The COPY verb

Introduction

Aims

In a large software system it is often very useful to be able keep record, file and table descriptions in a central source text library and then to import those descriptions into the programs that require them. In COBOL the COPY verb allows us to do this.

This unit introduces the COPY verb.

It describes some problems of large software systems which use of the COPY helps to alleviate.

It introduces the syntax of the COPY verb and discusses how the COPY verb, and in particular they COPY..REPLACING, works.

It introduces the notion of a text word and describes the role that these play in matching the text in the REPLACING phrase with the text in the library file.

The unit ends with a number of example programs.

Objectives

By the end of this unit you should -

1. Be aware of some of the advantages of using the COPY verb.
2. Understand what a text word and be able to identify text words in library text.
3. Be able to use the COPY.. REPLACING to copy source text from a library file and replace sections of it as required.

Prerequisites

Introduction to COBOL
Declaring data in COBOL
Basic Procedure Division commands

To top of page

Using the COPY verb

Introduction

COPY statements in a COBOL program are very different from other COBOL statements. While other statements are executed at run time, COPY statements are executed at compile time.

A COPY statement is similar to the "Include" used in languages like C and C++. It allows programs to include frequently used source code text.

When a COPY statement is used in a COBOL program the source code text is copied into the program from from a copy file or from a copy library before the program is compiled.

A copy file, is a file containing a segment of COBOL code.

A copy library, is a collection code segment each of which can be referenced using a name. Each program that wants to use items described in the copy library uses the COPY verb to include the descriptions it requires.

When COPY statements include source code in a program, the code can be included without change or the text can be changed as it is copied into the program. The ability to change the code as it being included greatly adds to the versatility of the COPY verb.

Using the COPY verb in large software systems

The COPY verb is generally used when creating large software systems. These systems are subject to a number of problems that the COPY verb helps to alleviate.

For instance, when data files are used by a number of programs in a large software system, it is easy for programmers describing those files to make errors in defining the key fields (Indexed files) or the file organization or the type of access allowed or the number, type and size of the fields in a record. Any errors in these descriptions may result in difficult-to-find bugs. For instance, if one program writes a file using an incorrect record description and the others read it using the correct description, a program crash may occur in in one of the correct subprograms rather than the one which actually has the problem.

Using copy libraries or files helps to reduce programmer transcription errors and also makes implementation simpler by reducing the amount of coding required. For instance, when a number of programs need to access the same file, the relevant file and record descriptions can be copied from a copy library instead of each programmer having to type their own (and possibly get them wrong).

Another advantage of using copy libraries is that they permit item descriptions such as file, record and table descriptions, to be kept and updated centrally in a copy library, often under the control of a copy librarian. This makes it more difficult for programmers to make ad hoc changes to file and record formats. Such changes generally have to be approved by the COPY librarian.

In a large software system using the COPY verb makes some maintenance tasks easier and safer. Certain changes may only require an update to the text in the copy library and a recompilation of any affected programs. If a copy library were not used each affected program would have to be changed

---

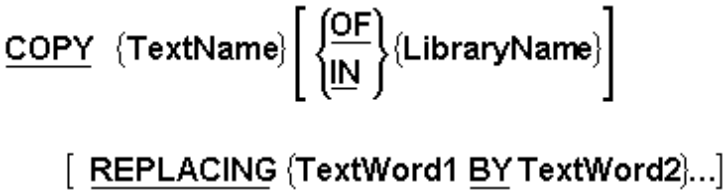
To top of page

The COPY verb
Syntax and Semantics

COPY Syntax


Detail

The syntax diagram shown opposite has been simplified for the sake of completeness and clarity. It is the author's opinion that the standard syntax diagram does not clearly identify all the replaceable objects.



Text Words
The matching procedure in the REPLACING phrase operates on text words.

A text word is defined as ;

- Any COBOL text enclosed in double equal signs (e.g. ==ADD 1==). This text is known as PseudoText and it allows us to replace a series of words or characters as opposed to an individual identifier, literal or word.
- Any literal including opening and closing quotes
- Any separator other than;□
  - A space
  - A Pseudo-Text delimiter□
  - A Comma
- Any COBOL reserved word
- Any Identifier.
- Any other sequence of contiguous characters bounded by separators.

