IBM Migration Utility for z/OS and OS/390

User's Guide and Reference

Release 1

SC27-1685-04

IBM Migration Utility for z/OS and OS/390

User's Guide and Reference

Release 1

SC27-1685-04

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 251.

Fifth Edition (July 2003)

This edition applies to IBM Migration Utility for z/OS and OS/390, Release 1, Program Number 5655-I18 and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

The information in this manual was furnished by Foundation Software, Inc.

This publication is available on the Web at:

http:/www.ibm.com/software/awdtools/migration

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
H150/090
555 Bailey Avenue
San Jose, CA
95141-1003
U.S.A.

or fax your comments from within the U.S.A., to 800-426-7773, or, from outside the U.S.A., to 408-463-2629.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this manual

This manual describes how to use the IBM Migration Utility for z/OS and OS/390 licensed program, hereafter referred to as Migration Utility.

Who should use this manual

This manual is for anyone who currently has Easytrieve Plus programs, and wishes to convert them to COBOL programs.

To use Migration Utility properly, you will need need:

- Some knowledge of job control
- Some knowledge of Easytrieve Plus. (Migration Utility does all the hard work, so you may be able to get away with minimal knowledge of Easytrieve Plus).

Structure of this manual

Chapter 1, "Introducing Migration Utility", on page 1 explains what Migration Utility is, and how it works. It also proves information you only need once.

Chapter 2, "Compatibility check", on page 9 points out the features of Easytrieve that Migration Utility is able to handle, and those few features that it can't.

Chapter 3, "Defining entities", on page 25 tells you how to define entities such as file and working storage.

Chapter 4, "Program instruction reference", on page 35 describes in detail each Easytrieve instruction that Migration Utility supports.

Chapter 5, "SQL/DB2 support", on page 87 and Chapter 6, "SQL File I/O statement reference", on page 97 describe SQL matters.

Chapter 7, "User exits", on page 105 tells you how to use and write user exit. User exits

Chapter 8, "Installation and Migration Utility options", on page 115 describes the last few steps of installing Migration Utility, and tells you about the options you can set when you install or when you are running Migration Utility

Chapter 11, "Messages", on page 149 lists all the messages that Migration Utility provides through all the steps of creating COBOL programs from Easytrieve programs.

Syntax notation

Throughout this book, syntax descriptions use the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ►►— symbol indicates the beginning of a statement.

  The —► symbol indicates that the statement syntax is continued on the next line.

The ►── symbol indicates that a statement is continued from the previous line.

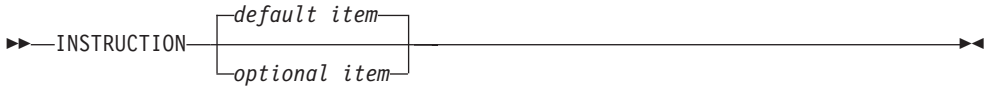
The ──►◄ indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the ►── symbol and end with the ──► symbol.

- **Keywords** appear in uppercase letters (for example, ASPACE) or upper and lower case (for example, PATHFile). They must be spelled exactly as shown. Lower case letters are optional (for example, you could enter the PATHFile keyword as PATHF, PATHFI, PATHFIL or PATHFILE).

  Variables appear in all lowercase letters in a special typeface (for example, *integer*). They represent user-supplied names or values.
- If punctuation marks, parentheses, or such symbols are shown, they must be entered as part of the syntax.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path. If the item is optional and is the default, the item appears above the main path.

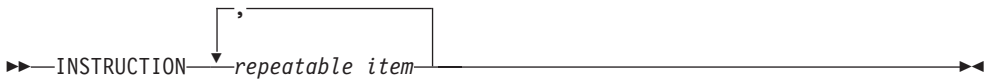


- When you can choose from two or more items, they appear vertically in a stack.

  If you **must** choose one of the items, one item of the stack appears on the main path.

  If choosing one of the items is optional, the whole stack appears below the main path.

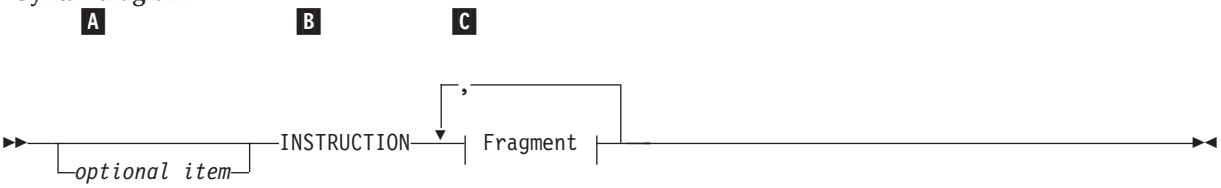
- An arrow returning to the left above the main line indicates an item that can be repeated. When the repeat arrow contains a separator character, such as a comma, you must separate items with the separator character.



  A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

The following example shows how the syntax is used.

Syntax diagram



Fragment:

Notes:

1 *operand choice2* and *operand choice3* must not be specified together

A The item is optional, and can be coded or not.

B The INSTRUCTION key word must be specified and coded as shown.

C The item referred to by "Fragment" is a required operand. Allowable choices for this operand are given in the fragment of the syntax diagram shown below "Fragment" at the bottom of the diagram. The operand can also be repeated. That is, more than one choice can be specified, with each choice separated by a comma. The note at the bottom of the syntax diagram indicates a restriction on the choice.

Summary of changes

Technical changes are marked in the text by a change bar in the left margin.

PTF UQ78189

- New features:
  - New EZPARAMS/EASYTRAN translator option:

    SPOOLOPT=NO/YES

    Optimizes temporary and sort work files record length by defining numeric display fields in packed-decimal format (COMP-3). Specifying SPOOLOPT=YES generates shorter record length and more efficient storage utilization.
  - Support for VSAM mixed I/O mode added to dynamic I/O modules.
  - Multiple reports handling to a single PRINTER file within the same JOB.
  - New DB2 Call Attachment module FSYDB250 uses external parameters for SSID and PLAN name.
  - DB2 host variables residing in a file using dynamic mode are off-loaded/loaded automatically into working-storage fields.
  - MOVE with MASK option.
  - Support for COBOL instructions using native Easytrieve syntax:
    - STRING

- INITIALIZE
- INSPECT

– Option for PAGE identifier (word) on report titles.
– Improved SQL BIND parameter punching for PACKAGE or PLAN BIND (depending on SQLBIND= option).
– PUT I/O to external tables.
– Ability to use POINT in START exit for synchronized files.
– I/O statistics for Sum, external tables, and SQL Files.
– Allow INSTREAM tables on JOB statement.
– Allow external tables on JOB statement.
– Allow external table PUT in BEFORE sort exit.

• Fixed problems:
– TALLY repaired when used in report exits to contain proper value.
– Problem with FETCH into host variables residing in records of dynamic I/O files.
– Access to RECORD-COUNT and RECORD-LENGTH adjusted to comply with Easytrieve rules when no file qualifier.
– Generate value for group items.
– Added MATCHED &file-1...&file-*n* syntax support.
– Field positioning fixed when +N coded before the first field on REPORT lines. **This change impacts field positioning. Users that report scraping in fixed positions must adjust the programs that do the scraping.**
– Parentheses put around the last term in an IF statement when comparison is on a list. For example, IF FIELDA EQ 10, 20, 30 translates to IF (FIELDA = 10 OR 20 OR 30)
– Allow quotes in literals.
– Allow table names to overlap file names when in dynamic I/O mode.
– Allow the use of the same SUM file in multiple JOBS.
– Change the way LINESIZE and PRINTER file size are used. **This change may impact field positioning and LRECL of print files. Users that report scraping in fixed positions must adjust the programs that do the scraping. Reports that are directed to a DASD file may have different LRECL.**

• Fixed syntax:
– Flag labels inside CASE statements.
– Handle ON/OFF field names.
– Allow &FIELD(&FILE) syntax for Sync and SORT files.
– Simulate Easytrieve macro parameters in Toolkit replacement macros.
– Accept SYSPRINT on DISPLAY statement.
– In GETJOB macro, corrected GETJOB-UERID to GETJOB-USERID.
– Other less significant syntax issues corrected.

PTF UQ75386

• New features:
– DB2 column definitions are retrieved from the SYSIBM.SYSCOLUMNS table if a DCLINCL for any given DB2 table is not coded in the Easytrieve Plus program. This option eliminates the need for SQL DCLGENs.
– A Call Attachment Facility (CAF) sample module for DB2 is distributed in the samples library along with the supporting load modules.

- Logic for finding DB2 Column References in Easytrieve Plus source changed to use Column Names as found in the DB2.SYSCOLUMNS or DCLGENs. This is compatible with Easytrieve Plus. Note that the previous versions of Migration Utility used COBOL field names as found in the DCLGENs, often resulting in undefined columns or names.
- Duplicate Reference Labels allowed at JOB Level.
- A new option, DOWHILE=PERFORM, handles Reference Labels inside DO/IF when labels are at the DO WHILE level.
- Partial support for floating-point fields added (COMP-1 and COMP-2 field types).
- SQLCA is included whenever PARM SQLID ('&owner') is present in Easytrieve Plus source.
- A separate FJSYSER file for displaying translator detected errors added to the translator job steps.
- Improved COBOL run time print I/O statistics.
- Support for COBOL copybooks with DYNAMIC I/O added.
- EZTABLE0 upgraded to handle more complex COBOL copybook parameters, including support for macros that contain W and S fields only.
- All COPY statements for macros found in EZTABLE0 generated with a prefix.
- VALUES (NO/YES) option added to PUNCH macro used by EASYCNV1.
- EZTPA00 program loader available for general use.

• New EZPARAMS/EASYTRAN translator options:

CAFPLAN=BATCH/&PLAN
Default Call Attachment Facility plan name

CAFOWNR=&USER/&OWNER
Default SYSIBM.SYSCOLUMNS table qualifier

COPYVERB=(COPY)
Option for COBOL COPY verb

COPYWRAP=('==')
Option for COBOL COPY replacing option wrap characters

DOWHILE=INLINE/PERFORM
Option for DO WHILE generating logic

• New Toolkit replacement macros:

ALPHACON Unstring an edited number into an internal numeric format

CONVAE Convert ASCII to EBCDIC

CONVEA Convert EBCDIC to ASCII

• New utility macros:

GETJOB Obtains JOB number and TSO user from the Job Scheduler Information Block.

• Converted COBOL runtime errors corrected:
  - Wrong LRECL on Dynamic I/O re-use.
  - I/O Buffers released after the file CLOSE call.

• COBOL runtime improvements corrected:
  - I/O buffers forced to be above the line (16MG) memory.
  - I/O statistics printed for all (except temporary) files.

• Syntax compatibility improvements:
  - Incomplete TITLE/LINE definitions accepted.

- BL1, BL3 and Packed unsigned fields can be used in the CALL and COBOL function statements.
- Old logic for handling one-byte indexed fields removed and logic added to handle out-of-range conditions automatically by generating a group field.
- Support added to support numeric groups with OCCURS.
- Support added to allow numeric and alphanumeric fields out of group range.
- -NN/+NN on report lines refined to allow negative overlap.
- Edit mask of all 9's repaired to use the actual mask length.
- OR/AND/XOR bit instructions allowed for any field type.
- In assignment formulae, the constant "0" is changed to WS-PENGI-ZERO to prevent COBOL Compiler errors.
- MASK insert characters logic improved to properly handle Z's, float and sign characters, following the decimal place.
- "DR" recognized as a special insert string in the mask definition.
- Invalid characters in Report Name are tolerated.
- Logic added to allow qualified DB2 column names in subscripts.
- Test for numeric on an alpha field forces the field to be generated as an elementary item to prevent COBOL compiler errors.
- Forced F sign on numeric fields in Report Exits inhibited.
- Alpha field value can be of any size.
- Inhibit Error 18 when a file name conflicts with a field name by assigning a new field name.
- Hexadecimal literal accepted on display and report lines.
- POS NN accepted back-to-back on report lines.
- COL NN accepted back-to-back on report lines.
- +NN and -NN accepted back-to-back on report lines.
- Use of LRECL=132 forced for REPORT files when record length is not specified.
- Field headings for subscripted fields compatible with Easytrieve Plus.
- Support for arithmetic expressions without parenthesis added.
- Selective hexadecimal values allowed for binary and packed unsigned fields.
- Long literal allowed on display and report lines.
- Enhanced layouts so that items belonging to a group are generated at end of the group, not at the end of the layout.
- Field title for RECORD-COUNT fixed.
- Working storage host VARCHAR variables generated as level-49 fields for proper DB2 use.
- Use of fields with OCCURS in subscripts flagged as an error.
- RECORD-COUNT, PAGE-COUNT, and LINE-COUNT can be used as subscripts.
- First I/O for a DISPLAY in the activity section to a PRINTER file no longer skips to channel 1.
- Parsing of Easytrieve comments enhanced to allow unpaired quotes or parenthesis.
- Edit mask of relative VSAM files key changed to 10 characters.
- Duplicate file key on synchronized job file is bypassed.

- COMP-3 and COMP fields used as host variables are generated with a sign picture.
- When punching COBOL copybooks, &field-9U is generated if FSIGN=ALL or FSIGN=YES is specified in EZPARAMS/EASYTRAN.
- Literal imbedded in SQL statements as host variable generated with a colon.
- Improved handling of VARYING fields. The mask corrected to reflect the length of the data area only.
- Other less significant syntax adjustments for better compatibility.

Summary of changes

Chapter 1. Introducing Migration Utility

Migration Utility is IBM's licensed version of the Foundation Software Program Engineering (FS/PEngi) family of COBOL development tools.

The components of the original set of tools are:

PEngiEZT
Easytrieve Translator.

PEngiCCL
Common Conditional (Macro) Language, sometimes referred to as CCL1.

PEngiBAT
Batch Programs Generator Subsystem for generating Batch COBOL Programs.

Throughout this manual there are references to PEngiCCL and PEngiBAT. However, when you use Migration Utility you are not able to use PEngiCCL or PEngiBAT in stand-alone mode. They are used internally by Migration Utility as a part of the over-all process.

Migration Utility converts Easytrieve programs to IBM mainframe COBOL or PEngiBAT. Its primary objective is to provide the ability to run Easytrieve programs in COBOL mode, eliminating Easytrieve inefficiencies. The benefits are:
- COBOL I/O handling is more efficient
- COBOL sorting and searching is more efficient
- COBOL better coexists with other languages and environments
- All COBOL debugging tools can be used for debugging
- More people are available for program support
- COBOL is portable to other platforms
- You can save money by eliminating Easytrieve costs

Migration Utility gives you a choice:

- You can continue developing programs using the Easytrieve format. The only thing that changes is JCL. That is, you use the Migration Utility translator and COBOL compiler in the place of Easytrieve. You can maintain the program source in Easytrieve format.
- You can convert the Easytrieve programs to PEngiBAT or COBOL. You can convert existing and newly developed programs. After converting, you maintain COBOL code or enjoy the power of PEngiBAT.

If you do not own Easytrieve you can use Migration Utility and enjoy the benefits without ever purchasing Easytrieve.

What is supported

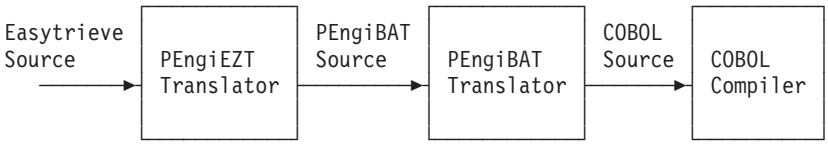
Migration Utility converts standard Easytrieve batch programs. It supports VSAM, QSAM, SAM, SQL/DB2, tape files and unit record devices. It also supports the Easytrieve Macro Language and COPY directive. In most instances there will be no changes required to your existing Easytrieve programs.

Translating concepts

Migration Utility translator reads in programs modeled (written) in Easytrieve Plus format and converts them to COBOLII/COBOL/S390 (COBOL). The COBOL programs are then compiled and linked as regular COBOL programs.

The translator is written in PEngiCCL macro language.

The translating process involves converting the Easytrieve source to PEngiBAT format. The generated PEngiBAT program is then converted to COBOL.



This process is transparent to the user. It is handled by the supplied procedures.

Translating guidelines

These guidelines are valid for translating existing or newly developed Easytrieve programs to COBOLII or COBOL/390.

Migration Utility JCL library (distributed with the product) contains standard procedures for running the translator. See "System information" on page 110 for PDS names. You need to run only one of the following procedures, depending on the level of completeness you want to obtain. The procedures are:

JCEZCOB1
Translates programs to PEngiBAT format and places them into a PDS.

JCEZCOB2
Translates programs to COBOL, and places them into a PDS. It does not compile.

JCEZCOB3
Translates programs to COBOL, compiles and links the load module.

JCEZCOB4
Translates programs to COBOL, compiles, links and executes (link and go).

JCEZC390
Translates programs to COBOL/390, compiles and links the load module.

JCEZDB2A
Translates programs to COBOL, translates SQL, compiles and links.

JCEZDB2B
Translates programs to COBOL, translates SQL, compiles, links and binds.

JCEZDB2R
Sample Run JCL for generated COBOL with DB2®.

JCBIND00
Sample BIND JCL for DB2.

JCEZE390
Translator JCL with external PROC for translate, compile and link.

JCEZL390
Translator JCL with external PROC for translate, compile, link and execute (link and go).

JCEZG390
Translator JCL with external PROC for translate, compile and run (link and go with program LOADER).

EZTCOB
External PROC used by JCEZE390 and JCEZL390 JCL.

EZTLKG
External PROC used by JCEZG390 JCL.

To install, follow these steps:

1. If your installation did not create standard procedures for running Migration Utility, copy the above procedures into a PDS and tailor them to run with your user ID. (Consult with System Administrator for JCL library.)

   DB2 users, refer to "Activating Call Attachment Facility (CAF) for DB2 users" on page 115.
2. The Easytrieve program source code must be placed into a PDS/PDSE or equivalent library that can be accessed as a PDS.

   Change `ISOURCE= symbolic` in the procedure to point to the PDS where the Easytrieve program is located. The program source is read from the SYSIN, in FSCCL1 step, if SYSIN DDname is provided. If SYSIN is not coded, the program source is read from the FJCPYLB DDname.

   There must be only one program per PDS member. Migration Utility does not translate multiple programs from a single PDS member.

   Note: When translating existing programs, verify if any tailoring is needed. See Chapter 2, "Compatibility check", on page 9 for more information.
3. When your program is read as a PDS member, you can leave JCL at the front of the program. You must remove any JCL at the end of the program (for example, `/*` or `//`). For instream SYSIN, you must remove the JCL and add `/*` and `//` to the bottom of the program source.

   Change `FJSYSJC= symbolic`, in the Proc, to an output data set name where program JCL will be created (JCEZCOB2 and JCEZCOB3 procedures only).
4. The Easytrieve Macros used by the program must be placed into a PDS or equivalent library that can be accessed as a PDS. One or more libraries can be concatenated in the JCL.

   Change `USERCPY= symbolic`, in the Proc, to point to the PDS where Easytrieve macros are located.

   If there is more than one macro library, concatenate additional libraries to the FJCPYLB DDname in the first (FSCCL1) step.
5. Member `EZPARAMS` in the Migration Utility library (`SYS1.SFSYEZTS`) contains Migration Utility default options. Make a copy of the `EZPARAMS` member and tailor it to your needs. It is essential to set the correct IOMODE= option in the EZPARAMS member as this parameter affects the amount of tailoring required to be made to Easytrieve Plus programs.

   Macro `EASYDTAB` in the Migration Utility library (`SYS1.SFSYCCLM`) contains the REPORT statement defaults. Make sure that EASYDTAB contains defaults compatible with your existing Easytrieve defaults, including edit masks for SYSDATE and SYSDATELONG. Refer to Chapter 8, "Installation and Migration Utility options", on page 115 for details.

Change `EZPARMS= symbolic`, in the Proc, to point to the PDS where EZPARAMS member resides.

6. Change Proc EXEC (located at the bottom of the Migration Utility Proc), to reflect the input program name, the output program name (if any), and the JCL option, for example:

   ```
   //STEP001 EXEC PROC=FSPENGI,IMEMBER=PROGXYZ,OMEMBER=PROGXYZ,JCL=YES
   ```

   The `JCL=YES` option punches a procedure for running the translated program. You can omit this option until the program translates clean. After a successful run, JCL can be found on the FJSYSJC file. This generated procedure contains JCL statements located in front of your program, and sample symbolic for any internally generated files. You can retrieve the sample procedure from the flat file into your PDS and massage it.

   Migration Utility tries to identify the file usage based on the top to bottom sequence of events in the program. The first OPEN determines the file type as an output or an input file. The assumption might be wrong for files that are opened more than one time in a single program.

   The DEBUG= switch located in the JCL can be used to generate a display statement of paragraph name in each generated COBOL paragraph.

   DEBUG=Y Generates active displays.
   DEBUG=I Generates inactive displays.
   DEBUG=N Does not generate any display statements.

   When DEBUG=I is specified, the statement `SOURCE COMPUTER.....WITH DEBUGGING OPTION` is generated with a "*" in C C 7. Subsequently, you can remove the `*` to activate the imbedded displays. When you specify DEBUG=Y, the statement is generated without a "*".

7. Submit the JOB. The Migration Utility translator prints the program and the diagnostics on the SYSLIST device.

   Depending on the procedure you are using, there can be up to six job steps involved:

   The first (FSCCL1) step ,common to all three procedures, is always the step that translates the Easytrieve program to the PEngiBAT format. Errors in the Easytrieve program are detected in this step. Errors and the input program source are printed on the SYSLIST device and FJSYSER file.

   The second (FSCCL2) step, common to JCEZCOB2 and JCEZCOB3, is always the step that translates the PEngiBAT program generated in the first step, to COBOL. Errors in this step indicate a flaw in the PEngiBAT translator. Some problems could probably be eliminated by rationalizing the origin of the problem back to the Easytrieve program, however. Errors and the generated PEngiBAT program source are printed on the SYSLIST device and FJSYSER file.

   The third (COB2) step , and the fourth (LKED) step, in JCEZCOB3, compile and link the generated COBOL program. Errors in COB2 step indicate a flaw in the PEngiBAT translator. These errors could be eliminated by rationalizing the origin of the problem back to the Easytrieve program. Some common errors that can be encountered are:

   - Field names that conflict with COBOL verbs
   - Undefined fields
   - Non-numeric fields used in arithmetic
   - Improper IF statement

Programs that contain SQL statements must be translated with JCEZDB2A or JCEZDB2B jobs. The SQL translator and BIND steps are standard DB2/SQL facilities. All messages should be handled as per DB2/SQL conventions.

8. When COB2 and LKED step run clean, test the program as per JCL as described in step number 6.

   Any file I/O errors that are detected by the program are printed on the FJSYABE and SYSOUT listings. The error report shows the file name that caused the error, status information and some suggestions as to the cause of the problem. Similar descriptions can be found in the COBOL Programmers Reference Manual.

Structure of Easytrieve programs

An Easytrieve Program has three sections. These are described below.

Environment Section
This section lets you alter Easytrieve Compiler options through the PARM statement. Except for DB2-related parameters, this section is ignored by Migration Utility (refer to "PARM statement parameters" on page 89).

Library Section
This section contains the FILE, RECORD and Work Field definitions. This section is fully supported by Migration Utility as described. All exceptions are clearly noted.

Activity Section
This section contains program procedures and statements that compose program processing logic and file I/O events. This section is fully supported by Migration Utility as described. All exceptions are clearly noted.

The Activity Section contains JOB and SORT subsections. There can be multiple JOB and SORT subsections within a single program. Each JOB or SORT subsection may contain one or more REPORT definitions.

Order of statements in an Easytrieve program

```
                    ___________________________________

Environment Section      PARM . . .

                    ___________________________________

Library Section          FILE . . .
                            DEFINE . .
                         . . .
                    ___________________________________

                           JOB . .
Activity Section             (statements)
                             (job procedures)
                              REPORT . .
                                 (report procedures)
                           SORT . .
                           (sort procedures)
                            . . .
                    ___________________________________
```

This sample program illustrates the order.

```
FILE FILEIN1 DISK (80)                              |
   CUST-NAME      01  15 A HEADING ('NAME')         |
   CUST-ADDRESS1  16  15 A HEADING ('ADDRESS1')     | Library Section
   CUST-ADDRESS2  32  15 A HEADING ('ADDRESS2')     |
   CUST-ADDRESS3  48  15 A HEADING ('ADDRESS3')     |

JOB INPUT FILEIN1
   PRINT REPORT1                                    |
                                                    |
REPORT REPORT1 LINESIZE 080                         |
TITLE 01 'NAME-ADDRESS REPORT EXAMPLE'              | Activity Section
LINE  01 CUST-NAME       +                          |
         CUST-ADDRESS1   +                          |
         CUST-ADDRESS2   +                          |
         CUST-ADDRESS3                              |
```

The program produces the following report (Xs represent real data):

```
05/30/95                  NAME-ADDRESS REPORT EXAMPLE                     PAGE    1


               NAME           ADDRESS1        ADDRESS2        ADDRESS3

           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
           XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
```

Of course, most real programs are a lot more complex.

Review of the Easytrieve punctuation rules

Easytrieve statements can be placed anywhere between columns 1 and 72. Each statement is separated by one or more spaces, or a comma followed by at least one space.

Each Easytrieve statement is followed by its relevant arguments. The arguments can be placed on the same line or on subsequent lines, but, each continued line must be terminated by a ″+″ or ″-″ symbol.

When a line is continued with a ″-″ symbol, the continuation is assumed to start at the beginning of the next line (Usually used for continuing literal).

When a line is continued with a ″+″ symbol, the continuation is assumed to start at the beginning of the text on the next line (first non-space).

The statement can be terminated with a ″.″ or by omitting the continuation symbol.

Examples

Here are some Easytrieve statements:

```
REPORT REPORT1       +                   |   single
      LINESIZE 80    +                   |   statement
      SUMMARY        +                   |   with + for continuation
      DTLCOPY
```

```
SEQUENCE  ACCOUNT DATE
CONTROL   ACCOUNT                        |   statements
LINE 01   ACCOUNT DATE LAST-NAME         |      with
                                          no continuation
```

Review of the Easytrieve punctuation rules

Chapter 2. Compatibility check

There are a number of items that you need to check to ensure a smooth translation. These are listed below.

File organization support

Migration Utility does not support DL1 and IDMS files. Programs accessing such files can be translated by tailoring your Easytrieve source to call a subprogram for I/O services, or the %COBOL statement can be used to add COBOL code to handle such situations.

DB2/SQL column definitions can be automatically accessed from the SYSIBM.SYSCOLUMNS catalog. Refer to "Activating Call Attachment Facility (CAF) for DB2 users" on page 115. If CAF is not available, then an `SQL DCLINCL &NAME` must be added to the programs that use SQL/DB2 tables. One statement is required for each SQL/DB2 table in use. These statements must be placed before the SQL file definitions (preferably before the first valid Easytrieve Definition).

SBCS and DBCS character support

Migration Utility supports single byte character set (SBCS) and K (DBCS) type of fields.

- DBCS Page number and the DBCS Date on REPORT statement are not supported.
- Conversion of DBCS to SBCS and SBCS to DBCS is not supported.
- Easytrieve K fields are converted to COBOL as G type of fields.
- Migration Utility automatically adds the shift-in (x'0e') and shift-out (x'0f') characters to K fields found on the DISPLAY and REPORT lines.
- When a constant (literal) is being assigned to a K field, Migration Utility automatically adds the shift-in (x'0e') and shift-out (x'0f') characters to the literal (constant).

Fixed-length records

When running in dynamic mode (IOMODE=DYNAM), record length is not a concern. However, when running in static mode (IOMODE=NODYNAM), you may need to make adjustments as described in this section.

Make sure that the record size of each file is fully defined. This can be done by coding the record length in the file statement, or by defining the record layout in full. This is because Easytrieve retrieves record size dynamically during the run time. COBOL does not. It is essential to have the correct file record length in the converted COBOL program.

NON-VSAM variable-length records

When running in dynamic mode (IOMODE=DYNAM), record length is not a concern. However, when running in static mode (IOMODE=NODYNAM), you may need to make adjustments as described in this section.

NON-VSAM variable-length record files can be:
- Variable
- Unblocked
- Variable blocked
- Spanned organization

Make sure that the record size of variable-length files includes 4 extra bytes for standard length as per IBM standards. This is a standard Easytrieve rule too, thus, you need to worry about only those files that do not have record size included in the File definition.

Record length can be coded in the FILE statement.

Migration Utility computes the usable record area by subtracting 4 from the declared length.

Easytrieve retrieves record size dynamically during the run time. COBOL does not. It is essential to have the correct file record length in the converted COBOL program.

VSAM variable-length records

When running in dynamic mode (IOMODE=DYNAM), record length is not a concern. However, when running in static mode (IOMODE=NODYNAM), you may need to make adjustments as described in this section.

A VSAM file is considered a variable-length file when the minimum record length, specified in the VSAM catalog, is not equal to the maximum record length. If the minimum and maximum lengths are equal, then the file is of fixed-length format.

For VSAM files, Easytrieve obtains record and file characteristics from the VSAM Catalog, and it does not allow record size in the FILE definition.

COBOL does not dynamically allocate VSAM file characteristics. Therefore, an option was added to Migration Utility to allow a record size on the FILE statement. If the size is not specified on the file statement, Migration Utility defaults to the size of the defined record.

If the minimum record size is equal to the maximum record size in the VSAM catalog, the specified size must be equal to the maximum value specified in the VSAM catalog.

If the minimum record size is not equal to the maximum record size in the VSAM catalog, the specified size must be equal to the maximum value minus 4.

All output and UPDATE VSAM variable-length files must be specified with `FILE . . . V (NNN)` where `NNN` is the LRECL (Maximum LRECL in the catalog - 4). variable-length Read Only VSAM files do not need to be coded with a V. Refer to "Defining VSAM files" on page 26 for full syntax.

VSAM key usage

When running in dynamic mode (IOMODE=DYNAM), the VSAM key is dynamically allocated at run time. However, when running in static mode (IOMODE=NODYNAM), you may need to make adjustments as described in this section.

COBOL requires that an alphanumeric VSAM file key for KSDS files is named in the FD statement.

Easytrieve retrieves the key characteristics from the catalog.

To overcome the problem, the Migration Utility convention is to use one of:
1. The first defined field in the record as the file key. This is the default.
2. The key field named on the file statement. The key must be defined in the file record.

   Example: `FILE FILEIN VS (KEY CUST-ACCOUNT)`

In the first case, the key must be defined as an alphanumeric key for the full key length. This method is also Easytrieve Plus compatible.

For relative (RRN) files, Migration Utility assigns an internal key in working storage. The key of RRN files can also be named on the file statement as shown in the second case. The named field must be previously defined as a four byte binary field, however.

VIRTUAL files

Easytrieve VIRTUAL files are handled as regular sequential files in the translated COBOL. No special handling is needed.

Extended printer support

Migration Utility does not support extended printing of Easytrieve. One way around it is to change the Easytrieve program to the standard printing.

Index usage

Easytrieve allows index usage for fields defined without the OCCURS. COBOL does not. Therefore, all such statements are flagged by Migration Utility.

Easytrieve allows the same index name to be used for more than one field. COBOL does not. To resolve the problem, Migration Utility assigns a unique internal index name to each indexed field.

These internal indexes are updated every time an index assigned by the Easytrieve is changed. Therefore, there could be a substantial overhead maintaining an index that is used for more than one field.

Resolve the problem by changing the program to use a unique index for each field.

Example:

Index usage

```
FILE FILEIN
NAME   27     40  A
SCAN1  27     10  A  INDEX (SUB1) OCCURS 4
SCAN2  27      3  A  INDEX (SUB1) OCCURS 13
SCAN3  27      9  A  INDEX (SUB1) OCCURS 5
```

In this example, three internal indexes are updated every time ″SUB1″ index is changed in the program, even if it accesses only one field. To correct the problem, assign a unique index to each of the fields:

```
FILE FILEIN
NAME   27     40  A
SCAN1  27     10  A  INDEX (SUB1) OCCURS 4
SCAN2  27      3  A  INDEX (SUB2) OCCURS 13
SCAN3  27      9  A  INDEX (SUB3) OCCURS 5
```

Restrictions

Packed Unsigned (U) fields, one (1) byte binary fields and three (3) byte binary fields are flagged as errors by the translator when used as index fields. This is because such fields must be set-up into a valid numeric field understood by COBOL before they can be used, adding substantial CPU overhead. To resolve the problem, move such fields into a 4 byte binary field and use the new field for indexing.

Note: SSOMDE=GEN of EZPARAMS/EASYTRAN allows the use of PU, BL1, and BL3 in subscripts.

Field naming conventions

Easytrieve allows up to 40 character field name length. Migration Utility reduces all field names that are longer than 16 characters to 16 characters, automatically. The field names are reduced by taking the first three characters of the words separated by a dash (-), until the name goes below 17 characters in length. Note that the INDEX names are not reduced. Long INDEX names should be manually reduced to the acceptable size.

The process above might yield undesired or ambiguous field names. To avoid the problem, a translate table can be provided in Migration Utility EZPARAMS options to translate specific words to a desired acronym, or ambiguous field names to acceptable names.

Use `NAMETAB` parameter of EZPARAMS to change any special characters found in Easytrieve field names to make field names COBOL compliant.

Use `COBVERBS=YES` option of EZPARAMS to alter names of Easytrieve fields that conflict with COBOL Reserved Words.

Migration Utility Options are described in Chapter 8, “Installation and Migration Utility options”, on page 115.

Ambiguous field position; fields with Index and OCCURS

The biggest challenge writing Migration Utility was to translate the Easytrieve defined record and working storage layouts to COBOL. The problem is that Easytrieve allows fields to be defined out of sequence. As a result, many layouts in Easytrieve programs are badly fragmented and out of order.

To overcome the problem, Migration Utility inserts fields or group of fields sequentially by group reference and position within the group they belong to.

If you do get errors during Migration Utility translation, re-arrange field definitions such that they are in the correct sequence and do not destructively overlap.

- Programs with orderly record definitions generate fewer and simpler COBOL layouts.
- Fields are grouped together and any fields out of group range are flagged by PEngiEZT.
- All numeric fields that destructively overlap another field within a group item are flagged.
- All alpha fields that destructively overlap another field within a group are generated with a REDEFINE.
- Fields coded with INDEX &INDEX usage without OCCURS, are automatically generated with OCCURS 1 TIME.

  Example

  ```
  WORKF1 W 10 A INDEX AINDEX
  ```

  is converted to

  ```
     02 FILLER OCCURS 1 TIMES INDEXED BY AINDEX-001.
        03 WORKF1                PIC X(10) VALUE SPACES.
  ```

- Alpha fields of length 1 defined with OCCURS NN INDEX &INDEX that have subordinate fields are generated with a group size of NN. OCCURS is altered to OCCURS 1 INDEX &INDEX.

  Example

  ```
  WFIELD1      W     1 A OCCURS 50 INDEX INDEXY
  WFIELD2 WFIELD1    1 A
  WFIELD3 WFIELD1 +1 9 A
  ```

  is converted to

  ```
     02 FILLER OCCURS 1 TIMES INDEXED BY INDEXY-002.
        03 WFIELD1.
           04 WFIELD2              PIC X(1).
           04 WFIELD3              PIC X(9).
           04 FILLER               PIC X(40) VALUE SPACES.
  ```

  In this example, `WFIELD1` is generated with the length of 50, not with the length of 1 as initially defined by the Easytrieve statement. The assumption is that `WFIELD1` was coded to serve as a reference for indexing, rather than for the use in data manipulation. Thus, any access from or to `WFIELD1` result in an erroneous outcome. To correct the problem, `WFIELD2` can be used in the place of `WFIELD1` in the Activity Section.

- Fields coded with OCCURS and without INDEX &INDEX usage that have subordinate fields out of group range are flagged as errors.

  Example

  ```
  FIELDA        1    5 A OCCURS 6
  FIELDB   FIELDA    6 A
  ```

  In this example, `FIELDB` is 6 bytes long and `FIELDA` is 5 bytes, thus `FIELDB` cannot exist in `FIELDA`. Assuming that `FIELDB` is a standalone field, the definition can be written as

  ```
  FIELDA        1    5 A OCCURS 6
  FIELDB        1    6 A
  ```

Ambiguous field position; fields with Index and OCCURS

- Fields coded with OCCURS and with INDEX &INDEX usage that have subordinate fields out of group range are flagged as errors.

  Example

```
WFIELDA        W   5  A OCCURS 10 INDEX INDEXY
WFIELDB  WFIELDA   5  A
WFIELDC  WFIELDA  +5  9 A
```

  Assuming that all fields are to be accessed via the INDEX INDEXY, the definition can be written as

```
WFIELDG        W   1  A OCCURS 50 INDEX INDEXY
WFIELDA  WFIELDG   5  A
WFIELDB  WFIELDG   5  A
WFIELDC  WFIELDG  +5  9 A
```

Binary field handling

Two byte and four byte binary fields are passed on to COBOL in Native mode.

The maximum value that can be accommodated by such fields in COBOL is different from the maximum value accommodated by Easytrieve. (For limits, see binary field description in “Defining Records and Working Storage” on page 32.) COBOL Compiler option TRUNC(BIN) is recommended.

One and three byte binary fields are not supported by COBOL. Migration Utility expands special logic for handling such fields.

(For limits, see binary field description in “Defining Records and Working Storage” on page 32.)

Assigning hex values

Migration Utility automatically resolves hex values usage whenever possible. However, there are situations when an automatic solution cannot be implemented.

Illegal hex values are flagged by Migration Utility. Resolve the problem by changing the hex value to a decimal equivalent, or converting the field to an alphanumeric field.

Common hex usage involves assigning low-value or high-value to the fields, such as binary zeros or all X″FF″. For such cases, replace the hex value by the ″LOW-VALUE″ or ″HIGH-VALUE″ statements respectively.

