**Legend :**

**Lexical rule**

**Syntax rule : not in grammar**

* Compare with grammar : may be more precise than grammar, additional checks to add after parsing
* Do not merge two syntax forms when runtime behavior is completely different

**Semantic rule – type checking**

* Check if the identifiers used in the statement are valid and are pointing to valid data
* Compute the type properties of the result data

**Runtime behavior – dangerous behavior**

* Create different statement objects if the behavior is completely different
* Properties to create on the statements objects
* Implement program flow and data impacts analysis
* Add warning at semantic rules level to prevent legal but dangerous behaviors

**MOVE statement**

The MOVE statement transfers data from one area of storage to one or more other

areas.

**Format 1: MOVE statement**

\_\_ MOVE *identifier-1 | literal-1* TO \_ *identifier-2* +

**Format 2: MOVE statement with CORRESPONDING phrase**

\_\_ MOVE CORRESPONDING | CORR *identifier-1* TO *identifier-2*

CORR is an abbreviation for, and is equivalent to, CORRESPONDING.

***identifier-1* , *literal-1***

The sending area.

***identifier-2***

The receiving areas. *identifier-2* must not reference an intrinsic function.

When format 1 is specified:

v All identifiers can reference alphanumeric group items, national group items, or

elementary items.

v When one of *identifier-1* or *identifier-2* references a national group item and the

other operand references an alphanumeric group item, the national group is

processed as a group item; in all other cases, the national group item is

processed as an elementary data item of category national.

v The data in the sending area is moved into the data item referenced by each

*identifier-2* in the order in which the *identifier-2* data items are specified in the

MOVE statement. See “Elementary moves” on page 370 and “Group moves” on

page 374 below.

When format 2 is specified:

v Both identifiers must be group items.

v A national group item is processed as a group item (and not as an elementary

data item of category national).

v Selected items in *identifier-1* are moved to *identifier-2* according to the rules for

the “CORRESPONDING phrase” on page 281. The results are the same as if

each pair of CORRESPONDING identifiers were referenced in a separate MOVE

statement.

Data items described with the following types of usage cannot be specified in a

MOVE statement:

v INDEX

v POINTER

v FUNCTION-POINTER

v PROCEDURE-POINTER

v OBJECT REFERENCE

A data item defined with a usage of INDEX, POINTER, FUNCTION-POINTER,

PROCEDURE-POINTER, or OBJECT REFERENCE can be part of an alphanumeric

group item that is referenced in a MOVE CORRESPONDING statement; however,

no movement of data from those data items takes place.

The evaluation of the length of the sending or receiving area can be affected by the

DEPENDING ON phrase of the OCCURS clause (see “OCCURS clause” on page

191).

If the sending field (*identifier-1*) is reference-modified or subscripted, or is an

alphanumeric or national function-identifier, the reference-modifier, subscript, or

function is evaluated only once, immediately before data is moved to the first of

the receiving operands.

Any length evaluation, subscripting, or reference-modification associated with a

receiving field (*identifier-2*) is evaluated immediately before the data is moved into

that receiving field.

For example, the result of the statement:

MOVE A(B) TO B, C(B).

is equivalent to:

MOVE A(B) TO TEMP.

MOVE TEMP TO B.

MOVE TEMP TO C(B).

where TEMP is defined as an intermediate result item. The subscript B has changed

in value between the time that the first move took place and the time that the final

move to C(B) is executed.

For further information about intermediate results, see *Appendix A. Intermediate*

*results and arithmetic precision* in the *Enterprise COBOL Programming Guide*.

After execution of a MOVE statement, the sending fields contain the same data as

before execution.

**Usage note:** Overlapping operands in a MOVE statement can cause unpredictable

results.

**Elementary moves**

An elementary move is one in which the receiving item is an elementary data item

and the sending item is an elementary data item or a literal.