Text Word examples

MOVE
1 Text Word

MOVE Total TO Print-Total
4 Text Words - **MOVE Total TO Print-Total**

MOVE Total TO Print-Total.
5 Text Words - **MOVE Total TO Print-Total .**

PIC S9(4)V9(6)
9 Text words - **PIC S9 (4) V9 (6)**

"PIC S9(4)V9(6)"
1 Text word - **"PIC S9(4)V9(6)"**

How the COPY works

If the REPLACING phrase is not used the compiler simply copies the text into the client program without change.

If the COPY does use the REPLACING phrase the text is copied and each occurrence of *TextWord1* in the library text is replaced by the corresponding *TextWord2* in the REPLACING phrase.

Some example COPY statements

```
COPY CopyFile2 REPLACING XYZ BY 120.
COPY CopyFile5 REPLACING X(3) BY X(19).
COPY CopyFile3 IN EGLIB.
COPY CopyFile5 REPLACING ==X(3)== BY ==X(19)==.
COPY CopyFile4 IN EGLIB.
```

How the REPLACING phrase works

The REPLACING phrase tries to match the TextWord1 items with text words in the library text. If a match is achieved, then as the text is copied from the library, the matched text is replaced by the TextWord2 items.

For purposes of matching, each occurrence of a separator comma, semicolon, space or comment line in the library text or in TextWord1 is considered to be a single space.

Each sequence of one or more space separators is also considered to be a single space.

Comment lines in the library text is or in TextWord2 are copied into the target text unchanged.

COPY Rules

1. When TextWord1 is PseudoText it must not be null, nor can it consist solely of either the character space(s) or comment lines.

2. When TextWord2 is PseudoText it can be null. Null PseudoText is indicated by four equal signs (e.g. ====)

3. A COPY statement can occur anywhere a character-string or a separator can occur except that a COPY statement must not occur within another COPY statement.

4. TextName defines a unique external filename which conforms to the rules for user defined words.

5. If the word COPY appears in a comment-entry or in the place where a comment entry can appear, it is considered part of the comment-entry.

6. The text produced as a result of the complete processing of a COPY statement must not contain a COPY statement.

7. Normally a textword in the copy text is replaced by a textword in the replacement text but it is possible to replace only part of a text-word by enclosing the text to be replaced by either parentheses or colons.
For example - (Prefix) or :Prefix:

To top of page

COPY examples

COPY Example1

Text in **green** is the copy statement in the source text before processing.

Text in **red** is the copied library text which may have been altered by the REPLACING phrase if one was present and found matching text in the library file.

Text in **grey** is the COPY statement which has been applied.

Source Text

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG1.
AUTHOR.  Michael Coughlan.

ENVIRONMENT DIVISION.
FILE-CONTROL.
    SELECT StudentFile ASSIGN TO "STUDENTS.DAT"
    ORGANIZATION IS LINE SEQUENTIAL.

DATA DIVISION.
FILE SECTION.
FD StudentFile.
COPY COPYFILE1.

PROCEDURE DIVISION.
BeginProg.
   OPEN INPUT StudentFile
   READ StudentFile
      AT END SET EndOfSF TO TRUE
   END-READ
   PERFORM UNTIL EndOfSF
      DISPLAY StudentNumber SPACE StudentName SPACE
              CourseCode SPACE FeesOwed SPACE AmountPaid
      READ StudentFile
         AT END SET EndOfSF TO TRUE
      END-READ
   END-PERFORM
   STOP RUN.
```

Text in COPYFILE1

```
01  StudentRec.
    88  EndOfSF      VALUE HIGH-VALUES.
    02  StudentNumber                PIC 9(7).
    02  StudentName                  PIC X(60).
    02  CourseCode                   PIC X(4).
    02  FeesOwed                     PIC 9(4).
```

```
    02  AmountPaid                  PIC 9(4)V99.
```

Source Text after processing

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG1.
AUTHOR.  Michael Coughlan.

ENVIRONMENT DIVISION.
FILE-CONTROL.
    SELECT StudentFile ASSIGN TO "STUDENTS.DAT"
    ORGANIZATION IS LINE SEQUENTIAL.

DATA DIVISION.
FILE SECTION.
FD StudentFile.
COPY CopyFile1.
01  StudentRec.
    88  EndOfSF      VALUE HIGH-VALUES.
    02  StudentNumber               PIC 9(7).
    02  StudentName                 PIC X(60).
    02  CourseCode                  PIC X(4).
    02  FeesOwed                    PIC 9(4).
    02  AmountPaid                  PIC 9(4)V99.

PROCEDURE DIVISION.
BeginProg.
   OPEN INPUT StudentFile
   READ StudentFile
      AT END SET EndOfSF TO TRUE
   END-READ
   PERFORM UNTIL EndOfSF
      DISPLAY StudentNumber SPACE StudentName SPACE
              CourseCode SPACE FeesOwed SPACE AmountPaid
      READ StudentFile
         AT END SET EndOfSF TO TRUE
      END-READ
   END-PERFORM
   STOP RUN.
```