Field headings

The maximum string length of a heading in Migration Utility is 58 characters. Easytrieve allows field headings longer than 58 characters. Reduce headings longer than 58 characters to 58 characters or less.

Paragraph-naming conventions

Easytrieve allows paragraph and procedure names to be over 30 characters. COBOL does not. Migration Utility alters paragraph names to conform to COBOL rules.

COBOLII/S390 and COBOL-VS compatibility

The Easytrieve programs are translated to COBOLII/COBOL/390 compatible. Called subprograms that are written in other COBOL dialects such as COBOL-VS, and contain VSAM file I/O routines, may experience a problem. This is strictly a COBOL compatibility problem. Such subroutines must be compiled with COBOLII or COBOL/390 compiler to make them compatible.

Calling subprograms

Migration Utility converts calls to subprograms as follows:

1. It generates a static call for program names that are enclosed in quotes.
2. It generates a dynamic call for program names that are not enclosed in quotes

Easytrieve does not accept quotes around the program name. By default, all called programs would be interpreted as dynamic calls by Migration Utility. To force a static call, enclose the program name in quotes.

Undetected errors

Easytrieve is a very forgiving language. It often ignores extraneous statements or it does not impose strict rules. This is especially visible with REPORT related statements like NOPRINT, NEWPAGE, and RENUM.

Migration Utility may flag such extraneous statements. If it does, remove them or resort to a simpler form of expression.

Sign of numeric fields

Numeric fields defined with decimal places (even if zero) are generated with a signed COBOL picture. Numeric fields defined with no decimal places are generated with an unsigned COBOL picture.

For example,

```
FIELD-A    W        5   N 2
```

is generated as:

```
02 FIELD-A      PIC S9(03)V99 VALUE ZEROS.
```

and

```
FIELD-A    W        5   N
```

is generated as:

```
02 FIELD-A      PIC  9(05) VALUE ZEROS.
```

Some COBOL instructions force a positive sign in unsigned numeric fields, even if the source field is negative. This can lead to runtime differences due to logic taking on a different path when unsigned fields are tested for positive or negative values. Make sure that arithmetic comparisons for negative values operate on signed fields.

Easytrieve forces an “F” sign in positive zoned decimal numeric fields. This is true for signed (fields defined with decimal places, such as quantity), and for unsigned fields (fields defined without decimal places).

Sign of numeric fields

COBOL forces a "C" sign in positive zoned decimal numeric fields that are defined with a sign. Refer to the FSIGN= option in "Migration Utility translator options" on page 119 for overriding options.

For example, if O-BALANCE field is defined as per below and it contains value 22222 then:

```
             Definition                  Hex Value
Easytrieve   O-BALANCE 1 5 N 0           F2F2F2F2F2
COBOL        O-BALANCE   PIC S9(05)      F2F2F2F2C2
```

The last byte is different, F2 instead of C2, This is numerically equal, but it is not equal if compared as an alphanumeric value. Altering the Easytrieve definition to a numeric unsigned field yields the same in COBOL:

```
             Definition                  Hex Value
Easytrieve   O-BALANCE 1 5 N             F2F2F2F2F2
COBOL        O-BALANCE   PIC 9(05)       F2F2F2F2F2
```

Make sure that your Easytrieve fields are properly defined. All quantitative fields should be defined with decimal places (even if zero) and all non-quantitative values should be defined without decimal places.

In general, the intermediate calculations are not a problem. The problem is visible on the output display numeric fields that are defined as quantity (with a sign) but they are truly not a quantity, such as account numbers, serial numbers, and item numbers.

Varying-length fields

In Easytrieve, fields defined as "varying" fields are composed of two byte binary length followed by the text area. The length is maintained by Easytrieve as the content of the field changes.

Migration Utility generates COBOL code that artificially maintains such fields, that is, the field is defined as a group item with two byte binary length followed by the text. The length value is automatically changed as the content of the field changes. Note that this is exactly what the Easytrieve does.

The difference exist in the maximum value that can be represented by the length. In COBOL, a two byte binary field can accommodate up to 9,999 in value (unless TRUNC(BIN) is specified), while Easytrieve can accommodate up to 32,767.

The difference also exists in the compare instructions. While the generated COBOL uses the length of the first argument and the second argument for comparison, Easytrieve compares the values for the length of the first argument only. The problem exists only if the fields being compared are not of the same length. The remedy to this is to make sure that the fields being compared are of the same length.

Also note that when a value is moved into the field length, the Easytrieve moves the value as is, while COBOL converts it to the binary equivalent. For example, moving 32 into the field length results in X'F3F2' when performed by Easytrieve, and x'0020' when performed by COBOL.

Uninitialized Working Storage fields

Be careful with Working Storage fields. Uninitialized fields may contain a different initial value in the translated program as it is not possible to predict what is in Working Storage at linking time. Using an uninitialized field without placing a value in it may result in erroneous outcome.

Migration Utility generates the initial values for Working Storage fields automatically, providing that the field is not an object of redefine. If there may be uncertainty, move a value into questionable fields in the START procedure of the first JOB.

When a VALUE is specified, and the field redefines another field, Migration Utility generates a MOVE of specified value, into the target field, at the beginning of the program. Value of an indexed field, or fields with occurs is moved into the first slot.

For initializing file records, refer to MEMINIT= parameter of EZPARAMS.

The MOVE statement

In Easytrieve, the MOVE statement moves data from left to right as if both areas were alphanumeric. The data moved is not converted, instead, it is moved as is, even if the from or to fields are packed or binary fields.

Migration Utility generates a standard COBOL MOVE from Easytrieve MOVE. The data is moved according to the standard COBOL conversion rules. So a move from a binary field into a display numeric field results in data conversion from binary to Display numeric format, yielding a result that is different to the Easytrieve Move. You can achieve compatible results by redefining the numeric field as an alpha field and using the alpha field name as the source or target in the move statement.

Migration Utility issues a Warning (MNOTE) message for questionable moves. The messages should be reviewed and problems should be corrected if deemed as problematic.

FILE-STATUS (STATUS) codes

When the IOCODE=EASYT option is used, Migration Utility generates logic that converts the COBOL status code to the Easytrieve Plus equivalent. No tailoring is needed.

> When IOCODE=NATIVE option is used, status code references must be tailored as described below.

Easytrieve I/O status codes are different from those in COBOL. In general, changes are not needed if your program is not checking for a specific non-zero value. If your program is testing for a specific non-zero value, you must adjust the value in your Easytrieve source to comply with the COBOL status codes. For more information, refer to “System-defined fields” on page 66.

- FILE-STATUS code in the generated COBOL program is a two byte alphanumeric field while in Easytrieve it is a full word numeric field. Instructions in an Easytrieve program that assign FILE-STATUS to a numeric field are flagged as errors.

  Example:

```
RETURN-CODE = FILEIIN:FILE-STATUS
```

This is flagged as an error. The statement can be written as

```
WRETURN-CODE W 2 N

MOVE FILEIN:FILE-STATUS TO WRETURN-CODE
RETURN-CODE = WRETURN-CODE
```

- When testing for a value other than zero, Migration Utility expects the value to be a two digit constant (literal) enclosed in quotes.

Labels inside a DO and IF pair of statements

Easytrieve allows labels (paragraph names) between a pair of DO and IF statements. Programmers use labels to loop within the DO IF, or to jump around the code. COBOL cannot handle such syntax.

Migration Utility automatically generates a separate COBOL paragraph if the label is coded inside a DO IF pair before the first `GO TO &LABEL` statement that refers to it. The generated paragraph is PERFORMED from within the original DO IF pair.

Migration Utility flags all labels that are coded after the first `GO TO &LABEL`, and the label is inside a DO IF pair. Such conditions must be massaged by the programmer.

Consider using the DOWHILE=PERFORM option. It will resolve the DO level labels. However, the generated logic will be more fragmented.

Example

This example shows the initial Easytrieve code, and the code after conversion:

```
FIELDB = 'N'
IF FIELDA = 'Y'
  DO WHILE FIELDB EQ  'N'
     FIELDC = FIELDC + 1
     IF FIELDC GT 100
        FIELDB = 'Y'
        GOTO LABEL1
     ELSE
        FIELDB = 'N'
     END-IF
LABEL1
  END-DO
END-IF
```

Before translating, the routine should be converted to something like this:

```
FIELDB = 'N'
IF FIELDA = 'Y'
  DO WHILE FIELDB EQ  'N'
     PERFORM LABEL1-CODE
  END-DO
END-IF
 .
 .
* THIS PROC MUST BE INSERTED OUTSIDE OF THE CURRENT JOB MAIN BODY.
* END OF CURRENT JOB STATEMENT BUT BEFORE FIRST REPORT WOULD DO.

LABEL1-CODE. PROC.
  FIELDC = FIELDC + 1
  IF FIELDC GT 100
     FIELDB = 'Y'
```

```
    GOTO LABEL1
  ELSE
    FIELDB = 'N'
  END-IF
LABEL1.
END-PROC.
```

In general, Migration Utility flags improperly coded labels inside a DO IF pair. A good practice is to run the translator to get errors first and then fix them.

You can code a GOTO *&label* statement from any IF, DO, or CASE nest level, providing *&label* is at level 0 in the nest. When you are writing new programs, avoid coding labels other than at level 0 inside IF, DO, or CASE statements.

External table record length

When running in static mode (IOMODE=NODYNAM), it is essential to have the correct file record length in the converted COBOL program.

Refer to "Defining tables" on page 27 for syntax rules.

JCL for converted program

Migration Utility can generate sample JCL for running the program.

To generate JCL, existing JCL can be optionally included in front of the program and JCL=YES can be specified on the Proc EXEC statement. For more information, see "Translating guidelines" on page 2.

Migration Utility generates an Instream sample Proc by merging the existing JCL to any new JCL needed by the generated COBOL program. The unneeded statements are bypassed.

The following rules are observed:

- An Instream Proc is generated. A JOB statement and additional libraries must be added to the Proc.
- All JCL for input and output files are included from the included JCL, if present, else a symbolic is generated with a dummy file name for each file. For concatenated input files, symbolic is used only for the first file. Other files are concatenated in the JCL.
- Statements for Temporary and Sort Work files are also generated, however, the Proc must be changed to include proper allocation.

  Temporary and Sort Work files are not transferred from the included JCL.
- System related files such as SYSPRINT, SYSDUMP, SYSOUT, etc. are also generated.

  The Proc is created on the FJSYSJC file as defined in the translator JCL.

Overlapping fields on report lines

Easytrieve allows field overlaps on the report headers, print lines and field titles.

Migration Utility allows limited overlapping. A warning message is issued for each encountered overlap.

Overlapping fields on report lines

When field titles overlap, Migration Utility strips the leading and trailing spaces from all field titles to make things fit better. However if the overlapping still cannot be resolved after all spaces are stripped, Migration Utility reduces the size of the previous title and issues a warning message (generally in the PEngiBAT step). The adjustment might not result in titles identical to those printed by Easytrieve.

When fields or literals on print lines overlap, Migration Utility generates a layout with REDEFINEs for proper placement of each field, starting with the intended column.

If the fields or titles on your report do not match exactly to those printed by Easytrieve, make manual adjustment to Easytrieve program Source to avoid overlaps. You can reduce the field size or title or shift its location to the right, or if possible reduce the mask size.

Caution: Any reduced field mask can cause a loss of leading data digits. Use extreme care.

Group fields for SQL/DB2 usage

Easytrieve allows group fields to be used as host variables for SQL operations. However, SQL/DB2 translator enforces strict rules on field types.

To overcome the problem, Migration Utility generates elementary field definitions for all host variables, except for 01-level (the record level) host variables. Migration Utility issues a warning message when a group field is changed to an elementary item. The change has no impact on processing logic not related to SQL operations.

If the outcome of the automated change does not solve the problem, make manual changes as shown in the following example.

Example

This example shows a group item and how it should be adjusted to make it an elementary item.

```
WS-DATE    W              8 A
WS-MM      WS-DATE        2 N
WS-DD      WS-DATE    +2  2 N
WS-YYCC    WS-DATE    +4  4 N
```

can be coded as:

```
WS-DATE-X  W              8 A
WS-MM      WS-DATE-X      2 N
WS-DD      WS-DATE-X  +2  2 N
WS-YYCC    WS-DATE-X  +4  4 N
WS-DATE    WS-DATE-X      8 A
```

In this example, `WS-DATE-X` is generated as a group item and redefined by `WS-DATE`, thus making `WS-date` an elementary item.

OCCURS fields for SQL/DB2 usage

Easytrieve allows fields defined with OCCURS to be used as host variables for SQL operations. SQL/DB2 enforces strict rules on field types. The subscripts cannot be passed on to the SQL/DB2 preprocessor.

When possible, Migration Utility generates logic that offloads or loads the subscripted host variables into elementary fields before and after the “EXEC SQL” operation. Working storage is generated from the host variable field attributes.

When the subscripted host variables are located in the “WHERE...” statement, the offloading or loading logic is invoked before and after the “EXEC SQL OPEN &CURSOR” operation.

When the subscripted host variables are located in the “INTO...” or the “FROM...” statement, the offloading or loading logic is invoked before and after the “EXEC SQL FETCH/INSERT/...” operations.

If the SQL/DB2 translator issues errors due to falsely generated statements for subscripted host variables, you must modify the statements by re-coding such fields without OCCURS. You can code *n* number of fields, each ending with a sequence number representing the field slot for clarity.

Example

This example shows an OCCURS item and how it can be adjusted to make it SQL-friendly.

```
WS-ITEMS   W              15 A
WS-ITEM    WS-ITEMS        3 N OCCURS 5
```

can be coded as:

```
WS-ITEMS   W              15 A
WS-ITEM-01 WS-ITEMS        3 N
WS-ITEM-02 WS-ITEMS   +3   3 N
WS-ITEM-03 WS-ITEMS   +6   3 N
WS-ITEM-02 WS-ITEMS   +9   3 N
WS-ITEM-03 WS-ITEMS   +12  3 N
```

The SQL FETCH/SELECT INTO host variables reference must be changed to reference non-subscripted fields.

Packed unsigned fields

Migration Utility generates statements that artificially control access to or from packed unsigned (PU) fields for all operations except when used as a subscript.

The maximum allowed length of a PU field is 8 bytes due to COBOL restrictions on numeric fields.

Easytrieve allows PU fields to be used as subscripts. COBOL does not support PU fields.

Migration Utility flags PU fields when used as subscript. To use a PU field as a subscript, define a packed sign field or a binary integer in your Easytrieve source for subscript use. Add a move where appropriate from the PU field into the newly defined field.

The reason for not supporting subscript usage of PU fields is because it would add to the complexity of the generated code and performance overhead.

Mask of numeric fields

In Easytrieve, numeric fields are printed with decimal places when the mask contains decimal places, even if the field is coded without the decimal places. In COBOL, the decimal places are padded with zeros.

Example

```
FIELDA     W          7 N   MASK('ZZZZ9.99')  VALUE 12345
```

Easytrieve prints 123.45 and COBOL prints 12345.00 which is obviously shifted left by two digits.

A clue to this type of problem is values that differ in multiples of 10 between the Easytrieve and COBOL outcomes.

Solution for OCCURS 1 problem

In Easytrieve Plus, OCCURS 1 is a valid statement. Programmers normally use the OCCURS 1 technique to establish a reference for subscripting. COBOL does not support such syntax.

To avoid manual adjustments to every statement in an Easytrieve Plus program, one of the following options via EZPARAMS/EASYTRAN can be supplied:

OCCURS1=0	Flags OCCURS as an error
OCCURS1=1	Generates the field without occurs
OCCURS1=2	Generates OCCURS 2 in the place of OCCURS 1

The default in EASYTRAN is OCCURS1=0.

Note: Changing OCCURS 1 to OCCURS 2 doubles the field length. This solution may not be valid for all programs, especially if the field defined with OCCURS 1 changes the length of its group item.

OCCURS1=1 is recommended if the field with OCCURS 1 is not referenced in your program.

Duplicate fields usage and reference

Duplicate fields usage

Migration Utility V1R1.0 did not handle the duplicate field names in compliance with Easytrieve Plus. Sometimes this resulted in a cleanly generated program, but false runtime outcome. The manual solution was to add an object qualifier to the field name when referenced in the program.

Migration Utility V1R1.1 uses the duplicate field names in compliance with Easytrieve Plus, using the following rules:

JOB INPUT &FILE
: If a referenced field is a duplicate name, and it is defined in the JOB file, Migration Utility uses the JOB file field; otherwise, it issues an EZT000-25 error.

JOB SORT& FILEIN TO &FILEOUT...
If a referenced field is a duplicate name, and it is defined in the SORT &FILEIN, Migration Utility uses the &FILEIN file field.

JOB INPUT (&FILE KEY (&KEY1...) &FILE2 KEY (&KEY1...))
If a referenced field is a duplicate name, and it is defined only in one JOB file, Migration Utility uses the JOB file field, otherwise it issues an EZT000-25 error.

JOB INPUT NULL
If a referenced field is a duplicate name, Migration Utility issues an EZT000-25 error.

Unavailable Field reference

In Migration Utility V1R1.0, referenced fields within files that are not used in the job were accepted. Easytrieve Plus flags such conditions as errors.

Migration Utility V1R1.1 flags referenced file fields of unused files within the same job.

File DDname considerations

Migration Utility generates sort work and temporary files for internal use whenever a SORT or a report with SEQUENCE statement is encountered. These generated file names may interfere with the file names defined in the Easytrieve program. If you encounter a problem, make a global change to the file name in your program. The standard internal names are: SORTWK*n*, SORTFL*n* and TEMPWK*n*, where *n* is the sequence number assigned to each file.

When running in static mode, Migration Utility interprets all files that begin with SORTWK and SORTFL as sort work files. The select statement for these files is generated without a FILESTATUS flag. Files that begin with SORTWK or SORTFL in your program should be changed to a different name to avoid complications.

VSAM files, mixed I/O mode

Easytrieve Plus allows mixed I/O mode (dynamic, sequential and SKIP sequential) for VSAM files within a single JOB. However, you must use mixed I/O mode in an orderly way such that the file is always appropriately positioned for the sequential I/Os.

Migration Utility allows mixed I/O mode, but switching back and forth from sequential to random or dynamic mode can lead to incompatible results.

For example, when a new record is added, Easytrieve Plus repositions the file from sequential mode to the point where the new record is inserted. In the same situation, COBOL remains positioned to read the next record for sequential mode, even if a record is inserted elsewhere.

When a file with mixed I/O mode is detected, the translator warning message “POSSIBLE I/O CONFLICT” is issued. If the VSAM file updated by Migration Utility does not match the VSAM file updated by Easytrieve Plus, check for translator warning messages to see if incorrect file positioning is the cause.

To properly position the file, use the POINT for sequential operations, or define a second file for dynamic/random operations.

Chapter 3. Defining entities

This chapter tells you how to define files, tables, records and working storage in Migration Utility.

Defining files

The FILE statement describes the files that are referenced in the program.

Various ways of defining files are described on the pages that follow.

Supported file organizations

INDEXED	VSAM KSDS File
RELATIVE	VSAM Relative File
VSAM-SEQ	VSAM Sequential File
TABLE	Instream or External Table
CARD	Card Reader
PRINTER	Printer
PUNCH	Card Punch
SQL	SQL/DB2
TAPE	Tape File
DISK	Sequential Disk File
SEQUENTIAL	Any Sequential File
VIRTUAL	Easytrieve Virtual File (treated as sequential file)
PDS	PDS and PDSE files

Supported sequential file record formats

F	Fixed Unblocked
V	Variable Unblocked
U	Undefined
FB	Fixed Blocked
VBS	Variable Blocked Spanned
VB	Variable Blocked

Non-supported file organizations

DLI	DLI Files are not flagged
IS	ISAM Files are flagged
IDMS	IDMS Files are flagged

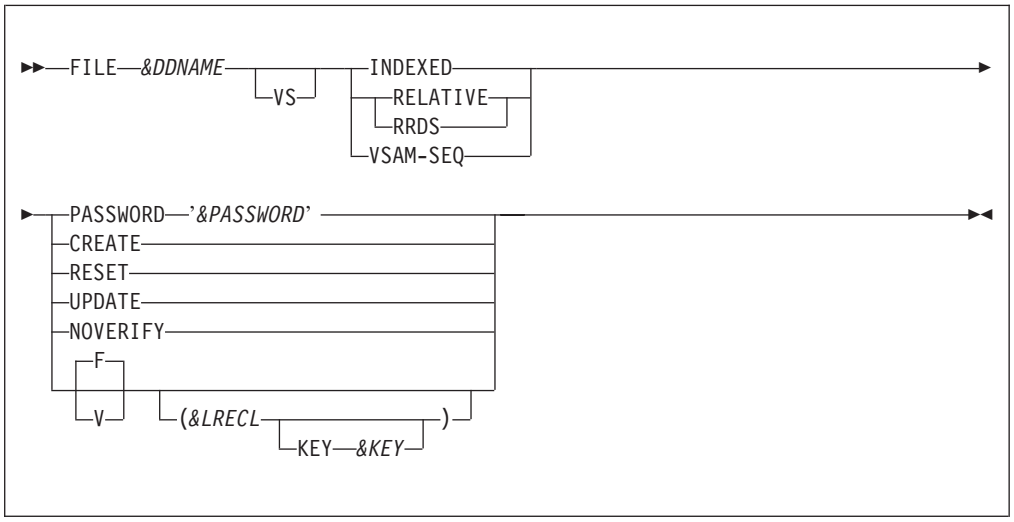
Non-supported file attributes (these attributes are bypassed)

ASA	Option is ignored
WORKAREA	Option is ignored
EXTENDED	Option is not supported
DBSCODE	Option is not supported
RETAIN	Option is ignored; you can control it via JCL.

Supported file attributes

BUFNO	Number of buffers used when IOMODE=DYNAM is specified

Defining VSAM files



Parameters

VS Indicates a VSAM File.

INDEXED
Defines KSDS VSAM File

RELATIVE
Defines Relative VSAM File

RRDS The same as RELATIVE

VSAM-SEQ
Defines ESDS VSAM File

CREATE
Defines an output file

PASSWORD '*&PASSWORD***'**
&PASSWORD is a one to eight character VSAM file password.

RESET
Resets file to starting point (ignored by Migration Utility)

UPDATE
Defines file for update mode Valid for INDEXED and RELATIVE files only.

NOVERIFY
Ignore File Verify (ignored by Migration Utility)

Record format.
Possible values are:
F Fixed
V Variable

This is a Migration Utility convention only. Easytrieve does not support it.

&LRECL
Record length. The default is the size of the defined record. Easytrieve does not support record length for VSAM Files. The length is obtained from the VSAM Catalog.

KEY *&KEY* is valid for VSAM KSDS (Indexed) files only. *&KEY* identifies the field name coded in the record layout to be used as the VSAM key. This option is not supported by Easytrieve. Use this syntax when defining file layout using the %CBLCNVRT macro from COBOL copybooks.

When running in IOMODE=DYNAM, record length and key locations are resolved at run time dynamically. The following customization applies only when running in static mode (IOMODE=NODYNAM).

If KEY *&KEY* is not coded, the File KEY for INDEXED files must be the first defined field in the record. The field must be an alphanumeric field or a group item. Numeric fields are flagged as errors. The File KEY for RELATIVE files is automatically generated by Migration Utility based on the key usage in the I/O statements.

Record format and record length are not allowed by Easytrieve for VSAM files. It is a Migration Utility option only. To retain Easytrieve compatibility, make sure that the length of the record you define is equal to the real file record length. The &LRECL option is provided as a safety feature if you want to prevent the program from ever being run using native Easytrieve.

Migration Utility depends on the value of the record format to recognize VSAM variable record format for OUTPUT and UPDATE Files. This is not an Easytrieve convention. You must code a “V” for VSAM variable-length records.

Examples

This example defines FILEIN1 VSAM INDEXED File with key length of 16 bytes and fixed record size of 500 bytes.

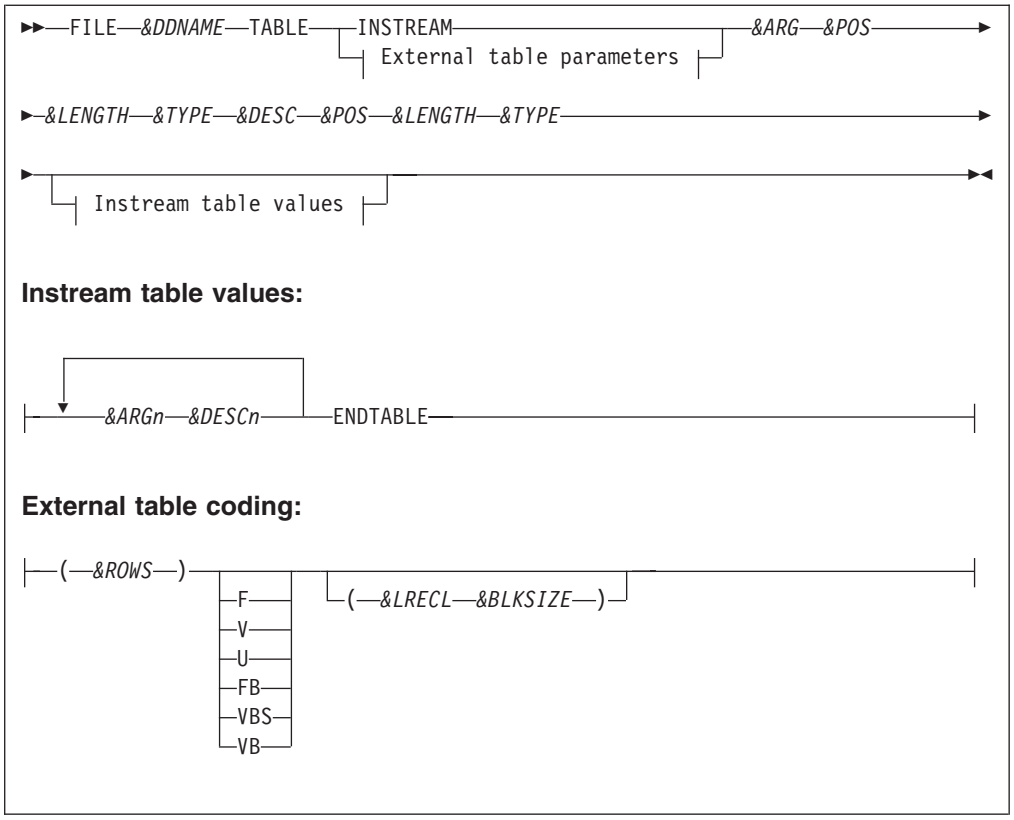
```
FILE FILEIN1 INDEXED UPDATE (500)

    IN1-KEY       1  16  A              | <= key is the first defined field
    IN1-RECORD    1  500 A              | <= ensures full size
         .                              |
         .                              | <= other layout can be coded
```

This example defines FILEIN1 VSAM INDEXED File with key length of 16 bytes and variable record size of 500 bytes.

```
FILE FILEIN1 INDEXED UPDATE V(500)

    IN1-KEY       1  16  A              | <= key is the first defined field
    IN1-RECORD    1  500 A              | <= ensures full size
         .                              |
         .                              | <= other layout can be coded
```

Defining tables

Migration Utility supports instream and external tables. The instream tables have data imbedded in the program following the table definition. For external tables, data is read from an external file. In either case, the data must consist of two fields, Argument and Description. Argument is the Table Key. Description is associated with the Key.



Parameters

&DDNAME
1 to 8 character file name

&ARG Field name for table key

&DESC
Description for field name

&ARGn
Table data for Argument field

&DESCN
Table data for Description field

&POS Start Position in the record

&LENGTH
Field Length

&TYPE
Field Type, A, N, P

ENDTABLE
Required end of data marker

&ROWS
Maximum number of Table Rows

Record format
Can be:
F Fixed Unblocked
V Variable Unblocked
U Undefined

FB Fixed Blocked
VBS Variable Blocked Spanned
VB Variable Blocked

This is a Migration Utility optional parameter.

&LRECL
Record length

When running in IOMODE=DYNAM, record length is resolved at run time dynamically. The following customization applies only when running in static mode (IOMODE=NODYNAM).

Record length is required for &RECFM V, U, VB. The length must include 4 extra bytes over the actual record size for all variable-length files (V or VB).

Record length is required when the actual record length of your file is not equal to the size of the defined layout. The default is the size of the defined record.

This is a Migration Utility optional parameter.

&ARG Field name for table key

&DESC
Description field name

&POS Start Position in the record

&LENGTH
Field Length

&TYPE
Field Type: A or N. Other formats are not supported by Easytrieve.

Note: &RECFM and &LRECL are Migration Utility parameters only. This convention provides the ability to define a table file that has different record length from the size defined by the layout.

Examples

This example defines WEEKDAY Instream Table for translating a day of the week:

```
FILE WEEKDAY TABLE INSTREAM
             ARG1   1 1 A
             DESC1  3 9 A
1 SUNDAY
2 MONDAY
3 TUESDAY
4 WEDNESDAY
5 THURSDAY
6 FRIDAY
7 SATURDAY
ENDTABLE
```

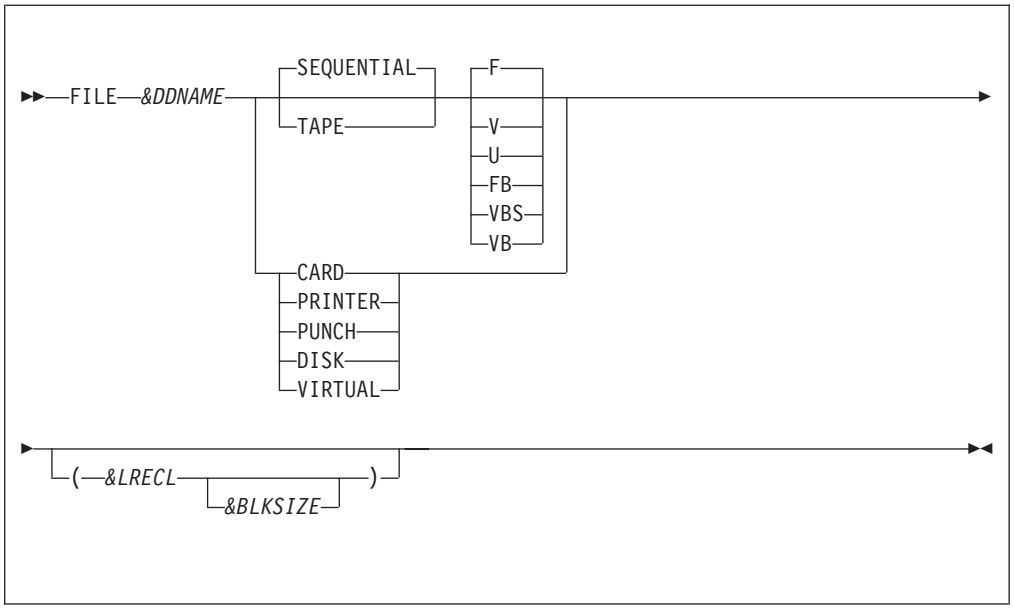
This example define an external Branch Table of 150 rows with a 2 digit Branch Number and a 15 digit Branch Name:

```
FILE BRTABLE TABLE (150)
             BRANCH         1 2  A
             DESCRIPTION    4 15 A
```

This example defines the same table as in Example 1 that resides on a variable length file of LRECL=60:

```
FILE BRTABLE TABLE (150) V(64)
             BRANCH         1 2  A
             DESCRIPTION    4 15 A
```

Defining unit record devices and sequential files



Parameters

&DDNAME
One to eight character file name

Device:

CARD	Card reader
PRINTER	Printer
PUNCH	Punch device
TAPE	Tape file
DISK	Sequential disk file
SEQUENTIAL	Any sequential file
VIRTUAL	Easytrieve virtual file

VIRTUAL files are handled as sequential disk files.

Record format:

F	Fixed Unblocked
V	Variable Unblocked
U	Undefined
FB	Fixed Blocked
VBS	Variable Blocked Spanned
VB	Variable Blocked

&LRECL
Record length

When running in IOMODE=DYNAM, record length is resolved at run time dynamically. The following customization applies only when running in static mode (IOMODE=NODYNAM).

Record length is required for records of format V, U, and VB. The length must include four extra bytes over the actual record size for all variable-length files (V or VB). The record length is required when the record layout is not coded. The default record length is the size of the defined record.

&BLKSIZE
Block size. Ignored by Migration Utility. The block size can be controlled via DCB in JCL.

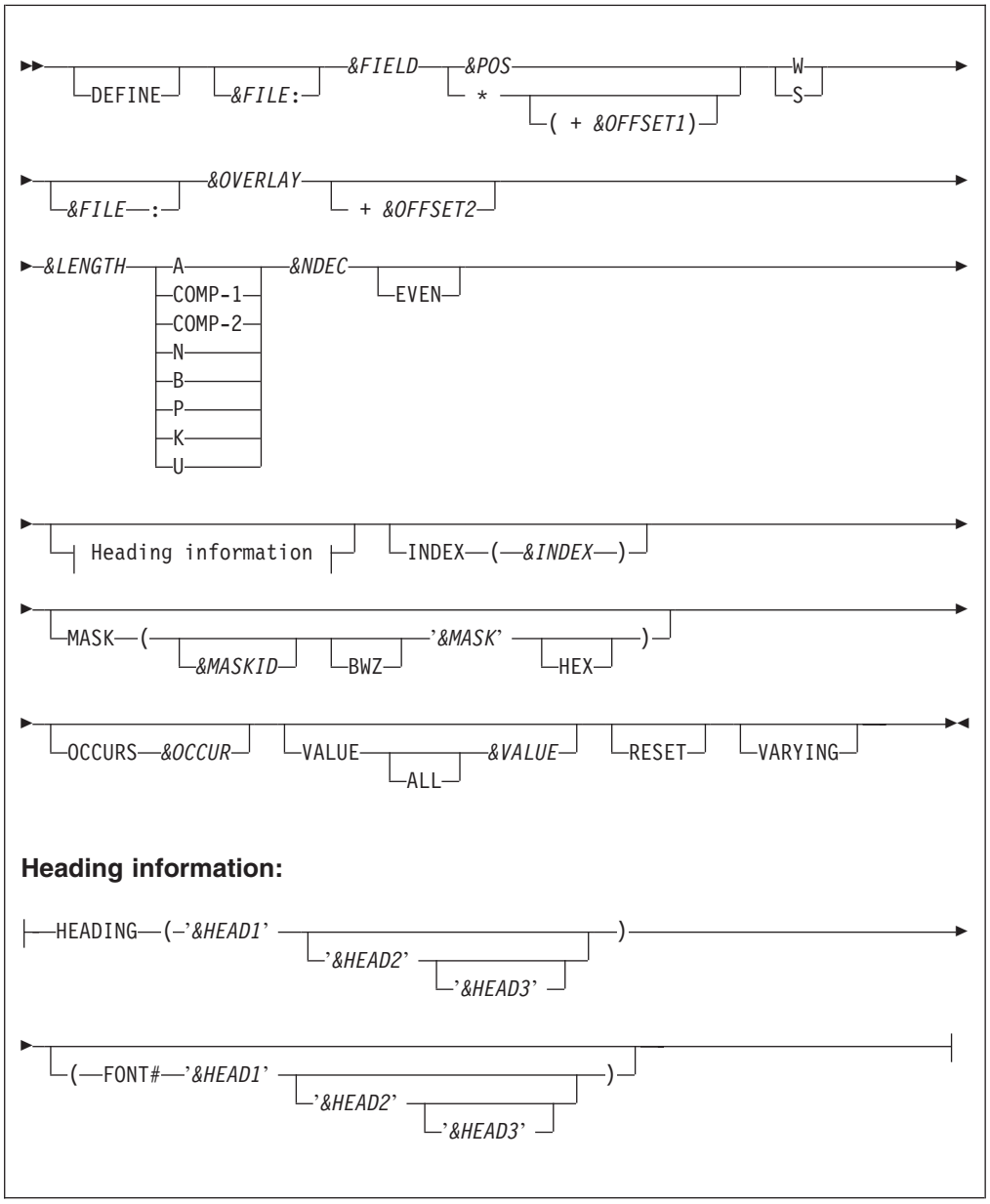
Note: Each file definition can be optionally followed by the record layout. For additional information refer to "Defining Records and Working Storage" on page 32.

Examples

Here are some variations of possible file definitions:

```
FILE INPUT1 CARD    (80)
FILE INPUT2 VB      (260 0)
FILE OUTFIL FB      (512 0)
FILE MASTER TAPE F  (2500)
FILE TRANFL DISK F  (3000)
FILE OUTFIL VIRTUAL
```

Defining Records and Working Storage



Parameters

DEFINE
This keyword denotes the beginning of a field. It is usually omitted.

&FILE File name. It is supported by Migration Utility but not recommended.

&FIELD
1 to 30 character field name

&POS Starting field position in the record

*** +** *&OFFSET1*
Relative offset to the last defined field (* for current location)

W Establishes a working storage field that can be changed

S Establishes a static working storage field (equivalent to a literal)

&OVERLAY
The group field name that this field belongs to

&OFFSET2
Displacement relative to the &OVERLAY. The displacement plus the field size must fit within the boundary of the &OVERLAY field.

&LENGTH
Field length

Field type:

A	Alphanumeric
COMP-1	Single-precision floating point number
COMP-2	Double-precision floating point number
N	Numeric
B	Binary (see comments below)
P	Packed decimal
K	Double character set
U	Packed unsigned

Note: Floating-point types, COMP-1 and COMP-2 fields cannot be printed or displayed. To print or display a COMP-1 or COMP-2 field, you must first move the contents into a valid numeric field. However, you can display the value using native COBOL.

&NDEC
Number of decimal places (numeric fields only)

EVEN
Valid for U fields only. Forces the number of characters represented by the field to be even.

&HEAD1, *&HEAD2*, *&HEAD3*
Field headings for report headers. The maximum length is 30 characters.

FONT#
Font Number (not supported by Migration Utility)

&INDEX
A unique index name used for accessing fields with OCCURS

&MASKID
Letters A through Z identify a previously defined mask.

