Liant Software Corporation

RM/COBOL®

First Edition Supplement A

LIANT

This document is a supplement to the First Edition manuals for Liant Software Corporation's RM/COBOL language. It is assumed that the reader is familiar with programming concepts and with the COBOL language in general.

The information contained herein applies to systems running under the Microsoft Windows operating system.

The information in this document is subject to change without prior notice. Liant Software Corporation assumes no responsibility for any errors that may appear in this document. Liant reserves the right to make improvements and/or changes in the products and programs described in this guide at any time without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted.

Liant Software Corporation
8911 N. Capital of Texas Highway
Austin, TX 78759
U.S.A.

Phone (512) 343-1010
(800) 762-6265
Fax (512) 343-9487

Website http://www.liant.com/

Documentation Release History for the RM/COBOL First Edition Supplement A:

Edition Number	Document Part Number	Applies To Product Version	Publication Date
1A	401242	RM/COBOL version 11 and later	January 2007

Contents

Introduction

Welcome to version 11 of RM/COBOL. The pre-existing features in earlier versions of RM/COBOL are fully described in the RM/COBOL First Edition documents provided with this release. New features added in version 11 are described in this supplement document. Additional information about version 11 of RM/COBOL is provided in the README file provided with the product.

Windows Vista

RM/COBOL 11 is the first version to support use on the new Microsoft Windows Vista operating system. RM/COBOL 11 also supports Windows 2000, Windows XP, and Windows 2003 Server.

Note RM/COBOL 11 does not support earlier Windows versions, including Windows 98, Windows 98 SE, Windows Me, and Windows NT 4.0.

Windows Vista no longer supports the WinHelp form of help files. In RM/COBOL 11, the CodeWatch help file and Syntax Summary help file are now HTML help files (**.chm** files). There are two known issues with HTML help files. The first is that the files must be on a local drive, not a network drive, in order to be able to display the help topics. The second is that the files cannot be renamed without causing some internal links in the file to fail.

COBOL CALL Statement

RM/COBOL now supports up to 2047 items in the USING list of a CALL statement and the USING list of the Procedure Division header to allow passing more parameters to a called program. Prior to version 11, this limit was 255 items. As a result of this enhancement, error message 459 has been changed from

```
"USING phrase in Procedure Division header exceeds 255 data-names."
```

to

```
"USING phrase in Procedure Division header or CALL statement exceeds
2047 data-names."
```

COBOL Source Format

There have been extensive revisions to the source format supported by the RM/COBOL compiler to allow longer source records:

- The source format is still fixed reference format as defined for standard COBOL, but the 80-character limit on total source record length has been extended to a maximum of 65000 characters with a default of 1024 characters.
- The right margin (also called margin R) for program-text has been extended to a maximum of the new total source record length, with a default of 72 characters for backwards compatibility.
- Compiler directives have been added to allow modifying margin R, control the listing of source lines, and start a new page in the listing.
- Compiler configuration keywords have been added to set the maximum source record length and to change the default initial right margin setting.

In fixed format, columns 1 through 6 are commentary, column 7 is the indicator column, margin A is after column 7, and margin B is after column 11. Thus, area A is columns 8 through 11 and area B is columns 12 through margin R. Margin R has traditionally been after column 72, with a commentary Identification area in columns 73 through 80.

While the total source record length is established at the start of compilation and cannot be changed during that compilation, the right margin can be varied with a compiler directive statement. The right margin determines where the compiler stops considering text as COBOL program-text to be compiled; characters following the right margin are the commentary area referred to as the Identification area. When the right margin is set to the maximum source record length, then there is no Identification area; in this case, columns 8 through the end of the source record are program-text to be compiled.

The details of the source format revisions and associated RM/COBOL compiler changes are described in the following list.

1. Source records can now vary in length from 0 to 65000 characters in length. By default, however, they can vary from 0 to 1024 characters in length. The maximum record length is determined by a configuration keyword:

   COMPILER-OPTIONS SOURCE-RECORD-MAX-LENGTH=*n*

   The maximum source record length is established at the beginning of a compilation and cannot be changed during that compilation. The default maximum source record length is 1024 in the absence of any configuration specification.

   If *n* is less than 80, it is changed to 80 with no warning or error diagnostic.

   If *n* is greater than 65000, a configuration value error E0009 occurs.

2. The program-text area for a source record is from margin A (after column 7) to margin R and is the area that can contain a comment, if the indicator column contains a fixed comment indicator ("*" or "/"), character-strings (COBOL words), separators, floating indicators (such as ">>" for a directive and "*>" for in-line comments), or all spaces. Following margin R, if margin R is less than the maximum source record length, is the commentary Identification area.

   Margin R can vary from after column 72 to after the end of the maximum record size. The default initial margin R is after column 72, matching traditional COBOL fixed reference format, but can be configured with a configuration keyword:

   COMPILER-OPTIONS INITIAL-MARGIN-R=*n*

   where *n* is the column number after which the Identification area (commentary) begins.

   If *n* is greater than or equal to the maximum source record length, then there is no Identification area; that is, the program-text area of a source record extends to the end of the record. The default initial margin R column is 72.

   If *n* is less than 72, it is changed to 72 with no warning or error diagnostic.

   If *n* is greater than the maximum source record length, it is changed to the maximum source record length with no warning or error diagnostic.

   Margin R can also be changed at any time during the compilation with the RM/COBOL implementer-defined directive:

   When this directive appears in copied library text, the margin R setting reverts to the value before the COPY statement when the end-of-file is reached on the copied file; that is, such a directive in copied library text has no effect on the source that specifies the COPY statement.

   If *integer-1* is less than 72, it is changed to 72 with no error or warning diagnostic.

   If *integer-1* is greater than or equal to the maximum source record length, the directive is equivalent to using the END OF RECORD format. In either of these cases, there is no Identification area; that is, the program-text area of a record extends to the end of the record.

3. Prior to the possibility of source records extending beyond column 72, there was a natural limit on the number of statements per source record. Now that source records can be quite large, a limit on the number of statements per source record is necessary to prevent issues with program debugging and instrumentation. A limit of 64 statements per source record was chosen to avoid such issues. More than 64 statements coded on a single source record is diagnosed with the compiler warning message 774:

   "Number of statements on one source record exceeds 64."