COPY Example2

Source Text

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG2.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 NameTable.
*See note1
COPY CopyFile2 REPLACING XYZ BY 120.

01 CopyData.
*See note2
COPY CopyFile3 IN EGLIB
     REPLACING ==R== BY ==4==.
```

```
*See note3
COPY CopyFile4 REPLACING ==V99== BY ====.

*See note4
COPY CopyFile5 REPLACING "KEY" BY "ABC".

*See note5
COPY CopyFile5 REPLACING KEY BY ABC.

*See note6
COPY CopyFile5 REPLACING "CustKey" BY "Cust".

*See note7
COPY CopyFile5 REPLACING "KEY" BY =="ABC".
     02 CustNum              PIC 9(8)==.


PROCEDURE DIVISION.
BeginProg.

STOP RUN.
```

Text in CopyFile2

```
    02 StudName PIC X(20) OCCURS XYZ TIMES.
```

Text in CopyFile3

```
    02  CustOrder           PIC 9(R).
```

Text in CopyFile4

```
    02 CustOrder2           PIC 9(6)V99.
```

Text in CopyFile5

```
    02 CustKey              PIC X(3) VALUE "KEY".
```

Source Text after processing

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG2.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 NameTable.
*See note1
COPY CopyFile2 REPLACING XYZ BY 120.
    02 StudName PIC X(20) OCCURS 120 TIMES.

01 CopyData.
*See note2
COPY CopyFile3 IN EGLIB
     REPLACING ==R== BY ==4==.
    02  CustOrder           PIC 9(4).

*See note3
```

Note1
The TextWord XYZ in the library text is replaced by 120.

Note2
The PseudoText symbols tell us that we are looking for the text word R in the library text. Other R's in the text which might be part of a word are not matched because they are not text words. The replaced R is a text word because it is bounded by two other text words i.e. the parentheses (and).

Note3
The PseudoText ==V99== is replaced by nothing; effectively deleting it.

Note4
The literal "KEY" in the library text is replaced by "ABC"

Note5
In this case the library text is copied without change because the literal "KEY" in the library text is not the same as the word *KEY* in TextWord2.

Note6
Similar to the previous example demonstrates the difference between CustKey and the literal "CustKey". A literal text word includes the opening and closing quotes.

Note7
Replaces "KEY" by two lines of text. Note that we are only replacing the literal "KEY" in the library text. So the full stop after "KEY" is not replaced and that is why there is no full stop after the PIC 9(8) in the TextWord2 replacement text.

```
COPY CopyFile4 REPLACING ==V99== BY ====.
    02 CustOrder2           PIC 9(6).

*See note4
COPY CopyFile5 REPLACING "KEY" BY "ABC".
    02 CustKey              PIC X(3) VALUE "ABC".

*See note5
COPY CopyFile5 REPLACING KEY BY ABC.
    02 CustKey              PIC X(3) VALUE "KEY".

*See note6
COPY CopyFile5 REPLACING "CustKey" BY "Cust".
    02 CustKey              PIC X(3) VALUE "KEY".

*See note7
COPY CopyFile5 REPLACING "KEY" BY =="ABC".
     02 CustNum              PIC 9(8)==.
     02 CustKey              PIC X(3) VALUE "ABC".
     02 CustNum              PIC 9(8).


PROCEDURE DIVISION.
BeginProg.


STOP RUN.
```

COPY Example3

Source Text

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG3.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 CopyData.
*See note1
COPY CopyFile5 REPLACING ( BY @.

*See note2
COPY CopyFile5 REPLACING 3 BY three.

*See note3
COPY CopyFile5 REPLACING ) BY &.

*See note4
COPY CopyFile5 REPLACING X BY ==Replace the X==.

*See note5
COPY CopyFile5 REPLACING ==X(3)== BY ==X(19)==.