BWZ Print option. Blank is printed when contents of the field is zero.

&MASK
Print mask

HEX Print option for printing in HEX

&OCCUR
Number of field occurrences

&VALUE
Initial field value. For alphanumeric fields, the value must be enclosed in quotes. ALL is a Migration Utility option only. It is not supported by Easytrieve.

RESET
Field is to be initialized at the beginning of each JOB.

VARYING
Field is a variable-length field (alphanumeric fields only).

The maximum value that can be contained in the binary fields when running with Easytrieve differs from the value that can be accommodated by COBOL as follows:

Note: Compiling COBOL with TRUNC(BIN) option will increase the maximum value of 2 byte and 4 byte fields to their maximum capacity. Refer to COBOL Compiler IBM manual for exact values.

Memory Size	Easytrieve Max-Value	COBOL Max-Value
4 bytes	2,147,483,647+ 2,147,483,647-	999,999,999+ 999,999,999-
3 bytes	8,388,607+ 8,388,607-	9,999,999+ n/a
2 bytes	32,767+ 32,767-	9,999+ 9,999-
1 byte	127+ 127-	99+ n/a

COBOL does not support one byte and three byte binary fields. For such fields, Migration Utility expands special code that prepares fields in working storage before they are accessed.

Example

These examples show some variations of possible field definitions.

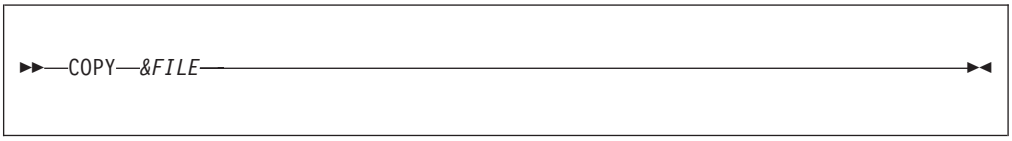
```
FILE FILEIN1 DISK (107)
   IN-ACCOUNT     01  10 N     MASK '99-99999999'   +                  |
                              HEADING ('ACCOUNT' 'NUMBER')             |    sample
   IN-NAME        11  15 A     HEADING ('SHORT' 'NAME')                 |
   IN-CUR-BAL     27  07 P 2   HEADING ('CURRENT' 'BALANCE')            |
   IN-INT-DATA    35  11 A                                              |     file
   IN-INT-DATE IN-INT-DATA    6 N    HEADING ('INTEREST' 'DATA')       |
   IN-INTEREST IN-INT-DATA +6 5 N 2  HEADING ('INTEREST' 'AMOUNT')     |
   IN-AMOUNTS     47  05 N 2  OCCURS 12 INDEX AMOUNT-INDEX              |    record
                                 HEADING ('MONTHLY' 'AMOUNTS')         |

WS-DATE   W     6  N    MASK ('99/99/99')                              |
WS-DATE-MM WS-DATE      2  N                                           |   sample
WS-DATE-DD WS-DATE +2   2  N                                           |   working
WS-DATE-YY WS-DATE +6   2  N                                           |   storage
                                                                       |
WS-REPORT-TITLE    S    40 A VALUE 'REPORT1 TITLE'                     | <= literal
WS-REPORT-TITLE2   W    40 A                                            <= static
```

Chapter 4. Program instruction reference

This portion of the manual lists program instructions, with the syntax, further explanation, and sometimes examples, for each instruction.

COPY statement

The COPY statement duplicates the field definitions of a named file.



Parameter

&FILE The name of the previously defined file whose fields you want to duplicate.

Easytrieve allows an unlimited number of COPY statements for any one file.

Migration Utility allows a maximum of 27 copy statements per program. Migration Utility alters the field naming conventions of the newly created file, by prefixing each field by a letter assigned to the file being defined.

Easytrieve requires the file qualifier to be placed before the field name when referenced.

Migration Utility requires the file qualifier to be placed before the field names expanded due to the COPY statement only. The original file fields are accessed without placing the file name before the fields.

The defined file must be of the same organization as the file being copied.

Examples

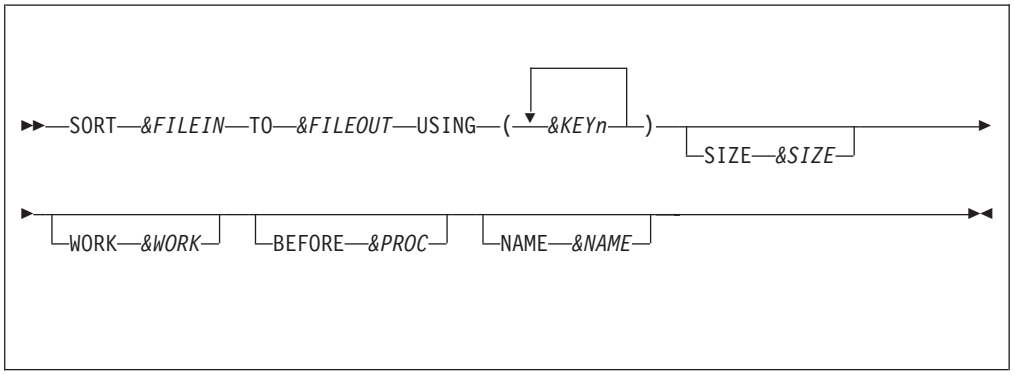
The examples show some variations of possible field definitions.

```
FILE FILEIN1 DISK (107)
   IN-ACCOUNT     01  10 N     MASK '99-99999999'   +
                                HEADING ('ACCOUNT' 'NUMBER')
   IN-NAME        11  15 A     HEADING ('SHORT' 'NAME')
   IN-CUR-BAL     27  07 P 2   HEADING ('CURRENT' 'BALANCE')
   IN-INT-DATA    35  11 A
   IN-INT-DATE IN-INT-DATA    6 N    HEADING ('INTEREST' 'DATA')
   IN-INTEREST IN-INT-DATA +6 5 N 2  HEADING ('INTEREST' 'AMOUNT')
   IN-AMOUNTS     47  05 N 2  OCCURS 12 INDEX AMOUNT-INDEX
                                HEADING ('MONTHLY' 'AMOUNTS')

FILE FILEIN2 DISK (107)                                          FILEIN2
 COPY FILEIN1                                                      COPY
                                                                  FILEIN1
JOB INPUT FILEIN1

GET FILEIN2 STATUS
IF EOF FILEIN2                                                   Some
```

```
   STOP
END-IF                                                                  References
                                                                        to
IF FILEIN1:IN-ACCOUNT NE FILEIN2:IN-ACCOUNT                             with file
   DISPLAY FILEIN1:IN-ACCOUNT ' ACCOUNTS DO NOT MATCH'
                                                                        names
END-IF
```

SORT Activity Section

You can code one or more Sort Activity Sections following the FILE and Working Storage definitions.



Parameters

&FILEIN
1 to 8 character input file name

&FILEOUT
1 to 8 character output file name

&KEYn
Fields to be sorted on (up to eight fields).

&SIZE Sort core size (ignored by Migration Utility)

&WORK
Work area name (ignored by Migration Utility)

&PROC
Input exit (taken after the Read of input record)

&NAME
Sort name (ignored by Migration Utility)

Example

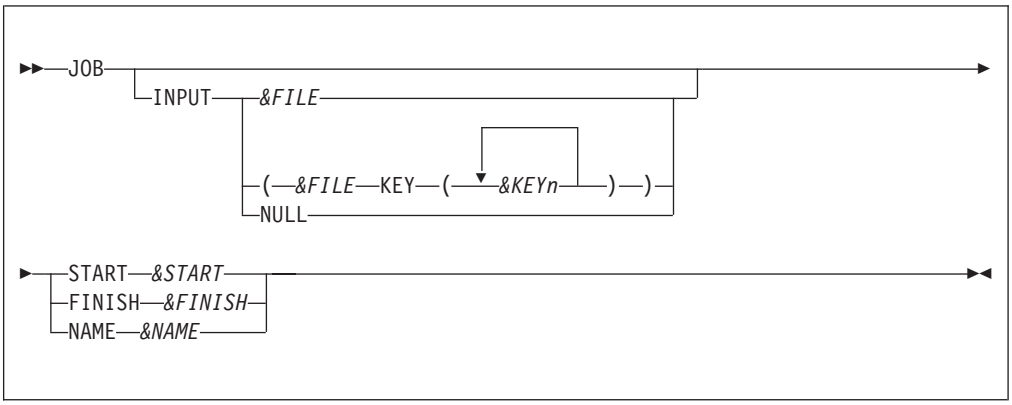
This example shows sorting input file FILEIN to output file FILOUT. Clip non-numeric accounts with all nines.

```
FILE FILEIN  (80)                                                       
   ICUS-ACCT      01  15 N                                             
   ICUS-NAME      16  15 A                                             
   ICUS-ADDRESS1  32  15 A                                             Input file
                                                                       
FILE FILOUT  (80)                                                       
   OCUS-ACCT      01  15 N                                             
   OCUS-NAME      16  15 A                                             
   OCUS-ADDRESS1  32  15 A                                             Output file
```

```
SORT FILEIN TO FILOUT  +
     USING  (ICUS-ACCT ICUS-NAME) +          | Sort statements
     BEFORE SELECT-FILEIN                    |
                                             |
SELECT-FILEIN. PROC.                         |
IF ICUS-ACCT NOT NUMERIC                     |
   ICUST-ACCT = 999999999999999              | Before Sort Exit
END-IF                                       | move all 9's into bad accounts
SELECT                                       | SELECT is needed to accept the
END-PROC.                                    | record.
```

JOB Activity Section

The JOB statement defines and initiates processing activities.



Parameters

INPUT
Identifies the automatic input to the activity. If omitted, the input file is assumed to be the output file from the SORT activity, if any, which immediately preceded the current JOB. Otherwise, the default input file is the first file named in the library section.

&FILE Identifies the automatic input file for sequential processing

&KEYn
Identifies one or more file keys for synchronized file processing (file match). This format requires that at least two input files are defined to the JOB activity. File keys must be of compatible format, for example, numeric or alphanumeric. For further information see “Synchronized file processing” on page 38.

NULL Inhibits the automatic input process. This option is used when input to the program is handled in the activity via READ or GET statements. When “NULL” is coded, a “STOP” must be provided in the activity processing section or the JOB will loop.

&START
The start procedure name. Identifies a procedure to be invoked during the initiation of the JOB. It is invoked before any automatic input file records are read, therefore, automatic input file data fields cannot be accessed. The START is usually handy for initializing fields or positioning files before input.

&FINISH
The finish procedure name. Identifies a procedure to be invoked before normal termination of the JOB. It is usually used to display information accumulated during the processing.

&NAME
Assigns a name to the current JOB. This statement is ignored by Migration Utility.

Example

```
FILE FILEIN1 DISK (80)                          |
   ICUS-ACCT      01  15 N                      |
   ICUS-NAME      16  15 A                      |
   ICUS-ADDRESS1  32  15 A                      | Input file 1
   ICUS-ADDRESS2  48  15 A                      |
   ICUS-ADDRESS3  62  15 A                      |
                                                |
FILE FILEIN2 DISK (80)                          |
   JCUS-ACCT      01  15 N                      |
   JCUS-NAME      16  15 A                      |
   JCUS-ADDRESS1  32  15 A                      | Input file 2
   JCUS-ADDRESS2  48  15 A                      |
   JCUS-ADDRESS3  62  15 A                      |

JOB INPUT NULL
     .                                          | JOB with no automatic input
     .

JOB INPUT FILEIN1
     .                                          | JOB WITH FILEIN1 as input
     .

JOB INPUT +
    (FILEIN1 KEY(ICUS-ACCT) +                   | JOB with synchronized files
     FILEIN2 KEY(JCUS-ACCT))                    | process (file match)
     .                                          |
     .

JOB INPUT FILEIN1 +
    START  A001-FILEIN1-START +                 | JOB with START and FINISH Procs
    FINISH Z999-JOB1-FINISH                     |
     .                                          |
     .

A001-FILEIN1-START. PROC.
     .                                          | JOB Start Proc
     .                                          |
END-PROC.

Z999-JOB1-FINISH. PROC.
     .                                          | JOB finish Proc
     .                                          |
END-PROC.
```

Synchronized file processing

Synchronized file processing lets you:

- Match or merge multiple input files
- Serially process a single keyed file

In either case, special conditional expressions help determine relationships among files, and file records on individual files. The special conditions are MATCHED,

DUPLICATE, FIRST-DUP, LAST-DUP, and file existence tests as described later in this section (see "Special IF statements in synchronized process" on page 41).

The synchronized file process is initiated via the JOB activity FILE statements.

Each file named in the JOB activity must be followed by one or more keys (field names) to be used in the comparison.

Corresponding keys of all files must be of the same type. Numeric keys must correspond to numeric keys and alphanumeric keys must correspond to alphanumeric keys.

Numeric keys can have different lengths.

Alphanumeric keys are expected to have the same length.

The files cannot be updated during the synchronized file processing because the algorithm reads records ahead.

Indexed and relative files can be positioned, before synchronization starts, using a POINT statement in the START procedure.

Example

This example shows JOB statements with Synchronized File Process:

```
FILE FILE1 DISK (80)
   I1-ACCT        01  15 N
   I1-NAME        16  15 A
   I1-ADDRESS     32  15 A                  Input FILE1

FILE FILE2 DISK (80)
   I2-ACCT        01  15 N
   I2-NAME        16  15 A
   I2-ADDRESS         15 A                  Input FILE2

FILE FILE3 DISK (80)
   I3-ACCT        01  15 N
   I3-NAME        16  15 A
   I3-ADDRESS         15 A                  Input FILE3


JOB INPUT (FILE1 KEY(I1-ACCT)  +             Match all three files
           FILE2 KEY(I2-ACCT)  +
           FILE3 KEY(I3-ACCT))
     .
JOB INPUT (FILE1 KEY(I1-ACCT)  +
           FILE3 KEY(I2-ACCT))               Match FILE1 to FILE2
     .
JOB INPUT (FILE1 KEY(I1-ACCT)  +
           FILE3 KEY(I3-ACCT))               Match FILE1 to FILE3
```

Record availability

During synchronization, file records are made available for input based on the relationships of the files' key. Records with the lowest key are made available first, and so on, following the hierarchy order of the files specified on the JOB statement.

Record availability

Duplicate key values affect record availability differently based on which file contains the duplicates. The matching algorithm is hierarchical. The lower level file key is exhausted before another record is processed from the next higher level file. The figure below depicts the concept:

```
Input Files Data

     FILE1 RECORD      FILE2 RECORD      FILE3 RECORD
      KEY   #           KEY   #           KEY    #

     AAAA   1          BBBB   1          AAAA    1
     BBBB   2          CCCC   2          CCCC    2
     CCCC   3          CCCC   3          DDDD    3
     CCCC   4          DDDD   4          EEEE    4
     HHHH   5          DDDD   5          GGGG    5
     HHHH   6          FFFF   6          HHHH    6
     IIII   7          GGGG   7          HHHH    7


  ------------- Record Availability during the JOB ------------

  JOB      FILE1 RECORD      FILE2 RECORD      FILE3 RECORD
 CYCLE      KEY   #           KEY   #           KEY    #

  1        AAAA   1               N/A         AAAA    1
  2        BBBB   2          BBBB   1               N/A
  3        CCCC   3          CCCC   2          CCCC    2
  4        CCCC   3          CCCC   3               N/A
  5        CCCC   4               N/A               N/A
  6             N/A          DDDD   4          DDDD    3
  7             N/A          DDDD   5               N/A
  8             N/A               N/A         EEEE    4
  9             N/A          FFFF   6               N/A
 10             N/A          GGGG   7          GGGG    5
 11        HHHH   5               N/A         HHHH    6
 12        HHHH   5               N/A         HHHH    7
 13        HHHH   6               N/A               N/A
 14        IIII   7               N/A               N/A
```

As per above, there are two CCCC keys on FILE1 and FILE2 and one CCCC key on FILE3.

In JOB Cycle #3, the first CCCC record of FILE1, FILE2 and FILE3 are available.

In JOB Cycle #4, the first CCCC record of FILE1, the second CCCC record of FILE2 are available only. A record from FILE3 is not available at all.

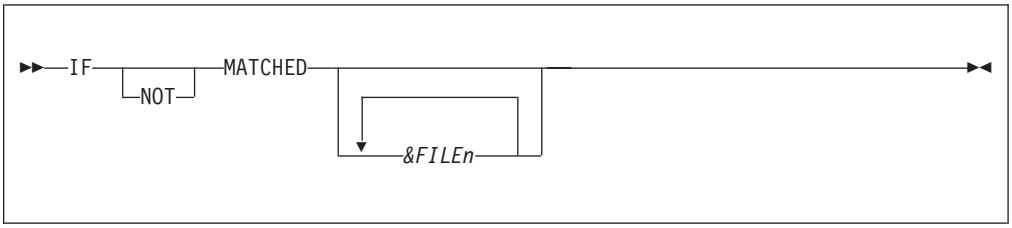
In JOB Cycle #5, the second CCCC record of FILE1 is available only. A FILE2 and FILE3 records are not available at all.

Special IF statements in synchronized process

The following special IF statements allow you to process records based on the match criteria.

MATCHED

Use the MATCHED test to determine the relationship between the current record of one file with the current record of one or more other files.



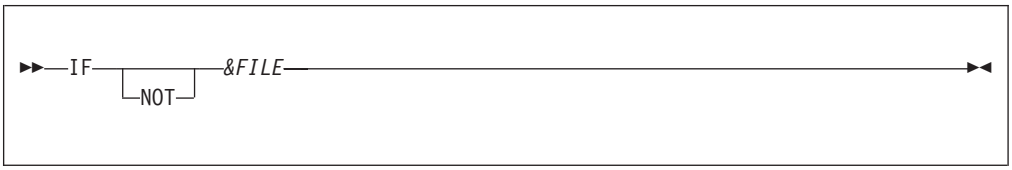
Parameter

&FILEn
The names of the file names being matched

If "MATCHED" is not followed by at least one file name, then all files are included in the test.

File existence

To determine presence of data from a specific file, use this special test.



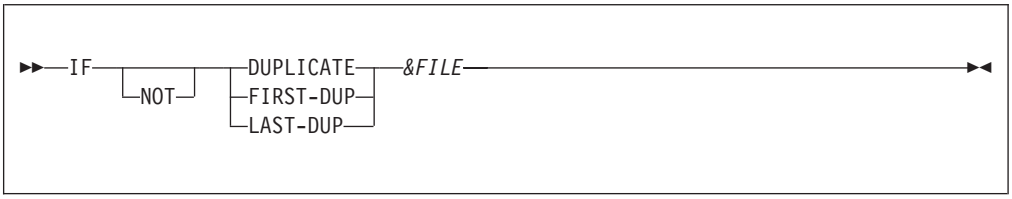
Parameter

&FILE The name of the file whose existence is being tested

If the `IF &FILE` test is true, then file record is available and can be processed. Otherwise, the &FILE record is not available for process.

DUPLICATE, FIRST-DUP, LAST-DUP

DUPLICATE, FIRST-DUP and LAST-DUP determine the relationship of the current record of a file to the preceding and following records in the same file.



Parameter

&FILE The name of the file being tested.

Special IF statements in synchronized process

IF DUPLICATE &FILE is true when duplicate key exists (JOB Cycle 3, 4 and 5 for FILE1 on the previous page).

IF FIRST-DUP &FILE is true when the first record containing duplicate key is processed (JOB Cycle 3 for FILE1 on the previous page)

IF LAST-DUP &FILE is true when the first record containing duplicate key is processed (JOB Cycle 5 for FILE1 on the previous page)

Example

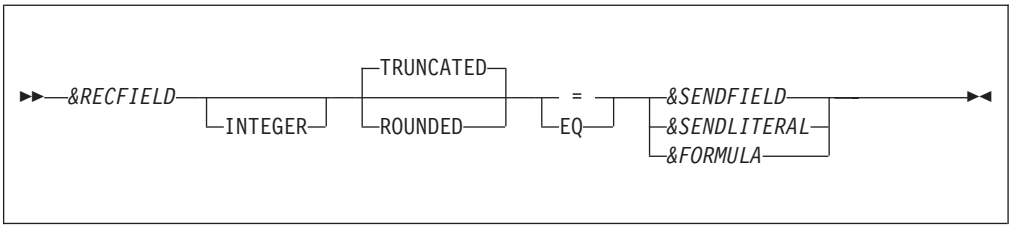
```
FILE FILE1   DISK (80)                          |
   ICUS-ACCT      01  15 N                      |
   ICUS-NAME      16  15 A                      |
   ICUS-ADDRESS1  32  15 A                      | Input file 1
   ICUS-ADDRESS2  48  15 A                      |
   ICUS-ADDRESS3  62  15 A                      |
                                                |
FILE FILE2   DISK (80)                          |
   JCUS-ACCT      01  15 N                      |
   JCUS-NAME      16  15 A                      |
   JCUS-ADDRESS1  32  15 A                      | Input file 2
   JCUS-ADDRESS2  48  15 A                      |
   JCUS-ADDRESS3  62  15 A                      |


JOB INPUT +                                     |
    (FILE1   KEY(ICUS-ACCT) +                   | JOB with Synchronized Files
     FILE2   KEY(JCUS-ACCT))                    | Process (file match)
                                                |

IF MATCHED                                      |
   PRINT REPORT1                                | Report all MATCHED Records
END-IF                                          |
                                                |
IF DUPLICATE FILE1                              |
   PRINT REPORT2                                | Report Duplicates on FILE1
END-IF                                          |
                                                |
IF DUPLICATE FILE2                              |
   PRINT REPORT3                                | Report Duplicates on FILE2
END-IF.                                         |
                                                |
REPORT REPORT1                                  |
TITLE 01 'REPORT OF MATCHED RECORDS'            |
LINE  01 ICUS-ACCT JCUS-ACCT                    |
                                                |
REPORT REPORT2                                  |
TITLE 01 'DUPLICATE RECORDS ON FILE1'           |
LINE  01 ICUS-ACCT ICUS-NAME                    |
                                                |
REPORT REPORT3                                  |
TITLE 01 'DUPLICATE RECORDS ON FILE2'           |
LINE  01 JCUS-ACCT JCUS-NAME                    |
                                                |
```

Assignment statement

The Assignment statement assigns a value to a field. The value can be another field, a literal or an arithmetic expression.

There are two types of assignment statements:

1. Normal assignment (assigns field values and arithmetic outcomes to a field). This type of assignment is supported by Migration Utility as described in this section.
2. Bit field assignments (used with XOR, AND, OR Logical operators). This type of assignment is supported by Migration Utility via a CALL to special subprogram. Because of its infrequent use, this type of assignment is not described in this manual. However, functionally the generated COBOL logic yields the same results as Easytrieve.



Parameters

&RECFIELD
The field name to which the value is assigned

INTEGER
Coding INTEGER drops the decimal digits from the assigned value

ROUNDED | TRUNCATED
Specify ROUNDED or TRUNCATED when the receiving field is too small to handle the fractional result of the assignment.

= | EQ
Use = or EQ to indicate assignment.

&SENDFIELD
Sending field (field to be copied)

&SENDLITERAL
Sending value can be a literal. Alphanumeric literals must be enclosed in quotes.

&FORMULA
Arithmetic expression. It can contain arithmetic operators (+, -, *, /). The outcome of the calculation is placed in the *&RECFIELD*.

Migration Utility supports exponentiation (**). Thus, you can exponentiate values before moving them into a field. Exponentiation is native to COBOL.

The data type being assigned to a field must be compatible with the fields' data type. That is, numeric fields require a numeric source and alphanumeric fields require an alphanumeric source. Alphanumeric literal must be enclosed in quotes. Numeric literal can be preceded by "+" or "-".

Example

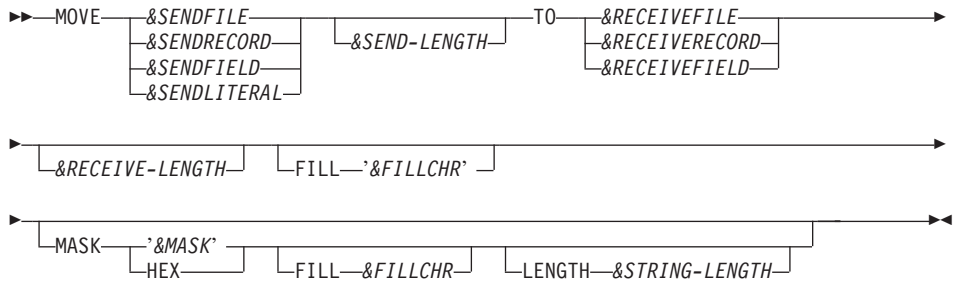
```
FILE FILEIN1
I-BALANCE 1   5  N 2
I-STATE   6   2  A

WS-AMOUNT W   5  N 2
WS-STATE  W  15  A
```

```
JOB INPUT FILEIN1                                    |
WS-AMOUNT = 0                                        | some
IF I-STATE = 'NJ'                                    | assignment
   WS-AMOUNT = I-BALANCE *  1.09                     | statements
   WS-STATE  = 'NEW JERSEY'
END-IF

IF I-BALANCE NOT NUMERIC
   I-BALANCE = ZERO                                  |
END-IF                                               |
                                                     | more
IF I-STATE = 'NY'                                    | assignment
   WS-AMOUNT = ((I-BALANCE + 10000) * 1.01))         | statements
   WS-STATE  = 'NEW YORK'                            |
END-IF                                               |
                                                     |
```

MOVE statement

The MOVE statement transfers data strings from one storage location to another. The MOVE statement is specially useful for moving data without conversion and for moving variable-length fields. There are two MOVE statement formats.

Format 1



Parameters

Source data identifier

You can use one of the following:

&SENDFILE

A file name defined in the Library Section. Referencing a file name results in a move of the current file record.

&SENDRECORD

A record name or working storage area

&SENDFIELD

A currently available field name

&SENDLITERAL

A literal. Alphanumeric literal must be enclosed in quotes.

&SEND-LENGTH

Length of the sending field. It can be a numeric literal or a field name.

Target location identifier

You can use one of the following:

&RECEIVEFILE

A file name defined in the Library Section. Referencing a file name results in a move into the current file record.

&RECEIVERECORD
A record name or working storage area
&RECEIVEFIELD
A currently available field name

&RECEIVE-LENGTH
Length of the receiving field. It can be a numeric literal or a field name.

&FILLCHR
A pad character. This character is used to pad the target object if the sending object is shorter than the receiving object. The default is spaces.

&MASK
A mask for the receiving field. (This is a Migration Utility extension to MOVE.)

The mask can be:
- Any valid edit mask with insert characters up to 30 characters long.
- "HEX" for conversion to hexadecimal.

The sending field can be a numeric or an alphanumeric field. The receiving field must be an alphanumeric field (a type A field).

When you use '&MASK', the contents of the sending field are edited into the receiving field according to the mask.

When you specify HEX, the contents of the sending field are converted to the hexadecimal equivalent and placed into the receiving field.

If you do not specify a mask, the contents of the sending field are edited into the receiving field according to the default mask of the sending field.

The MASK option is useful for formatting fields for a spreadsheet or for inserting special characters into a data string.
&FILLCHR
A pad character. This character is used to replace trailing spaces in *&RECEIVEFIELD*. Trailing spaces are replaced for alphanumeric masks only.

For example:

```
FIELDA   W     10  VALUE '12345'
FIELDB   W     10  VALUE SPACES

MOVE FIELDA TO FIELDB MASK 'X(10)' FILL '*'
```

After completion, FIELDB contains 123456****.
&STRING-LENGTH
After the MOVE statement has been completed, contains the length of *&RECEIVEFIELD*, excluding the trailing spaces.

Can be a binary, display, or packed decimal field.

If FILL *&FILLCHR* are specified, *&STRING-LENGTH* contains the length of *&RECEIVEFIELD*, excluding the *&FILLCHR* pad character.

For numeric masks, *&STRING-LENGTH* contains the length of *&MASK*.

Example 1:

```
FIELDA   W     10  VALUE '123456'
FIELDB   W     10  VALUE SPACES
WLENGTH  W      2  B

MOVE FIELDA TO FIELDB MASK 'X(10)' FILL X'*' LENGTH WLENGTH
```

After completion, FIELDB contains 123456**** and WLENGTH contains 6.

Example 2:

```
FIELD1   W     10  VALUE '123456'
FIELD2   W     10  VALUE SPACES
WLENGTH  W      2  B

MOVE FIELD1 TO FIELD2 MASK 'ZZZ,ZZ9' LENGTH WLENGTH
```

After completion FIELD2 contains 123,456 and WLENGTH contains 7.

The following example show the effect of using a mask:

```
FIELD-1      W       5   N    VALUE  -123   MASK ('99-99-99')
FIELD-2      W      10   A    VALUE SPACES
FIELD-3      W      10   A    VALUE SPACES
FIELD-4      W      10   A    VALUE SPACES
FIELD-5      W      10   A    VALUE SPACES

MOVE FIELD-1 TO FIELD-2 MASK '----9'
MOVE FIELD-1 TO FIELD-3 MASK HEX
MOVE FIELD-1 TO FIELD-4 MASK
MOVE FIELD-1 TO FIELD-5 MASK 'ZZZZZCR'
```

After the move, the contents of the receiving fields are:

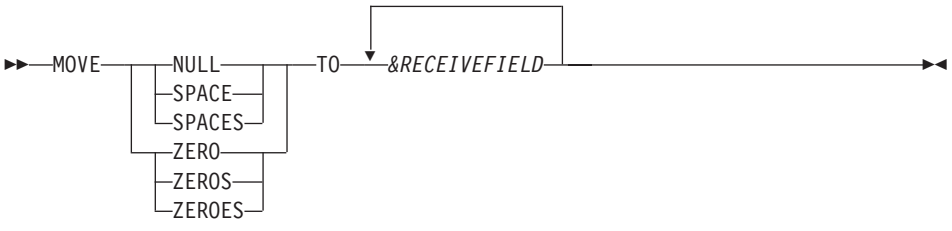
FIELD-2 -123

FIELD-3 F0F0F1F2D3

FIELD-4 00-01-23

FIELD-5 123CR

Format 2



Parameters

The first parameter identifies the sending data area.

The default length is the length of the receiving field. Moving SPACE or

SPACES fills the field with all spaces. Moving NULL fills the field with low values, and moving ZERO, ZEROES or ZEROS moves all zeros to the field.

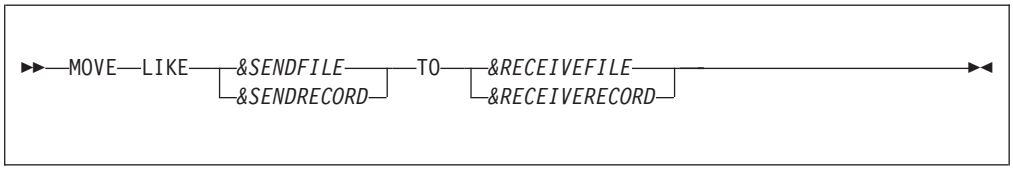
&RECEIVEFIELD
One or more receiving fields. The receiving field is set to the proper data format. However, you cannot move spaces into a packed field or a binary field.

Easytrieve Format 1 data is moved from left to right as if both areas were alphanumeric. The data moved is not converted. It is moved as is, even if the from and to fields are packed or binary fields.

Migration Utility Format 1 generates standard COBOL MOVEs. The data is moved according to the standard COBOL Conversion rules so a move from a binary field into a display numeric field results in data conversion from binary to Display Numeric format, yielding a result that differs from the Easytrieve Move. Compatible results can be achieved by redefining the numeric field as an alpha field and using the alpha field name as the source or target in the move statement.

MOVE LIKE statement

The MOVE LIKE statement moves the contents of fields with identical names from one file, record or working storage to another. Data movement and conversion follow the rules of the Assignment statement.



Parameters

Source file identifier

You can use one of:

&SENDFILE

A file name defined in the Library Section. Referencing a file name results in the move of the current file record.

&SENDRECORD

A record name or working storage area.

Target location identifier

You can use one of :

&RECEIVEFILE

A file name defined in the Library Section. Referencing a file name results in the move into the current file record.

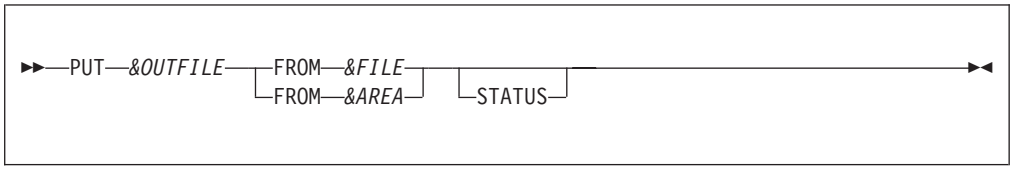
&RECEIVERECORD

A record name or working storage area

The moves are generated starting with the last target field backward. Thus, the order in which overlapping fields are defined is important.

PUT statement

The PUT statement writes a record to an output sequential file. It also adds consecutive records to a VSAM Indexed or Relative file.



Parameters

&OUTFILE

Output file name

Input source

You can use one of:

FROM *&FILE*
FROM *&AREA*

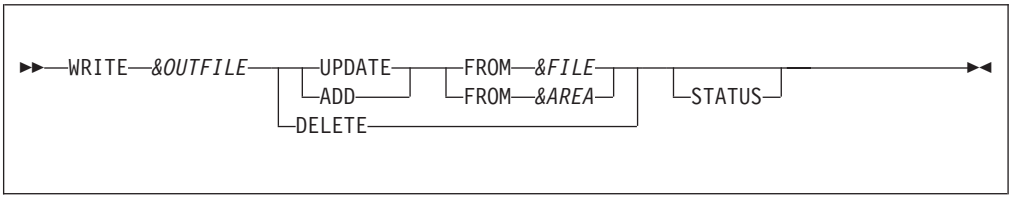
For variable-length records, the length of the output record is equal to the length of the input record. For fixed-length records, the output file record is a fixed-length as defined in the library section. If the FROM object

length is shorter than the output record, only the length of the input object is moved. The remaining length remains uninitialized.

STATUS
Specify if you want to test for a successful I/O. Normally, zero in the file status indicates a successful I/O and a non-zero indicates an I/O error.

WRITE statement

The WRITE statement writes a record to an output INDEXED or RELATIVE file in random mode.



Parameters

&OUTFILE
The output file name must be a VSAM Indexed or Relative file. The UPDATE option must be coded on the FILE statement in the Library Section.

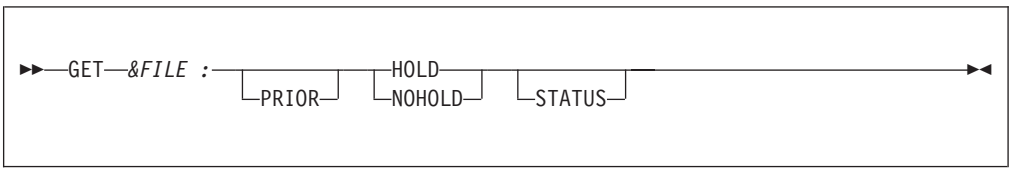
I/O operation
This can be UPDATE, ADD or DELETE.

The name of the input source
&FILE is the file name, *&AREA* is the area name. For variable-length records, the length of the output record is equal to the length of the input record. For fixed-length records, the output file record is of a fixed-length as defined in the library section. If the FROM object length is shorter than the output record, only the length of the input object is moved, and the remaining length remains uninitialized.

STATUS
Specify if you want to test for a successful I/O. Normally a zero in the file status indicates a successful I/O and a non-zero indicates an I/O error.

GET statement

The GET statement reads the next sequential record from the specified file.



Parameters

&FILE The name of the input file to be read.

PRIOR
Reads the previous record from the named file. If the position in the file is not established, the last record on the file is read.

PRIOR is not supported by Migration Utility.

HOLD
Protects the record from a concurrent update

NOHOLD
Does not protect the record from a concurrent update.

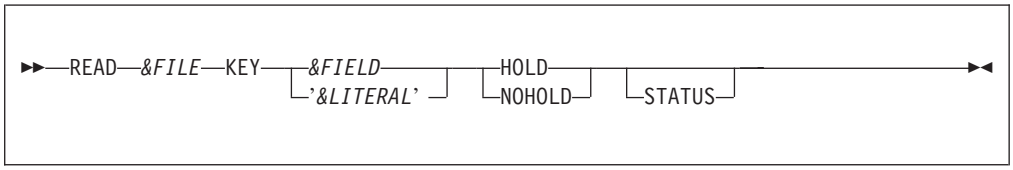
STATUS
Specify if you want to test for a successful I/O. Normally, zero in the file status indicates a successful I/O and a non-zero indicates an I/O error.

HOLD and NOHOLD are not supported by Migration Utility. Such amenities can be accomplished via JCL DISP= parameter and VSAM SHARE Options.

You must test for End OF File (EOF) or file presence (IF &FILE) to ensure record availability.

READ statement

The READ statement performs RANDOM access to INDEXED and RELATIVE VSAM files.



Parameters

&FILE The name of the input file to be read

&FIELD
A field name that contains the file key to be read

&LITERAL
A literal that identifies a record on the file

HOLD
Protects the record from a concurrent update

NOHOLD
Does not protect the record from a concurrent update.

STATUS
Specify if you want to test for a successful I/O. Normally, zero in the file status indicates a successful I/O and a non-zero indicates an I/O error.

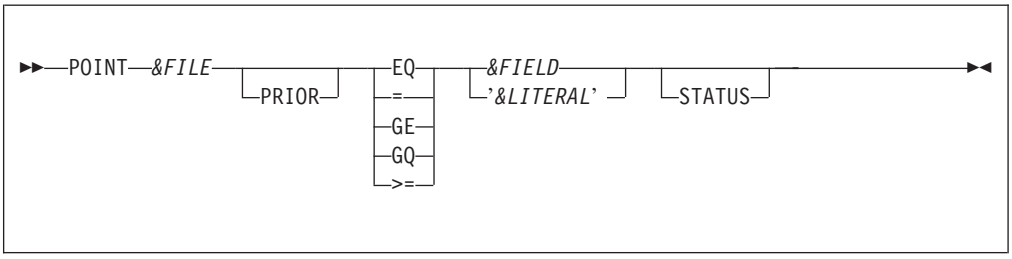
HOLD and NOHOLD are not supported by Migration Utility. Such amenities can be accomplished via JCL DISP= parameter and VSAM SHARE options.