   The remaining statements on the source record are still compiled, but are treated as if they were part of the 64th statement for purposes of debugging and instrumentation. For example, performing a single statement step in the debugger on the 64th statement will step to the first statement on the next line after executing all the remaining statements on the source record.

4. The compiler updates the source column header for the listing file when the margin R setting is changed, but does not automatically force a new page. A new listing page can be forced with the "/" comment indicator or by using the PAGE directive after the directive that changed margin R if the new header is desired immediately. (See item 8 in this list for a description of the PAGE directive.)

   If there is an Identification area, the listing source column header shows the Identification area start with "IDENTFCN". Unless configured differently (see item 5 in this list), the Identification area, if present, is separated from the program-text area by a "|" character in the header and each source line that is printed in the listing. The separator character is suppressed for comment lines that have nonblank characters within two characters of the Identification area (to avoid changing comments that continue from the program-text area into the Identification area) and for directives.

   When there is no Identification area, the listing source column header simply shows a column ruler to the configured listing line length (as described in item 23 in this list), or the maximum source record length if the maximum source record length is less than the configured listing line length.

   Here is an example listing with no Identification area:

```
RM/COBOL (Version 11.00) for 32-Bit Windows                    01/30/2007  11:44:38    Page 1
Source file: stoprunx  Options: L A

  LINE   DEBUG     PG/LN -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7....+....8..

     1                    >>IMP MARGIN-R IS AFTER END OF RECORD
     2                    IDENTIFICATION DIVISION.
     3                    PROGRAM-ID.  STOPRUN.
     4
     5                    ENVIRONMENT DIVISION.
     6
     7                    DATA DIVISION.
```

5. The Identification area separator character value can be configured as follows:

   COMPILER-OPTIONS LISTING-ID-AREA-SEPARATOR=*char*

   If *char* is 0, then no Identification area separator character is printed in the listing source column header nor in source records printed in the listing, which matches the behavior of previous RM/COBOL compilers. The default separator is "|" (or, equivalently, 0x7C).

   Here is an example listing with the default Identification area separator and the default margin R setting:

```
RM/COBOL (Version 11.00) for 32-Bit Windows                    01/30/2007  14:27:01    Page 1
Source file: stoprun  Options: L A

  LINE   DEBUG     PG/LN -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..|IDENTFCN

     1                    IDENTIFICATION DIVISION.                                          |Comment1
     2                    PROGRAM-ID.  STOPRUN.                                             |Comment2
     3                                                                                      |Comment3
     4                    ENVIRONMENT DIVISION.                                             |Comment4
     5                                                                                      |Comment5
     6                    DATA DIVISION.                                                    |Comment6
```

6. The treatment of end of program-text area spaces for unterminated and continued nonnumeric literals depends on whether an Identification area exists for the continued source record.

   When there is an Identification area field in source records, that is, when margin R is less than the maximum source record length (see item 2 in this list for margin R considerations and item 1 for maximum source record size considerations), an unterminated and continued nonnumeric literal is considered to be space filled on the continued record through margin R, regardless of the actual source record length as read from the input device.

   When there is no Identification area, that is, when margin R is greater than or equal to the maximum source record length, only spaces included in the continued record as read from the input device are included in an unterminated and continued nonnumeric literal.

   In order to know the record length as read from disk so that trailing spaces included in the record can be recognized, the compiler's source input file has been changed to use a device type of DISK instead of the prior device type of INPUT. This device type change results in a requirement of a mass storage file for the source input file. In cases where requiring a mass storage file for the input source file causes an incompatibility with existing methods of source input, the configuration:

   COMPILER-OPTIONS SOURCE-ON-INPUT-DEVICE=YES

   has been added to cause the compiler to use the old method of using an INPUT device type for source input. When an INPUT device is configured, the file manager adds trailing spaces to records shorter than the maximum source record length and there is no information available about the actual record length as read from the input device, even if it happens to be a disk. Thus, unterminated and continued nonnumeric literals will have spaces included up to and including the current margin R column following the last non-space character on the continued line when source is on an INPUT device.

7. The directive:

   >> LISTING { ON / OFF }

   can be used to turn the listing of source records on or off at any point in the source file.

   The compiler starts with a default of listing source records in the on state. A LISTING directive without either ON or OFF is equivalent to >>LISTING ON. LISTING directives are listed, even if the listing state is off. When a LISTING directive occurs in copied library text, the listing state reverts at the end of the copy file to what it was before the COPY statement.

   Note The SUPPRESS phrase of the COPY statement, the C Compile Command Option, the E Compile Command Option, and the LISTING-ATTRIBUTES configuration keyword values that suppress certain lines from being printed override a LISTING ON directive and the presence of LISTING directives in the listing.

8. The directive:

   >> PAGE [*comment-text-1*]

   can be used to force a new listing page.

   comment-text-1, if present, is printed with the PAGE directive on a new listing page if the listing state is on.

   Note The PAGE directive in fixed format is redundant with the "/" comment indicator. Use of the PAGE directive may ease conversion to future source formats that might not have an indicator column.

9. The compiler's default behavior for replaced lines has been changed to suppress listing replaced lines when no C Compile Command Option or corresponding LISTING-ATTRIBUTES configuration keyword values are specified. This behavior is as if C=2 had been specified in the Compile Command. If this new behavior is undesired, the C Option can be specified or the configuration:

   COMPILER-OPTIONS LISTING-ATTRIBUTES=KEEP-REPLACED-LINES

   can be specified. Specification of any C Option, other than C=0 or ~C, eliminates the default of C=2 (for example, C=4 does not default to C=6).

10. The value specified in the C Compile Command Option can now be 0 through 15. When the binary value includes the 4 bit (0100), then replacement lines are suppressed in the listing. When the binary value includes the 8 bit (1000), then COPY statement lines are suppressed in the listing.