Valid operands belong to one of the following categories:

v **Alphabetic**: includes data items of category alphabetic and the figurative

constant SPACE

v **Alphanumeric**: includes the following items:

– Data items of category alphanumeric

– Alphanumeric functions

– Alphanumeric literals

– The figurative constant ALL *alphanumeric-literal* and all other figurative

constants (except NULL) when used in a context that requires an

alphanumeric sending item

v **Alphanumeric-edited**: includes data items of category alphanumeric-edited

v **DBCS**: includes data items of category DBCS, DBCS literals, and the figurative

constant ALL DBCS-literal.

v **External floating-point**: includes data items of category external floating point

(described with USAGE DISPLAY or USAGE NATIONAL) and floating-point

literals.

v **Internal floating-point**: includes data items of category internal floating-point

(defined as USAGE COMP-1 or USAGE COMP-2)

v **National**: includes the following items:

– National group items (treated as elementary item of category national)

– Data items of category national

– National literals

– National functions

– Figurative constants ZERO, SPACE, QUOTE, and ALL *national-literal* when

used in a context that requires a national sending item

v **National-edited**: includes data items of category national-edited

v **Numeric**: includes the following items:

– Data items of category numeric

– Numeric literals

– The figurative constant ZERO (when ZERO is moved to a numeric or

numeric-edited item).

v **Numeric-edited**: includes data items of category numeric-edited.

**Elementary move rules**

Any necessary conversion of data from one form of internal representation to

another takes place during the move, along with any specified editing in, or

de-editing implied by, the receiving item. The code page used for conversion to or

from alphanumeric characters is the one in effect for the CODEPAGE compiler

option when the source code was compiled.

The following rules outline the execution of valid elementary moves. When the

receiving field is:

**Alphabetic**:

v Alignment and any necessary space filling or truncation occur as described

under “Alignment rules” on page 166.

v If the size of the sending item is greater than the size of the receiving item,

excess characters on the right are truncated after the receiving item is filled.

**Alphanumeric** or **alphanumeric-edited**:

v If the sending item is a national decimal integer item, the sending data is

converted to usage DISPLAY and treated as though it were moved to a

temporary data item of category alphanumeric with the same number of

character positions as the sending item. The resulting alphanumeric data item is

treated as the sending item.

v Alignment and any necessary space filling or truncation take place, as described

under “Alignment rules” on page 166.

v If the size of the sending item is greater than the size of the receiving item,

excess characters on the right are truncated after the receiving item is filled.

v If the initial sending item has an operational sign, the unsigned value is used. If

the operational sign occupies a separate character, that character is not moved,

and the size of the sending item is considered to be one less character than the

actual size.

**DBCS:**

v If the sending and receiving items are not the same size, the sending data is

either truncated on the right or padded with DBCS spaces on the right.

**External floating-point**:

v For a floating-point sending item, the floating-point value is converted to the

usage of the receiving external floating-point item (if different from the sending

item's representation).

v For other sending items, the numeric value is treated as though that value were

converted to internal floating-point and then converted to the usage of the

receiving external floating-point item.

**Internal floating-point**:

v When the category of the sending operand is not internal floating-point, the

numeric value of the sending item is converted to internal floating-point format.

**National** or **national-edited**:

v If the representation of the sending item is not national characters, the sending

data is converted to national characters and treated as though it were moved to

a temporary data item of category national of a length not to cause truncation or

padding. The resulting category national data item is treated as the sending data

item.

v If the representation of the sending item is national characters, the sending data

is used without conversion.

v Alignment and any necessary space filling or truncation take place as described

under “Alignment rules” on page 166. The programmer is responsible for

ensuring that multiple encoding units that together form a graphic character are

not split by truncation.

v If the sending item has an operational sign, the unsigned value is used. If the

operational sign occupies a separate character, that character is not moved, and

the size of the sending item is considered to be one less character than the actual

size.

**Numeric** or **numeric-edited**:

v Except when zeros are replaced because of editing requirements, alignment by

decimal point and any necessary zero filling take place, as described under

“Alignment rules” on page 166.

v If the receiving item is signed, the sign of the sending item is placed in the

receiving item, with any necessary sign conversion. If the sending item is

unsigned, a positive operational sign is generated for the receiving item.

**372** Enterprise COBOL for z/OS, V5.1.1 Language Reference

v If the receiving item is unsigned, no operational sign is generated for the

receiving item and the absolute value of the sending item is used in the move.

v When the category of the sending item is alphanumeric, alphanumeric-edited,

national, or national-edited, the data is moved as if the sending item were

described as an unsigned integer.

v When the sending item is floating-point, the data is first converted to either a

binary or internal decimal representation and is then moved.

v When the receiving item is numeric-edited, editing takes place as defined by the

picture character string or BLANK WHEN ZERO clause associated with the

receiving item.

v When the sending item is numeric-edited, the compiler de-edits the sending data

to establish the unedited value of the numeric-edited item (this value can be

signed). The unedited numeric value is used in the move to the receiving

numeric or numeric-edited data item.

