IBM i CL Command Parameter Grammar and Syntax

This document contains two parts: 1. The original descriptive text provided. 2. A cleaned and structured grammar version with MAX constraints and ASCII diagrams.

Original Text

```
CL Prompting is done based on the type of parameters.
All parameters begin with a base level PARM statement.
A PARM statement may be defined as either a primitive type, such as *CHAR, *INT, *DEC and oth
In addition, PARM statements may contain a MAX attribute. The default is 1, but when Max > 1
When Max > 1, then a simple PARM or a PARM that is made up of only top-level QUAL or non-nes
For example a simple PARM named OBJTYPE with TYPE(*CHAR) and LEN(10) and MAX=50 would look s
A PARM named OBJ that is also a 2-QUAL parameter with MAX=50, might look like this: OBJ(QGPL
A PARM named MSGID that is made up of 2 ELEM statement where the first one is TYPE(*CHAR) LE
If we then add Max > 1 (let's use MAX=50) then the dynamics of the syntax for this parameter
MSGID((CPF9898 QSYS/QCPFMSG) (MCH3601 QCPFMSG))
Where each repeatable ELEM group is enclosed in parens. The paren requirement occurs when 1 

Max=1 ->  MSGID(CPF9898 QSYS/QCPFMSG)
Max=50 -> MSGID((CPF9898 QSYS/QCPFMSG))
Similarly, the CPYF command's INCREL parameter has a 4-part ELEM parameter that has Max=50. S
INCREL((*IF IMDEL *EQ 'D') (*OR IMDEL *EQ ' ') (*OR IMDEL *NE '1'))

When a simple type is not enough, and a structure or repeating list (or repeating list of st
An ELEM is 2 or more parameter definitions passed as one value to the called program. But on
JOB(4 0 *SECLVL) is a 3-ELEM parameter.
Each ELEM can be a primitive type or another ELEM statement (called a nested ELEM) or a QUAL
Nested ELEM statement are grouped to that nesting level with parens on input and should be en
A QUAL is 2 or more items that are similar to ELEM except they are joined together with a qua
QUAL statement cannot contain any child members. They are also prompted in reverse order. Fo
q3
q2
q1
If only a subset of the number of QUAL children is specified, then populating the prompter as
```

Revised Grammar and Syntax

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

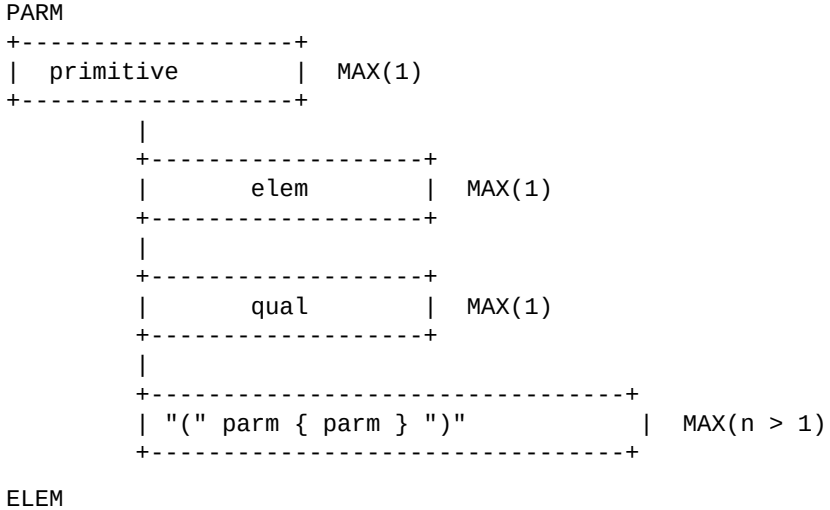
```
<qual> ::= <qual-value> { "/" <qual-value> }

<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



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

   OR

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

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