11. The REPLACE and COPY … REPLACING behavior has been changed to never modify a line as was previously done in RM/COBOL compilers. Instead, the original line is marked as replaced, and then a copy of the line is modified and marked as a replacement line. Since REPLACE and REPLACING were completely rewritten because of variable length records, several improvements were incorporated during the rewrite. Records are now enlarged, if necessary and possible, in order to do "in-place" replacements on the replacement record. (Note that records can more often be increased in size when there is no Identification area as described in item 2 in this list.) The compiler also produces fewer redundant replaced line copies when an "in-place" replacement is not possible.

12. The DATE-COMPILED paragraph treatment has been made consistent with the REPLACE statement. That is, the original DATE-COMPILED paragraph line or lines are marked as replaced and the new DATE-COMPILED paragraph line is marked as a replacement. The summary listing messages explaining the replaced and replacement line indicators were adjusted to include the case of DATE-COMPILED as follows:

```
"   <n>  Source was replaced because of REPLACE, REPLACING or DATE-COMPILED."

"    >n<  Source was inserted by REPLACE, REPLACING or DATE-COMPILED."
```

13. COPY statements are now marked in the listing with the source indicator "[]". (Previously, this indicator was used for modified lines, but those no longer exist as noted in item 11 in this list). There are now Compile Command C Option values and a COMPILER-OPTION LISTING-ATTRIBUTES configuration value to suppress COPY statement lines in the listing, if desired, as described in items 10 and 22c) of this list. The summary listing explanation of the listing source indicators "[]" was changed to reflect their new use:

    ```
    "   [n]  Source is a COPY statement, which is logically replaced
    by copied library text."
    ```

    The compiler now uses a routine common to REPLACE and REPLACING logic to logically replace a COPY statement.

Here is an example of a COPY statement in the listing:

```
 8                 DATA DIVISION.                                              |
 9                 WORKING-STORAGE SECTION.                                    |
10                                                                            |
11                * Single record with leading and trailing text.             |
13          >0<     01 COPY01A-G01.                                           |
14          [0]                     COPY "COPY01A.CPY".                       |
15          +1+   * Begin COPY01A.CPY.                                        |
16          +1+      02 Copy01A-D01       PIC X.                              |
17          +1+      02 Copy01A-D02       PIC X.                              |
18          +1+   * End COPY01A.CPY.                                          |
19          {0}                                          02 COPY01A-D10 PIC X.|
20                                                                            |
21                * Single record with only leading text.                     |
23          >0<     01 COPY01B-G01.                                           |
24          [0]                     COPY "COPY01B.CPY".                       |
25          +1+   * Begin COPY01B.CPY.                                        |
26          +1+      02 Copy01B-D01       PIC X.                              |
27          +1+      02 Copy01B-D02       PIC X.                              |
28          +1+   * End COPY01B.CPY.                                          |
29                                                                            |
30                * Single record with only trailing text.                    |
32          [0]         COPY "COPY01C.CPY".                                   |
33          +1+   * Begin COPY01C.CPY.                                        |
34          +1+    01 Copy01C-G01.                                            |
35          +1+      02 Copy01C-D01       PIC X.                              |
36          +1+      02 Copy01C-D02       PIC X.                              |
37          +1+   * End COPY01C.CPY.                                          |
38          {0}                              02 COPY01C-D10 PIC X.            |
39                                                                            |
40                * Single record with no leading or trailing text.           |
41          [0]         COPY "COPY01D.CPY".                                   |
42          +1+   * Begin COPY01D.CPY.                                        |
43          +1+    01 Copy01D-G01.                                            |
44          +1+      02 Copy01D-D01       PIC X.                              |
45          +1+      02 Copy01D-D02       PIC X.                              |
46          +1+   * End COPY01D.CPY.                                          |
47                                                                            |
48                 PROCEDURE DIVISION.                                        |
```

14. Hexadecimal literal continuation has been changed to ignore trailing spaces. Previously, the compiler would give an error if a continued hexadecimal literal had trailing spaces on the continued record. Now, the trailing spaces are ignored on the continued record. The hexadecimal literal continues with the first hexadecimal digit following the opening quote on the continuation record. Hexadecimal literals specified as replacement text are preserved in the form given in a REPLACE statement or a REPLACING phrase of the COPY statement. Previously, the compiler normalized hexadecimal literals to the form X'<*hex-digit*>…' with each hexadecimal digit that was a lowercase letter converted to the corresponding uppercase letter. In addition, error message 258 was changed from

    “Hexadecimal literal has wrong character.”

    to

    “Hexadecimal literal contains character other than hexadecimal digit (0-9 or A-F).”

15. The compiler now diagnoses a continuation line that begins in area A with warning message 761:

    ```
    "Continuation line should begin in area B, but this line begins in
    area A."
    ```

    Note This warning can be suppressed with configuration if desired.

    Previously, the compiler ignored this violation of COBOL fixed reference format rules.

16. The compiler now recognizes and processes a REPLACE statement anywhere it occurs. Previously, the compiler recognized a REPLACE statement only when it followed a period space separator or was the first statement in the source file (there were some subtle errors in this earlier approach). The compiler diagnoses the failure of a REPLACE statement to follow a period space separator, other than as the first statement in a source file, with warning message 762:

    ```
    "Preceding period separator is required before this REPLACE
    statement."
    ```

17. The compiler now better diagnoses incorrect use of the words COPY or REPLACE within a COPY or REPLACE statement. Incorrect use of the word COPY in a COPY or REPLACE statement is diagnosed with warning message 767:

    ```
    "The reserved word COPY must not be used in the replacement text
    of a COPY or REPLACE statement."
    ```

    Incorrect use of the word REPLACE in a REPLACE statement is diagnosed with warning message 768:

    ```
    "The reserved word REPLACE must not be used in the replacement
    text of a REPLACE statement."
    ```

18. Compiler directives, that is, source records beginning with ">>" in the program-text area of a source record, which have an unrecognized directive, are diagnosed with the new error message 763:

    ```
    "Compiler directive required here, but this word is not recognized
    as a compiler directive."
    ```

    The currently recognized directives are IMP (explained in item 2), LISTING (explained in item 7), and PAGE (explained in item 8). This error is also generated when the IMP directive is not followed by a recognized implementer-defined directive. The only current implementer-defined directive is MARGIN-R (explained in item 2).