**Usage notes:**

1. If the receiving item is of category alphanumeric, alphanumeric-edited,

numeric-edited, national, or national-edited and the sending field is numeric,

any digit positions described with picture symbol P in the sending item are

considered to have the value zero. Each P is counted in the size of the sending

item.

2. If the receiving item is numeric and the sending field is an alphanumeric literal,

a national literal, or an ALL literal, all characters of the literal must be numeric

characters.

**Valid and invalid elementary moves**

The table shows valid and invalid elementary moves for each category.

In the table:

v YES = Move is valid.

v NO = Move is invalid.

v Column headings indicate receiving item categories; row headings indicate

sending item categories.

*Table 42.* **Valid and invalid elementary moves**

**Valid and**

**invalid**

**elementary**

**moves**

**Alphabetic**

**Alphanumeric**

**Alphanumeric**

**edited Numeric**

**Numericedited**

**External**

**floatingpoint**

**Internal**

**floatingpoint**

**DBCS1**

**National,**

**nationaledited**

Alphabetic and

SPACE sending

item

Yes Yes Yes No No No No No Yes

Alphanumeric

sending item2

Yes Yes Yes Yes3 Yes3 Yes8 Yes8 No Yes

Alphanumericedited

sending

item

Yes Yes Yes No No No No No Yes

Numeric integer

and ZERO

sending item4

No Yes Yes Yes Yes Yes Yes No Yes

Numeric

noninteger

sending item5

No No No Yes Yes Yes Yes No No

Chapter 20. PROCEDURE DIVISION statements **373**

*Table 42.* **Valid and invalid elementary moves** *(continued)*

**Valid and**

**invalid**

**elementary**

**moves**

**Alphabetic**

**Alphanumeric**

**Alphanumeric**

**edited Numeric**

**Numericedited**

**External**

**floatingpoint**

**Internal**

**floatingpoint**

**DBCS1**

**National,**

**nationaledited**

Numeric-edited

sending item

No Yes Yes Yes Yes Yes Yes No Yes

Floating-point

sending item6

No No No Yes Yes Yes Yes No No

DBCS sending

item7

No No No No No No No Yes Yes

National sending

item9

No No No Yes Yes Yes Yes No Yes

National-edited

sending item

No No No No No No No No Yes

1. Includes DBCS data items.

2. Includes alphanumeric literals.

3. Figurative constants and alphanumeric literals must consist only of numeric characters and will be treated as

numeric integer fields.

4. Includes integer numeric literals.

5. Includes noninteger numeric literals.

6. Includes floating-point literals, external floating-point data items (USAGE DISPLAY or USAGE NATIONAL),

and internal floating-point data items (USAGE COMP-1 or USAGE COMP-2).

7. Includes DBCS data-items, DBCS literals, and figurative constant SPACE.

8. Figurative constants and alphanumeric literals must consist only of numeric characters and will be treated as

numeric integer fields. The ALL literal cannot be used as a sending item.

9. Includes national data items, national literals, national functions, and figurative constants ZERO, SPACE,

QUOTE, and ALL national literal.

**Moves involving file record areas**

The successful execution of an OPEN statement for a given file makes the record

area for that file available. You can move data to or from the record description

entries associated with a file only when the file is in the open status.

Execution of an implicit or explicit CLOSE statement removes a file from open

status and makes the record area unavailable.

**Group moves**

A group move is any move in which an alphanumeric group item is a sending

item or a receiving item, or both.

The group moves are:

v A move to an alphanumeric group item from one of the following items:

– any elementary data item that is valid as a sending item in the MOVE

statement

– a national group item

– a literal

– a figurative constant

v A move from an alphanumeric group item to the following items:

**374** Enterprise COBOL for z/OS, V5.1.1 Language Reference

– any elementary data item that is valid as a receiving item in the MOVE

statement

– a national group item

– an alphanumeric group item

A group move is treated as though it were an alphanumeric-to-alphanumeric

elementary move, except that there is no conversion of data from one form of

internal representation to another. In a group move, the receiving area is filled

without consideration for the individual elementary items contained within either

the sending area or the receiving area, except as noted in the OCCURS clause. (See

“OCCURS clause” on page 191.)