The &FIELD is normally a working storage field or a field in another file. The contents of the &FIELD must be established before READ is issued.

You can use file presence (IF &FILE) to ensure a successful read.

POINT statement

The POINT statement establishes the position in an INDEXED or RELATIVE file for subsequent sequential retrieval. The data is available only after the next sequential retrieval.



Parameters

&FILE Name of input file. It must be an INDEXED or RELATIVE VSAM file.

PRIOR
Specify PRIOR if you want to use PRIOR on the GET statement. See "GET statement" on page 49for more information.

Note: PRIOR is not supported by Migration Utility.

Relational operator for search condition
= and EQ search for the exact key value, GE, GQ and >= search for the first key that is greater than or equal to the key value.

&FIELD
A field name that contains the file key to be searched.

&LITERAL
A literal that identifies a record on the file

STATUS
Specify if you want to test for a successful I/O. Normally, zero in the file status indicates a successful I/O and a non-zero indicates an I/O error.

&FIELD is normally a working storage field or a field in another file. The contents of &FIELD must be established before POINT is issued.

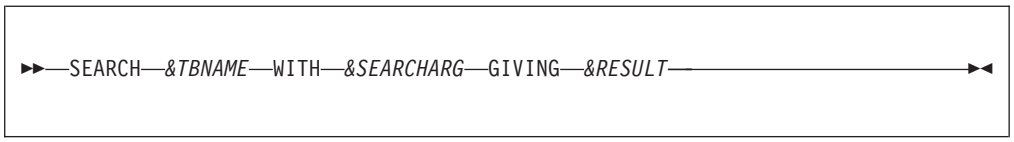
For the KSDS file, the field length or literal value must have the same length as the file key. For the RELATIVE files, the key must be a 4 byte binary integer field.

You cannot use file presence (IF &FILE) to ensure a successful point.

PRIOR is not supported by Migration Utility.

SEARCH statement

The SEARCH statement accesses external or instream table information. Special tests of the IF statement can be used to validate the results of SEARCH.



Parameters

&TBNAME
Name of the TABLE (FILE) that describes table resources

&SEARCHARG
Identifies the field containing the search argument

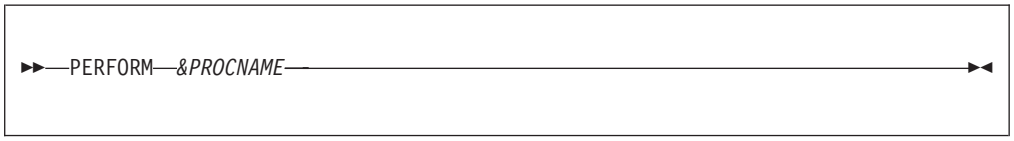
&RESULT
Identifies the receiving field into which data is retrieved

&SEARCHARG is normally a working storage field or a field in another file. The contents of &SEARCHARG must be established before SEARCH is issued.

You can use file presence (IF &FILE) to ensure a successful read.

PERFORM statement

The PERFORM statement executes a procedure, and, after execution, returns to the next statement after PERFORM.



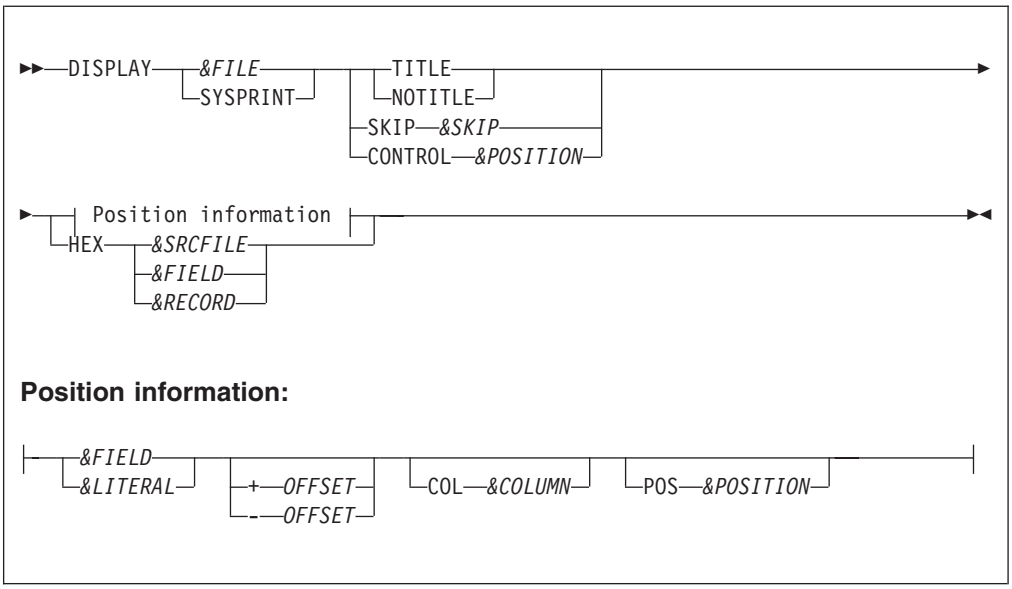
Parameters

&PROCNAME
The name of the Procedure to be executed

PERFORM statements in a procedure can invoke other procedures; however, recursion is not permitted. Recursion can cause unpredictable results.

DISPLAY statement

The DISPLAY statement formats and transfers data to a system output device or to a file.



Parameters

&FILE A sequential file name. Use a file name if you wish to write data to a file.

SYSPRINT
Directs output to the system output device. Migration Utility normally displays to the SYSLIST system file.

Report title control:

TITLE Specify TITLE if you wish to print a report title for the display coded in a report exit. TITLE will skip to a new page on page overflow and print report titles if any.

NOTITLE Specify NOTITLE if you wish to skip to a new page but inhibit printing of the report title.

&SKIP An integer from 0 to ″N″. The number of lines to be skipped before printing. Zero overlays the existing display line.

Migration Utility &skip integer range is 0 to 3.

&CC The print carriage control for controlling spacing. Valid characters are 0 through 9, +, -, A, B, or C depending on the make and model of the printer.

&SRCFILE
The name of a file whose record is to be displayed. Specifying the name results in displaying the most current record contained in the file.

&RECORD
Specifies a record or working storage area to be displayed

&FIELD
Specifies a field name to be displayed

&LITERAL
A character string (literal) to be displayed. Alphanumeric literal must be enclosed in quotes.

OFFSET
The space adjustment parameters modify the normal spacing between display items. + or – indicate the direction in which the spacing is adjusted.

&COLUMN
Specifies the print column number where the next display item is to be placed. The number can be from 1 to nn, but it cannot force the next line item beyond the end of the line.

&POSITION
Specifies the position of display line items in respect to the items on line 1 within report procedures. The position corresponds to the line item number of line 1 under which the line item is placed.

Unless positioning is specified, the first data entry of each display line begins in column 1. Each data item that follows is printed following the previous one with no spaces between data items.

HEX produces five formatted 100 byte lines per record/field.

When DISPLAY is used in the REPORT procedure, the output line is always in the appropriate place in that report, unless you specify a print file that is not the report file to which the procedure applies.

The displayed data is in an edited format. Thus, displaying to an edited file will result in a file that contains edited fields.

Easytrieve Plus does not allow DISPLAY HEX in Report Exits. Migration Utility does but if ″SEQUENCE″ is coded on the report statement, Migration Utility issues a Warning Message about potential ″Undesired″ Sort File record length.

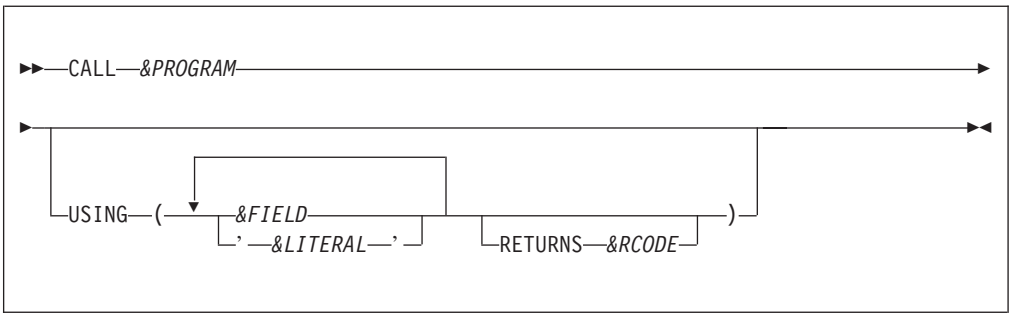
Warning: Doing DISPLAY HEX for SEQUENCE-d reports in report exits can result in large Sort File record length and processing overhead. Use it with caution.

Examples

```
DISPLAY 'CURRENT BALANCE ' WS-CURRENT-BALENCE

DISPLAY HEX FILEIN-RECORD

DISPLAY PRINTER1 CONTROL 1 HEX FILEIN-RECORD
```

CALL statement

The CALL statement allows you to invoke subprograms written in a language other than Easytrieve.



Parameters

&PROGRAM
Name of program to be invoked. From one to 8 characters.

Parameters for passing to the subprogram:
&FIELD
&LITERAL

&RCODE
The field name for the Return Code returned by the called program. It must be a valid numeric field. The COBOL Return Code can be always found in the COBOL RETURN-CODE field. RETURNS is supported by Migration Utility but is not needed.

Easytrieve calls programs dynamically.

Migration Utility generates program calls as follows:

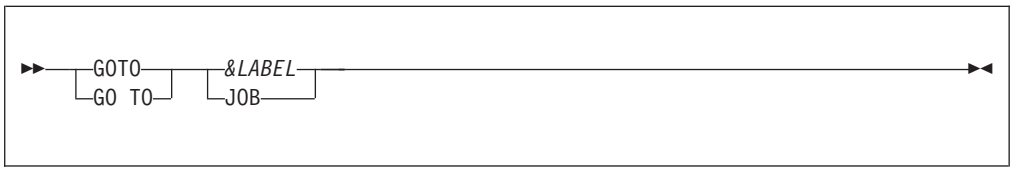
- Generates a static call when the program name is enclosed in quotes.
- Generates a dynamic call when the program name is not enclosed in quotes.

Examples

```
CALL  FSABE01

CALL  FSABE01 USING ('1000' PROGRAM-NAME)
```

GOTO statement

The GOTO statement alters the flow of processing.



Parameters

&LABEL
Specify a label in the current JOB activity section to which control is to be transferred. Processing continues with the first statement following the named label.

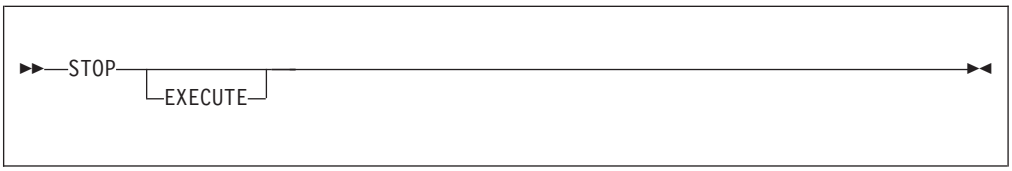
JOB Transfer control back to the first statement of the current JOB activity. When processing the automatic file input, GOTO JOB results in a read of the next sequential record on the input file.

Example

```
FILE FILEIN1 (80)                                    |
   CUST-NAME      01  15 A HEADING ('NAME')          |
   CUST-ADDRESS1  16  15 A HEADING ('ADDRESS1')     | Library Section
   CUST-ADDRESS2  32  15 A HEADING ('ADDRESS2')     |
   CUST-ADDRESS3  48  15 A HEADING ('ADDRESS3')     |

JOB INPUT FILEIN1
  IF CUST-NAME = 'JOHN DOE'
     GOTO PRINT-DATA                                 | Activity section
  ELSE                                               |
     GOTO JOB                                        |
  END-IF                                             | with
                                                     |
PRINT-DATA.                                          |  GOTO statements
   PRINT REPORT1
   GOTO JOB

REPORT REPORT1 LINESIZE 080
TITLE 01 'NAME-ADDRESS REPORT EXAMPLE'               |
LINE  01 CUST-NAME                                   |   REPORT definitions
         CUST-ADDRESS1                               |
         CUST-ADDRESS2                               |
         CUST-ADDRESS3                               |
```

STOP statement

The STOP statement terminates current job activity or program execution.



Parameter

EXECUTE
Immediately terminates all processing. This is equivalent to a Forced-End-of-Job.

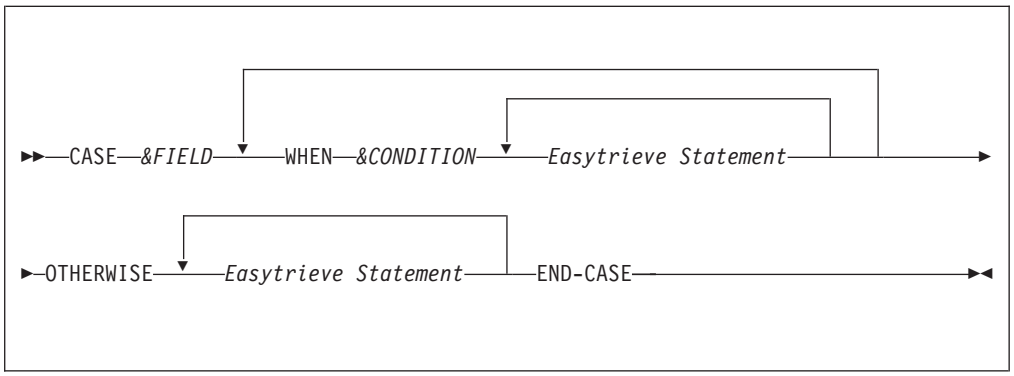
STOP without EXECUTE terminates only current job activity. Any other jobs subsequent to the current one continue processing.

Example

```
FILE FILEIN1 (80)                                    |
   CUST-NAME      01  15 A HEADING ('NAME')          |
   CUST-ADDRESS1  16  15 A HEADING ('ADDRESS1')     |  Library section
   CUST-ADDRESS2  32  15 A HEADING ('ADDRESS2')     |
   CUST-ADDRESS3  48  15 A HEADING ('ADDRESS3')     |

JOB INPUT FILEIN1
  IF FILEIN1:RECORD-COUNT GT 100
     STOP                                            | Activity section
  END-IF                                             |
  IF CUST-NAME GT 'F'                                |
     STOP EXECUTE                                    | with
  END-IF                                             |
                                                     |  STOP statements
  PRINT REPORT1
  GOTO JOB                                           |

REPORT REPORT1 LINESIZE 080
TITLE 01 'NAME-ADDRESS REPORT EXAMPLE'               |
LINE  01 CUST-NAME                                   |   REPORT definitions
         CUST-ADDRESS1                               |
         CUST-ADDRESS2                               |
         CUST-ADDRESS3                               |
```

CASE, WHEN, OTHERWISE and END-CASE statements

The CASE statement provide an elegant way to test values.



Parameters

&FIELD
The field name to be evaluated.

&CONDITION
Value (&LIT) to be tested for. It must be a literal or &LIT1 THRU &LIT2.

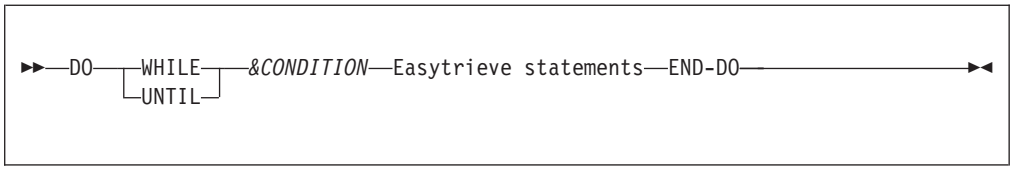
OTHERWISE
Must be the last statement after a series of tests. The statements following OTHERWISE are executed only when all previous tests fail.

END-CASE
Terminates the CASE

The CASE statement is translated by Migration Utility to COBOL EVALUATE statement. The OTHERWISE statement is translated to WHEN OTHER of EVALUATE statement.

DO and END-DO statements

The DO and END-DO statements define the scope of repetitive program logic.



Parameters

WHILE
Evaluates the condition at the top of a group of statements

UNTIL
Evaluates the condition at the bottom of a group of statements

&CONDITION
Specifies the condition for the continuous execution of the loop. Refer to "Conditional expressions" on page 60 for conditional expression syntax.

END-DO
Terminates the DO statement

For DO WHILE, the truth value of the conditional expression determines whether statements bound by the DO END-DO pair are to be executed. When the conditional expression is true, the statements are executed. When the conditional expression is false, the processing continues with the next statement following the END-DO.

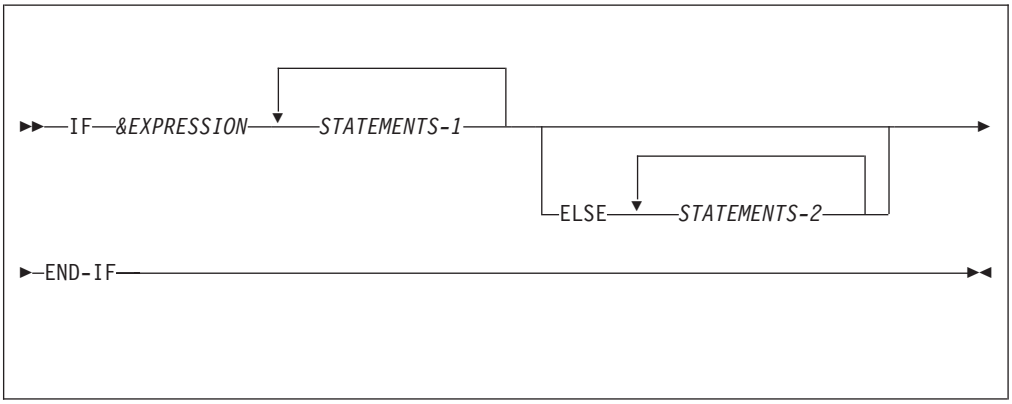
For DO UNTIL, the statements bound by the DO..END-DO are executed. The truth value of the conditional expression (evaluated at end of the statements) determines whether statements bound by the DO..END-DO are to be re-executed. When the conditional expression is true, the statements are re-executed. When the conditional expression is false, the processing continues with the next statement following the END-DO.

Example

```
FILE FILEIN1
FIELD-A   1   10 A  OCCURS 10

WS-COUNT  W    2 N

JOB INPUT FILEIN1
WS-COUNT = 1
 DO UNTIL WS-COUNT GT 10
   DISPLAY FIELD-A (WS-COUNT)
   WS-COUNT = WS-COUNT + 1
END-DO
```

IF, ELSE, and END-IF statements

The IF statement conditionally controls execution of the statements bound by the IF..END-IF.



Parameters

&EXPRESSION
Conditional expression (see "Conditional expressions" on page 60)

STATEMENTS-1
The statements executed if *&EXPRESSION* is evaluated to be true.

STATEMENTS-2
The statements executed if *&EXPRESSION* is evaluated to be false. If ELSE is not specified, then no statements are executed.

END-IF
Terminates the logic associated with the previous IF statement.

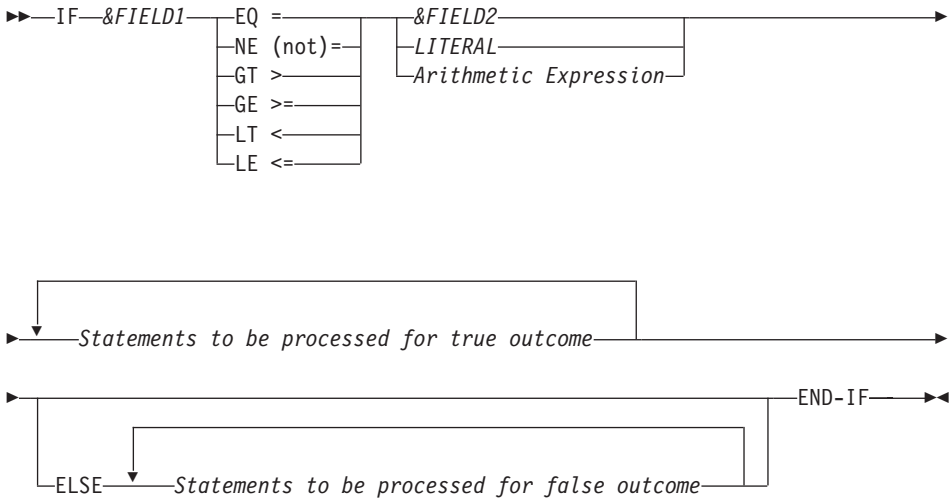
Conditional expressions

Conditional expressions are used in combination with the IF and DO statements to manipulate and select data in the Job Activity section.

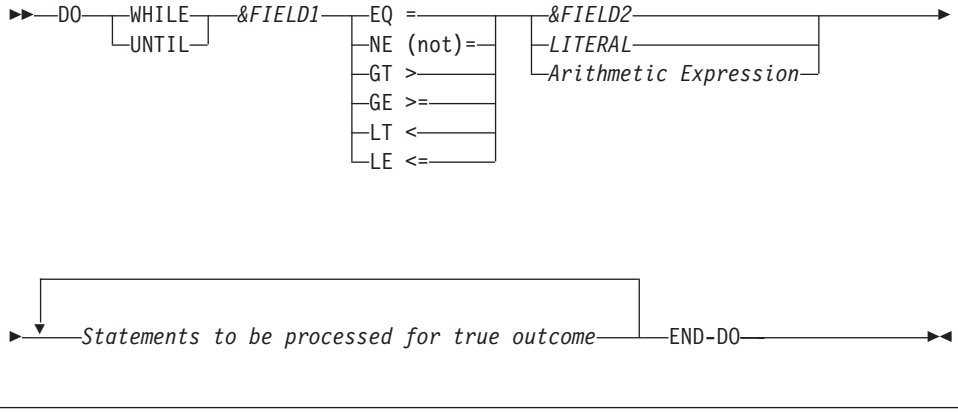
When an IF statement is present, the statements following the IF statement are processed based on the truth of the conditional expression. Statements are processed until an END-IF or an ELSE statement is encountered.

When a DO statement is present, all statements following the DO statement are processed, based on the truth of the conditional expression, until and END-DO statement is encountered.

The IF statement syntax



The DO statement syntax



Parameters

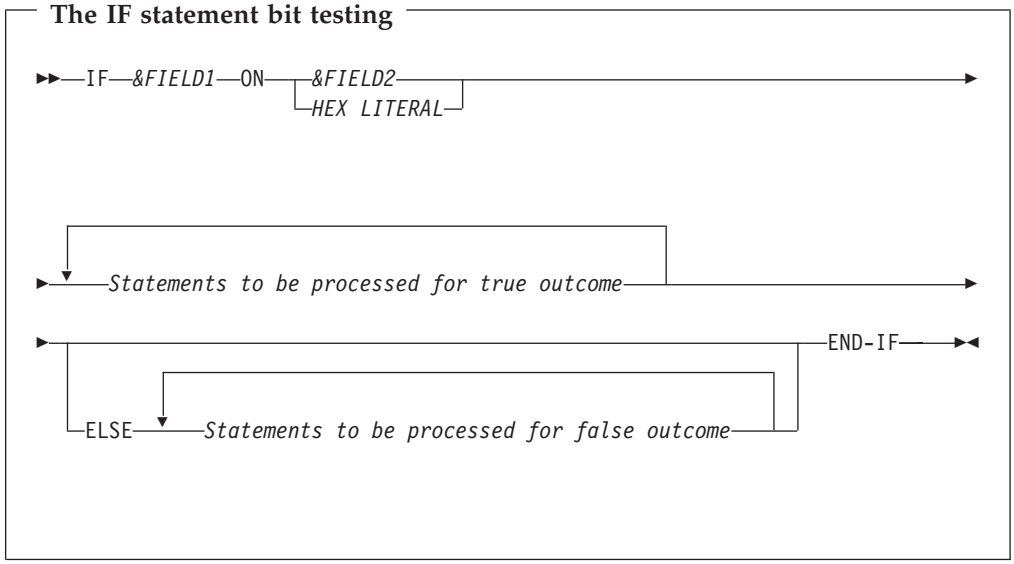
&FIELD1
A field name used as argument 1 in comparison

&FIELD2
A field name used as argument 2 in comparison. The field must be of the same type as &FIELD1. So if &FIELD1 is numeric then &FIELD2 must be numeric.

LITERAL
A numeric or an alphanumeric literal, depending on the type of &FIELD1. Numeric literals can have a leading "+" or "-". Multiple literals can be listed. Also, the THRU statement can be used to denote a range of low to high values.

Arithmetic Expression
Can be any arithmetic expression. Valid only when &FIELD1 is numeric.

The IF statement bit testing



The IF statements can be nested. Migration Utility supports up to NESTS=NN of IF nests (refer to EXPARAMS NESTS=parameter). Any expressions that contain unreasonable levels of IF nests have to be split into multiple expressions to satisfy the limit.

Migration Utility does limited checking for compatible Fields Class of IF arguments. Any missed non-compatible arguments are flagged by the COBOL compiler.

Conditional expressions should be kept as simple as possible. More complex expressions are harder to understand and, sometimes, can lead to absurd outcomes.

Easytrieve allows comparison on a range of values via a THRU statement. The THRU range is translated by Migration Utility to a COBOL equivalent expression, depending on the last interpreted relational/logical operator.

For example, Easytrieve statement

```
IF FIELDA EQ 10 THRU 55
```

is converted to COBOL as

```
IF (FIELDA NOT > 55 AND NOT < 10)
```

whereas

```
IF FIELDA NE 10 THRU 55
```

is converted to COBOL as

```
IF (FIELDA < 10 AND > 55)
```

Easytrieve allows comparison on a list of values. The list is translated by Migration Utility to a COBOL equivalent expression, depending on the last interpreted relational/logical operator.

For example, Easytrieve statement

```
IF FIELDA EQ 10, 15, 20, 25
```

is converted to COBOL as

```
IF (FIELDA = 10 OR 15 OR 20 OR 25)
```

whereas

```
IF FIELDA NE 10, 15, 20, 25
```

is converted to COBOL as

```
IF (FIELDA NOT = 10 AND 15 AND 20 AND 25)
```

There are some differences in the way COBOL and Easytrieve Plus evaluate the IF statement. For example, Easytrieve Plus compares alphanumeric fields using the length of the first argument, whereas COBOL considers the length of both arguments. When converting existing Easytrieve Plus programs, you should perform several parallel runs to ensure the output is the same.

The table below lists allowed Relational and Logical Operators, allowed in Easytrieve, and their equivalent in COBOL as translated by Migration Utility.

Easytrieve	COBOL	Explanation
EQ	=	Test for Equal condition
=	=	
NE	NOT =	Test for Not Equal condition
NQ	NOT =	
*=	NOT =	
LT	<	Test for Less Than condition
LS	<	
<	<	
*<	NOT <	Test for Not Less Than condition
LE	NOT >	Test for Not Greater condition
LQ	NOT >	
<=	NOT >	
GT	>	Test for Greater Than condition
GR	>	
>	>	

```
     *>          NOT >

     GE          NOT <          Test for Greater or Equal condition
     GQ          NOT <
     >=          NOT <

     OR          OR             Logical Operator OR

     AND         AND            Logical Operator AND

     NOT         NOT            Logical Operator NOT

     AND NOT     AND NOT        Logical Operator AND NOT

     OR  NOT     OR NOT         Logical Operator OR NOT
```

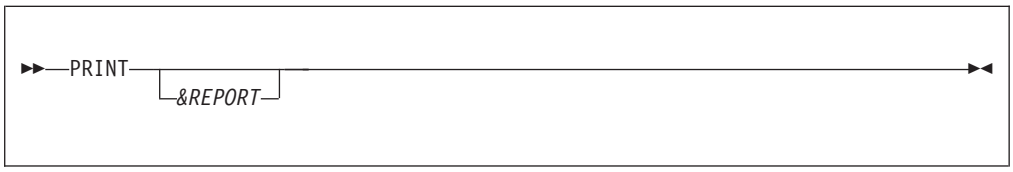
Examples

Here are some examples of IF and END-IF statements:

```
FILE FILEIN1
I-BALANCE 1   5  N 2

WS-AMOUNT W   5  N 2

JOB INPUT FILEIN1
WS-AMOUNT = 0
IF I-BALANCE > 5000                                  |  Non-nested
   WS-AMOUNT = I-BALANCE * 1.10                      |
ELSE                                                 |
   WS-AMOUNT = I-BALANCE * 1.09                      |  IF statement
END-IF

IF I-BALANCE > (WS-AMOUNT + 55)
   WS-AMOUNT = I-BALANCE                             |  Arithmetic in
ELSE                                                 |
   WS-AMOUNT = I-BALANCE * 1.55                      |  IF statement
END-IF

IF I-BALANCE NOT NUMERIC
   DISPLAY 'BALANCE NOT NUMERIC'                     |
   DISPLAY HEX I-BALANCE                             |
 ELSE                                                |  Nested
   IF I-BALANCE EQ 5000, 5500, 5200                  |
      WS-AMOUNT = I-BALANCE * 1.10                   |
   ELSE                                              |   IF statements
       WS-AMOUNT = I-BALANCE * 1.09                  |
   END-IF                                            |
END-IF
```

PRINT statement

The PRINT statement produces report output.



Parameter

&REPORT
REPORT name specified on a report statement. If not supplied, it is assumed to be the first report defined in the JOB activity.

In general, report output is not written directly to a report's printer file. Formatting and printing is usually deferred until end of JOB activity and perhaps after sorting.

When a work file is specified for a report, executing PRINT causes fixed format records to be spooled to the work file. The format of the fields is determined by Easytrieve.

Migration Utility generated COBOL program uses similar concepts when the SEQUENCE statement is included in the REPORT group. However, if the SEQUENCE is not specified, the report is produced directly to the printer. Sort is not invoked.

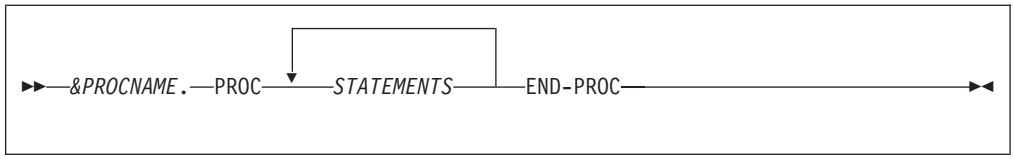
Example

```
FILE FILEIN1 (80)                                  |
   CUST-NAME      01  15 A HEADING ('NAME')       |
   CUST-ADDRESS1  16  15 A HEADING ('ADDRESS1')   | Library Section
   CUST-ADDRESS2  32  15 A HEADING ('ADDRESS2')   |
   CUST-ADDRESS3  48  15 A HEADING ('ADDRESS3')   |

JOB INPUT FILEIN1
   PRINT REPORT1                                   |
                                                   |
REPORT REPORT1 LINESIZE 080                        |
TITLE 01 'NAME-ADDRESS REPORT EXAMPLE'             | Activity Section
LINE  01 CUST-NAME     +                           |
         CUST-ADDRESS1 +                           |
         CUST-ADDRESS2 +                           |
         CUST-ADDRESS3                             |
```

PROC and END-PROC statements

The PROC statement defines the beginning of a procedure in a JOB or SORT activity section. The END-PROC terminates the scope of the PROC.

A procedure can be perceived as a group of statements that perform a specific processing function.



Parameters

&PROCNAME
A label that identifies the procedure. It can:
- Be 128 characters in length
- Contain any character other than a delimiter
- Begin with A-Z, 0-9, or a national character (#, @, $)
- Not consist of all numeric characters

STATEMENTS
Any Easytrieve statements that are valid in the JOB or SORT activity section

END-PROC
Indicates the end of the defined procedure. END-PROC is required for each declared procedure name.

File I/O statements such as PUT or GET cannot be coded in procedures coded during SORT or REPORT processing.

Perform statement can be used to invoke other procedures from any given proc. Recursion is not permitted.

COBOL paragraph names can be 1 to 30 characters in length. All paragraph names longer than 30 characters are truncated by Migration Utility to conform to COBOL Standards.

RETRIEVE statement

The RETRIEVE statement identifies the IDMS database records that are input to the JOB activity.

Retrieve is not supported by Migration Utility, and thus is not described in this manual.

SELECT statement (SORT and REPORT selection)

The SELECT statement can be coded in REPORT-INPUT procedure or BEFORE procedure of a SORT statement.

When coded in REPORT-INPUT procedure, it must be used to select records of interest. Only those records marked SELECT are passed on for printing. If REPORT-INPUT procedure is not coded, then SELECT cannot be used for selecting report records, however, all records are selected in such cases.

SELECT statement (SORT and REPORT selection)

When coded in BEFORE procedure of a SORT statement, it must be used to select records of interest from the input file (file being sorted). Only those records marked SELECT are returned to the sort for further processing.

Example

```
FILE FILEIN1 (80)                                  |
   INPT-NAME      01  15 A HEADING ('NAME')        |
   INPT-ADDRESS1  16  15 A HEADING ('ADDRESS1')    | Library Section
   INPT-ADDRESS2  32  15 A HEADING ('ADDRESS2')    |
   INPT-ADDRESS3  48  15 A HEADING ('ADDRESS3')    |

FILE SORTED1 (80)                                  |
   SORT-NAME      01  15 A HEADING ('NAME')        |
   SORT-ADDRESS1  16  15 A HEADING ('ADDRESS1')    | Library Section
   SORT-ADDRESS2  32  15 A HEADING ('ADDRESS2')    |
   SORT-ADDRESS3  48  15 A HEADING ('ADDRESS3')    |

SORT FILEIN1 TO SORTED1 USING (INPT-NAME) +
               BEFORE INPUT-SORT-EXIT              |  SORT statements

INPUT-SORT-EXIT. PROC
IF INPT-NAME = 'JOHN DOE'                          |
    SELECT                                         |  BEFORE sort procedure
END-IF                                             |
END-PROC.                                          |

JOB INPUT SORTED1
    PRINT REPORT1                                  |
                                                   |
REPORT REPORT1 LINESIZE 080                        |
TITLE 01 'NAME-ADDRESS REPORT EXAMPLE'             |  Report statements
LINE  01 SORT-NAME      +                          |
         SORT-ADDRESS1  +                          |
         SORT-ADDRESS2  +                          |
         SORT-ADDRESS3                             |

REPORT-INPUT. PROC
IF SORT-NAME = 'JOHN DOE'                          |
    SELECT                                         |  REPORT-INPUT procedure
END-IF                                             |
END-PROC.                                          |
```

System-defined fields

Easytrieve provides three categories of system-defined fields:

- General fields
- File related fields
- Report related fields

The fields for each category are described in the sections that follow. Fields not described are not supported by Migration Utility.

General fields (available globally)

SYSDATE

An 8 byte date field in MM/DD/YY format. The date is obtained at the start of program execution.

Easytrieve replaces the leading zero by a space when printing on Report Headings or Detail Lines.

Migration Utility replaces the leading zero by a space when printing on Report Headings only. The leading zero is printed on Detail Lines, however.

SYSDATE-LONG
A 10 byte date field in MM/DD/CCYY format. The date is obtained at the start of program execution.

See SYSDATE above for printing rules.

SYSTIME
An 8 byte time field in HH:MM:SS format. The time is obtained at the start of program execution.

See SYSDATE above for printing rules.

RETURN-CODE
A 4 byte binary field which is returned to the MVS™ operating system at end of job (program termination).

File fields (available globally in each activity section)

RECORD-LENGTH
A 4 byte binary field available for all file types. It contains the length of the last accessed or written file record. For variable-length records, the field contains only the length of the data (the 4 length related bytes are excluded). For variable-length records, the length must be assigned before the WRITE or PUT operations.

RECORD-COUNT
A read-only 4 byte binary field that contains the number of logical input operations performed.

FILE-STATUS
This is a read only field that contains the status of the most recent I/O operation. FILE-STATUS is available when STATUS is coded on the I/O statement.

Note: FILE-STATUS is always available in the generated COBOL program.

The Easytrieve status codes are:
0000 Operation is successful
0004 End Of File Reached (EOF)
0008 Record with a duplicate alternate key exists
0012 Duplicate key
0016 Record not found
0020 File is Locked (work stations only)
0024 Logical or physical I/O error

For information about converting COBOL status codes to Easytrieve equivalent codes, refer to the IOCODE=EASYT option on page 123. FILE-STATUS customization applies only when running with IOCODE=NATIVE.

In general, testing for ZERO, NOT ZERO or EOF is sufficient. Testing for any other specific values must be adjusted manually in Easytrieve Source or the translated COBOL programs. The COBOL values can be found in the COBOL Programmer Reference Manual (FILE-STATUS information).

FILE-STATUS code in the generated COBOL program is a two byte alphanumeric field while in Easytrieve it is a full word numeric field. Instructions in Easytrieve Program that assign FILE-STATUS to a numeric field are flagged as errors.

When testing for a value other than zero, the value must be a two digit constant (literal) enclosed in quotes. For convenience, here are some more frequently checked COBOL status codes:

00 Operation is successful
02 Record with a duplicate alternate key exists (Read)
04 Wrong Length Record
10 End Of File Reached (EOF)
22 Duplicate Key (Write)
23 Record not found
30 Permanent I/O Error
34 Permanent I/O error, file is full (out of space)
39 Incompatible File DCB / Organization
96 DD Statement is missing in JCL (VSAM only)

Report fields (available only in report exits)

LINE-COUNT
A two byte binary field that contains the number of lines printed on the current page.

LINE-NUMBER
A two byte field that contains the number of the line being printed within the line group.