    Compiler directives that have a syntax error other than "unknown directive" are diagnosed with the new error message 764:

    ```
    "Compiler directive has a syntax error here."
    ```

19. The compiler now diagnoses the error of a replaced region that starts on a debug line where the replacement requires a continuation record with warning message 765:

    ```
    "Replaced text began on debug line, but replacement text requires
    continuation."
    ```

    Because of internal changes to how source is represented, the compiler still knows that the line is a debug line, even though a "D" cannot be placed in the indicator area of the line, thus minimizing additional errors that previously occurred during a non-debug compilation in this situation.

20. In the compilation summary listing, if the W Compile Command Option is specified or configured, the option value is now listed in addition to just the fact that the workspace size was specified.

21. In the compilation summary listing of options specified, the listing (L Compile Command Option), object (O Compile Command Option), and configuration file (G and H Compile Command Options) now include the pathname in addition to just the fact that the option was specified.

22. The LISTING-ATTRIBUTES configuration keyword can now specify several new values, including the following values:

    a) RENUMBER-SEQUENCE-AREA. This is a more consistent and clearer alternative to the RESEQUENCE-LINE-NUMBERS keyword that specifies a YES or NO value.

    b) SUPPRESS-COPIED-LINES. This is a more consistent and clearer alternative to the SUPPRESS-COPIED-FILES value, which remains for compatibility with existing configuration files.

    c) SUPPRESS-COPY-STATEMENT-LINES. This is a new configuration capability to suppress COPY statements in the listing. It corresponds to the C Compile Command Option with a value of 8 through 15, as described in item 10 of this list.

    d) SUPPRESS-REPLACEMENT-LINES. This is a new configuration capability to suppress replacement lines in the listing (lines inserted because of REPLACE, COPY…REPLACING, or DATE-COMPILED). It corresponds to the C Compile Command Option with a value of 4, 5, 6, 7, 12, 13, 14, or 15, as described in item 10 in this list.

    e) KEEP-REPLACED-LINES. This is a new configuration capability to override the new default of suppressing replaced lines as described in item 9 of this list.

23. The compiler configuration now includes:

    COMPILER-OPTIONS LISTING-LINE-LENGTH=*n*

    for specifying the maximum length of listing lines. The default value is 132.

    The minimum value is 80 and the maximum value is 65535. Values outside this range cause a configuration value error.

    When printing records to the listing or print file, the compiler limits the line length to the specified value, truncating the record if necessary.

24. The compiler detects if source records are truncated on input and produces summary warning message 766:

    ```
    "One or more source records were truncated during compilation:
    total truncations = n."
    ```

    This aids the user in determining the need to adjust the maximum source record size configuration, as described in item 1 in this list. When this warning is produced, the total warning count for the compilation is incremented by *n* instead of the usual 1 per warning message.

    Note The warning can be suppressed with configuration, in which case the warning count is not modified.

25. Two new context-sensitive words were added to the compiler to support directives: IMP and MARGIN-R. These new words are reserved only in the context of a directive and, thus, will not conflict with user-defined words in any existing source program.

    The floating indicator ">>" was also added to introduce a directive source record.

26. Because of changes to how source records are maintained in memory without unnecessary trailing blanks, and because of some fixes to source scanning low-level routines needed to handle variable length records, it is expected that the compiler is now somewhat faster on large programs.

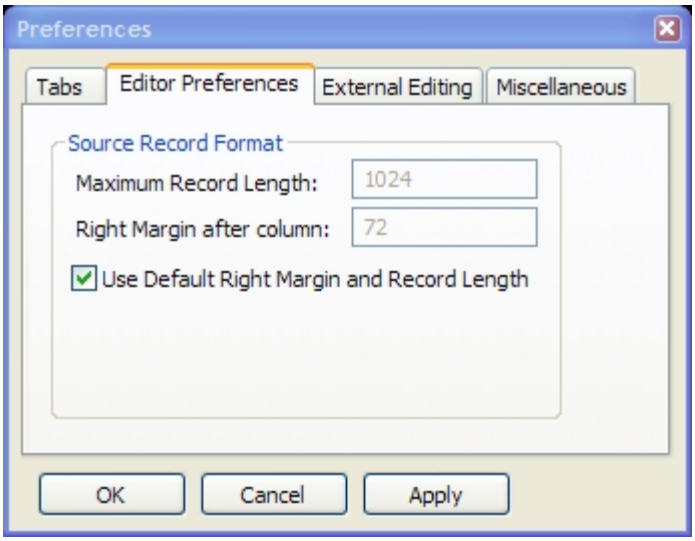
27. The compiler configuration now includes:

    COMPILER-OPTIONS LISTING-DIAGNOSTIC-PREFIX=*prefix-string*

    for specifying the prefix string that precedes diagnostic messages in the source listing. The default value is ">>>>>". (Previously, the default was "*****".)

    The string has a maximum length of 15 characters. Longer prefix strings are truncated to the first 15 characters.

28. CodeWatch now includes an Editor Preferences tab in the Preferences dialog box to set the maximum record length and right margin for purposes of syntax coloring and animation in the debugger.



If the **Use Default Right Margin and Record Length** check box is selected, then the default values of 1024 and 72 are set and cannot be changed. If the check box is not selected, then the **Maximum Record Length** can be entered, and a Right **Margin after column** value that is less than or equal to the maximum record length value can be entered. See also CodeWatch Debugger Enhancements.

