evaluates COND at "top" of loop
- A simple example:

```
DCL VAR(&LGL) TYPE(*LGL) VALUE('1')
:
DOWHILE COND(&LGL)
: (group of CL commands)
ENDDO
```

DOUNTIL Loop

- Same COND support as IF statement in CL
- Evaluates COND at "bottom" of loop
- A simple example:

```
DCL VAR(&LGL) TYPE(*LGL) VALUE('0')
:
DOUNTIL COND(&LGL)
: (group of CL commands)
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

LEAVE and ITERATE

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

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
- May also have an OTHERWISE

SELECT Group

- WHEN
  - Has COND and THEN support (like IF)
  - To execute multiple statements must use DO/ENDDO
- OTHERWISE
  - Run if no WHEN statement COND = True
  - Single parm of CMD (like ELSE)
  - Again needs DO/ENDDO for multiple statements

SELECT Example

```
SELECT

 WHEN  COND((&COUNT *EQ 4) *AND (&COUNT2 *EQ 2)) THEN(DO)
   ..some important stuff...
  ENDDO

 WHEN  COND(&COUNT *EQ 6) THEN(DO)
   ..some different important stuff..
  ENDDO

 OTHERWISE  CMD(DO)
   ..default important stuff..
  ENDDO
ENDSELECT
```

Multiple File Support

- Supports up to 5 file "instances"
- Instances can be for the same file 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 10-character name (*SNAME)

Multiple File Support (continued)

- If OPNID name specified, declared CL variables are prefixed by this name and an underscore (e.g. &OPENIDENT5_FLDA)
- OPNID also added to existing file input/output CL statements
  - RCVF
  - ENDRCV
  - SNDF
  - SNDRCVF
  - WAIT

Increased size for *CHAR var

- Previous limit was 9999 bytes for CL variables declared as 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

Incr. max 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 stay at 99

Parameter passing "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 OS/400 procedure APIs
- Maximum numbers of parameters still 300

Follow-on CL Compiler 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 They’re not done yet!**
- Rochester is currently working on the next set of enhancements
- They are looking for early feedback & missed function
- Cards and letters to Guy Vig gwvig@us.ibm.com

Subroutines

- Simple code block between SUBR and ENDSUBR statements
- Invoked by new GOSUBR statement
  - No argument/parameter passing
  - No local scoping of subroutine variables
  - No nesting allowed (subroutines in subroutines)
- Return to caller via RTNSUBR or ENDSUBR
- Would not allow GOTO from outside of subroutine to label within the subroutine
- Using GOTO to leave SUBR gives warning

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 *BASED attribute on DCL statement
- Add *DEFINED attribute on DCL statement
- Allow pointer to be used with %SUBSTRING
- Makes many functions available to ILE CL
  - Full record-level file I/O
  - String functions

Faster CL program startup

- Might support option to not initialize CL variables that don't have initial VALUE
- Investigating not initializing compiler temporary variables (many done twice today)
- Might allow variables to be in **static** storage (currently all variables in automatic storage)
- Could support **static external** CL variables for ILE CL
  - Would enable sharing across procedures and languages (e.g. between CL and C)

Other possible improvements

- Provide higher precision for *DEC variables
- Provide 8-byte integers (ILE CL only)
- DCLF support for large character fields & integer fields
- Arrays (may limit to single-dimension)
- Structures (may limit substructure nesting)
- Date, Time, Timestamp, Float data types
- Enhanced generic name parameter values
  - Generic suffix support
  - Single-character wildcard support
- Proxy command support

Continuing to deliver improvements

- Intention is to keep adding improvements
- Rochester wants to deliver enhancements that will delight iSeries customers, including business partners
  - If They're hitting the mark, tell an IBM exec
  - If They've missed, tell Guy Vig (gwvig@us.ibm.com)
- Funding at risk if little or no positive customer feedback