Note: This counter is not supported by Migration Utility.

PAGE-COUNT
A two byte binary field that contains the number of the page being printed

TALLY
A ten byte packed decimal field that contains the number of detail records in a control break

LEVEL
Indicates the control break level. This field is available on the "BEFORE-BREAK" and "AFTER-BREAK" report exits only.

BREAK-LEVEL
Indicates the break level number

&field BREAK
Tests control break on &field field, where &field is ″FINAL″ or a CONTROL field name.

WS-PENGI-DATE-9
A 6-digit date (SYSDATE) as obtained by a COBOL ACCEPT statement without insert characters. This is a numeric field and can be used in computations. The format is YYMMDD.

WS-PENGI-DATE-LONG-9
An 8-digit date (SYSDATE-LONG) as obtained by a COBOL ACCEPT statement without insert characters. This is a numeric field and can be used in computations. The format is CCYYMMDD.

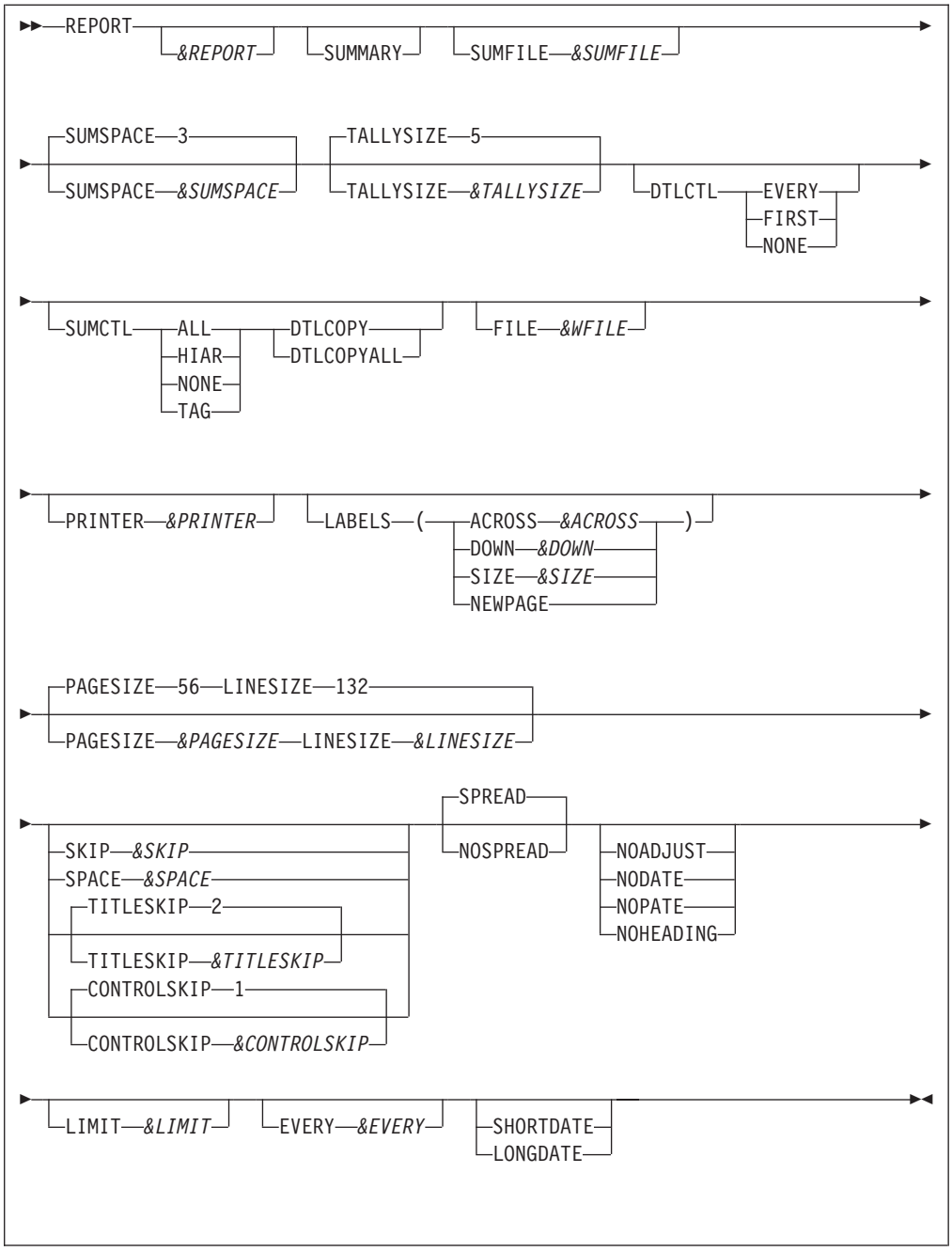
&FILE:KEY
The file key of RRN and PDS/PDSE files can be accessed by coding the file name as a qualifier to the KEY field. For example: FILEIN:KEY.

Easytrieve reserved keywords

Keyword	Description
EOF	Used to test end of file mark. It can optionally be followed by the file name the test applies to (preferred way of coding). If file is not supplied, the outcome of the last file accessed in sequential mode is used.
NUMERIC	Used to test data fields for numeric contents
MATCHED	Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 38 for more information.
DUPLICATE	Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 38 for more information.
FIRST-DUP	Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 38 for more information.
LAST-DUP	Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 38 for more information.
SPACE, SPACES	Used to test a field for all spaces or assign a field to all spaces
ZERO, ZEROS, ZEROES	Used to test a field for all zeros or move all zeros to a field
LOW-VALUES	Used to test a field for all binary zeros or move all binary zeros to a field
HIGH-VALUES	Used to test a field for all hex ″FF″ or move all hex ″FF″ to a field

REPORT statement

The REPORT statement defines the type and characteristics of a report. Multiple reports can be specified per single JOB activity section. REPORT statement with its parameters is placed at the end of each JOB activity section. It also must be followed by the SEQUENCE, CONTROL, TITLE, HEADING and LINE statements as described on the pages that follow.



Parameters

&REPORT

Report name. Easytrieve allows an up to 128 character report name. Because of COBOL restrictions, Migration Utility assigns its own internal name for each declared report. However, you reference the name as declared. The internal naming conventions are REPORTNN, where NN is report sequence number relative to zero.

SUMMARY

Prints a summary report by minor control break. The detail report is not printed.

&SUMFILE
A one to eight character Optional Summary File name for recording Control Break field values and summary totals. All data is as of minor control break.

&SUMSPACE
The number of digits to be added to the size of the summary field buckets. This is necessary to prevent overflow on accumulated values.

&TALLYSIZE
The size of the TALLY field. Valid values are 1 to 18 (digits).

DTLCTL
Indicates the printing method of control fields on detail line. Possible values are:
EVERY
Prints value of all control fields on every detail line.
FIRST Prints value of all control fields on the first detail line of each page, and on the first detail line after each control break. Printing of control field values is inhibited on all other detail lines.
NONE
Inhibits printing of control fields on every detail line

SUMCLT
Indicates printing method of control fields on total lines. Possible values are:
ALL Prints control field values on every total line
HIAR
Prints control field values in the hierarchical fashion on the total lines. Only values of control fields on the same hierarchical level, or higher than the breaking control field, are printed on the related total line.
NONE
Inhibits printing of control fields on total lines.

TAG Prints *&FIELD* TOTAL annotation to the left of the totals. Keep in mind that there must be enough space, on the left, for the totals literal. *&FIELD* is the field name that caused the break.

DTLCOPY
Prints detail information on minor level total lines.

DTLCOPYALL
Prints detail information on all control breaks. (This option is not supported by Migration Utility)

&WFILE
Work file DDname. This statement is ignored by Migration Utility.

&PRINTER
Printer file DDname. Normally Easytrieve directs all reports to the SYSPRINT. This statement lets you redirect output to a designated file. Make sure you define the file in the Library Section.

LABELS
Identifies mailing label printing. Possible values are:
&ACROSS
A number indicating the number of labels printed across the page
&DOWN
Specifies the number of print lines for each label

&LSIZE
Specifies the horizontal length of each label

NEWPAGE
Forces top of page (channel 1) for first label line

The NOHEADING and NOADJUST options are automatically activated for LABELS type of reports. You cannot use TITLE, HEADING and SUMMARY when you print labels.

&PAGESIZE
The number of lines per logical printed page. It can be 1 to 32767 and it must be at least as large as the sum of:
Number of the TITLE Lines
Number of TITLESKIP
Number of HEADING lines plus 1
Number of LINE statements
Number of lines of SKIP

&LINESIZE
The length of the printed line from 1 to 32767. One extra character is added, by the compiler, for print carriage control.

&SKIP The number of blank lines to insert between line groups, that is, before the last printed LINE NN and the first LINE 01. This statement is ignored by Migration Utility.

&SPACE
The default number of spaces (blank characters) between print fields. Note that the amount of space that each field occupies is the field (edited) length or the field Title, whichever is longer. The SPREAD parameter overrides this option.

&TITLESKIP
The number of blank lines between the last TITLE line and the first HEADING line. The default is 2 lines.

&CTLSKIP
The number of blank lines between the last total line and the first detail line (detail line post totals).

SPREAD
Insert maximum number of spaces between fields (columns).

NOSPREAD
Deactivates the SPREAD option.

NOADJUST
Deactivates automatic centering of report TITLES and LINES. When specified, all printed lines are left justified on the page. NOADJUST and SPREAD are mutually exclusive.

NODATE
Inhibits printing of System Date (CPU date) on the first title line.

NOPAGE
Inhibits printing of the Page Number (PAGE nnnnn) on the first title line.

NOHEADING
Inhibits printing of the column (field) headings. If this option is not specified, column headings are automatically printed.

&LIMIT
The maximum number of lines to print. This is a development option. It is ignored by Migration Utility.

&EVERY
Every *&EVERY*th line is to be printed. This is a development option. It is ignored by Migration Utility.

SHORTDATE
The SYSDATE date format is to be used.

LONGDATE
The SYSDATE-LONG date format is to be used.

SEQUENCE statement

The SEQUENCE statement optionally defines the order of a report. The SEQUENCE must be coded Immediately following the REPORT statement. When SEQUENCE is coded, REPORT data is sorted in the specified order before printing.

Parameters

&FIELD
One or more field names to sort by. A maximum of 16 fields can be sorted. The fields are sorted in major to minor order as listed.

D The field is sorted in descending order. If not specified, the field is sorted in ascending order.

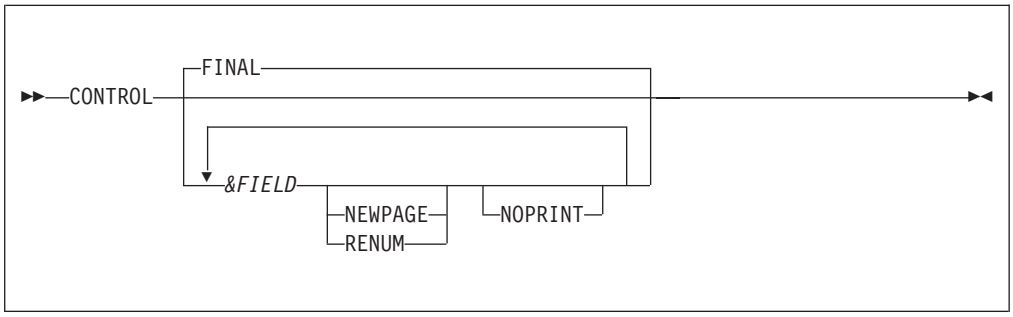
The SEQUENCE statement causes sorting of the report items before printing. The sort adds a substantial amount of processing overhead, therefore the SEQUENCE should be used with care. SEQUENCE is not needed if your input file is in the same order as your reports.

SEQUENCE fields do not have to be a part of the printed report.

CONTROL statement

The CONTROL statement identifies the fields used for control breaks. That is, a total line is printed for each field identified by the CONTROL statement.

The fields are listed in major to minor sequence. The first listed field is the major control break and the last one the most minor control break. The final total can be printed by specifying the ″FINAL″ keyword as the first entry in the CONTROL list.



Parameters

FINAL
The Final totals are to be printed at End-of-Report. If omitted, Final totals are implied.

&FIELD
Control break fields in major to minor order. You can specify up to 16 fields.

NEWPAGE
Forces new page post control break totals for the specified field.

RENUM
The same as NEWPAGE except it resets page counter to 1 following the control break.

NOPRINT
Suppresses printing of the total line for the specified field. Note that all processing to accommodate the total line is performed, but the line is not printed.

A maximum of 16 control breaks can be specified. The FINAL break is implied when no control fields are supplied.

A break level number is assigned to each control break field, with most minor break assigned to level 1, the one before it to 2, and so on. Final control break has the highest level number of N+1, where N is the number of fields specified.

A Level can be tested via the LEVEL keyword in the BEFORE-BREAK and AFTER-BREAK report exits. (See “Report exits” on page 78 for details).

All fields are compared according to the field type. For example, packed numeric fields are compared as packed decimals with sign.

SUM statement

The SUM statement explicitly names the fields to be totaled for a report with control breaks.

Parameter

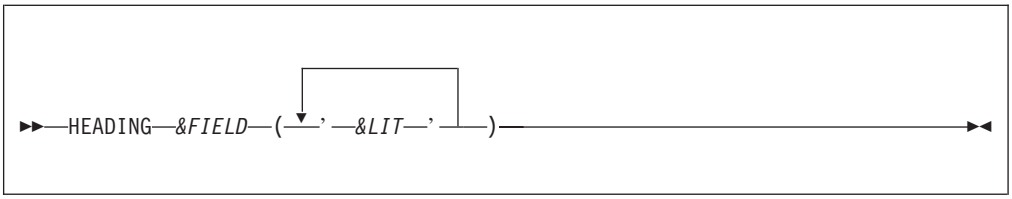
&FIELD
A list of one or more fields to be summed. The fields must be quantitative contained in an active file or working storage.

Normally, all quantitative fields specified on LINE statements are totaled. The SUM overrides the automatic summing by forcing fields specified on a SUM statement to be totaled only.

Note: The quantitative fields are all numeric fields defined with decimal places. One can force fields that do not have any decimal places to be treated as quantitative fields by coding zero for the number of decimals in the field definition.

HEADING statement

The HEADING statement optionally defines an alternate heading for a print field. The HEADING statement must be coded following the REPORT statement but before the LINE statements of a report definition. Thus, the HEADING overrides the original heading coded for field definition.



Parameter

FIELD Field name for which the heading is to be used

&LIT A heading literal. Can be up to 128 characters long. Migration Utility allows literals up to 58 characters long. Literals are stacked vertically over the print field or column.

Example

Here are various report headings:

```
 FILE FILEIN1 (80)                                  |
    CUST-NAME      01  15 A HEADING ('NAME')       |
    CUST-ADDRESS1  16  15 A HEADING ('ADDRESS1')  | Library Section
    CUST-ADDRESS2  32  15 A HEADING ('ADDRESS2')  |
    CUST-ADDRESS3  48  15 A HEADING ('ADDRESS3')  |
```

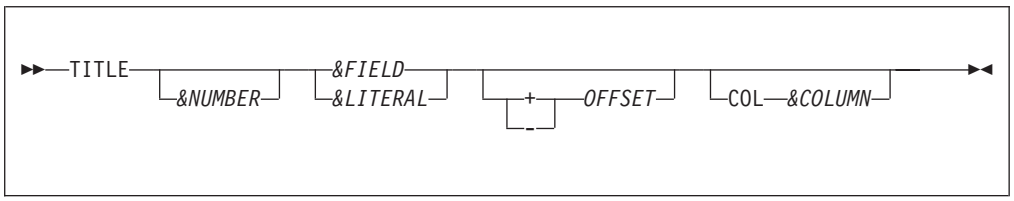
```
JOB INPUT FILEIN1
   PRINT REPORT1                                    |
                                                    |
REPORT REPORT1 LINESIZE 080                         |
HEADING  CUST-NAME ('CUSTOMER' 'NAME')              |
HEADING  CUST-ADDRESS2 ('CUST' 'ADDRESS' 'TWO')     |
TITLE 01 'NAME-ADDRESS REPORT EXAMPLE'              | Activity Section
LINE  01 CUST-NAME      +                           |
         CUST-ADDRESS1  +                           |
         CUST-ADDRESS2  +                           |
         CUST-ADDRESS3                              |
```

The program produces the following report (where Xs represent real data):

```
05/30/95                 NAME-ADDRESS REPORT EXAMPLE                  PAGE    1

            CUSTOMER                          CUST
              NAME          ADDRESS1         ADDRESS         ADDRESS3
                                               TWO

          XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
          XXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
```

TITLE statement

The TITLE statement defines report titles (lines printed on the top of each page). One or more TITLE statements can be specified. Each TITLE statement defines a single title line. The TITLE statements must be placed following the REPORT and CONTROL statement and before the first LINE statement.



Parameters

&NUMBER
Title sequence number. Specifies the position of the title in the title area. Valid numbers are 1 to 99. The numbers must be in ascending sequence. The first title number must be 1 or unspecified.

Note: Migration Utility ignores the sequence numbers.

&FIELD
A field name to be printed on the title

&LITERAL
A character string (literal). An alphanumeric literal must be enclosed in quotes.

OFFSET
The space adjustment parameters modify the normal spacing between title items. + or - indicates the direction in which the SPACE statement is applied.

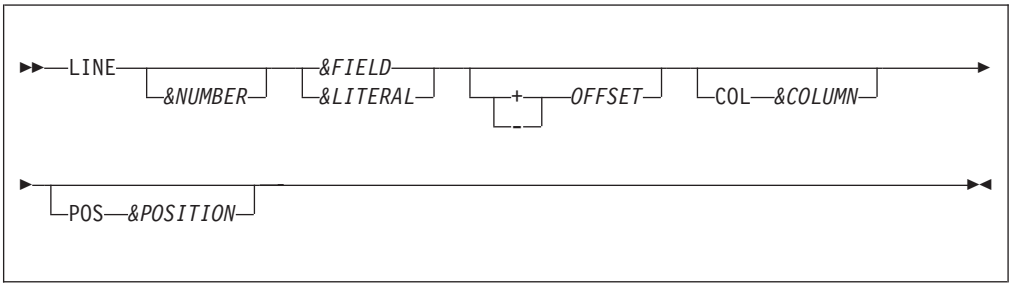
&COLUMN
The print column number where the next title item is to be placed. The number can be 1 to nn, but it cannot force the next title beyond the end of the title LINESIZE.

The system date and the current page number are automatically inserted on the first TITLE line. You can inhibit the date and page printing by coding NODATE and NOPAGE options on the REPORT statement.

Each TITLE line is centered within the title area of the report by default. You can inhibit centering by specifying NOADJUST on the REPORT statement.

LINE statement

The LINE statement defines report detail lines. One or more LINE statements can be specified. Each LINE statement defines a single report line. The LINE statements must be placed following the TITLE statements.



Parameters

&NUMBER
Line sequence number. Specifies the position of the line in the line group. Valid numbers are 1 to 99. The numbers must be in ascending sequence. The first line number must be 1 or unspecified.

Note: Migration Utility ignores the sequence numbers.

&FIELD
A field name to be printed on the line

&LITERAL
A character string (literal). An alphanumeric literal must be enclosed in quotes.

OFFSET
The space adjustment parameters modify the normal spacing between line items. + or - indicates the direction in which the SPACE statement is applied.

&COLUMN
The print column number where the next line item is to be placed. The number can be 1 to nn, but it cannot force the next line beyond the end of the line LINESIZE.

&POSITION
The position of line items on line 2 through 99 in respect to the items on line 1. The position corresponds to the line item number of line 1 under which the line item is placed. In simple terms, this parameter allows one to align items of line 2 through line 99 with items on line 1.

Any quantitative fields listed on the LINE statements are automatically totaled on each summary line for reports that contain CONTROL statements (control breaks). The automatic totaling can be overridden by coding the SUM statement of the REPORT definition.

Report exits

Migration Utility supports Easytrieve report exits as described in this section. Report exits are always placed after the REPORT statement (the last LINE statement). Each exit is declared in a form of a PROC and terminated by a PROC-END scope terminator.

Exits are recognized by their standard PROC name as follows:

REPORT-INPUT. PROC
Report input exit. This procedure is entered before entering report logic. It can be used to massage the input data before it is acquired by the print logic.

BEFORE-LINE. PROC
Before LINE printing exit. This exit is entered before each line is printed. It enables the user to print a literal string before the detail line. The content of the print fields cannot be changed.

AFTER-LINE. PROC
After line exit. This exit is entered after printing of each LINE. It is usually used to print a literal string after a detail line on a report.

BEFORE-BREAK. PROC
Before Break exit. This exit is entered before control break occurs. It can be used to calculate percentages and average totals that must be calculated immediately before printing. The exit is entered for each control break specified by the CONTROL statement.

The value of LEVEL system variable can be used to determine which control break is being processed. The value of 1 indicates the most minor break, 2 the one before it, etc. The FINAL break contains the value of N+1, where N is the number of fields specified on the CONTROL statement.

TALLY system variable contains the number of records in a particular control group.

BREAK-LEVEL system variable contains the field name that caused the break.

Note: If NOPRINT is specified on a CONTROL statement, the BEFORE-BREAK exit is still invoked.

AFTER-BREAK. PROC
After break exit. This exit is entered after printing of the last total line of each control break. It can be used to produce special annotations on control breaks. The exit is entered for each control break specified by the CONTROL statement.

The value of LEVEL system variable can be used to determine which control break is being processed. The value of 1 indicates the most minor break, 2 the one before it, etc. The FINAL break contains the value of N+1, where N is the number of fields specified on the CONTROL statement.

TALLY system variable contains the number of records in a particular control group.

BREAK-LEVEL system variable contains the field name that caused the break.

Note: If NOPRINT is specified on a CONTROL statement, the BEFORE-BREAK exit is still invoked.

ENDPAGE. PROC
End of Page exit. This exit is entered when end of page is reached. It can be used to print page foot information such as special annotations or totals.

Easytrieve allows the ENDPAGE exit for all types of reports. However, Migration Utility allows the ENDPAGE exit for all types of reports, except when printing LABELS. This may require you to relocate the ENDPAGE logic to the AFTER-BREAK exit.

TERMINATION. PROC
Termination exit. This exit is entered at the end of the report. This exit can be used to print report footing, including control totals and distribution information.

Each exit must be terminated by an END-PROC scope terminator. The exits are optional, thus, only those exits that are needed are coded.

Any quantitative fields referenced in the BEFORE-BREAK and AFTER-BREAK exits contain the total value accumulated for the specific break level. These values can be used in calculations but cannot be altered. The rules vary with each Easytrieve version, therefore pay extra attention to the final outcome. Migration Utility lets you use total values but it does not alter their contents.

Native COBOL support

Migration Utility (Translator) allows the programmer to embed COBOL code in the Easytrieve Plus program between %COBOL and %END macros (statements).

All code is punched out unchanged. Procedure statements are placed in the PROCEDURE DIVISION, and Working Storage fields in the WORKING-STORAGE SECTION.

In addition, the following COBOL instructions are supported in native Easytrieve Plus syntax:
- STRING (fully supported except for ON OVERFLOW option)
- INITIALIZE
- INSPECT (see also "INSPECT VREPLACING statement" on page 83

You can code any of the above COBOL instructions in a program just as you would for any other Easytrieve Plus instruction. The %COBOL is not needed and the restrictions described below do not apply. Refer to COBOL-II or a later version of the COBOL Reference manual for the coding rules. Other COBOL instructions can be coded according to the %COBOL rules below.

Examples:

```
STRING FIELDA ',' FIELDB INTO FIELDC

STRINGC-POINTER = 1
STRING FIELDA DELIMITED BY SIZE  +
       ','    DELIMITED BY SIZE  +
       FIELDB DELIMITED BY SIZE  +
              INTO FIELDC WITH POINTER STRINGC-POINTER

INITIALIZE FILEIN1

INSPECT FILEIN1 REPLACING ALL '*' BY SPACES

INSPECT FILEIN1 REPLACING ALL '*' BY SPACES BEFORE INITIAL '?'
```

```
INSPECT FILEIN1 REPLACING ALL '*' BY SPACES AFTER INITIAL 'A'

INSPECT FIELDC CONVERTING '4' TO 'X'

INSPECT FIELDC CONVERTING '4' TO 'X' AFTER INITIAL 'A'

WCOUNT = 0
INSPECT FIELDC TALLYING WCOUNT FOR CHARACTERS

WCOUNT = 0
INSPECT FIELDC TALLYING WCOUNT FOR CHARACTERS BEFORE 'A'

WCOUNT = 0
INSPECT FIELDC TALLYING WCOUNT FOR ALL 'AA' BEFORE 'A'

WCOUNT = 0
INSPECT FIELDC TALLYING WCOUNT FOR LEADING SPACES BEFORE 'A'
```

Benefits

COBOL provides for programming instructions not supported by Easytrieve Plus, such as STRING and UNSTRING. In addition, PEngiBAT Functions and Macros can be used where appropriate for optimum productivity.

%COBOL facility should be used by those users who use Migration Utility as a Reports Generator, or to combat unsupported Easytrieve statements when translating Easytrieve Plus programs (such as IMS and IDMS).

Restrictions

COBOL support should be used with caution when referencing Easytrieve Plus field names (those fields defined using the Easytrieve Plus syntax). This is because Migration Utility sometimes changes the field names to comply with COBOL rules, but COBOL code is passed on unchanged, thus causing or creating undefined field names.

The best way to combat this situation is to define working storage fields needed for COBOL logic using COBOL syntax. Information can be moved into COBOL defined fields from Easytrieve Defined fields at the beginning of the routine, and from COBOL defined fields into Easytrieve Defined fields before exiting.

Coding conventions

```
%COBOL &syntax
     .
     .
     .
%END
```

Where:

- &syntax FULL = COBOL code is according to Standard COBOL rules:
  - Area A starts in position 8
  - Area B starts in position 12
  - Comments are all lines that contain a ″*″ in position 7
- NONE = COBOL Code is according the following rules:
  - Area A starts in position 2
  - Area B starts in position 6
  - Comments are all lines that contain a “#” in position 1

- All statements are transformed by Migration Utility to comply with Standard COBOL rules. Data on each line must be limited to 64 characters to prevent spanning beyond CC 72.
- The default is NONE

Example

This example shows the use of %COBOL without COBOL syntax rules.

```
FILE FILEIN1
INPUT-TEXT   1 80 A

%COBOL
 WORKING-STORAGE SECTION.
#****************************************************************
# THIS IS %COBOL WORK AREA IMBEDDED IN Easytrieve.
*
#****************************************************************
 01  WORK-AREA.
     02 FIELD-A    PIC 9(05).
     02 FILLER     PIC X(05).
     02 FIELD-B    PIC 9(02).

 01  WORK-AREA2.
     02 RESULT-1   PIC ZZ,ZZZ,ZZZ.99-.
     02 RESULT-2   PIC ZZ,ZZZ,ZZZ.99-.

%END

JOB INPUT FILEIN1

DISPLAY INPUT-TEXT

%COBOL
#****************************************************************
# THIS IS %COBOL CODE IMBEDDED IN Easytrieve
*
#****************************************************************
 PROCEDURE DIVISION.
     MOVE INPUT-TEXT TO WORK-AREA
     COMPUTE RESULT-1 = (FIELD-A ** FIELD-B)
     COMPUTE RESULT-2 = (FIELD-A ** FIELD-B) / 12

    DISPLAY 'RESULT-1: ' RESULT-1
     DISPLAY 'RESULT-2: ' RESULT-2

# THE END OF %COBOL TEST CODE THAT IS IMBEDDED IN Easytrieve
*
%END

DISPLAY 'END-OF-JOB'
STOP
```

This example show the use of %COBOL with FULL COBOL syntax rules.

```
FILE FILEIN1
INPUT-TEXT   1 80 A

%COBOL FULL
       WORKING-STORAGE SECTION.

      ****************************************************************
      * THIS IS %COBOL WORK AREA IMBEDDED IN Easytrieve.
      *
      ****************************************************************
       01  WORK-AREA.
```

```
           02 FIELD-A    PIC 9(05).
           02 FILLER     PIC X(05).
           02 FIELD-B    PIC 9(02).

       01  WORK-AREA2.
           02 RESULT-1   PIC ZZ,ZZZ,ZZZ.99-.
           02 RESULT-2   PIC ZZ,ZZZ,ZZZ.99-.

%END
JOB INPUT FILEIN1
DISPLAY INPUT-TEXT
%COBOL FULL
      ****************************************************************
      * THIS IS %COBOL CODE IMBEDDED IN Easytrieve
      *
      ****************************************************************
       PROCEDURE DIVISION.
           MOVE INPUT-TEXT TO WORK-AREA
           COMPUTE RESULT-1 = (FIELD-A ** FIELD-B)
           COMPUTE RESULT-2 = (FIELD-A ** FIELD-B) / 12

           DISPLAY 'RESULT-1: ' RESULT-1
           DISPLAY 'RESULT-2: ' RESULT-2
         * THE END OF %COBOL TEST CODE THAT IS IMBEDDED IN Easytrieve    *
%END

DISPLAY 'END-OF-JOB'
STOP
```

Support for COBOL and PEngi Functions in ASSIGN statement

Migration Utility allows for COBOL and PEngi (CCL1) functions in Easytrieve Plus ASSIGN statements.

To do so, code FUNCTION &FUNAME(&ARG1 &ARG2...) for the second argument.

COBOL supports numerous functions. For information about COBOL functions, refer to the COBOL reference manual.

The use of PEngi (CCL1) functions is beyond the scope of this document.

Generating rules

When the target field in the assign is a numeric field, Migration Utility generates a COBOL COMPUTE statement.

When the target field in the assign is an alphanumeric field, Migration Utility generates a COBOL MOVE statement.

Example:

```
WS-RANDOM-NUMBER = FUNCTION RANDOM (SEED)
```

INSPECT VREPLACING statement

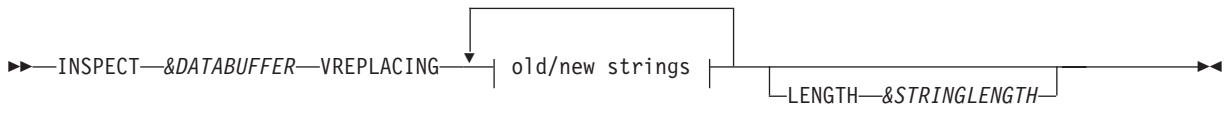
The INSPECT VREPLACING statement replaces one or more specified character strings in a data string with replacement character strings.

The INSPECT VREPLACING statement:

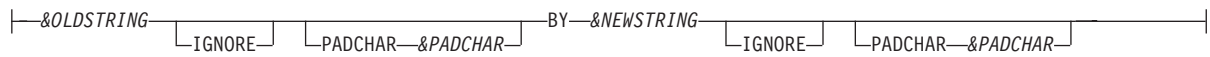
- Replaces all occurrences of specified characters or character strings in a data string with specified replacement characters or character strings.
- Deletes all occurrences of specific characters or character strings in a data string.
- Calculates the length of the modified data string and places the length in a user-specified field (if supplied).

INSPECT VREPLACING is a modified version of the COBOL statement, INSPECT REPLACING. However, unlike INSPECT REPLACING (which can only be used when the lengths of the old and new strings are the same), INSPECT VREPLACING allows you to replace strings of the same or different lengths.

Format 1



old/new strings:



Parameters

&DATABUFFER

A buffer that initially holds the data string to be inspected and which, after replacement, holds the new string with the replaced items.

&DATABUFFER must be defined in working storage as an alphanumeric data item, large enough to hold the resulting changed string. The maximum buffer length is 32,767.

To determine the length of the string to be searched, Migration Utility uses either the length you supply (*&STRING-LENGTH*) or, if not supplied, the length of *&DATABUFFER*.

&OLDSTRING

The data string to be replaced. Must be a display numeric or an

alphanumeric field, or a literal. The maximum field length is 256 bytes. The maximum literal length is 160 characters.

&NEWSTRING
The data string that is to replace *&OLDSTRING*. Must be a display numeric or an alphanumeric field, or a literal. The maximum field length is 256 bytes. The maximum literal length is 160 characters.

&PADCHAR
The pad character used to pad the trailing part of the field.

If specified for *&OLDSTRING*, the length used in the comparison excludes these trailing pad characters; otherwise, Migration Utility use the length of *&OLDSTRING* in the comparison.

If specified for *&NEWSTRING*, the length used in the replacement excludes these trailing pad characters; otherwise, Migration Utility use the length of *&NEWSTRING* in the replacement.

&STRINGLENGTH
A binary, display, or packed-decimal field initially containing the length of the string (in *&DATABUFFER*) to be inspected.

After replacement, Migration Utility sets *&STRINGLENGTH* (if supplied) to the length of the new string.

Replacement rules:

- The replacement process uses two internal work buffers, one to hold the input data string, the other to hold the modified output data string.
- The contents of *&DATABUFFER* are moved to the internal input buffer.
- The contents of the internal input buffer are inspected (by indexing through the buffer) for a match with the contents of *&OLDSTRING*.

  The length used in the comparison is the derived length of *&OLDSTRING*.
- If a match with *&OLDSTRING* is found in the internal input buffer, that section of the input data string is replaced by the contents of *&NEWSTRING* and the result placed in the internal output buffer.

  The length used in the replacement is the derived length of *&NEWSTRING*. If the derived length of *&NEWSTRING* is zero, the matched section of the input data string is deleted in the internal output buffer.

  The index for the internal input buffer is incremented by the length of *&OLDSTRING*.

  The index for the internal output buffer is incremented by the length of *&NEWSTRING*.
- If no match is found with *&OLDSTRING* in the internal input buffer, one character is moved from the internal input buffer to the internal output buffer unchanged.

  The index for the internal input buffer is incremented by 1.

  The index for the internal output buffer is incremented by 1.
- The contents of the internal input buffer are again inspected for a match with the contents of *&OLDSTRING* and the above process repeated until the end of the internal input buffer is reached.
- The process is repeated for each pair of replacement strings specified.
- On completion, the contents of the internal output buffer are moved to *&DATABUFFER*, and *&STRINGLENGTH* (if specified) is set to the length of the updated string in *&DATABUFFER*.
- If the length of the updated string exceeds:
  – the length of *&DATABUFFER*, or

– 32,767 characters

the updated string is truncated.

If *&STRINGLENGTH* is greater than the length of *&DATABUFFER*, overflow has occurred.

Example 1:

(Assume that the fields prefixed by "I" are in a record named HTMLIN.)

```
WBUFFER         W  252 A
VAR0            W   10 A VALUE '::INSERT::'
NUL0            W    1 A VALUE X'00'
VAR1            W   10 A VALUE '@C.'
VAR2            W   10 A VALUE '@B.'
VAR3            W   10 A VALUE '@W.'
VAR4            W   10 A VALUE '@R.'
VAR5            W   10 A VALUE '@A.'

COMPANY         W   10 A
BRANCH          W   10 A
WAGE            W   10 A
RATE            W   10 A
BONUS           W   15 A
BONUS1          W    5 P 2
WLENGTH         W    2 B

MOVE ICOMPANY TO COMPANY
MOVE IBRANCH  TO BRANCH
MOVE IWAGE    TO WAGE MASK 'ZZZ,ZZZ.99'
MOVE IRATE    TO RATE MASK 'ZZ.999'
BONUS1 = IWAGE * (IRATE / 100)
MOVE BONUS1 TO BONUS MASK 'ZZZ,ZZZ.99'

MOVE HTMLIN TO WBUFFER MASK 'X(80)' LENGTH WLENGTH
```

WLENGTH now contains the length of the input data.

```
INSPECT WBUFFER VREPLACING   +
   VAR0 PADCHAR SPACE BY NUL0    PADCHAR X'00'  +
   VAR1 PADCHAR SPACE BY COMPANY PADCHAR SPACE  +
   VAR2 PADCHAR SPACE BY BRANCH  PADCHAR SPACE  +
   VAR3 PADCHAR SPACE BY WAGE    PADCHAR SPACE  +
   VAR4 PADCHAR SPACE BY RATE    PADCHAR SPACE  +
   VAR5 PADCHAR SPACE BY BONUS   PADCHAR SPACE  +
 LENGTH WLENGTH
```

WLENGTH now contains the length of the replaced data, and WBUFFER contains the new string.

Example 2:

If the field lengths for VAR0...VAR5 in Example 1 were defined with exact VALUE length, you could write the INSPECT VREPLACING statement as:

```
INSPECT WBUFFER VREPLACING   +
   VAR0 BY  NUL0    PADCHAR X'00'  +
   VAR1 BY  COMPANY PADCHAR SPACE  +
   VAR2 BY  BRANCH  PADCHAR SPACE  +
   VAR3 BY  WAGE    PADCHAR SPACE  +
   VAR4 BY  RATE    PADCHAR SPACE  +
   VAR5 BY  BONUS   PADCHAR SPACE  +
 LENGTH WLENGTH
```

Example 3:

If the field lengths for *all* the fields in Example 1 were defined with exact VALUE length, you could write the INSPECT VREPLACING statement as:

```
INSPECT WBUFFER VREPLACING   +
           VAR0 BY  NUL0    PADCHAR X'00'  +
           VAR1 BY  COMPANY  +
           VAR2 BY  BRANCH   +
           VAR3 BY  WAGE     +
           VAR4 BY  RATE     +
           VAR5 BY  BONUS    +
 LENGTH WLENGTH
```

Note that the pad character for the NUL0 field was retained to remove "::INSERT:" (the value in the VAR0) from the buffer.

Chapter 5. SQL/DB2 support

Easytrieve Plus supports two methods for accessing database:

- Using native SQL statements to manage cursors
- Using Easytrieve method to automatically manage cursors

Migration Utility supports both methods as described in this section.

Translating concepts

DB2/SQL column definitions can be automatically accessed from the SYSIBM.SYSCOLUMNS catalog. Refer to "Activating Call Attachment Facility (CAF) for DB2 users" on page 115.

If CAF is not available, then a DCLINCL must be supplied for each accessed table.