START WHILE KEY LIKE

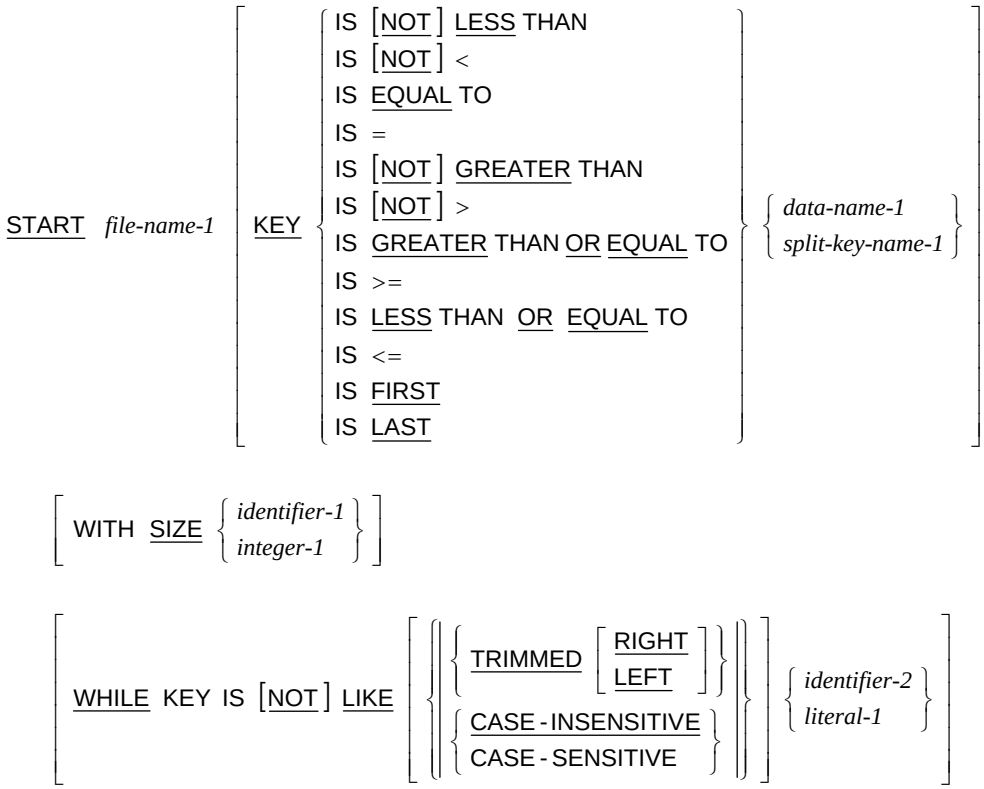
The START statement has been enhanced with a new phrase, which is introduced by the context-sensitive word WHILE. This phrase specifies a filter to be applied when sequentially reading records (READ NEXT or READ PREVIOUS) in an indexed organization file subsequent to successful execution of the START statement. The filter is expressed as a regular expression in the same form as used for the LIKE condition in relation conditions.

The implied subject of the WHILE KEY LIKE filter in a START statement is the key value of the key of reference in the record that would be accessed by the READ statement.

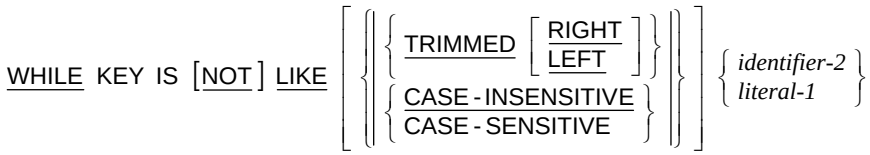
- Records with key of reference values that match the specified pattern regular expression are returned during subsequent sequential READ statements.
- Records with key of reference values that do not match the specified pattern regular expression are skipped during subsequent sequential READ statements.

If the word NOT is specified in the WHILE phrase, then filtering is reversed: records with key of reference values that match the pattern are not returned and records with key of reference values that do not match the pattern are returned in subsequent sequential READ statements.

The new format for the START statement is as follows:



The optional WHILE phrase in the START statement has the following format and rules.



literal-1 must be a nonnumeric literal.

identifier-2 must refer to an alphanumeric data item or a pointer data item.

The key value of the key of reference established by the START statement from the KEY phrase is the subject of the filter condition. Selection of the key of reference is not affected by specification of the WHILE phrase in a START statement. The key of reference may be a split key; in this case, the complete split key value is the subject of the filter condition.

The SIZE phrase does not affect the pattern matching done when the WHILE phrase is specified. Whether or not the SIZE phrase is specified, the entire key value, except as modified by the

TRIMMED phrase, is used when applying the pattern for filtering records during subsequent sequential READ statements. The SIZE phrase only affects the initial positioning established by the START statement.

Unless otherwise specified by use of the TRIMMED phrase, the entire contents of the key value must match the pattern value for a record to pass the filter.

- If the TRIMMED LEFT phrase is specified, leading spaces in the key value are ignored.
- If the TRIMMED RIGHT phrase is specified, trailing spaces in the key value are ignored.
- If the TRIMMED phrase is specified without either the LEFT or RIGHT modifiers, both leading and trailing spaces in the key value are ignored.

Note The TRIMMED phrase must not be used if the key value may contain significant spaces that would be ignored as a result of its specification.

Case is significant for the filter if the CASE-SENSITIVE phrase is specified or implied; that is, a case-sensitive match of the key value to the pattern value is done when filtering records. Case is not significant for the filter if the CASE-INSENSITIVE phrase is specified; that is, a case-insensitive match of the key value to the pattern value is done when filtering records.

The data item referenced by *identifier-2* or the value of *literal-2* is the pattern of the filter. If *literal-2* is specified, its value must be a syntactically correct regular expression for the desired pattern. If *identifier-2* is specified and refers to an alphanumeric data item, the value of that data item must be a syntactically correct regular expression for the desired pattern. If *identifier-2* refers to a pointer data item, then the value of that data item must point to a compiled pattern at the time the START statement is executed. A compiled pattern may be obtained by using the C$CompilePattern library routine. The syntax and semantics of a regular expression that may be used in a WHILE KEY LIKE pattern are the same as for the LIKE condition. This is fully described in the topic “LIKE Condition (Special Case of Relation Condition)” in the Procedure Division chapter of the *RM/COBOL Language Reference Manual, First Edition*.

