IBM i CL Command Parameter Grammar and Syntax

This document describes the structure of IBM i CL command parameters, including PARM, ELEM, and QUAL definitions. It provides grammar rules, usage of MAX(n) constraints, and examples of syntax. The purpose is to support AI prompting and VS Code CL Prompter extensions.

1. Command

```
<command> ::= <parm>+
```

2. PARM

```
<parm> ::= <primitive>                        // MAX(1)
         | <elem>                             // MAX(1)
         | <qual>                             // MAX(1)
         | <parm-group>                       // MAX(n > 1)

<parm-group> ::= "(" <parm> { <parm> } ")"

- MAX(1): value appears without parentheses.
- MAX(n > 1): entire parameter group must be enclosed in parentheses, and may repeat up to n
```

```
Examples:
OBJTYPE(*FILE *PGM *DTAARA)          // MAX(50), primitive repeated
OBJ(QGPL/DATAF1 QTEMP/DATAF2 CHGJOB) // MAX(50), QUAL repeated
MSGID((CPF9898 QSYS/QCPFMSG) (MCH3601 QCPFMSG)) // MAX(50), ELEM repeated
```

3. ELEM

```
<elem> ::= <primitive> { <primitive> }        // single group
         | "(" <elem-group> { <elem-group> } ")"

<elem-group> ::= <primitive> { <primitive> }

- An ELEM is two or more primitives grouped together.
- With MAX(n > 1), each group is enclosed in parentheses.
```

```
Examples:
JOB(4 0 *SECLVL)                              // 3-element ELEM, MAX(1)
MSGID((CPF9898 QSYS/QCPFMSG) (MCH3601 QCPFMSG)) // repeating ELEM, MAX(50)
```

4. QUAL

```
<qual> ::= <qual-value> { "/" <qual-value> }

<qual-value> ::= <primitive>

- QUALs are slash-delimited.
- Number of QUAL children is fixed.
- When prompting:
  - Values are presented in reverse order (q1/q2/q3 → prompt fields: q3, q2, q1).
  - Missing values are assumed from the left.
```

```
Examples:
OBJ(QGPL/DATAF1)     // library/object
OBJ(MYLIB/MYFILE/MYMBR) // library/file/member
```

5. ASCII Railroad Diagrams

```
PARM
+-------------------+
|  primitive        |  MAX(1)
+-------------------+
         |
         +-------------------+
         |       elem        |  MAX(1)
         +-------------------+
         |
         +-------------------+
         |       qual        |  MAX(1)
         +-------------------+
         |
         +---------------------------------+
         | "(" parm { parm } ")"            |  MAX(n > 1)
         +---------------------------------+

ELEM
primitive -- primitive -- [ primitive ... ]        MAX(1)

   OR

"(" elem-group { elem-group } ")"                  MAX(n > 1)

QUAL
primitive "/" primitive [ "/" primitive ... ]      MAX(1)
```