The DECLGENs are included via the ″SQL DCLINCL &NAME″ Migration Utility statement. One statement is required for each SQL/DB2 table in use. These statements must be placed before SQL file definitions (preferably before the first valid Easytrieve Definition in the program but after the leading comments).

Note: The DECLGENs can also be included via the ″EASYTRAN:″ comment in your program to preserve Easytrieve Syntax compatibility. In this way, the same Easytrieve Source can be used as input to Easytrieve Plus and Migration Utility. Refer to the ″EASYTRAN:″ coding rules in this document.

Migration Utility generates an SQL INCLUDE or partial Column Definitions in the generated COBOL for each included DECLGEN in Easytrieve Plus Source. For details refer to the DECLGEN=FULL/PART and SQLPFIX=EZPARAMS option.

Example

This example shows a real DECLGEN of a DB2 table. COBOL users usually have similar DECLGENs available for use by COBOL programmers. Oracle or other Database Users should create a similar DECLGEN for each table to make translating possible.

```
************************************************************
* DCLGEN TABLE(CUST_TB)
*
*        LIBARARY(&SYS1.SFSYEZTS(DECLADDR)
*
*        ACTION(REPLACE)
*
************************************************************
     EXEC SQL DECLARE CUST_TB TABLE
     ( CUST_CO_NBR                DECIMAL(5, 0) NOT
     NULL,
       CUST_ID                    CHAR(9) NOT NULL,
       CUST_NUMBER                SMALLINT NOT NULL,
       CUST_ACCOUNT               DECIMAL(5, 0) NOT
       NULL,
       CUST_PRODUCT               CHAR(3) NOT NULL,
       CUST_METHOD                CHAR(23) NOT NULL,
       CUST-RELATION              CHAR(3) NOT NULL,
```

```
      CUST_PRRIM_IND            CHAR(1) NOT NULL,
    ) END-EXEC.

*************************************************************
* COBOL DECLARATION FOR CUST_TB.
*
*************************************************************
 01  CUST-TB.
     10 CUST-CO-NBR          PIC   S9(5)V COMP-3.
     10 CUST-ID              PIC   X(09).
     10 CUST-NUMBER          PIC   S9(04) COMP.
     10 CUST-ACCOUNT         PIC   S9(5)V COMP-3.
     10 CUST-PRODUCT         PIC   X(03).
     10 CUST-METHOD          PIC   X(23).
     10 CUST-RELATION        PIC   X(03).
     10 CUST-PRIM-IND        PIC   X(01).

*************************************************************
*  THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 8.   *
*************************************************************
```

Note 1: COBOL 01 level field name must match the table name it represents. Since the table names are usually coded with an underscore, the underscores should be changed to dashes to preserve COBOL field naming conventions.

Note 2: Migration Utility generates the COBOL field names with the prefix specified by the SQLPFIX= option. A different sequence number is attached to each new table to preserve uniqueness. Thus if SQLPFIX=(Q-) is specified, the prefix attached to the field names would be Q1-, Q2-, etc. It is important to recognize that the field names for holding table column information are hard generated in the COBOL code. The field names located in the original DECLGENs are not referenced in the generated code.

Note 3: Table names in Easytrieve programs can be coded as &owner.&table. Migration Utility searches DECLGENs for a table name coded in the Easytrieve program with the qualified &owner.&table first. If not found, then the search is conducted without the &owner qualifier. Your DECLGEN table name can be qualified or unqualified. To ensure a smooth translation, code table name in the DECLGEN without a qualifier, unless you need a specific owner in which case the &owner must also be coded in your Easytrieve program.

Native SQL statements

With minor adjustments to the Host Variables names, Migration Utility interprets native SQL statements exactly as written in the Easytrieve Program. The Host Variable names are adjusted to reflect the changes that take place during the translating process.

As per Easytrieve Plus rules, Migration Utility treats all Easytrieve statements that start with SQL keyword as the Native SQL statements. Using these Native SQL statements, the programmer can code fully SQL-compliant programs and have complete SQL cursor control.

Automatic cursor management

Easytrieve Plus can manage the SQL cursor in two ways:

- Easytrieve files defined as SQL files
- Automatic retrieval without a file

Migration Utility supports both methods as per Easytrieve Plus rules described in the paragraphs that follow.

Easytrieve file defined as an SQL file

SQL Files can be accessed:

- Via JOB INPUT &FILE for Automatic Input. In this case, a new row is automatically fetched or retrieved from the table into the file's data area. The method is ideal for users that do not have advanced knowledge of SQL, that is, users do not have any Cursor control.
- Via SQL-like I/O statements. The following I/O statements are available:
  - CLOSE
  - DELETE
  - FETCH
  - INSERT
  - SELECT
  - UPDATE

Automatic retrieval without a file

In this case, SQL must be coded on the JOB statement in place of a file name. A SELECT statement must be coded immediately after the JOB statement to specify the columns to be retrieved and the Host Variables to receive the data. Each time the JOB Activity is iterated, another row of data is fetched or retrieved.

Automatic retrieval functions in read-only mode.

SQL statements syntax rules

The following syntax must be observed when coding SQL statements in Easytrieve Programs:

- Operators must be separated by blanks.
- Standard Easytrieve Plus continuation conventions must be followed.
- Commas are considered when parsing and are not ignored.
- The period is used for qualifiers not to signify end-of-statement.
- The colon (:) identifies host variables, and is not a qualification separator
- SQL statement cannot be followed by another statement on the same line.

PARM statement parameters

The following Easytrieve Plus PARM statements set the SQL environment for the program:

For DB2:

SQLID
SSID
PLAN
LINK

PARM statement parameters

For SQL/DS™:
USERID
PREPNAME

The QUAL Migration Utility PARM statement supplements the generation of the BIND parameters. It provides a way of supplying a value for the DB2 BIND QUALIFIER. The QUAL parameter can be coded with the existing PARM parameters. For example:

```
PARM QUAL('SYS2') SSID ('TESTDB2') PLAN('TESTDB2P') LINK('TESTPROG'
```

The PARM statements information is extracted by Migration Utility and BIND parameters are generated for potential BIND. The BIND file is used as an input to the BIND step in JCEZDB2B JCL, or you can tailor it for custom use, for example, as an input to a separate BIND Job. Parameters are interpreted as follows (You can view SQLBIND macro source):

```
&SYSTEM  = SSID from the Easytrieve PARM statement
&PLAN    = PLAN from the Easytrieve PARM statement
&OWNER   = SQLID from the Easytrieve PARM statement
&QUAL    = QUAL from the Easytrieve PARM statement
&MEMBER  = LINK from the Easytrieve PARM statement
&SQLMODE = SQLMODE Option from EASYPARM/EASYTRAN
               BIND    = Do not use CAF to connect to DB2
               &PROGRAM = Program to use for connecting to DB2 via CAF
```

Library section for SQL processing

Before the SQL data can be accessed, you must define the fields to hold the columns to be retrieved from the database. These fields are referred to as the Host Variables.

For native SQL statements and Automatic Retrieval without a file, these fields are usually defined as Working Storage fields.

For SQL Files, fields are defined within the file (as if it were a regular file). The fields defined within the file correspond to the selected columns of the SQL table. The table columns are retrieved into the file fields.

SQL catalog INCLUDE facility

The SQL INCLUDE FROM &owner.&table statement names the SQL table or view from which column names and data types are to be included, and it defines the location at which the fields are to be generated.

The SQL INCLUDE statement must be coded in the library section of your Easytrieve Plus program and precede any other statements that access the included table, but must be coded after the SQL DCLINCL &NAME statement, if DCLINCL is provided.

Migration Utility utilizes SYSIBM.SYSCOLUMNS catalog whenever it encounters an “SQL INCLUDE FROM &owner.&table” in the Easytrieve Plus program, and a DCLINCL was not previously supplied. Refer to “Activating Call Attachment Facility (CAF) for DB2 users” on page 115.

If the &owner.&table exists in the catalog, column definitions are obtained from the catalog, and the field names are generated from the column names.

If the &owner.&table does not exist in the catalog, a DCLINCL must be supplied for the table. The field names are obtained from the COBOL definitions in the DECLGENs.

When to use SQL INCLUDE

SQL include is used to automatically define the necessary host variables into which DB2/SQL table information is fetched. SQL INCLUDE is not needed in every Easytrieve program that uses DB2. An alternative is to select/fetch column information into manually-defined working storage fields.

Processing nullable fields

Easytrieve supports the SQL nullable columns. Easytrieve determines if a column or field is nullable from the information extracted from the SQL catalog.

Migration Utility determines if a field is a nullable field from the DECLGEN.

When a column is declared as nullable, and NULLABLE is specified in SQL INCLUDE definition of an SQL File, a two byte null indicator (2 B 0) is automatically generated by Easytrieve and placed before the field name. Each retrieval places a negative value into the null flag for empty fields (fields that have no value assigned).

After the retrieval, you can use special processing statements:

IF NULL to determine if column/field contains a null value. MOVE NULL to set a column/field to a null value.

When using Native SQL or automatic input without a file, null indicator can be defined as two byte signed binary field in working storage (2 B 0). This indicator is then used in the INTO clause in the native or automatic SELECT statement.

SQL data types

Migration Utility accepts SQL data types as defined by the COBOL definitions in the included DECLGEN. Data Types are not checked for proper SQL syntax. However, SQL pre-processor does so.

SQL syntax checking

For Native SQL statements, Migration Utility does minimal syntax checking. With the exception of host variables, statements are passed to SQL pre-processor as coded. Host variables are potentially re-named and adjusted to avoid unresolved references.

For Easytrieve SQL look alike I/O statements, Migration Utility generates standard SQL for DB2.

System-defined fields

RECORD-COUNT
Reflects the number of rows returned (fetched or by automatic means)

RECORD-LENGTH
The sum of lengths of all fields within a file.

EOF processing

When the end of table is reached, either with automatic (JOB) or Fetch processing, the file is marked EOF (end of file). In automatic processing, execution stops and FINISH procedure (if present) is executed. In controlled processing you can test for file EOF (IF EOF &FILE) to determine an end of file condition.

Communication Area fields

Easytrieve automatically generates SQL Communication Area (SQLCA) fields if at least one SQL or SQL table statement is encountered in your program.

Migration Utility automatically generates an SQL INCLUDE for SQLCA in the generated COBOL source. An Easytrieve copybook of SQLCA is included in the distributed library. The SQLCA copybook is located in the SYS1.SFSYEZTC PDS.

Easytrieve Plus SQL files

To process data from an SQL table via Easytrieve SQL file method, you must code the following:

1. A file statement specifying one or more table names. If all columns defined in the file are subject to update, specify the UPDATE keyword on the FILE statement.

   Define one of more fields for the columns within the table(s) that you want to retrieve. The definitions can be defined using the DEFINE statement or by using the SQL INCLUDE statement. When SQL INCLUDE is used, field definitions are automatically generated from the SQL Catalog. Selective columns can be updated by coding UPDATE on the SQL INCLUDE and omitting the UPDATE on the file statement.
2. Code a SELECT statement that defines the result set for the cursor. If the SELECT statement is omitted, a default SELECT is generated automatically for all table columns. The SELECT statement, if coded, must be the first statement following the JOB statement. Coding your own SELECT gives you the choice of customizing the result set for the cursor.

Note: A SELECT statement for an SQL file is similar to opening the file. SELECT coded for a file that is already open first closes the file and the re-opens it based on the new SELECT.

Examples

This example shows automatic processing with SELECT:

```
SQL DCLINCL DCLCTXAB

FILE FILEIN1 SQL

 SQL INCLUDE               +
          (CUST_CO_NBR,    +
           CUST_ID,        +
           ACCT_CO_NBR,    +
           ACCT_PRDCT_CD)  +
     LOCATION *            +
     HEADING               +
     UPDATE                +
     NULLABLE              +
     FROM CUST_B_ACCT_TB
```

```
JOB INPUT FILEIN1
SELECT FROM  FILEIN1         +
       WHERE (CUST_ID = 315)
       .
```

Note: SQL DCLINCL is a required Migration Utility statement.

This example shows automatic processing without SELECT:

```
SQL DCLINCL DCLCTXAB

FILE FILEIN1 SQL
 SQL INCLUDE                +
          (CUST_CO_NBR,     +
           CUST_ID,         +
           ACCT_CO_NBR,     +
           ACCT_PRDCT_CD)   +
     LOCATION *             +
     HEADING                +
     UPDATE                 +
     NULLABLE               +
     FROM CUST_B_ACCT_TB

JOB INPUT FILEIN1
       .
       .
```

Note: SQL DCLINCL is a required Migration Utility statement.

Using DEFER with SELECT

Coding DEFER on the SQL FILE statement gives you an opportunity to code SELECT anywhere in the logic. SELECT does not have to be coded immediately after the JOB statement. For example, SELECT can be coded in the START procedure after the host variable values used in selection have been set.

Be careful. If the DEFER is not specified, and the SELECT is coded elsewhere (not immediately after the JOB statement), a default SELECT is generated in addition to the coded SELECT, thus causing duplication and performance problems.

Multiple tables

Easytrieve SQL files can be defined with multiple tables, that is, tables can be joined. Referencing a file that was defined with multiple tables results in a JOIN for all defined tables.

Example

```
FILE FILEIN (TABLE1, TABLE2)
```

Controlled processing

You can use the FETCH statement (with the SELECT and CLOSE) to retrieve the records from an SQL file. These statements can be coded within JOB activity with or without automatic input.

Controlled statements cannot be used in SORT or REPORT procedures.

Fetch cannot be used on automatic input file within the same JOB activity.
However, you can FETCH from a file other than the file subject to automatic input.

Automatic retrieval without a file

In this method, a special JOB and SELECT statements are coded to retrieve the data.

The retrieval without a file is a read-only method that usually retrieves data into working storage fields.

The method allows some selection techniques not available for cursors associated with SQL Files.

The following is required when processing an SQL table using this method:

1. One or more field definitions for the columns within the tables that you want to retrieve. The definitions can be coded using the DEFINE statement or SQL INCLUDE statement in working storage. Fields can be also defined within a file.
2. A JOB statement with the JOB INPUT SQL parameter. SQL denotes that the input does not involve an SQL File.
3. A non-file based SELECT that defines result set for the cursor. Only one non-file SELECT statement is allowed within a single JOB activity.

   This SELECT statement is different from the FILE based SELECT used with SQL files. It is more similar to the true SQL SELECT. For example, the tables to be accessed are named and more advanced functions can be performed such as UNIONs.

The SQLCODE is tested following each execution of the SELECT statement. The end of data condition results in the end-of-input processing with all amenities associated with it.

Example

This example shows selecting all rows from the USERTAB table (assume that DECLGEN name is USERTAB).

```
SQL DCLINCL USERTAB

DEFINE USER-NAME    W  20 A
DEFINE USER-DEPT    W   2 P 0
DEFINE USER-PHONE   W   3 P 0
DEFINE NULL-PHONE   W   2 B 0

JOB INPUT SQL
SELECT * FROM USERTAB   +
   INTO  :USER-NAME, :USER-DEPT, :USER-PHONE :NULL-PHONE
     .
     .
```

Note: SQL DCLINCL is a required Migration Utility statement.

Native SQL processing

Native SQL statements equivalent to those used in COBOL can be imbedded in the Easytrieve programs. Using these native SQL statements, the programmer can code fully compliant SQL program.

Migration Utility fully supports all SQL statements. With the exception of the host variables, the coded statements are punched out unchanged. Thus the user can

code a variety of SQL dialects. The host variable names are adjusted to prevent potential problems and conflicts with the naming conventions in COBOL.

The following processing requirements must be adhered to:

1. The SQL DECLARE &CURSOR CURSOR and SQL INCLUDE must be coded in the Library Definition. All other statements must be coded in the Activity Section.
2. The SQLCODE must be tested after each operation for successful completion. SQLWARN0 field should be tested whenever SQLCODE of zero is returned.

Coding native SQL requires an advanced knowledge of SQL statements and database.

Native SQL statements cannot be coded in the SORT and REPORT procedures.

The following native SQL statements are supported:

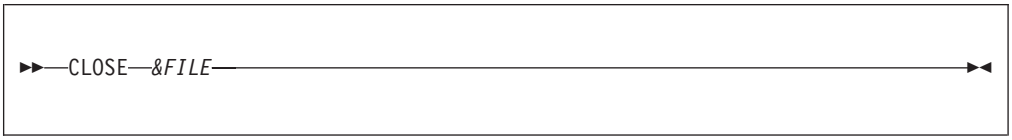
CLOSE
COMMIT
CONNECT
DECLARE
DELETE
FETCH
INSERT
OPEN
PUT
ROLLBACK
SET CURRENT SQLID
UPDATE

For further information and syntax rules of native SQL statements refer to the appropriate SQL reference manual.

Chapter 6. SQL File I/O statement reference

CLOSE statement

The CLOSE statement closes an SQL File.



Parameter

&FILE The file to be closed.

At the termination of each activity, all files opened during the activity are automatically closed. The CLOSE statement can be used to close the file before the activity terminates. The next I/O statement using the file re-opens it.

A file can also be closed and re-opened to create a new cursor.

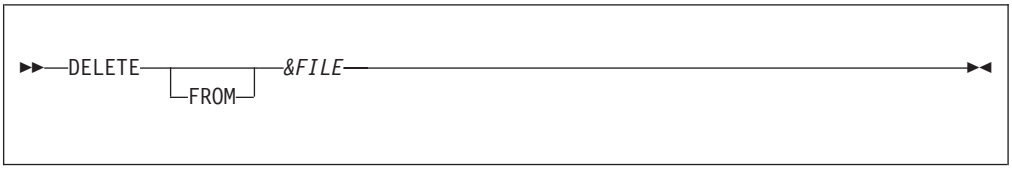
The CLOSE statement cannot be used to close a printer file or to close an automatic input/output file.

Example

```
CLOSE FILEIN
```

DELETE statement

The DELETE statement deletes a row from an SQL File.



Parameters

&FILE The file from which to delete

FROM
Is available for readability.

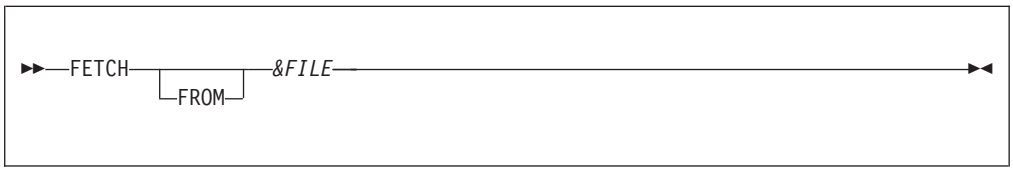
DELETE perform a DELETE WHERE OF cursor. The file must be defined with the UPDATE parameter.

Example

```
DELETE FILEIN
```

FETCH statement

The FETCH statement retrieves a row from an SQL File.



Parameters

&FILE The name of the SQL file.

FROM
Is available for readability.

The FETCH statement retrieves rows from the open cursor and places the data into the file's data area. If there is no cursor associated with the file, the cursor previously selected is re-opened. If no cursor was previously selected, then a default cursor for all fields FROM &table is opened.

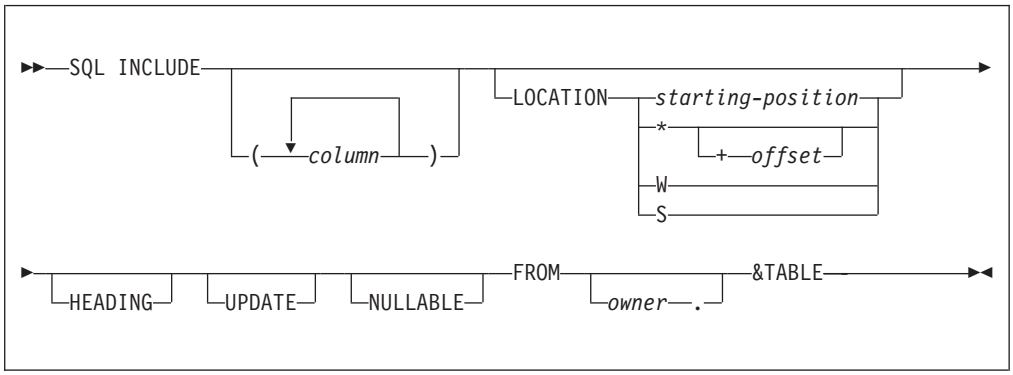
FETCH cannot be used in a SORT or REPORT procedure. The FETCH cannot reference an automatic input file in the same JOB activity.

Example

```
FETCH FILEIN
```

SQL INCLUDE statement

The SQL INCLUDE specifies the SQL table information to be used to generate field definitions. It names the table and gives the location where the field definitions are to be generated.



Parameters

column A list of columns to be included. The Easytrieve field names are generated for these columns. If no columns are specified, all columns from the table are included.

LOCATION
The location at which the field definitions are generated.
starting-position
The starting position relative to position one of the record or file.

***** Indicates that the field begins in the next available starting position.

offset The offset you want to add to the * position. There must be at least one blank between the * and the "+".

W, S Establishes working storage fields.

HEADING

This statement is not supported by Migration Utility. In Easytrieve Plus, it causes remarks in the DBMS system catalog to be used as HEADINGS.

UPDATE

The generated columns are updated. If UPDATE is coded on the FILE statement, all columns in the file are modifiable.

NULLABLE

Causes default indicator fields to be generated for columns that contain NULL. The indicator field is defined as a 2 B 0 field preceding the field being defined. If the column being defined is used as a host variable, then the default indicator is used as the null indicator unless overwritten by coding an indicator variable.

The indicator variable preceded the data portion of the field in storage. This field cannot be directly referenced. The IF NULL statement must be used.

owner 1 to 18-character alphanumeric qualifier

&TABLE

1 to 18-character alphanumeric name.

SQL INCLUDE must precede any other SQL statements and must be coded in the Library Section of the program.

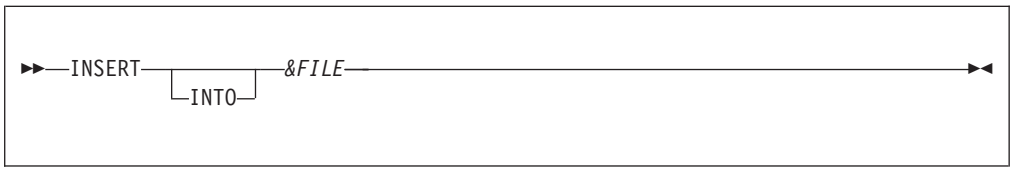
Example

```
SQL DCLINCL DCLCTXAB

FILE FILEIN1 SQL
 SQL INCLUDE               +
          (CUST_CO_NBR,    +
           CUST_ID,        +
           ACCT_CO_NBR,    +
           ACCT_PRDCT_CD)  +
     LOCATION *            +
     HEADING               +
     UPDATE                +
     NULLABLE              +
     FROM CUST_B_ACCT_TB

JOB INPUT FILEIN1
SELECT FROM  FILEIN1         +
       WHERE (CUST_ID = 315)
       .
       .
```

Note: SQL DCLINCL is a required Migration Utility statement.

INSERT statement

The INSERT statement inserts a row into an SQL file.



Parameters

&FILE The name of the SQL file.

INTO Included for readability.

INSERT does not require an open cursor. If the cursor for the file is not open, one is not opened automatically. If a cursor is open, the inserted row does not appear in the cursor's result set until the cursor is closed and re-opened with a new SELECT statement.

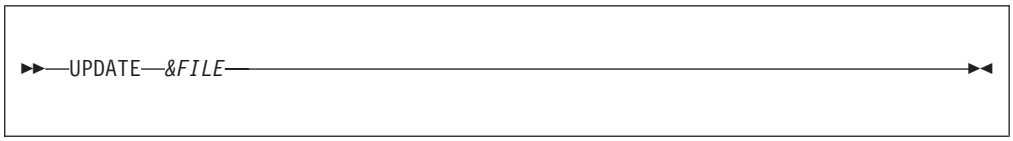
The file must be specified with the UPDATE parameter.

Example

```
INSERT FILEIN
```

UPDATE statement

The UPDATE statement updates a row from an SQL file.



Parameter

&FILE The name of the SQL file.

UPDATE issues an UPDATE WHERE CURRENT OF cursor.

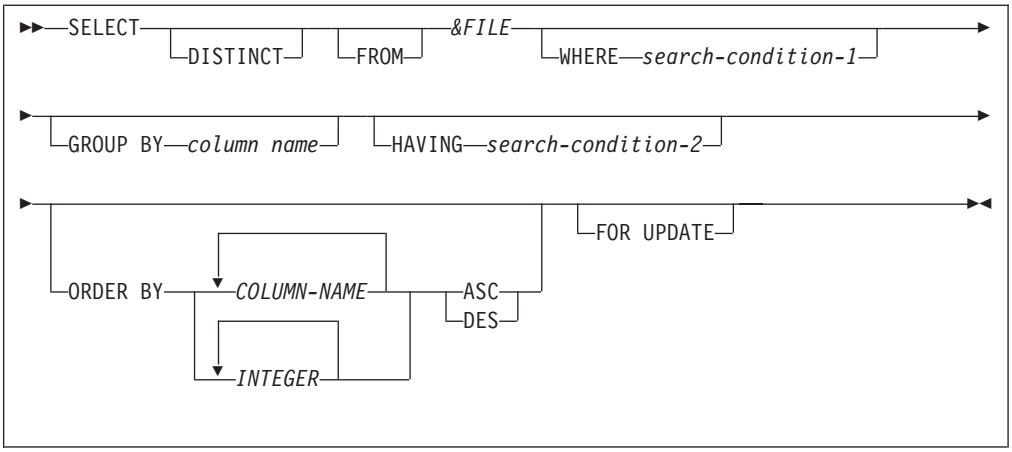
When the file is defined with the UPDATE, all defined columns are updated, otherwise only columns defined with the UPDATE are updated. Refer to the description of the SQL INCLUDE statement.

Example

```
UPDATE FILEIN
```

SELECT statement

A SELECT statement issued for an SQL file causes a cursor to be automatically declared and opened as a file. The resulting cursor can then be fetched and updated by subsequent commands for the file. The cursor can also be used for automatic input using the JOB statement.



Parameters

DISTINCT
Eliminates duplicate rows. If omitted, all rows are supplied.

FROM
Code for readability.

&FILE An SQL file.

search-condition-1
Conditions for the retrieval of data.

column name
Columns for group fetches of data into the file.

search-condition-2
Condition specifying the data to be returned to the user, for example, a range of values.

ORDER BY
Returns the rows in the sequence of specified columns. ASC is ascending order, DESC is descending order.

FOR UPDATE
Allow updates of the updateable fields in the &FILE.

If no SELECT is issued for the &FILE, the default SELECT is used (all rows are selected).

If SELECT is the first statement in a JOB activity that matches an SQL file in automatic input, it overrides the default SELECT.

SELECT can be coded in a JOB's START procedure. If so, DEFER should be coded on the FILE statement to avoid duplication and performance problems.

If a SELECT is specified for a file that already has an open cursor, the cursor is closed and a new one is opened.

Example

```
SQL DCLINCL USERTAB

DEFINE USER-NAME    W  20 A
DEFINE USER-DEPT    W   2 P 0
DEFINE USER-PHONE   W   3 P 0
```

```
DEFINE NULL-PHONE   W   2 B 0

JOB INPUT SQL
SELECT FROM USERTAB WHERE USER-NAME = 'JOHN'
     .
     .
```

Note: SQL DCLINCL is a required Migration Utility statement.

Easytrieve macros

The Easytrieve macros allow the user to have record definitions and more frequently used Easytrieve routines defined externally of the program.

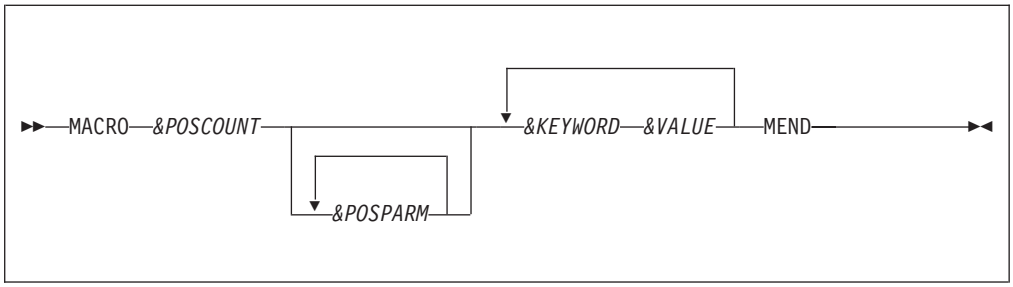
Macros are traditionally kept in a separate PDS or Librarian library.

Migration Utility fully supports all Easytrieve Macro Language conventions.

Macros allow symbolic replacement of the symbols imbedded in the macro source. Thus, macros can be created for frequently used program routines with the ability to mold the code by the external parameters coded in the program. This saves you creating new routines, and provides you with reliable, already tested, routines.

All Easytrieve macros must start with a MACRO statement and terminate with an MEND statement. User coded statements are placed between the MACRO and the MEND statement.

The macro name is the name of the Partitioned Data Set (PDS) member you create.



Parameters

&POSCOUNT
The number of positional parameters on the prototype statement. If the macro contains only keyword parameters, you must code ZERO.

&POSPARM
Positional parameter identifier. The number of parameter identifiers must match the number specified by the &POSCOUNT.

&KEYWORD
A keyword name to be used as a symbol in the replacement scheme.

&VALUE
Default value associated with the keyword.

MEND
The terminating keyword of the macro.

One or more keywords with associated values can be specified. Values with imbedded blanks must be enclosed in quotes.

User supplied statements are coded between the MACRO prototype and the MEND statements. The imbedded statements can include symbolic parameters (positional or keyword) as declared on the prototype. Each keyword is preceded by & (see the example).

Example

This example macro is the "MSTFILE" macro for defining Master File layout:

```
MACRO 2 FILE SIZE ORG KSDS PREFIX I

 FILE &FILE &ORG (&SIZE)
 &PREFIX-ACCOUNT      1   10 A
 &PREFIX-NAME         11  15 A
 &PREFIX-ADDRESS1     27  30 A
 &PREFIX-ADDRESS2     58  30 A

MEND
```

In the example, there are two positional parameters, FILE and SIZE, and two keyword parameters ORG and PREFIX. The keyword parameter ORG defaults to the value KSDS and the keyword parameter PREFIX defaults to the value I.

Notice the symbolic parameters imbedded in the FILE and the record definitions. Each positional and keyword identifier is preceded by a "&".

Invoking macros

Macros are imbedded in the Easytrieve source by prefixing the macro name by a "%".

Macro parameters are coded following the macro name.

Positional parameters are coded first (if any). The number of positional parameters must always match the number of positional parameters declared in the macro prototype.

Keyword parameters are optional, but if coded, they must be coded following the positional parameters. A value must be coded for each keyword parameter.

Any keyword parameters not supplied, when invoking macros, assume their default values as declared in the macro prototype.

When macro parameters span over multiple lines, each line must end with a ″+″ or a ″-″ as per Easytrieve punctuation rules.

Example

This example uses the MSTFILE macro above to define two master files:

```
%MSTFILE FILEIN 130 ORG KSDS PREFIX I1

%MSTFILE FILEIN2 130 ORG KSDS PREFIX I2

%MSTFILE FILEIN3 130 PREFIX I3
```

```
Valid

MSTFILE

Macro Invocations
```

Chapter 7. User exits

Optionally, user who are familiar with PEngiCCL macro language can write their own PEngiCCL macro to extract information collected by the translator. The macro is invoked by the translator at End of Job before exiting the main logic when no errors exist.

The Implementing process is as follows:

1. Make a copy of EASYUXIT macro located in SYS1.SFSYCCLM library.
2. Change the macro prototype name ″EASYUXIT″ in your new macro to reflect the new name.
3. Make the desired changes. Note that the EASYUXIT macro inherits relevant variable queues from the main translator macro, therefore, all variables with an ″I:″ in the definition located in EASYUXIT macro can be accessed.
4. Add USERXIT=&MACNAME parameter to your EZPARAMS (EASYTRAN) options, where &MACNAME is the name of your new macro.
5. Make sure that you have FJSYSP0 DDname defined in your JCL (first step). You can direct the output to SYSOUT or to a real file. LRECL is FB 80 characters long.

Alternatively, you can use EASYUXIT macro without changes. This macro produces a list of files with related information and a list of Easytrieve macros detected in the program. FJSYSP0 is required in the first step of the translator JCL.

Note: Writing PEngiCCL macros is not suitable for all users. In depth knowledge of PEngiCCL macro language is required.

CBLCNVRT macro

Use this macro to convert COBOL copybooks to Easytrieve Plus macros in two ways:

- A standalone job
- Coding CBLCNVRT

Running a standalone job to do the conversion.

1. Tailor and use JCEZCNV0 JCL located in SYS1.SFSYJCLS to do so.
   JCEZCNV0 uses sample EASYCNV0 file located in SYS1.SFSYEZTS.
2. Follow directions in the EASYCNV0 for updating rules.
3. Note that the %PUNCH is always needed as coded in the EASYCNV0 file.
4. Use standard Easytrieve Syntax to add entries to the EASYTCNV0 file.
   The Easytrieve Plus macros are punched to FJSYSP2 file in IEBUPDAT format.
5. Last, run a standard IBM IEBUPDAT job to add macros to a PDS.

Your generated Easytrieve Plus macros are now ready for use.

Example

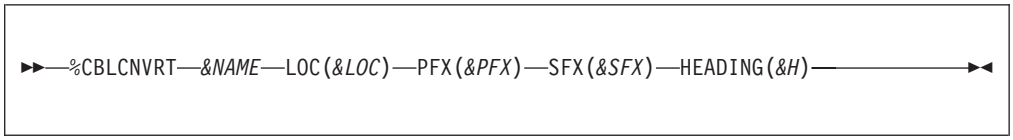
This is an example of EASYCNV0 entries:

```
%PUNCH EASYT (FJSYSP2).   * PUNCH OPTION AND DDNAME (DO NOT CHANGE)
%CBLCNVRT COBCOPYA.       * PUNCH COPY BOOK 1
```

Coding CBLCNVRT in Easytrieve Plus programs.

You can code CBLCNVRT to pull in COBOL copybooks to be used in the program. To do so, follow the standard Easytrieve Plus syntax for coding macros. Code %CBLCNVRT macro followed by the copybook name and CBLCNVRT macro options.

This format of CBLCNVRT is unique to Migration Utility. It is not supported by Easytrieve Plus.



Parameters

&NAME
The name of the COBOL copybook to be used.

&PFX Optional Fields Prefix, the default is ().

&SFX Optional Fields Suffix , the default is ().

&LOC Location: (*) or (W) or (S)

&H Field Title Option:
() Produces field titles found in the copybook.
(*) Produces commented out field titles found in the copybook.

Refer to "Generating COBOL COPY statements" on page 108 for the copybook format requirements.

Field titles are extracted from the copybook `" *: HDR ('&TITLE',...)` record.

The use of the %CBLCNVRT macro for VSAM Indexed file requires the use of the `KEY &KEY` option on the FILE statement.

Example

```
FILE FILEIN F (80)
%CBLCNVRT COBCOPYA LOC(*) PFX().
%CBLCNVRT COBCOPYA LOC(W) PFX(W-).

FILE FILEIN2 VS (KEY VCOMPANY)
%CBLCNVRT COBCOPYA LOC(*) PFX(V).
```

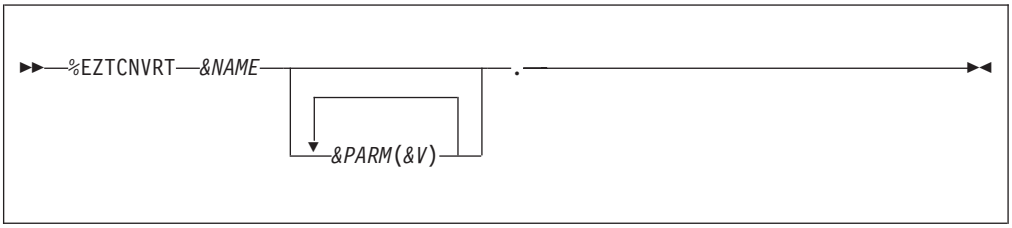
EZTCNVRT macro

The EZTCNVRT macro can be used to convert Easytrieve Plus macros to COBOL copy books.

EZTCNVRT is unique to Migration Utility. It is not supported by Easytrieve Plus.

A standalone job must be run to do the conversion:

1. Tailor and use JCEZCNV1 JCL located in SYS1.SFSYJCLS.
2. Make sure that COBOLCOPY=NO is coded in the EZPARAMS option.
3. Provide COPYCHAR='$S..' in EZPARAMS for translating special characters.
4. JCEZCNV1 uses sample EASYCNV1 file located in SYS1.SFSYEZTS.
5. Follow directions in the EASYCNV1 for updating rules.
   Note that the %PUNCH is always needed as coded in the EASYCNV1 file.
6. Use standard Easytrieve Syntax to add entries to the EASYTCNV1 file.
7. The COBOL copybooks are punched to FJSYSP2 file in IEBUPDAT format.
8. Last, run a standard IBM IEBUPDAT job to add copybooks to a PDS.
   Your generated COBOL copybooks are now ready for use.



Parameters

&NAME
The Easytrieve Plus macro name.

&PARM
Macro parameter (maximum of 8).

&V Value for each keyword/parameter.

Example

This is an example of EASYCNV1 Entries:

```
%PUNCH COBOL (FJSYSP2).      * PUNCH OPTION AND DDNAME (DO NOT CHANGE)
%EZTCNVRT EZTCOPYA LOC(*).   * PUNCH COPY BOOK 1
%EZTCNVRT $ZTCOPYB LOC(*).   * PUNCH COPY BOOK 2
```

Special considerations

Punch one copybook at a time to avoid complications with duplicate field names.

If the generated copybook is used as working storage, use the punch option %PUNCH COBOL (FJSYSP2) VALUES(YES) to force default field values to be generated in the copybook.