Syntax errors in a literal pattern are detected during RM/COBOL compile time and the compilation error fully describes the syntax error. Syntax errors in an alphanumeric data item pattern cause the START statement to be unsuccessful and result in a 23,*nn* I/O status value when the START statement is executed. The value of *nn* indicates the type of error as follows:

1. Pattern class character range cannot include multi-character escape.
2. Pattern class character range cannot be hyphen '-' except at beginning or end of positive character group.
3. Pattern class character range cannot be opening bracket '['.
4. Pattern class character range cannot specify decreasing range.
5. Pattern character class subtraction cannot be followed by additional class specification.
6. Pattern escape sequence (initiated by '\') is not valid.
7. Pattern compilation requires more memory than is available.
8. Pattern quantifier opened with '{' is missing the closing brace '}'.
9. Pattern character class expression is missing the closing bracket ']'.
10. Pattern parenthesized subexpression is missing the closing parenthesis ')'.
11. Pattern category escape '\p{' or '\P{' is missing the closing brace '}'.
12. Pattern category escape '\p{' or '\P{' is missing the opening brace '{'.
13. Pattern category escape '\p{' or '\P{' contains an unknown category specification.
14. Pattern quantifier maximum count is less than the minimum count.

15. Pattern quantifier maximum count is missing; at least one decimal digit was expected.
16. Pattern quantifier maximum count is too large (> 65535).
17. Pattern quantifier minimum count is missing; at least one decimal digit was expected.
18. Pattern quantifier minimum count is too large (> 65535).
19. Pattern contains an unexpected closing brace '}'.
20. Pattern contains an unexpected closing bracket ']'.
21. Pattern contains an unexpected closing parenthesis ')'.
22. Pattern contains an unexpected quantifier '*' that is not preceded by a valid atom.
23. Pattern contains an unexpected quantifier '+' that is not preceded by a valid atom.
24. Pattern contains an unexpected quantifier '?' that is not preceded by a valid atom.
25. Pattern contains an unexpected quantifier '{' that is not preceded by a valid atom.
26. Pattern is too large or complex to compile.
27. A pointer to a compiled pattern either does not point to accessible memory or points to memory that does not contain a valid compiled pattern.
28. Compiled pattern contains an unrecognized pattern matching instruction code; the compiled pattern is not valid.

The values 1 through 26 for *nn* in an I/O status 23,*nn* correspond to compilation errors 682 through 707, respectively, of the RM/COBOL compiler. See the Compiler Messages appendix of the *RM/COBOL Language Reference Manual, First Edition* for additional information about each of these errors. Values 27 and 28 represent execution time errors with no corresponding compiler error message.

The WHILE phrase does not affect the order of records returned by subsequent sequential READ statements; the phrase only affects which records are returned. Records are returned in the order established by the collating sequence for the file. The collating sequence of the file does not affect pattern matching done while filtering records; the filter pattern is applied to the uncollated values of the key of reference. That is, the filter pattern is defined independent of the collating sequence for the file as if matching against the character values of the key in the data record. However, there are efficiency concerns because key values in the index are stored as collated values for quick comparison when searching the index.

A collating sequence for the file that maps multiple characters to the same collating value may impact performance of the filtering. Such a collating sequence may result in false positive matches when matching the pattern to the key value in the index. If this is the case, a filter is applied on the key value in the record before returning the record in order to eliminate the false positive matches. If the NOT LIKE form of the WHILE phrase is used, filtering is done only before records are returned; that is, no filtering occurs using the index, because false positive matches would be false non-matches. Filtering on the key value in the record is slightly less efficient than filtering on the key value in the index, but should still be more efficient than applying filtering logic in the COBOL program after a record is read. In summary, the best efficiency is obtained by following these guidelines:

- If possible, avoid using a collating sequence on the indexed file that collates multiple characters to the same collating position.
- If the file does have a collating sequence that maps multiple characters to the same collating position, use a pattern with classes that match on all the character values that map to the same collating position. This results in no false positives on matching and the implementation optimizes this case.

- If possible, avoid using the NOT LIKE form of the WHILE phrase on indexed files with a collating sequence that maps multiple characters to the same collating position and a pattern that can result in false positive matches.

A subsequent START statement or random READ statement cancels any existing filter established by the WHILE phrase in a START statement. The cancelling of the existing filter, if any, is done at the beginning of the START or random READ operation. Thus, a random READ is never filtered.

Reaching the end of file (READ NEXT) or beginning of file (READ PREVIOUS) does not cancel the filter; the filter continues to apply if the direction of reading is reversed and continued sequentially. Reading past the end of file, that is, a READ NEXT at end of file or a READ PREVIOUS at the beginning of the file, or any operation that fails in such a manner that the current record is not set will cancel the current filter, if any.

The WHILE phrase of the START statement requires runtime support that did not exist prior to RM/COBOL version 11. Therefore, use of this phrase causes object version 14 to be produced. RM/COBOL versions 11 and later support object version 14 at runtime. Prior versions of RM/COBOL runtime systems do not support object version 14. The Z Compile Command Option can specify values from 9 to 14 in RM/COBOL version 11. If the Z Compile Command Option specifies a value less than 14, then the WHILE phrase of the START statement cannot be used in the program being compiled.

The WHILE phrase is an RM/COBOL extension to the standard COBOL language. Therefore, if the F Compile Command Option specifies flagging of extensions, use of the WHILE phrase will be flagged in the program being compiled.

Since the WHILE phrase is only allowed for an indexed organization file, use of the WHILE phrase in a START statement for a relative organization file is diagnosed with error message 769:

```
"The WHILE phrase in START statement is not permitted for relative
file."
```

The LIKE condition, both in relation conditions and in the START statement, was incorrectly allowing the figurative constant NULL as the pattern specifier. This error is now diagnosed with error message 770:

```
"The figurative constant NULL is not allowed here."
```

The LIKE condition in relation conditions was ignoring the possible memory overflow when a literal subject was matched to a literal pattern at compile-time. At runtime, a memory overflow causes procedure error 258. The memory overflow in the compiler is now diagnosed with error message 771:

```
"Pattern match memory overflow occurred while evaluating conditional
expression."
```

The compiler previously diagnosed a *data-name-1* error in the KEY phrase of the READ and START statement with the same error message 356. The text of the message 356 indicates something that is not true of the data-name-1 in the KEY phrase of a READ statement. A data-name-1 error in the KEY phrase of a READ statement is now diagnosed with error message 772:

```
"Data-name-1 must refer to prime or alternate record key associated
with file-name-1."
```