*See note6
COPY CopyFile5 REPLACING X(3) BY X(19).

*See note7
COPY CopyFile5 REPLACING PIC BY ==Pic is Replaced==.
```

```
*See note8
COPY CopyFile5 REPLACING P BY ==But P in PIC not replaced==.

PROCEDURE DIVISION.
BeginProg.

   STOP RUN.
```

Note that these COPY statements, which are themselves syntactically correct, result in statements which cause syntax errors. This is a clear indication that COPY statements are processed before the resulting program is compiled.

Note1
The text word (in the library text is replaced by @. Demonstrates that (is a text word.

Note2
Demonstrates that the 3 between (and) is a text word.

Note3
Demonstrates that) is a textword.

Note4
Demonstrates that the X before (3) is a textword and is replaced by the PseudoText "Replace the X"

Note5
The PseudoText X(3) is replaced by X(19)

Note6
This looks the same as number 5 but what is actually happening is that the series of textwords **X** and **(** and **3** and **)** is replaced by the series **X (19)**

Note7
The textword PIC (bounded by separator spaces) is replaced by the PseudoText *Pic is Replaced*

Note8
But the letter P in PIC is not a textword by itself and so is not replaced.

Library Text in CopyFile5

```
    02 CustKey              PIC X(3) VALUE "KEY".
```

Source Text after processing

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG3.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.

01 CopyData.
*See note1
COPY CopyFile5 REPLACING ( BY @.
    02 CustKey              PIC X@3) VALUE "KEY".

*See note2
COPY CopyFile5 REPLACING 3 BY three.
    02 CustKey              PIC X(three) VALUE "KEY".

*See note3
COPY CopyFile5 REPLACING ) BY &.
    02 CustKey              PIC X(3& VALUE "KEY".

*See note4
COPY CopyFile5 REPLACING X BY ==Replace the X==.
    02 CustKey              PIC Replace the X(3) VALUE "KEY".

*See note5
COPY CopyFile5 REPLACING ==X(3)== BY ==X(19)==.
    02 CustKey              PIC X(19) VALUE "KEY".

*See note6
COPY CopyFile5 REPLACING X(3) BY X(19).
    02 CustKey              PIC X(19) VALUE "KEY".

*See note7
COPY CopyFile5 REPLACING PIC BY ==Pic is Replaced==.
    02 CustKey              Pic is Replaced X(3) VALUE "KEY".

*See note8
COPY CopyFile5 REPLACING P BY ==But P in PIC not replaced==.
    02 CustKey              PIC X(3) VALUE "KEY".

PROCEDURE DIVISION.
BeginProg.

   STOP RUN.
```

COPY Example4

Source Text

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG4.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 Num1     PIC 9 VALUE 3.
01 Num2     PIC 9 VALUE 5.
01 Result   PIC 99 VALUE ZEROS.

01 CopyData.
*See note1
COPY CopyFile6 REPLACING "Mike" BY "Tony"
                         ??    BY 15.

PROCEDURE DIVISION.
BeginProg.
*See note2
COPY CopyFile7 REPLACING N1 BY Num1
                         N2 BY Num2
                         R1 BY Result.
   DISPLAY "The result is = " Result.
   STOP RUN.
```

Library Text in CopyFile6

```
    02 CustKey              PIC X(4) VALUE "Mike".
01  NameTable              PIC X(10)  OCCURS ?? TIMES.
```

Library Text in CopyFile7

```
    ADD N1, N2 GIVING R1.
```

Note1
This example demonstrates that a single COPY statement can be used to replace more than one set of items.

Note2
As above.

Source Text after processing

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG4.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 Num1     PIC 9 VALUE 3.
01 Num2     PIC 9 VALUE 5.
01 Result   PIC 99 VALUE ZEROS.

01 CopyData.
*See note1
COPY CopyFile6 REPLACING "Mike" BY "Tony"
                         ??    BY 15.
    02 CustKey              PIC X(4) VALUE "Tony".

01  NameTable              PIC X(10)  OCCURS 15 TIMES.
```

```
PROCEDURE DIVISION.
BeginProg.
*See note2
COPY CopyFile7 REPLACING N1 BY Num1
                         N2 BY Num2
                         R1 BY Result.
    ADD Num1, Num2 GIVING Result.
    STOP RUN.
```

To top of page

Copyright Notice

Last updated : May 2002

e-mail : CSISwebeditor@ul.ie