When the EZTCNVRT macro is used, all imbedded (nested) macros are ignored. The copybook is punched for the specified level-01 macro only.

All field names in the generated copybook are prefixed by the ″:AA:″ special characters. In this way, the REPLACING option of COBOL COPY statement can be used to assign unique field names when a copybook is used multiple times in the same program.

While the generated copybooks can be used in Native COBOL, the primary purpose of EZTCNVRT is to create COBOL copybooks that can be used in the translated programs. Thus, after generating all copybooks, you can translate Easytrieve Plus programs with COPYBOOK=YES option for a cleaner and leaner outcome.

Be aware, all field names are reduced to 16 characters or less to accommodate prefixing. You can enhance field names by providing the TRANSLATE WORDS and TRANSLATE FIELDS tables (see EZPARAMS Options). However, if you do so, the translate tables must be subsequently used for every program being translated to preserve naming conventions. The file with FJNAMES DDname in your JCEZCNV1 JCL will contain a table of reduced field names after the first pass. You can tailor this table and use it for translating field names (as TRANSLATE FIELDS table).

The generated copybook name is the macro name from which it was created. COBOL does not allow special characters in the copybook name and Easytrieve Plus does. Use COPYCHAR='....' of EZPARAMS to replace the bad characters.

Generating COBOL COPY statements

By default, Migration Utility generates hard-coded file and working storage layouts in the generated programs. With minor effort, it is possible to generate COBOL COPY statements for your Easytrieve Plus macros and then direct Migration Utility to punch out COBOL COPY statements instead of the hard-coded layouts.

To do this:

1. Make an inventory of all Easytrieve Plus macros that qualify to be a copybook. In general, these would be those macros that fully define a record or working storage area. Macros that contain FILE statement along with record layout also qualify. Note that COBOL copybooks pulled in using CBLCNVRT are automatically considered.
2. Make a table of macro names similar to the EZTABLE0 table located in SYS1.SFSYEZTS pds.
3. Generate COBOL copybooks out of Easytrieve Plus macros using the EZTCNVRT macro (refer to “EZTCNVRT macro” on page 107).
4. Prepare EASYTRAN/EZPARAMS options:

```
COPYBOOK=YES        YES for generating copybooks
COPYNTAB=&NAME      The table that contains the list of Easytrieve macros
COPYCHAR='$S'       Character replacement for fixing bad copybook names
NCOPIES=nnn         nnn is the maximum number of copybooks in a single program
```

5. Concatenate the PDS that contains your version of EZTABLE0 to FJCPYLB DDname (the first step of the translator JCL).
6. Concatenate the PDS where your COBOL copybooks are located to FJCPYLB in the second step (PEngiBAT step) of the translator JCL, and to SYSLIB of the COBOL Compiler step.

Example

This is an example of EZTABLE0 table:

```
*--------------------------------------------------------------------*
* EASYTRIEVE PLUS MACROS THAT QUALIFY FOR COBOL COPY BOOKS.           *
*                                                                    *
* THIS TABLE IS USED BY THE EASYTRIEVE PLUS TRANSLATOR WHEN           *
* OPTION COPYNTAB=EZTABLE0 IS CODED IN EZPARAMS.                     *
*                                                                    *
* FILE LAYOUT:                                                       *
*                                                                    *
* CODE EZT MACRO NAMES FOLLOWED BY:                                  *
* ---------------------------------                                  *
*   A. PREFIX KEYWORD                                                *
*   B. SUFFIX KEYWORD                                                *
*   C. SUB LIST OF ALLOWED ADDITIONAL KEYWORDS ON MACRO STATEMENT    *
*   D. '*' MUST BE PLACED BEFORE COMMENTS                            *
*                                                                    *
* RULES:                                                             *
*   1. PREFIX IS OPTIONAL                                            *
*      PREFIX CAN BE CODED ALONE                                     *
*                                                                    *
*   2. SUFFIX IS OPTIONAL                                            *
*      IF SUFFIX IS NEEDED, AND THERE IS NO PREFIX, A FAKE PREFIX    *
*      KEYWORD MUST BE SUPPLIED. YOU CAN USE ANY FAKE KEYWORD.       *
*                                                                    *
*   3. SUB LIST OF ALLOWED KEYWORDS IS OPTIONAL                      *
*     +IF SUB LIST IS CODED, THE TRANSLATOR WILL GENERATE A COPY     *
*      STATEMENT ONLY WHEN MACRO IS CODED WITH THE KEYWORDS IN THE   *
*      SUB LIST. THE PREFIX AND SUFFIX ARE AUTOMATICALLY INCLUDED.   *
*     +IF SUB LIST IS NOT CODED, THE TRANSLATOR WILL GENERATE A COPY *
*      STATEMENT UNCONDITIONALLY.                                    *
*                                                                    *
* NOTES:                                                             *
*   EASYTRIEVE PLUS MACROS THAT QUALIFY FOR COPY BOOKS CAN CONTAIN   *
*   RECORD OR WORK AREA DEFINITIONS. MACROS WITH RECORD DEFINITIONS  *
*   CAN CONTAIN FILE STATEMENTS TOO.                                 *
*                                                                    *
*   THE PREFIX AND SUFFIX PARAMETERS ARE USUALLY USED TO MODIFY      *
*   NAMES SO THAT MULTIPLE LAYOUTS CAN BE GENERATED FROM A SINGLE    *
*   MACRO. LOOK AT YOUR MACRO PROTOTYPE STATEMENTS TO SEE IF THE     *
*   PREFIX AND SUFFIX PARAMETERS ARE USED.                           *
*                                                                    *
*   MACROS CAN BE CODED TO USE OTHER KEYWORD PARAMETERS BUT SUCH     *
*   PARAMETERS SHOULD NOT BE MODIFYING FIELD NAMES. CODE SUCH EXTRA  *
*   PARAMETERS AS A SUB LIST OF ALLOWED KEYWORDS. THE SUB LIST CAN BE *
*   CONTINUED ON MULTIPLE LINES. USE STANDARD EASYTRIEVE PLUS RULES.  *
*--------------------------------------------------------------------*
COBCOPYA PFX             * DEMO FILE COBOL COPYA BOOK USED BY %CBLCNVRT
COBCOPYB PFX             * DEMO FILE COBOL COPYB BOOK USED BY %CBLCNVRT
EZTCOPYA PFX SFX (LOC)   * DEMO FILE EZT COPYA FORMAT MACRO
EZTCOPYB PFX SFX (LOC)   * DEMO FILE EZT COPYB FORMAT MACRO

*----------- ADD ADDITIONAL MACROS AFTER THE LAST LINE ---------------*
```

Note: It is critical to code the correct prefix, suffix and the allowed keywords for each macro you specify. In this example, the prefix keyword is PFX, the suffix keyword is SFX and the additional parameter allowed on EZTCOPYA and EZTCOPYB macros is LOC.

Multiple COPY statements are generated when the layout consists of more than one macro.

Migration Utility has a complex algorithm for determining if a macro qualifies for a COPY statement. A hard-coded layout is forced whenever a listed macro in EZTABLE0 is used in a manner that would cause errors. For

example: a layout is composed of hard-coded fields and a macro that qualifies for a copybook, or a redefine of a field defined in a macro that qualifies for a copybook, is coded outside of the macro scope (that is, activity section).

Special considerations

If you run with COPYBOOK=NO and COPYNTAB= is supplied, the generator will take advantage of the table information and generate a prefix for copy books that are used multiple times in the same program, if such copybook is not coded with a different prefix. Thus the duplicate field names would be prefixed rather than made unique via a sequence number. Therefore, it is a good practice to build a table of all macros that qualify to be copybooks, even if COPYBOOK=NO is in effect.

System information

Migration Utility files

The following files are Migration Utility product PDS files:

Data set name	Description
SYS1.SFSYJCLS	Sample JCL distributed with the product
SYS1.SFSYCCLC	Translator COPY commands
SYS1.SFSYCCLM	Translator macros
SYS1.SFSYEZTC	Sample Easytrieve Plus macros and COBOL copybooks
SYS1.SFSYEZTS	Sample programs and translator parameter files
SYS1.SFSYFJCC	Translator pre-processed (byte code) macros
SYS1.SFSYLOAD	Translator load modules

Called by the translated COBOL programs

The following COBOL modules are located in SYS1.SFSYLOAD Library:

Name	Description
FSABECOB	Batch programs ABEND message handler
FSDATEZ0	Date services main interface program
FSDATSRV	Date services program for base 365
FSDATSR2	Date services program for base 360
FSDIMAGE	Unstring an edited number
FSMOVE00	Move Long Common module
FSPLOT00	Plot program (called by some batch programs)
FSSPACE0	Common modules used by Easytrieve Plus translated programs
FSYDB200	Call Attachment Interface (CAF) sample program
FSYDB250	Call Attachment Interface (CAF) module

The following BAL modules are located in SYS1.SFSYLOAD Library:

Name	Description
FSYDB202	Load DSNHFECP CAF Module
FSYGJOB0	Get Job Information from JOB Scheduler
FSDYNCNV	Convert ASCII to EBCDIC and EBCDIC to ASCII
FSYVLN90	Get file information block (FIB) at open time. This module replaces FSVLNT90.
FSVLNT90	Get File Information Block (FIB) at Open time
FSVLNT03	Get File Record Length after and I/O
FSFILL00	Pad a field with a pad characters
FSLOPER0	Perform Logical Operation, AND, OR, XOR
FSLOPER1	Perform Bit Testing for ON condition

FSSHIFTL Shift a field 4 bits left
FSSHIFTR Shift field 4 bits right

A number of other modules that may be needed at run time are included in the SYS1.SFSYLOAD library.

Runtime requirements

If Migration Utility is run with IOMODE=DYNAM, SYS1.SFSYLOAD library is required at run time.

If the translated COBOL program is compiled with the ″DYNAM″ option, then SYS1.SFSYLOAD must be concatenated to your JOBLIB or STEPLIB DDnames.

If the translated COBOL program is compiled with the ″NODYNAM″ option, then SYS1.SFSYLOAD is not needed at run time.

The FSABECOB program is invoked by the generated batch programs whenever there is a file I/O error. It can be also invoked for other reasons that require an abnormal program termination.

The FSABECOB prints a description of the detected error on the FJSYABE file if supplied in the JCL. The error description is usually based on the file status returned by the COBOL I/O routines. The description printed is as per FILE-STATUS code.

Summary of DDnames

To run Migration Utility, the following files must be defined in the JCL:

DDname	Description
FJSYSIN	Input file which contains the EZPARAMS File, normally SYS1.SFSYEZTS
FJSYSPW	PASSWORD file SYS1.SFSYCCLM(PASSWORD)
FJSYSJC	Optional Output Sequential JCL file. This file is generated by the translator when JCL=YES is specified. The record length of this file is 80 bytes. The block size can be coded in the JCL via the DCB statement.
FJBIND0	This file contains DB2/SQL BIND parameters skeleton. The record length of this file is 80 bytes. Refer to SQLMODE= parameter in EZPARAMS.
FJCCLLB	Pre-Compiled Translator CCL1 Macros SYS1.SFSYFJCC
FJCPYLB	PDS which contains copybooks. The user can also concatenate the PDS which contains the user written COBOL copybooks or other (copy) members (including Easytrieve Plus macros).
FJMACLB	PDS which contains Translator CCL1 standard macros. You can also concatenate the PDS which contains the user written CCL1 macros (including CCL1 macros for Easytrieve Plus).
FJSYSPH	The output generated program source code. This can be a sequential data set or a PDS library. The record length of this file is 80 bytes long. The block size can be coded in the JCL via the DCB statement. The created program can be passed on to the COBOL compiler in the same job stream or permanently saved.

FJSYSP0	The output statistics record normally contains a list of files and macros found in Easytrieve Plus program.
FJSYS01	This is a temporary sequential work file. The record length of this file is 94 bytes long. The block size can be coded in the JCL via the DCB statement.
FJNAMES	Refer to DDFNAME= option in EZPARAMS. This file contains a table of reduced field names. The record length of this file is 80 bytes.
SYSIN	File which contains an Easytrieve program. This can be a PDS or a sequential file. The record length of this file must be 80 bytes long. The block size cannot exceed 32,767 bytes.
SYSLIST	Translator diagnostics and generated program listing. This is a standard print file. For more information on this file, see the ″LIST=″ parameter in the Translator Options described in this section.
FJSYSER	Translator error file. This file contains a summary of errors and MNOTEs issued by the translator during the translating process. The LRECL of this file is 89, DSORG=PS.

Translator CCL1 preprocessor options

COPTION parameters

Migration Utility CCL1 preprocessor options can be overridden using the COPTION statement in two ways:

1. COPTION statement can be placed at the beginning of the EZPARAMS file. One or more COPTION statements can be supplied with multiple keywords on each statement depending on the requirements.

   Example:

   ```
   COPTION LIST=CND,LINES=60
   ```

2. COPTION statement can be coded in the PARM= parameter of an exec statement for MVS/ESA™ operating systems. Note that the maximum number of characters that can be used in a PARM= statement on an MVS/ESA operating system is 54.

   Example:

   ```
   //PENGI  EXEC PGM=FSCCOBOL,
   //       PARM='(COPTION(LIST=CND,LINES=60)'
   ```

The parameters are coded with “=” (keyword parameters) followed by the value.

The following options can be used in the COPTION statement:

Keyword	Description
BUFSIZE=2048	The size of the internal CCL1 work buffers. The maximum is 32,767. The work buffers are used by CCL1 to collect and decode all macro parameters. PEngiCCL allocates seven internal work buffers for the size specified in BUFSIZE. Note that the size should not be overestimated to avoid excessive use of storage.
ERRLIMT=32	Error limit count as NNN. Controls the maximum number of CCL1

detected printable errors. This parameter is designed to simplify error debugging by limiting the number of errors printed in a single preprocessing.

When the number of errors exceeds the ERRLIMT, all subsequent errors are suppressed.

LINES=56 Number of lines per page for preprocessed program

The LINES parameter value should not exceed the physical page capacity of 66 lines.

LIST=NO The preprocessor list option. Can have these values:

YES A listing of the pre-processed program and all generated code is produced, including internally generated functions. This type of listing is also referred to as the expanded listing.

NO A listing of the pre-processed program is not produced.

CND Only a listing of the input program source is produced. This type of listing is also referred to as a condensed listing.

FUN A listing of the input program and the internally generated functions is produced. This type of listing is also referred to as a condensed/function listing.

System information

Chapter 8. Installation and Migration Utility options

For step by step Easytrieve source conversion refer to the “Translating guidelines” on page 2.

Installation

Migration Utility is installed using SMP/E. Please refer to the *Program Directory* (GI10-8469) for installation instructions.

Migration Utility works on the OS/390® and z/OS™ operating systems.

Activating Call Attachment Facility (CAF) for DB2 users

DB2/SQL users must choose:
- The method for retrieving DB2/SQL column definitions.
- The method for running a converted DB2 program.

Note: COBOL programs are located in SYS1.SFSYEZTS pds. JOBS are located in SYS1.SFSYJCLS pds.

1. DB2/SQL column definitions can be retrieved by Migration Utility:
   - Automatically from SYSIBM.SYSCOLUMNS catalog via CAF.
   - By coding EASYTRAN: DCLINCL &DCLGEN in Easytrieve.

   To access SYSIBM.SYSCOLUMNS table automatically, CAF interface must be activated using the steps below.

   To use DCLINCL, refer to the description of DCLINCL in this manual.

   Tailor the JCCOBSQL job and compile the FSYDB2D1 COBOL program. JCCOBSQL is an instream procedure. The EXEC is at the bottom of the procedure.
   - If you do not have a BIND plan, create and bind a plan that can be accessed by each user using Migration Utility. Bind to as many systems as you need (test, production, and so on).
   - Bind FSYDB2D1 to your systems using the JCCOBBND job. The BIND parameters are at the bottom of the procedure. You must change the BIND parameters to your system requirements. Note that you must bind FSYDB2D1 as a PACKAGE for Call Attachment Facility (CAF) use.

   Change the CAFPLAN= EZPARAMS/EASYTRAN option to the bound plan name.

   Make sure the JCEZDB2A procedure has the proper DSNLOAD and DSNEXIT load libraries concatenated in the FSCCL1 and DB2 translator steps.

   The translator utilizes the SYSIBM.SYSCOLUMNS table whenever it encounters an “SQL INCLUDE FROM &owner.&table” in the Easytrieve Plus program, and a DCLINCL was not previously supplied.

Try translating a DB2 program using the JCEZDB2A job. Find a program that contains "SQL INCLUDE FROM &owner.&table" statements to test CAF functionality. Look at the SYSOUT file of the FSCCL1 step for CAF messages.

2. Converted DB2/SQL programs can be run in two ways:
   - By using the standard DB2 IJKEFT1B program (see JCEZDB2R job)
   - By attaching to DB2 via CAF from the generated COBOL programs

   If you want to use the DB2 Call Attachment Facility (CAF) to run your generated DB2 programs, and you do not have your own CAF module, you can chose from the two supplied modules:

   FSYDB200 Attaches to DB2 via CAF from the generated COBOL program at run time using a hard-coded plan name.

   To use this program you must:
   - Change the plan name in the FSYDB200 COBOL program to your installation standard plan name and compile the program using JCBATCOB supplied JCL. Note that this is not a DB2 program.
   - Link the generated COBOL program with FSYDB200 included. Change the EZPARAMS option to SQLMODE=FSYDB200 before translating Easytrieve programs. A call is generated to the CAF module at the beginning of the generated COBOL program. Make sure that the DB2SQL step in the translator JCL contains ATTACH(CAF) on the PARM statement.
   - Bind the generated COBOL program as a PACKAGE.
   - Provide SSID (DB2 system name) using the DSNHDECP load module at run time. This is a DB2 module. Consult with your DB2 system administrator for the proper load library.

   FSYDB250 (This is the preferred module.)

   Attaches to DB2 via CAF from the generated COBOL program at run time using an external parameter file pointed to by the //SQLDD statement. To use this program you must:
   - Link the generated COBOL program with FSYDB250 included. Change the EZPARAMS option to SQLMODE=FSYDB250 before translating Easytrieve Programs. A call is generated to the CAF module at the beginning of the generated COBOL program. Make sure that the DB2SQL step in the translator JCL contains ATTACH(CAF) on the PARM statement.
   - Bind the generated COBOL program as a PACKAGE.
   - Provide SSID=&SYSTEM,PLAN=&PLAN using //SQLDD JCL at run time.

   Note: SQLDD file LRECL=80,RECFM=F.

   You can place comments in the input SQLDD file by placing a '*' in position 1.

   Example:

   ```
   * THIS IS FSYDB250 PARAMETER FILE FOR DB2 PRODUCTION SYSTEM DB2P
   SSID=DB2P,PLAN=USERPLAN
   ```

3. How does IBM CAF for DB2 work?

The IBM DSNALI program connects to the DB2 system and plan name passed to it by the calling program.

The FSYDB200 program uses the SSID from the DSNHDECP load module located in a load library, the hard-coded plan name, and calls the DSNALI program at run time. Consult with your DB2 administrator for the exact location of DSNHDECP. The module is usually located in the &HQUAL.DSNLOAD or &HQUAL.DSNEXIT library. This load library must be concatenated to your JOBLIB or STEPLIB at run time.

The FSYDB250 program obtains the SSID and plan name from the SQLDD file and calls DSNALI at run time.

Using EZTPA00 program loader

The Easytrieve Plus compiler program name is EZTPA00. It normally resides in the Easytrieve Plus load library.

Migration Utility's EZTPA00 is a general purpose program loader that extracts and executes (fetches) the program name found in the SYSIN statement. It resides in the SYS1.SFSYLOAD library.

Note: EZTPA00 is a program loader not a compiler as the Easytrieve's module.

Some Easytrieve Plus users run in "compile and go" mode by pointing to programs, located in a pds, using a SYSIN.

For example,

```
//JNAME JOB ...
//STEP01 EXEC PGM=EZTPA00, ...
:
//SYSIN DD DSN=&DSNAME(TESTPGM1),DISP=SHR
```

Suppose that TESTPGM1 in the SYSIN above was converted to COBOL and linked into a load library and you want to make minimal changes to your JCL. You can invoke Migration Utility's EZTAP00 by pointing to &HQUAL.PENGI401.LOADLIB at run time. The TESTPGM1 program name is extracted and fetched by EZTPA00.

The benefits from using this technique may be limited. It is beneficial only to installations that run a large number of programs in "link and go" mode as described above, during the testing phase.

This technique is not recommended for use in a production environment as it adds another layer of complexity.

REPORT default options

EASYDTAB macro located in SYS1.SFSYCCLM pds contains default options that are similar to those of EZTOPT table of Easytrieve Plus. Update this macro to match the defaults of your EZTOPT table. The updating instructions can be found at the beginning of the macro.

The following options are available:

```
&GDTLCTL,FIRST;
       DTLCTL
          FIRST
          EVERY
```

```
        NONE
&GLINESIZE,132;
```
Length of Standard Print Line

```
&GPAGESIZE,58;
```
Number of report lines per page

```
&GPAGEWORD,'PAGE';
```
Page identifier (if DBCS, make sure SI/SO are present)

```
&GDSPLSIZE,66;
```
Number of DISPLAY lines per page

```
&GCNTRLSKP,1;
```
Number of lines after Control Breaks

```
&GSKIP,0;
```
Number of blank lines between Detail Lines

```
&GSPACE,3;
```
Number of spaces between fields

```
&GSPREAD,0;
```
SPREAD Option:
```
0=NOSPREAD
1=SPREAD
```

```
&GSUMCTL,HIAR;
```
Control Break Line options:
```
HIAR
ALL
NONE
TAG
DTLCOPY
```

```
&GSUMSPACE,3;
```
Size expansion for SUM fields: 0 TO 9

```
&GTALLYSIZE,2;
```
TALLY Counter Size. SUMSPACE is added to this value.

```
&GTITLESKIP,3;
```
Number of lines between Headings and Titles

```
&GNOADJUST,0;
```
NOADJUST Option:
```
0=ADJUST
1=NOADJUST
```

```
&GDATESIZE,SHORT;
```
Reports Date usage:
```
SHORT=SYSDATE
LONG=SYSDATE-LONG
```

```
&GDATSFORM,MMDDYY;
```
Reports SHORT Date format:
```
MMDDYY
YYMMDD
DDMMYY
```

```
&GDATSMASK,Z9/99/99;
```
SHORT Date edit mask:
```
Z9/99/99
```

99/99/99

&GDATLFORM,MMDDCCYY;
Reports Long Date format:
MMDDCCYY
CCYYMMDD
DDMMCCYY

&GDATLMASK,Z9/99/9999;
SYSDATE-LONG edit mask:
ZZ/99/9999
99/99/9999
9999/99/99

Mask identifier table to facilitate Easytrieve USERMASK

Easytrieve Plus provides for establishing default Mask Identifiers via the EZTOPT table. A new option has been added to Migration Utility EASTDTAB to provide compatibility.

To create default mask identifiers, masks can be added to the EASYDTAB defaults table located in SYS1.SFSYCCLM PDS. The system is shipped with a commented example at the bottom of EASYDTAB. The SETVT instruction can be un-commented and masks added as per instructions in the EASYDTAB.

Example:

```
ACCL SETVT &GUSERMASK
           A,'99/99/99'
           B,'99:99:99'
           C,'ZZZZZZZ9';  ";" MUST BE AFTER THE LAST ENTRY
```

Migration Utility translator options

The member EZPARAMS in the Migration Utility library (SYS1.SFSYEZTS) can be tailored to provide a global override for Migration Utility default options. A good way of doing this is to copy the distributed parameters into your own PDS. Do not forget to change the EZPARMS= symbolic in the JCL to point to the PDS that contains the new member.

Also, options can be imbedded in each Easytrieve Plus program source. This method lets you have specific options for each Easytrieve Plus program. The parameters are supplied at the beginning of the program as comment statements. The method is fully described in “Embedding options in the program source” on page 126

The first line in the EZPARAMS member is the COPTION statement. The COPTION statement describes PEngiCCL options such as the output listing, paragraph re-sequencing options, etc. The COPTION parameters are fully described in the FS/PEngi Installation. The defaults as set in the distributed EZPARAMS member are sufficient, thus there is no need for change.

Options are processed by the EASYTRAN macro. All options are keyword parameters, except the ″TRANSLATE″ option.

Options must be coded using the standard PEngiCCL conventions for coding macro instructions. That is, the word EXCCL can start in position 8 followed by the macro name. Any keyword and positional parameters must be coded starting in position 12 or after, on subsequent lines as illustrated in this example.

```
1...!....10...!....20...!....30...!....40....!....50...!....60...!..
       COPTION LIST=CND,ERRLIMT=015,PARASEQ=(NON,1,10)
       EXCCL EASYTRAN
             FILES=64
             FIELDS=2000
             INDENT=4
             TRANSLATE WORDS
                      (BALANCE  BAL)
                      (AMOUNT   AMT)
             TRANSLATE FIELDS
                        (INTEREST-AMOUNT INT-AMT)
                      (CURRENT-BALANCE CUR-BAL)
             END-TRANS ;  <= END SEMICOLON IS REQUIRED
```

Here is a list of keywords (the value indicated is the default):

CAFOWNR=(&USER)
Default DB2 table creator/owner to be used when the &owner is not provided in the "SQL INCLUDE FROM &owner.&TBname" statement. This parameter is the default for PARM SQLID ('&owner') Easytrieve Plus parameter.

When CAFOWNR=('&USER') is coded (with ampersand exactly as shown), the TSO User ID submitting the job is used. Any other value is used explicitly as coded. For example, CAFOWNR=OD uses "OD" as the &owner for retrieving column definitions if the &owner is not coded.

CAFPLAN=BATCH
Default Call Attachment Facility plan name for retrieving SQL/DB2 column definitions. The specified plan must match the BIND plan name of FSYDB2D1 program. For details, refer to "Activating Call Attachment Facility (CAF) for DB2 users" on page 115. Coding CAFPLAN=(<NO>) disables automatic retrieval of column information from the DB2 catalog. In such a case, the translator expects to find a DCLINCL statement for each DB2 table in use.

COBOL=COBOL390
Type of COBOL. COBOLII for COBOL-II, COBOL390 for COBOL/390

COBVERBS=YES YES causes Migration Utility to scan Easytrieve field names for Reserved COBOL Words and append -Y1 to reserved words to prevent COBOL compiler errors. Code "COBVERBS=NO" to inhibit this process.

COPYBOOK=NO The COBOL copybook option. Values are YES and NO.

When COPY=YES is coded, Migration Utility generates a copy statement for all files that are defined using %COPYNAME in Easytrieve source.

To use this option, a COBOL copybook must be defined with the identical field names defined in the Easytrieve copy book. In addition, each field must be defined with special replacement characters :AA:. For example:

```
01 FILEIN-RECORD.
     02    :AA:FIELDA        PIC  X(03).
     02    :AA:FIELDB        PIC S9(9) COMP.
```

The COBOL copybook must be placed in a PDS accessible by Migration Utility step2 and COBOL.

Using the copybook option may not be suitable for all programs since Migration Utility alters the generated names when duplicate fields names are detected. Use it with caution.

COPYCHAR='$S'

COBOL copybooks bad characters replacement string written as XY pairs.

When COPYBOOK=YES is specified, the translator generates a COBOL COPY statement using the actual Easytrieve Macro name. Such a name could contain characters not allowed by COBOL. Use this string to replace character "X" by character "Y". Multiple pairs can be coded. All pairs must be enclosed in the single quotes.

COPYNTAB=

Table name that contains the list of Easytrieve macros that qualify for File Record or Working Storage copybook (layout). The table must be a PDS member of LRECL 80. The PDS must be concatenated to FJCPYLB in the first step of your translator JCL. This table determines which macros are to be generated as COBOL copybooks.

When coded, this table is searched whenever an Easytrieve macro is encountered and the macro contains FILE or Working Storage field definitions. If located and:

COPYBOOK=YES
: A COBOL COPY is generated in the place of hard-coded layout.

COPYBOOK=NO
: The translator uses this fact and generates a meaningful fields prefix when the same macro is used in the program more than one time.

To optimize translating, it is recommended that you create this table even if COBOL COPY statements are not being generated.

The default is no table name. Refer to "Generating COBOL COPY statements" on page 108.

COPYVERB=(COPY)

COBOL copy to be used as COPY statement when COPYBOOK=YES is specified.

Change this default if you have a special copybook pre-processor that recognizes a different verb as a COPY statement. For example, SQL host variables are flagged by the DB2 pre-processor when located in a copybook. To allow copybooks, you can create your own pre-processor that recognizes some other verb as a COPY statement to expand copybooks before the DB2 translator step.

COPYWRAP=('==')

COBOL copy verb REPLACING string wrap characters. These characters are wrapped around the replacing strings of the COPY statement when COPYBOOK=YES is specified.

For example:

```
COPY &NAME
      REPLACING ==:AA:== BY ==K=== .
```

Migration Utility translator options

CURRENCY=($) Currency sign

DATEABE=NO Date routines abend option:
- **NO** Do not abend on invalid date
- **YES** Abend on invalid date
- **RC** Use RETURN-CODE for invalid dates

DDFNAME= DDname of file for reduced field names. This must be a valid 1-8 characters DDname. If coded, all field names that are reduced in length by the translator are punched out to this file. While this is an informational file, the punched information can be massaged for more meaningful names and used for creating a translate table for translating Field Names in the subsequent translating attempts. (Refer to the ″TRANSLATE FIELDS″ option below).

The file is not created unless this DDname is specified.

The file is not created if there are no reduced fields even if DDname is specified.

DECIMAL=PERIOD
Decimal point: PERIOD or COMMA

DECLGEN=FULL SQL INCLUDE generation.
- **FULL** Generates an SQL INCLUDE &NAME for each referenced table in the program.
- **PART** Generates hard-coded SQL INCLUDE for the column definitions only.

DOWHILE=INLINE/PERFORM
The DO WHILE code generating method. INLINE generates an inline PERFORM for the DO WHILE. PERFORM generates a separate paragraph for each DO WHILE nest. In general, the INLINE option generates logic that is less fragmented and easy to follow, but the reference labels inside the DO WHILE nests are not allowed. Manual adjustments may be required. The PERFORM option resolves reference labels at the DO WHILE level by creating a separate paragraph for each DO WHILE nest.

ENDCOL=72 End column on input source: 72 or 80

ETBROWS=512 Default number of rows for external tables. This value is used when the table rows is not coded on an external table file definition.

FIELDS=1500 Maximum number of field definitions

FILES=128 Maximum number of supported files

FSIGN=YES FSIGN handling method for numeric display format fields:
- **YES** Force F sign on fields located in file records
- **NO** Do not force F sign at all
- **ALL** Force F sign on all display numeric fields (records and working storage)

HEADERS=128 Maximum number of fields for Report Heading statement

HFIELDS=256 Maximum number of Title fields in a single report

INARGS=128 Number of input arguments from a single parsed string.

Note: The number of INARGS should not be overestimated due to the impact on the memory utilization. Only use a value greater than 128 if absolutely necessary.

INDENT=3 Standard indentation (3 means three spaces)

INDEXS=256 Maximum number of index entries due to OCCURS

IOCODE=NATIVE

Option for FILE-STATUS translation:

NATIVE

Migration Utility creates COBOL Status Codes in the generated program when referencing FILE-STATUS. When this option is in use, you must ensure that any hard-coded values that are checked against FILE-STATUS fields are properly adjusted before translating an existing Easytrieve Plus program. For restrictions, refer to "FILE-STATUS (STATUS) codes" on page 17.

EASYT

Migration Utility translates COBOL Status Code to Easytrieve Plus equivalent after each I/O. This option generates extra code, but the status codes checking remains compatible with those of Easytrieve Plus. No tailoring is required.

IOERC=1000 Return Code when file I/O error is detected. Code "IOERC=(NNNN,AUTO)" to generate unique abend code for each I/O routine. NNNN is used as the base. One is added to return coded in each I/O abend routine. If AUTO is not specified, the same code is used for all I/O errors.

IOMODE=NODYNAM

I/O mode:

DYNAM	Use dynamic I/O
NODYNAM	Use static I/O

LINES=256 Maximum number of report lines for a single report

MAXARG=256 Maximum number of arguments in a single IF statement

MAXINDENT=27 Maximum indentation (applies when nested IFs are processed)

MAXPROC=256 Maximum PROC paragraphs

MAXSTR=1024 Maximum string size for a single bracketed expression

MEMINIT=SPACES

Initialization for File Records:

SPACES Clears file records to spaces

VALUES Clears file records to spaces and then with COBOL INITIALIZE.

MNESTS=16 Maximum number of macro nests

MPARMS=064 Maximum number of supported macro parameters

NAMETAB='$S/-+A#N@V*-_-%P'

Translate table for special characters found in the field names. These are coded in pairs, so "$" is translated to "S", "/" is translated to "-", and so on.

NCOPIES=256 Maximum number of copybook names (macros) that qualify as record or working storage definition that can be used in a single

program. This is the limit of the queue that keeps track of macros and copybooks used in the program that are also listed in the table identified in the COPYNTAB= statement.

NESTS=16 Maximum number of nested IFs

OBJECTS=1024 Maximum number of Objects for COBOLBAS (passed on to PEngiBAT step)

OCCURS1=0 &field OCCURS 1 handling:
- **0** Flags OCCURS as an error
- **1** Generates the field without OCCURS
- **2** Generates OCCURS 2 in the place of OCCURS 1

OVERFLOW=NOTAG

Fields overflow tag option for Report Totals:
- **NOTAG** Do not place an asterisk (*) on overflow condition
- **TAG** Place an asterisk (*) in the last position of overflowed field

RFIELDS=768 Maximum number of fields on a single report

SPOOLOPT=NO Optimizes the record length for temporary and sort work files by defining numeric display fields in packed-decimal (COMP-3) format.
- **NO** Do not generate COMP-3 fields
- **YES** Generate COMP-3 fields

Note: For programs previously translated with SPOOLOPT=NO, the record length for temporary files may be different when translated with SPOOLOPT=YES. The LRECL value, if coded in the JCL, must be adjusted to reflect the new length. When you specify SPOOLOPT=YES, it generates shorter record lengths and utilizes storage more efficiently.

SQLFLDS=768 Maximum number of SQL fields

SQLMODE=BIND SQL/DB2 BIND Option:
- **BIND** Generate BIND parameters
- **PGMNAME** Generate a call at the beginning of the Procedure Division for attaching to DB2/SQL at run time (that is, CAFATTCH generates CALL 'CAFATTCH'.)

SQLPFIX=(Q-) Prefix for SQL file fields (host variables derived from DECLGENs). A sequence number is inserted into the prefix to yield unique field names, that is, Q1-&FIELD, for the first table, Q2-&FIELD for the second table, and so on. The COBOL fields defined in the DECLGENs are not used in the generated code in order to preserve the original location of the host variables.

SSMODE=FLAG Subscript Usage option for BL1, BL3, and PU fields:
- **FLAG** Flag as an error BL1, BL3, and PU if used in subscripts
- **GEN** Allow the use of BL1, BL3, and PU fields in subscripts

Note: Migration Utility generates special logic when accessing BL1, BL3 and PU fields. There is a substantial overhead when these fields are used as subscripts. The recommended option is FLAG.

THRESMOD=FIX Date threshold option:

FIX Fixed threshold (hard-coded at 40). This is the default. If the input date 2-digit year is less or equal to 40, the century is set to 2000. If the input date 2-digit year is greater than 40, the century is set to 1900.

Note: This option obviously has limitations and programs may have to be changed at one point to keep the proper threshold tolerance.

ROLL Rolling threshold whereby the CPU 2-digit year is added to 40. The formula is as follows:

&THRESHOLD = (40 + CPU Year)

If &THRESHOLD is less than 100:
- &ADJ1=1900
- &ADJ2=2000

If &THRESHOLD is greater than 99:
- &ADJ1=3000
- &ADJ2=2000
- &THRESHOLD = (&THRESHOLD - 100)

If the date 2-digit year is greater than &THRESHOLD, the century is set to &ADJ1.

If the date 2-digit year is less or equal &THRESHOLD, the century is set to &ADJ2.

Note: The ROLL option is valid for the entire century.

TRANSLATE WORDS

The section starting with this line is the words translate table. The full section is:

```
TRANSLATE WORDS
          (&FROMWORD-1 &TOWORD-1)
               .
               .
          (&FROMWORD-N &TOWORD-N)
END-TRANS
```

This table is used to reduce the field names to 16 characters or less. Up to 256 pairs of words can be coded. Notice that the "TRANSLATE" is not coded with an "=" like other keywords and that the list of keywords must be enclosed in parentheses (see example on the previous page).

TRANSLATE FIELDS

The section starting with this line is the fields translate table. The full section is:

```
TRANSLATE FIELDS
          (&FROMFIELD-1 &TOFIELD-1)
               .
               .
          (&FROMFIELD-N &TOFIELD-N)
END-TRANS
```

This table is used to replace ambiguous field names. The new names must be 16 characters or less. Up to 256 pairs of fields can be coded. Notice that the "TRANSLATE" is not coded with an "="

like other keywords and that the list of fields names must be enclosed in parentheses (see example on the previous page).

USERXIT= The name of the user written PEngiCCL macro to be invoked at end of job. Refer to the ″User Exits″ section in this document.

WARNDUP=GROUP

Warning message option for duplicate field definitions:

GROUP	A warning message is issued for W and S fields when the duplicate field is not within the same group
ALL	A warning message is issued for W and S fields when a duplicate field exists, and the re-definition of the field is not of the same attributes (for example, length, type)
NONE	No warning message for duplicate fields is issued

Embedding options in the program source

All EASYTRAN parameters described on the previous pages can be coded at the beginning of each Easytrieve Plus program as comments.

In addition, SQL DECLGEns can be included via the ″EASYTRAN: DCLINCL″ in a comment form to preserve the Easytrieve syntax compatibility. This is an alternative to ″SQL DCLINCL &NAME″ form of DECLGEN inclusion. One or more ″EASYTRAN: DCLINCL &NAME' statements can be included and multiple DECLGEN copybook names can be specified on a single line, each separated by at least one space. (See example below).