The compiler now allows for *data-name-1* in the KEY phrase of a START statement to refer to the first segment of a split-key or to a data item aligned on the first character of the first segment of a split key as long as that uniquely identifies a record key of the file for establishing the key of

reference. If this new feature is used and the reference does not uniquely identify a record key of the file, that is, two or more record keys would qualify for the key of reference based on the data item referenced by *data-name-1*, the lack of uniqueness is diagnosed with error message 773:

```
"Data-name-1 does not uniquely specify a record key associated with
file-name-1."
```

The compiler now implicitly qualifies *data-name-1* in the KEY phrase of a READ or START statement for an indexed organization file with *file-name-1* when *data-name-1* does not include *file-name-1* as its last qualifier. As a result, certain error paths at compilation-time have been modified. For example, if *data-name-1* was not defined before this change, the compiler-produced error message 263:

```
"Identifier is not defined."
```

With this change, the compiler generates the following error messages:

- For the READ statement, message 772:

  ```
  "Data-name-1 must refer to prime or alternate record key
  associated with file-name-1."
  ```

- For the START statement, message 356:

  ```
  "Identifier must refer to a record key or to a data item aligned
  on a record key associated with file-name."
  ```

Configuration Enhancements

In addition to the configuration changes for the compiler long source record format feature added in version 11 of RM/COBOL described in this document, other configuration enhancements are included.

Configuration Records Keyword Changes

The following configuration record keywords have either been modified or added in this release.

ENABLE-LOGGING Keyword SUB-CALLS Value

The RUN-OPTION configuration record included the SUB-CALLS value for the ENABLE-LOGGING keyword in version 10. The SUB-CALLS value for the ENABLE-LOGGING keyword causes logging of subprogram calls to the **RMCALLS.LOG** file. This logging has been enhanced in version 11 to include the annotation "(RM_STOP)" at the end of the log entry line for a called non-COBOL subprogram that returns an RM_STOP function return value.

SKIP-INITIAL-CWD-SEARCH Keyword

The RUN-FILES-ATTR configuration record now includes the SKIP-INITIAL-CWD-SEARCH keyword. This keyword controls whether or not the current working directory (CWD) is searched for a filename when RM/COBOL is locating a file. The default value is NO, which specifies previous behavior of always looking in the CWD for filenames without a full directory path. The two other values that may be specified, UNQUALIFIED-NAME and NOT-FULL-PATHNAME, cause the initial CWD search to be skipped when a path search string (for example, RUNPATH) is also specified. This means that only a path search will occur. If desired, the user can place a dot (.) in the path search string (for example, RUNPATH) to look in the CWD at the desired place in the search sequence.

Specifying a value of UNQUALIFIED-NAME means that filenames not containing a forward slash (/) nor a backward slash (\) will skip the initial CWD search.

Specifying a value of NOT-FULL-PATHNAME means that both unqualified names as well as any pathname that does not begin with a forward slash nor a backward slash nor, on Windows, a drive letter (c:) will skip the initial CWD search. The NOT-FULL-PATHNAME value is more useful when EXPANDED-PATH-SEARCH=YES is also specified.

For more information on how RM/COBOL locates files, please see the *RM/COBOL User's Guide, First Edition*. The topics "Locating RM/COBOL Files on UNIX" in the "Installation and System Considerations for UNIX" chapter and the "Locating RM/COBOL Files on Windows" in the "Installation and System Considerations for Windows" chapter describe how the CWD is searched first for unqualified file names. This new configuration keyword modifies the search sequence described in those topics by skipping the initial CWD search that they specify.

SUBSCRIPT-CHECKING Keyword

The COMPILER-OPTIONS configuration record now includes the SUBSCRIPT-CHECKING keyword. This keyword specifies subscript checking at runtime. The default value is NO, which means that subscripts are checked only to the extent of insuring that data outside the program-accessible memory is not accessed or modified; failure of this check results in program termination with a data reference error 104 at runtime. A value of NO allows the possibility of accessing or modifying data that is not part of the data item referenced, but does not allow accessing or modifying data that belongs to another program in the run unit or any other run unit.

The value YES may be specified for this keyword to check that the composite subscript for a data reference does not exceed the maximum values possible for the data item referenced; failure of this check results in program termination with a data reference error 109 at runtime. A value of YES causes additional code to be generated for the subscript checking and requires suppression of some optimizations that could otherwise be done at compile-time, thus resulting in slightly larger programs that have lower performance at runtime.

Configuration Scope on Windows

In RM/COBOL version 11 for Windows, it is now possible to set configuration properties with the **RMCONFIG** and **INI2REG** utilities (see the discussions beginning on page 20), either for all users or for the current user (this user). In prior versions of RM/COBOL for Windows, the properties from these two utilities applied to all users. The introduction of these two levels of properties requires further explanation for various properties.

Control Properties

Control properties, other than the Command Line Options property, are used in the following order: Program Specific Properties for the Current User, Program Specific Properties for All Users, Default Properties for the Current User, and Default Properties for All Users. The first setting of a particular property from this ordered search is used and the search is terminated.

Note A call to C$GUICFG to set a control property will temporarily override this search. The value specified in C$GUICFG will be used instead until the next update of control properties from the registry.

For the Command Line Options property, the options are processed cumulatively in the following order: Default Properties for All Users, Default Properties for the Current User, Program Specific Property for All Users, and Program Specific Property for the Current User. Any options from the Command Line Options properties are processed before options on the actual command line are processed so that the actual command-line options can override any options specified from the Command Line Options properties.

Note 1 Some options for the runtime system may not be overridden by the actual command-line options because the options themselves are cumulative; that is, multiple options of this type may be specified on the command line. The L Option (for library loads) is an example of such a parameter. For additional information, see the Runtime Command description (on page 177) and the description of the L Option (on page 183) in the *RM/COBOL User's Guide, First Edition*.

Note 2 The environment variable RM_IGNORE_GLOBAL_RESOURCES may be defined if you wish the compiler, runtime system, or recovery utility not to access the Command-Line Options Properties defined for All Users. This may be useful if you are trying to develop at the same time others are running an application in live "production mode."

Synonym Properties

During initialization, the synonym *names* and their *values* are set into the environment in the following order: Default Properties for All Users, Default Properties for the Current User, Program Specific Properties for All Users, and Program Specific Properties for the Current User. When duplicate synonym *names* occur in this ordering, the last setting of a synonym *name* is the result setting in the environment.

Note The environment variable RM_IGNORE_GLOBAL_RESOURCES may be defined if you wish the compiler, runtime system, or recovery utility not to access the Synonym Properties defined for All Users. This may be useful if you are trying to develop at the same time others are running an application in live "production mode."

C$GetSyn obtains the specified synonym for the Current User, if the synonym is defined for the Current User. If the specified synonym is not defined for the Current User, then C$GetSyn gets the synonym for All Users. If the RM_IGNORE_GLOBAL_RESOURCES environment variable is defined, the All Users setting is ignored when the synonym is not defined for the Current User.

C$SetSyn always sets the synonym for the Current User, that is, does not attempt to change the synonym for All Users.

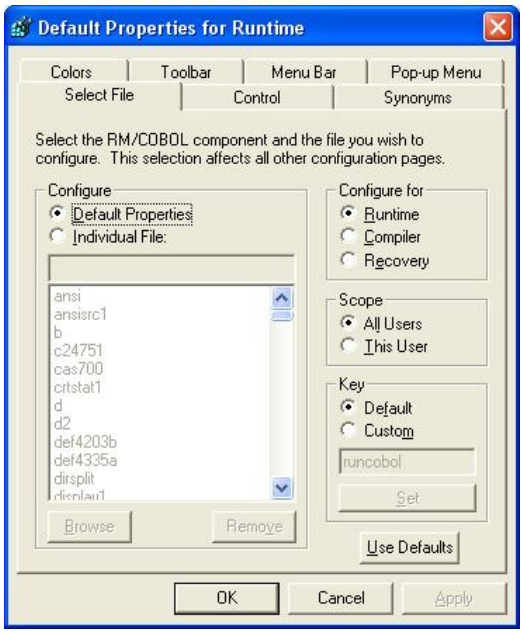
Note This is a change in RM/COBOL behavior on Windows. C$SetSyn previously always set the synonym for All Users. The old behavior would not be possible on Windows Vista without running as Administrator.

Colors, Toolbar, Menu Bar and Pop-up Menu Properties

Colors, Toolbar, Menu Bar and Pop-up Menu properties are used in the following order: Program Specific Properties for the Current User, Program Specific Properties for All Users, Default Properties for the Current User, and Default Properties for All Users. The first setting of a particular property from this ordered search is used and the search is terminated.

RMCONFIG Utility Scope Control

In RM/COBOL version 11 for Windows, the **RMCONFIG** utility now allows specifying the scope of its configuration settings to the current user or all users. In prior versions of RM/COBOL, the scope of RMCONFIG settings was all users by default and could not be changed. The RMCONFIG utility Select File tab now contains the Scope group of option buttons:



The All Users option button, when selected, specifies that options being modified through the RMCONFIG utility will apply to all users. To use this setting, you must have Administrator privileges and, on Windows Vista, the RMCONFIG utility must be running as Administrator. On Windows Vista, if the RMCONFIG utility is not running as Administrator, the All Users setting will appear to work, but will actually affect only the current user; in this case, the resulting settings will override, for the current user, any later changes for All Users made when running as Administrator.

The This User option button, when selected, specifies that options being modified through the RMCONFIG utility will affect only the current user.

INI2REG Utility Scope Control

In RM/COBOL version 11 for Windows, the **INI2REG** utility now allows specifying the scope of its configuration settings to the current user or all users. In prior versions of RM/COBOL, the scope of INI2REG settings was all users by default and could not be changed.

The INI2REG utility has a new command line option, -t, which indicates “this user” as opposed to “all users”. If the –t option is not specified, meaning configuration for All Users, you must have administrator privileges. Furthermore, on Windows Vista, INI2REG must be running as

Administrator to affect all users; otherwise, even without the –t option, only the current user will be affected; in this case, the resulting settings will override, for the current user, any later changes for All Users made when running as Administrator.

InfoExpress Client Enhancement on Windows

On Windows, the **rmixclnt.ini** configuration file can now be specified via the IXCONFIG environment variable. This is similar to existing UNIX behavior, but unlike UNIX, the value of IXCONFIG can be a full pathname to the file or it can be a folder name when the last character is a backslash (\). In the latter case, **rmixclnt.ini** will be appended to the folder name.

In addition, the search sequence for locating the "rmixclnt.ini" configuration file has been modified, for the Windows client only, to the following order:

1. IXCONFIG environment variable.
2. The execution directory.
3. The directory where rmtcp32.dll is located.
4. The main Windows directory.

PDFlib Support

PDFlib, a licensed product of GmbH, a German company, is a program that allows a COBOL program to create PDF files with CALL statements. Until a license is purchased from PDFlib GmbH and applied, PDFlib will print a "demo stamp" of **www.pdflib.com** across every page, and some functions or features may not be available.

RM/COBOL version 11 supports PDFlib versions 6.0.3 and 7.0.0 via support modules **rmpdflib6.dll** and **rmpdflib7.dll** on Windows and **rmpdfli6.so** and **rmpdflib7.so** on AIX 5.2, HP-UX 11, Sun SPARC, SUN i386, and Linux. RM/COBOL does not include the license for PDFlib, which must be purchased separately from PDFlib GmbH if use of this functionality is desired in an RM/COBOL program. Additional information on PDFlib and licenses may be obtained by visiting the PDFlib GmbH web site at www.pdflib.com.

For supported platforms, the required support module files are in the PDFlib subdirectory of the installation directory. That subdirectory also includes a readme file with additional information about PDFlib and some example programs and files.

CodeWatch Debugger Enhancements

The CodeWatch debugger has been enhanced to remember the workspace main window size and position from one session to another. As part of this enhancement, multiple monitor support has been implemented in CodeWatch. The main window can be restored to a secondary monitor if the main window was there when the debugger was last closed. If the secondary monitor is no longer available, the window will be restored to its current size and position on the primary monitor.

CodeWatch has a new Editor Preferences tab in the Preferences dialog. This new tab supports the RM/COBOL version 11 compiler changes that allow for longer source records. For further information on the Editor Preferences tab, see item 28 on page 12 of the "COBOL Source Format" topic.