This method lets you mold the translating process according to program requirements. Example below demonstrates the method.

The member ″EZTEMPLE″ located in the SYS1.SFSYEZTS can be used as a template.

The keyword parameters need not be coded with an ″=″ sign; FILES=64 is the same as FILES 64.

COBOL compiler options can be supplied via the ″PROCESS″ option as shown. Multiple PROCESS options can be coded if needed.

EASYTRAN keyword parameters follow, followed by the ″TRANSLATE WORDS″ and ″TRANSLATE FIELDS″ lists. Comments can be placed along the side of each parameter (comments start with a ″*″).

The ″END-EASYTRAN″ statements must be the last statement as shown.

Since statements are commented out, they do not interfere with the Easytrieve Plus syntax.

Note: There must be at least one space between the first ″*″ and the EASYTRAN statement, otherwise the statement is treated as a comment.

```
*********************************************************************
* EASYTRAN PROCESS LIST,ADV,OPTIMIZE     * COBOL COMPILER OPTION     *
* EASYTRAN DCLINCL DCLTAB2 DCLTAB3       * SQL DECLGEN (OPTIONAL)    *
* EASYTRAN DCLINCL DCLTAB4               * SQL DECLGEN (OPTIONAL)    *
```

```
* EASYTRAN FILES  64                    * MAX NUMBER OF FILES       *
*          FSIGN  YES                   * F SIGN YES/ALL/NO         *
*          FIELDS 2000                  * MAX NUMBER OF FIELDS      *
*          COBVERBS YES                 * COBOL VERBS TABLE YES/NO  *
*          TRANSLATE WORDS              * WORDS ALTERING OPTIONS    *
*                    (BALANCE   BAL)                                *
*                    (INTEREST  INT)                                *
*          TRANSLATE FIELDS             * NAME ALTERING OPTIONS     *
*                    (COMPANY   CO )                                *
*                    (OFFICER   OFF)                                *
*                    (BR~NCH    BR )                                *
*          END-TRANS                                                *
* END-EASYTRAN                                                      *
********************************************************************

          Easytrieve statements follow here
```

Embedding options in the program source

Chapter 9. Dynamic I/O mode and PDS/PDSE support

This chapter describes Dynamic I/O mode and support for PDS and PDSE libraries.

Dynamic I/O mode

The Dynamic I/O mode resolves the input/output file record length and VSAM files key at program run time. This option is ideal for users who wish to continue using the Easytrieve Plus source for ongoing development.

How does it work?

COBOL file SELECT and FD statements are not generated in the COBOL code. Instead, a File Information Block (FIB) is generated in working storage. Record layouts are generated in the LINKAGE section with a variable tail end to provide access to defined area plus the tail (up to 32,760 maximum on MVS). A CALL to a supplied I/O module (FSDYNIO0) is generated in the place of standard COBOL I/O instructions. FSDYNIO0 determines the file organization and dynamically loads the appropriate I/O module.

Messages and returned File Status codes are COBOL-compliant. Any unrecoverable I/O errors are trapped by MVS and a standard IBM message is issued. In some instances, the Migration Utility I/O error handler for VSAM files displays VSAM feedback codes found in the RPL.

Access to SYS1.SFSYLOAD load library is needed at run time.

You can activate dynamic I/O by:

- Coding `IOMODE=DYNAM` in the EZPARAMS table to establish the default for your installation.
- Using EASYTRAN conventions to override the IOMODE set in EZPARAMS to activate Dynamic I/O for a specific program.

  Example:

  ```
  * EASYTRAN: IOMODE DYNAM
  ```

- Coding `IO FDYNIOR` on the file statement to designate a specific file for Dynamic I/O.

  Example:

  ```
  FILE FILEIN F (80) IO FDYNIOR
  ```

Dynamic I/O considerations

- Print files do not run in dynamic I/O mode.
- When sorting using the SORT statement, the input file record length must be greater than, or equal to, the record length of the output file. If this is not the case, any portion of the output file record that spans beyond the input file record length becomes inaccessible.
- COBOL must be compiled and linked as RMODE(24). AMODE can be AMODE(ANY) or AMODE(24).

Benefits of Dynamic I/O

- The VSAM file key is accessed from the catalog at run time.
- You do not need to be concerned with record length, except as noted for the SORT statement. This lets you point to different file lengths at run time, providing the data being accessed is defined in the layout.
- Migration Utility recognizes input files with record formats F, FB, V, VB, and VBS at file open time. It recognizes VSAM and undefined length files from the file definition, VS, RELATIVE, ESDS or U respectfully.
- The output file record format (RECFM) is determined by the definition in the program. However, if provided, Migration Utility extracts the record length from the JCL.

Support for PDS/PDSE libraries

Migration Utility supports access to PDS and PDSE libraries whereby selected, or all, library members can be accessed from Migration Utility programs. Many system tasks that are too complex for panel-driven utilities can be easily accomplished with a simple Migration Utility program.

Guidelines for accessing PDS/PDSE libraries

- Files are defined as PDS/PDSE files by specifying PDS file organization on the FILE statement. The I/O module determines which file it is working with.

  Note: PDS files always work in dynamic I/O mode. Migration Utility forces dynamic I/O on PDS files.
- Migration Utility assigns a field for member name (key) in working storage. The key can be accessed using the &FILE:KEY field. The key is returned for every record read.
- The record format of an input PDS can be F, FB, V, VB, VBS, or U.
- The record format of an output PDS can be F, FB, V, VB, or VBS. Migration Utility does not support output PDS or PDSE files with an undefined length.
- PDS files can be used on JOB, SORT or standard I/O (GET/ PUT and POINT) statements.
- When accessing a PDS using a JOB or SORT statement, the system positions the file using the value found in the &FILE:KEY field.
- When accessing a PDS using a GET or PUT statement, a POINT statement must be issued first to position the file to the desired member.

  Example:

  ```
  POINT FILEIN KEY EQ WS-MEMBER-NAME
  ```
- When you specify the name of a PDS or PDSE member in the &FILE:KEY, you can use an asterisk (*) as a wildcard to:
  – Specify a pattern

  or
  – Represent a string of zero or more characters

  When you specify the name of a PDS or PDSE member using one or more asterisks (*), Migration Utility selects members as follows:
  – If you specify an 8-character member name containing more than one asterisk, all members that match the pattern are selected,

  otherwise

– Members are selected using a generic compare.

For example:

If you specify...	Migration Utility selects...
*	All members
FS*	All members whose name starts with “FS”
DYN	All members whose name contains “DYN”
****A**D***	All members whose name contains an “A” in position 4 and a “D” in position 7

- The PDS directory is read by specifying the DIRECTORY option on the GET statement.

  Example:

  ```
  GET FILEIN DIRECTORY STATUS
  ```

- To create a PDS member, a POINT statement must be issued to establish reference to the output member. Thereafter, the PUT statement is used to add records to the member.

  The demonstration program TESTPDS0 shown in Figure 1 on page 132 (a copy of TESTPDS0 can be found in SYS1.SFSYEZTS) demonstrates the use of PDS files. The JCRUNPD0 job to run this program is located in SYS1.SFSYJCLS.

- Use the %COBOL technique to take advantage of STRING, UNSTRING, INSPECT, and other COBOL instructions to perform the parsing.

```
************************************************************************
* EASYTRAN: PROCESS LIST,ADV,OPTIMIZE     * COMPILER OPTIONS         *
* EASYTRAN: IOMODE DYNAM                  * I/O MODE                 *
* END-EASYTRAN                                                       *
************************************************************************
* TESTPDS0: EASYTRIEVE PROGRAM ACCESSING PDS/PDSE LIBRARIES.         *
*                                                                    *
* THIS PROGRAM DEMONSTRATES:                                         *
*                                                                    *
* 1. HOW TO USE POINT TO ESTABLISH WILD CARD SELECTION               *
* 2. HOW TO READ PDS/PDSE DIRECTORY WITH WILD CARD                   *
* 3. HOW TO COPY PDS MEMBERS FROM INPUT FILE TO OUTPUT FILE          *
* 4. HOW TO OBTAIN DATASET NAME VIA A CALL TO FSDYNDSN PROGRAM       *
*                                                                    *
*  INPUT: FILEIN - PDS/PDSE LIBRARIE(S)                              *
*           PARM - WILD CARD FOR MEMBER SELECTION                    *
*                                                                    *
* OUTPUT: PDSOUT  - PDS/PDSE LIBRARY OF COPIED MEMBERS               *
*         REPORT1 - STATISTICAL DATA AND LIST OF SELECTED PROGRAMS   *
*                                                                    *
* NOTES: THIS IS A DEMO PROGRAM. USE YOUR OWN IMAGINATION TO DESIGN  *
*        PROGRAMS THAT ACCOMMODATE YOUR NEEDS.                       *
************************************************************************

FILE REPORT1 PRINTER
FILE FILEIN  PDS F (80)
ITEXT        1   5   A

FILE PDSOUT  PDS F (80)
OTEXT        1   5   A

WS-PDS-MEMBER   W    8 A  VALUE ' '
WS-SAV-MEMBER   W    8 A  VALUE ' '
WS-COUNT        W    8 N  VALUE 0
WS-RCOUNT       W    8 N  VALUE 0
WS-DSNAME       W   44 A

JOB INPUT NULL
*--------------------------------------------------------------------*
* OBTAIN PARM VALUE FROM THE EXEC STATEMENT FOR GENERIC PROCESS.     *
*--------------------------------------------------------------------*
%GETPARM WS-PDS-MEMBER 8
DISPLAY REPORT1 'PARM VALUE: ' WS-PDS-MEMBER

** OBTAIN INPUT FILE DATASET NAME
%GETDSN 'FILEIN' WS-DSNAME
DISPLAY REPORT1 NEWPAGE 'GETDSN: ' WS-DSNAME
DISPLAY REPORT1 ' '

*--------------------------------------------------------------------*
* READ PDS/PDSE DIRECTORY AND CREATE ADD STATEMENTS TO REPORT1.      *
*--------------------------------------------------------------------*
POINT FILEIN EQ WS-PDS-MEMBER STATUS
DO WHILE FILEIN:FILE-STATUS EQ 0
   GET FILEIN DIRECTORY STATUS
   IF NOT EOF FILEIN
        WS-COUNT = WS-COUNT + 1
        DISPLAY REPORT1 'FILEIN MEMBER=' FILEIN:KEY
   END-IF
END-DO
DISPLAY REPORT1 ' '
DISPLAY REPORT1 'TOTAL PDS MEMBERS LOCATED: ' WS-COUNT
DISPLAY REPORT1 SKIP 2
```

Figure 1. PDS/PDSE program example (Part 1 of 2)

```
*--------------------------------------------------------------------*
* LIST ALL MEMBERS IN INPUT PDS/PDSE SELECTED BY THE PARM AND COPY   *
* ALL SELECTED MEMBERS TO PDSOUT PDS FILE.                           *
*--------------------------------------------------------------------*
** OBTAIN OUTPUT PDSOUT DATASET NAME
%GETDSN 'PDSOUT' WS-DSNAME
DISPLAY REPORT1 NEWPAGE 'PDSOUT: ' WS-DSNAME

WS-COUNT = 0
POINT FILEIN EQ WS-PDS-MEMBER STATUS
DO WHILE FILEIN:FILE-STATUS EQ 0
   GET FILEIN STATUS
   IF NOT EOF FILEIN
      IF WS-SAV-MEMBER NE FILEIN:KEY
        WS-COUNT = WS-COUNT + 1
        DISPLAY REPORT1 'PDSOUT MEMBER=' FILEIN:KEY
        WS-SAV-MEMBER = FILEIN:KEY
        POINT PDSOUT EQ FILEIN:KEY
      END-IF
      PUT PDSOUT FROM FILEIN
   END-IF
END-DO
DISPLAY REPORT1 ' '
DISPLAY REPORT1 'TOTAL PDS MEMBERS CREATED: ' WS-COUNT
DISPLAY REPORT1 ' '
STOP
*---------------------- END OF PROGRAM -----------------------------*
```

Figure 1. PDS/PDSE program example (Part 2 of 2)

Chapter 10. Toolkit replacement macros

This chapter describes Toolkit and date-handling replacement macros, and enhanced date threshold handling.

Toolkit and date-handling replacement macros

Easytrieve Plus provides some special macros for date calculation and some more commonly-used functions. These macros are known as "Toolkit" macros.

Migration Utility provides the Toolkit replacement macros listed below. The macros are written in CCL1 language (PEngiCCL). Macros are DISTRIBUTED in byte code and macro source in the SYS1.SFSYFJCC and SYS1.SFSYCCLM libraries respectfully.

All date routines use the Gregorian Leap Year formula as follows:

The year is a leap year if:
- It is divisible by 4 and not divisible by 100,

or
- It is divisible by 400

The following Toolkit date-replacement macros are provided:

ALPHACON Unstring a edited number into an internal numeric format

CONVAE Convert ASCII to EBCDIC

CONVEA Convert EBCDIC to ASCII

DATECALC Add or subtract a number of days to a date

DATECONV Convert a date of any format to any format

DATEVAL Date validation

DATEVALE A Migration Utility special macro called by DATEVAL (internal use only)

DATEMASK A Migration Utility special macro called by all date macros (internal use only)

DAYSAGO Calculate days elapsed from User date to today

DAYSCALC Calculate the number of days between two dates

GETDATE Get the 6-digit system date

GETDATEL Get the 8-digit system date

WEEKDAY Obtain the name of the day of the week (for example, MONDAY)

The following Toolkit special purpose replacement macros are provided:

DIVIDE Module N division

EXPO Exponentiate a number

NUMTEST Test field for numeric, and count bad fields

RANDOM	Random number generator
SQRT	Square root calculation
UNBYTE	Decode a byte into 8 bytes of 0 and 1 flags

The following special purpose Migration Utility macros are provided:

GETDSN	Get data set name from the JCL
GETJOB	Obtains JOB number and TSO User from the Job Scheduler
GETPARM	Get PARAMETERS from the PARM= statement in the JCL
PARSE	Special parsing macro (uses the COBOL UNSTRING statement)

The coding conventions for the above macros are described later in this chapter.

The date macros generate code that call FSDATEZ0 and FSDATSRV modules at run time. Access to SYS1.SFSYLOAD is needed at run time.

Macros search sequence

Migration Utility normally obtains Easytrieve Macros from libraries defined for the FJCPYLB DDname in the FSCCL1 step of your JCL.

Make sure that the SYS1.SFSYCCLM PDS is concatenated in FJCPYLB and FJMACLB before any other Easytrieve Plus macro libraries. It is used as a finder PDS for replacement macros.

When Migration Utility locates a macro in FJCPYLB, it determines the macro format (Easytrieve or CCL1) and invokes the appropriate interpreter.

CCL1 macros are executed from the byte code library (FJCCLLB) if found there. Otherwise, the copy found in FJMACLB SYS1.SFSYCCLM source is used.

Enhanced date threshold handling

Migration Utility provides enhanced handling of the date threshold for deriving the century from a 2-digit year. It provides for a *fixed* threshold or a *rolling* threshold.

Note: All Migration Utility date calculation macros default to THRESHOLD 0.

The program default date threshold options are generated in the COBOL program at translation time as follows:

- If the THRESHOLD value coded in the date macros (default or user-supplied) is not zero, the macro threshold value is used.
- If the THRESHOLD value coded in the date macros (default or user-supplied) is equal to zero, the EZPARAMS/EASYTRAN coded threshold value is used as follows:

  `THRESMOD=FIX/ROLL`

  – FIX for fixed threshold (hard-coded at 40). This is the default.

    If the input date 2-digit year is less or equal to 40, the century is set to 2000.

    If the input date 2- digit year is greater than 40, the century is set to 1900.

    Note: This option obviously has limitations and programs may have to be changed to maintain the correct threshold tolerance.

– ROLL for the rolling threshold whereby the CPU 2-digit year is added to 40.
  &THRESHOLD = (40 + CPU year)
  If &THRESHOLD is less than 100:
  - &ADJ1=1900
  - &ADJ2=2000

  If &THRESHOLD is greater than 99:
  - &ADJ1=3000
  - &ADJ2=2000
  - &THRESHOLD = (&THRESHOLD - 100)

  If the input date 2-digit year is greater than &THRESHOLD, the century is set to &ADJ1.

  If the input date 2-digit year is less or equal &THRESHOLD, the century is set to &ADJ2.

  Note: The ROLL option accommodates this century without having to change programs.

- Runtime options allow you to override the hard-coded threshold generated in the program by coding DD statements in the application run JCL:

//FJTHRES0 DD DUMMY	Forces a fixed threshold of 40
//FJTHRES1 DD DUMMY	Forces a rolling threshold of (40 + CPU two digit year)

The search priority is FJTHRES0 but, if not present, FJTHRES1 is used.

Available date masks

The special macro, DATEMASK, changes the user-supplied mask to a "proper" mask that is understood by FSDATEZ0 and FSDATSRV (Migration Utility) programs.

The DATEMASK macro is invoked internally by all DATE macros. Do not use it in stand-alone mode.

The available masks are shown below. COL1 shows the user masks; COL2 shows the equivalent masks understood by the FSDATEZ0 program. Any "YYYY" found in the macros is changed to "CCYY".

```
 COL1     COL2

MMDDYY   MMDDYY,    .STANDARD FORMATS FOR FSDATEZ0
MMDDCCYY MMDDCCYY
MDY      MMDDYY
MDCY     MMDDCCYY

MMYYDD   MMYYDD,    .SPECIAL FORMATS FSDATEZ0
MMCCYYDD MMCCYYDD
MYD      MMYYDD
MCYD     MMCCYYDD

DDMMYY   DDMMYY,    .SPECIAL FORMATS FOR FSDATEZ0
DDMMCCYY DDMMCCYY
DMY      DDMMYY
DMCY     DDMMCCYY

DDYYMM   DDYYMM,    .SPECIAL FORMATS FOR FSDATEZ0
DDCCYYMM DDCCYYMM
```

Enhanced date threshold handling

```
DYM      DDYYMM
DCYM     DDCCYYMM

YYMMDD   YYMMDD,    .STANDARD FORMAT FOR FSDATEZ0
CCYYMMDD CCYYMMDD
YMD      YYMMDD
CYMD     CCYYMMDD

YYDDMM   YYDDMM,    .SPECIAL FORMAT FOR FSDATEZ0
CCYYDDMM CCYYDDMM
YDM      YYDDMM
CYDM     CCYYDDMM

YYDDD    YYDDD,     .JULIAN STANDARD FORMAT
CCYYDDD  CCYYDDD
YD       YYDDD
CYDDD    CCYYDDD
```

ALPHACON macro: coding rules

Purpose: Unstrings an edited number into a numeric field suitable for arithmetic.

Usage: %ALPHACON &alphafield &numfield DECIMAL ('&dec') CALLCOUNTER (&count)

where:

&alphafield	An alpha field that contains an edited number.
&numfield	A receiving numeric field.
&dec	Decimal point (character). The default is "." (period).
&count	Call counter (not used by Migration Utility). The default is 1.

Notes: The ALPHACON macro invokes the FSDIMAGE program at run time dynamically. Access to the SYS1.SFSYLOAD library is required at run time. On completion, the ALPHACON-FLAG field contains:

YES	Successful conversion
LEFT	Integer portion cannot fit into the &numfield
RIGHT	Decimal portion cannot fit into the &numfield decimals
BOTH	Neither the integer or the decimals can fit into the &numfield

Examples:

```
DEFINE WS-EDITED-FIELD W 15 A VALUE '123.55'
DEFINE WS-NUMERIC-FIELD W 9 N 2
⋮
%ALPHACON WS-EDITED-FIELD W S-NUMERIC-FIELD DECIMAL ('.')
```

CONVAE macro: coding rules

Purpose: Converts ASCII characters to EBCDIC.

Usage: %CONVAE &file STARTPOS &field LENGTH &length

where:

&file	A file name. If coded, file record is used.
&field	Field to be converted.
&length	Field length.

Notes: The CONVAE macro invokes the FSDYNCNV program at run time dynamically. Access to the SYS1.SFSYLOAD library is required at run time.

Examples:

```
DEFINE FILEIN F (500)
⋮
%CONVAE FILEIN LENGTH (500)


DEFINE WS-ALPHA-FIELD W 15 A VALUE 'ABCDE'
⋮
%CONVAE STARTPOS WS-ALPHA-FIELD LENGTH (15)
```

CONVEA macro: coding rules

Purpose: Convert EBCDIC characters to ASCII.

Usage: %CONVEA &file STARTPOS &field LENGTH &length

where:

&file	A file name. If coded, file record is used.
&field	Field to be converted.
&length	Field length.

Notes: The CONVAE macro invokes the FSDYNCNV program at run time dynamically. Access to the SYS1.SFSYLOAD library is required at run time.

Examples:

```
DEFINE FILEIN F (500)
⋮
%CONVEA FILEIN LENGTH (500)

DEFINE WS-ALPHA-FIELD W 15 A VALUE 'ABCDE'
⋮
%CONVEA STARTPOS WS-ALPHA-FIELD LENGTH (15)
```

DATECALC macro: coding rules

Purpose: Adds or subtracts a number of days to any date and places the result into the target user date of any format.

Usage: %DATECALC &fdate &fmask &sign &days &tdate &tmask THRESHOLD YY

where:

&fdate	The input date.
&fmask	Input date format. For example, MMDDYY.
&sign	PLUS when adding days, MINUS when subtracting days.
&days	Number of days to add or subtract.
&tdate	Output date.

&tmask	Output date format. For example, YYMMDD.
THRESHOLD YY	Threshold year. This is an optional parameter. The default is 00.

Notes: Coding 0 for the threshold value for this macro results in the usage of the default threshold value in the FSDATSRV module. Currently, the default threshold in FSDATSRV program is 40. The threshold of zero is recommended. Refer to the THRESMOD= option of EZPARAMS/EASYTRAN for additional ROLLING or FIXED threshold flexibility.

Examples:

```
%DATECALC F-DATE YYMMDD PLUS 15 T-DATE MMDDYY

%DATECALC F-DATE YYMMDD PLUS 15 T-DATE MMDDYY THRESHOLD 40
```

DATECONV macro: coding rules

Purpose: Converts dates from any format to any other format.

Usage: %DATECONV &fdate &fmask &tdate &tmask THRESHOLD YY

where:

&fdate	The input date.
&fmask	Input date format. For example, MMDDYY.
&tdate	Output date.
&tmask	Output date format. For example, YYMMDD.
THRESHOLD YY	Threshold year. This is an optional parameter. The default is 00.

Notes: Coding 0 for the threshold value for this macro results in the usage of the default threshold value in the FSDATSRV module. Currently, the default threshold in FSDATSRV program is 40. The threshold of zero is recommended. Refer to the THRESMOD= option of EZPARAMS/EASYTRAN for additional ROLLING or FIXED threshold flexibility.

Examples:

```
%DATECONV F-DATE YYMMDD T-DATE MMDDYY

%DATECONV F-DATE YYMMDD T-DATE MMDDYY THRESHOLD 40
```

DATEVAL macro: coding rules

Purpose: Validates input date for the given mask.

Usage: %DATEVAL &fdate &fmask THRESHOLD YY

where:

&fdate	The input date.
&fmask	Input date format. For example, MMDDYY.
THRESHOLD YY	Threshold year. This is an optional parameter. The default is 00.

Notes: Coding 0 for the threshold value for this macro results in the usage of the default threshold value in the FSDATSRV module. Currently,

the default threshold in FSDATSRV program is 40. The threshold of zero is recommended. Refer to the THRESMOD= option of EZPARAMS/EASYTRAN for additional ROLLING or FIXED threshold flexibility.

On completion, the DATEVAL-FLAG contains 'YES' for valid date, and 'NO' for invalid date. The flag can be tested and programming decisions can be made based on the outcome.

Examples:

```
%DATEVAL I-DATE YYMMDD

IF DATEVAL-FLAG EQ 'YES'
 .
 .
END-IF

%DATEVAL I-DATE CCYYMM THRESHOLD 50

IF DATEVAL-FLAG EQ 'YES'
 .
 .
END-IF
```

DAYSAGO macro: coding rules

Purpose: Calculates the number of days elapsed between two dates.

Usage: %DAYSAGO &date &format &operator &operand THRESHOLD YY

where:

&date	The input date.
&format	Input date format. For example, MMDDYY.
&operator	Code relational operator: EQ, =, NE, GT, ,LT, GE, LE
&operand	A numeric field name or constant to compare with. This value is the number days that you wish to verify.
THRESHOLD YY	Threshold year. This is an optional parameter. The default is 00.

Notes: Coding 0 for the threshold value for this macro results in the usage of the default threshold value in the FSDATSRV module. Currently, the default threshold in FSDATSRV program is 40. The threshold of zero is recommended. Refer to the THRESMOD= option of EZPARAMS/EASYTRAN for additional ROLLING or FIXED threshold flexibility.

On completion, the DAYSAGO-FLAG contains "YES" if the criteria is met, otherwise it contains "NO". The flag can be tested and programming decisions can be made based on the outcome.

The DAYSAGO-DIFF field contains the number of days between today's CPU date and the input date. If the input date is higher than the CPU date the value returned in the DAYSAGO-DIFF will be negative.

Examples:

```
%DAYSAGO I-DATE MMDDYY EQ 15

IF DAYSAGO-FLAG EQ 'YES'
 .
 .
END-IF

%DAYSAGO I-DATE CCYYMMDD EQ 30 THRESHOLD 45
```

DAYSCALC macro: coding rules

Purpose: Calculates the number of elapsed days between two dates.

Usage: %DAYSCALC &fdate &fmask &tdate &tmask &result THRESHOLD NN

where:

&fdate	The input date.
&fmask	Input date format. For example, MMDDYY.
&tdate	Output date.
&tmask	Output date format. For example, YYMMDD.
&result	A numeric field for returned number of days.
THRESHOLD YY	Threshold year. This is an optional parameter. The default is 00.

Notes: Coding 0 for the threshold value for this macro results in the usage of the default threshold value in the FSDATSRV module. Currently, the default threshold in FSDATSRV program is 40. The threshold of zero is recommended. Refer to the THRESMOD= option of EZPARAMS/EASYTRAN for additional ROLLING or FIXED threshold flexibility.

The &result field contains the number of days between &fdate and &tdate dates. If the &tdate is higher than the &fdate, the value returned in the &result is negative.

Examples:

```
%DAYSCALC F-DATE MMDDYY T-DATE YYMMDD WS-DAYS

%DAYSCALC F-DATE MMDDYY T-DATE YYMMDD WS-DAYS THRESHOLD 45
```

DIVIDE macro: coding rules

Purpose: Divides an input number, giving a quotient and a remainder.

Usage: %DIVIDE &number &divisor &quotient &remainder

where:

&number	A numeric field to be divided.
&divisor	The divisor.
&quotient	Output quotient numeric field.
&remainder	Output remainder numeric field.

Example:

```
%DIVIDE I-NUMBER 15 O-QUOTIENT O-REMAINDER
```

EXPO macro: coding rules

Purpose: Exponentiates a number.

Usage: %EXPO &value &exponent &result

where:

&value	The numeric field to be exponentiated.
&exponent	The exponent.
&result	The outcome of the exponentiation.

Notes: This macro uses the COBOL COMPUTE statement to perform the exponentiation.

Migration Utility provides an alternative way to code an assign statement using ** for exponentiation.

Example:

```
%EXPO I-NUMBER 3.5 O-RESULT
```

GETDATE macro: coding rules

Purpose: Gets a 6-digit current date in numeric format (without insert characters).

Usage: %GETDATE &date

where:

&date	A numeric field for the retrieved date.

Notes: The returned date is the date retrieved at the program start-up time in YYMMDD format.

Example:

```
%GETDATE WS-DATE
```

GETDATEL macro: coding rules

Purpose: Gets an 8-digit current date in numeric format (without insert characters).

Usage: %GETDATEL &date

where:

&date	A numeric field for the retrieved date.

Notes: The returned date is the date retrieved at the program start-up time in CCYYMMDD format.

Example:

```
%GETDATEL WS-DATE-LONG
```

GETDSN macro: coding rules

Purpose: Obtains the data set name for a specified DDname (MVS only).

Usage: %GETDSN &DDname &dsname

where:

&DDname	File DDname (1 to 8 characters).
&dsname	A 44-byte field for the retrieved data set name.

Notes: GETDSN macro invokes FSDYNDSN program at run time dynamically. Access to SYS1.SFSYLOAD library is required at run time.

On completion, RETURN-CODE can be tested for a successful call:

- When RETURN-CODE equals zero, the &DDname was located in the JCL and was placed into the &dsname field.
- When RETURN-CODE is not equal to zero, &DDname is not in the JCL. The &dsname is cleared to spaces.

Example:

```
DEFINE WS-DSNAME    W   44  A

%GETDSN 'FILEIN' WS-DSNAME
```

GETJOB macro: coding rules

Purpose: Obtains Job Number and TSO User from the JOB Scheduler Information Block.

Usage: %GETJOB

Notes: The GETJOB macro invokes the FSYGJOB0 program at run time dynamically. Access to the SYS1.SFSYLOAD library is required at run time.

On completion, the following information is available:

```
DEFINE GETJOB-DATA  W 80 A
DEFINE GETJOB-WORKID  GETJOB-DATA +00 8 A
DEFINE GETJOB-JOBID   GETJOB-DATA +08 8 A
DEFINE GETJOB-JOBNAME GETJOB-DATA +16 8 A
DEFINE GETJOB-JOBSTEP GETJOB-DATA +24 8 A
DEFINE GETJOB-PREFIX  GETJOB-DATA +32 8 A
DEFINE GETJOB-UERID   GETJOB-DATA +40 8 A
```

Example:

```
%GETJOB
```

GETPARM macro: coding rules

Purpose: Gets parameter information from the EXEC PARM= statement in the JCL.

Usage: %GETDATEL &field &length

where:

&field	A field to hold parameter information.
&length	The length of the field.

Notes: The PARM information is moved from the system area passed to the COBOL program via the LINKAGE SECTION into the designated field.

Example:

```
DEFINE WS-PARAMETER W   8  A

%GETPARM WS-PARAMETER 8
```

NUMTEST macro: coding rules

Purpose: Tests a field for numeric content, and counts the number of non-numeric occurrences.

Usage: %NUMTEST &field &description &field-id

where:

&field	Input field to be tested.
&description	Description for the DISPLAY message when not numeric.
&field-id	Identifier for the DISPLAY message when not numeric.

Notes: On completion, the NUMTEST-FLAG contains "YES" when the field is numeric, otherwise it contains "NO". The flag can be tested and programming decisions can be made based on the outcome.

Examples:

```
%NUMTEST I-FIELD 'NUMERIC FIELD TEST' 'I-FIELD'

IF NUMTEST-FLAG NE 'YES'
   .
   .
 END-IF
```

PARSE macro: coding rules

Purpose: Parses a string and places the contents into an array of strings with the corresponding lengths.

Usage: %PARSE &string &into &occurs DELIM(&delim) PFX(&pfx) SIZE(&size)

where:

&string	A quoted string or a field to parse. This is a required parameter.
&into	Name of the array (tokens) to parse into. This is a required parameter.
&occurs	Maximum number of words (tokens). This is a required parameter
&delim	Delimiter character(s). The default is spaces
&pfx	Prefix for &into for unique array names. The default is no prefix.
&size	Maximum size of each word (token). The default is 80.

Notes: Parsing is done using the UNSTRING COBOL statement. Each parsed word is placed into the &pfx&into array and the length into the corresponding &pfx&into-LEN field. Working storage is generated for each unique &pfx&into array as follows:

```
DEFINE  &PFX&INTO         W &SIZE A OCCURS &OCCURS
DEFINE  &PFX&INTO.-LEN    W  2    B 0 OCCURS &OCCURS
DEFINE  &PFX&INTO.-COUNT  W  4    B 0
DEFINE  &PFX&INTO.-ERRCD  W  4    B 0
```

On completion:

- &PFX&INTO array contains the extracted words.
- &PFX&INTO-LEN contains the length of each word respectfully.
- &PFX&INTO-COUNT field contains the number of extracted words.
- &PFX&INTO-ERRCD contains the error code (currently always set to zero).

Examples:

```
%PARSE I-STRING TOKEN 50 DELIM (PARSE-CHAR)

%PARSE I-STRING TOKEN 80 PFX('A') SIZE(30)
```

If the input string is a subscripted field, enclose the &string in quotes with the necessary subscript:

```
%PARSE 'I-STRING (SUB1)' TOKEN 80 DELIM (',' '$' '!') PFX('B') SIZE(45)
```

RANDOM macro: coding rules

Purpose: Generates a random number (of 1 to 18 digits) based on an initial random SEED.

Usage: %RANDOM &rand &seed &digits

where:

&rand	A numeric field in which the random number is returned.
&seed	Random function seed (must be a number or a numeric field).
&digits	The length of the returned &rand number.

Notes: This macro uses the COBOL RANDOM function number generator.

SQRT macro: coding rules

Purpose: Calculates the square root of a number.

Usage: %SQRT &number &result

where:

&number	The input numeric field.
&result	A numeric field for output.

Notes: This macro uses the COBOL COMPUTE statement to perform the exponentiation.

Migration Utility provides an alternative way to code an assign statement using ** for exponentiation.

Example:

```
%SQRT I-NUMBER .5 O-RESULT
```

UNBYTE macro: coding rules

Purpose: Generates 8 digits of “0” or “1” representing each bit in the input byte (using left-to-right decoding).

Usage: %UNBYTE &ibyte

where:

&ibyte	One-byte input field to be decoded.

Notes: On completion, the field names BIT0, BIT1, BIT2, BIT3, BIT4, BIT5, BIT6 and BIT7 correspond to the byte bits of the input field. ALLBITS is the 8-byte group field name for these elementary fields.

Example:

```
%UNBYTE I-BYTE

IF BIT0 EQ 1 OR BIT7 EQ 0
.
.
.
END-IF
```

WEEKDAY macro: coding rules

Purpose: Validates the given date and returns the name of the day of the week.

Usage: %WEEKDAY &fdate &fmask &day THRESHOLD NN

where:

&fdate	The input date.
&fmask	Input date format. For example, MMDDYY.
&day	An alphanumeric field for the returned name of the day of the week.
THRESHOLD YY	Threshold year. This is an optional parameter. The default is 00.

Notes: Coding 0 for the threshold value for this macro results in the usage of the default threshold value in the FSDATSRV module. Currently, the default threshold in FSDATSRV program is 40. The threshold of zero is recommended. Refer to the THRESMOD= option of EZPARAMS/EASYTRAN for additional ROLLING or FIXED threshold flexibility.

On completion, the &day field contains the day of the week (such as MONDAY, TUESDAY, and so on).

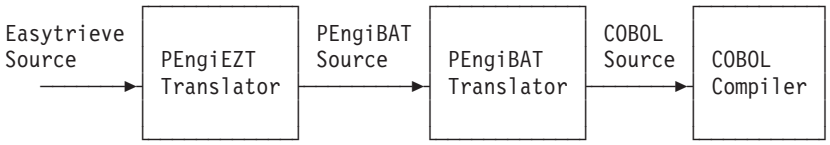
Examples:

```
%WEEKDAY I-DATE MMDDYY WS-WEEK-DAY

%WEEKDAY I-DATE MMDDYY WS-WEEK-DAY THRESHOLD 40
```

Chapter 11. Messages

Migration Utility works in two steps:

1. The PEngiEZT translator converts Easytrieve source files to PEngiBAT source files.
2. The PEngiBAT translator translates the PEngiBAT files to COBOL source files.



These two steps relate to the error messages that Migration Utility produces, and the action that you take in response to the messages.

Some of the messages produced by the first step are described in “Migration Utility (macro) generated error messages” on page 151. These messages relate to user/syntax problems. The description of the message also gives pointers as to how you fix the problem.

The rest of the messages produced by the first step are described in “PEngiCCL generated messages” on page 199. From the description of the message, you will have to determine whether you caused the message with a bad syntax (in which case you can fix the problem), or whether the problem is an error in macro definition. If the latter, then you need to report the problem to IBM.

Messages produced by the second step are described in “Migration Utility macro generated messages” on page 165 and “Migration Utility function generated messages” on page 188. In Migration Utility, you cannot intervene in Step 2 to produce errors, so any messages that are reported must relate to an error produced by Step 1. This should not happen. If you have a message reported by Step 2, then you need to report the problem to IBM.

The PEngiEZT error messages are preceded by the word *ERROR*. The messages are described in the error number sequence. The word "*ERROR*" and the condition code are not shown as part of the message since they do not change.

```
*ERROR* EZT000-01,012  MAXIMUM OF NN OBJECTS EXCEEDED
          |      |                |
          |      |                |
        Error  Condition      Message
        Number Code             Text
```

Messages are included in the SYSOUT file produced by the PEngiCCL preprocessor. Every message is written in two places:

- Immediately following the statement or macro that caused the error.
- At end of the listing, showing the page and the line number of the statement in error

The first page of the PEngiCCL preprocessor program listing contains the preprocessor options in effect and the errors summary, that is, the highest severity code and the number of errors detected during preprocessing.

To check for errors, look at the error summary on the first page of the preprocessor program listing. If the highest error severity code and the number of errors detected are not zero, then you had errors.

To locate errors, you can either scroll to the last page of the listing where the errors are shown and use the statement and/or page number to locate the actual error message and the statement in error, or you can browse through the listing.

Error messages are displayed following the statement or macro in error.

It is possible to get PEngiCCL preprocessor messages due to the previously detected errors. You should resolve all obvious errors by elimination process first. PEngiCCL preprocessor errors are usually caused by problems such as long data strings, missing parameters, null data strings, and so on.

MNOTEs (Warnings) are of informational nature. They do not inhibit code generation.

PEngiCCL error messages are composed of the error number, error severity code and a descriptive message. These messages are described in “PEngiCCL generated messages” on page 199, in error number sequence. Typically, each PEngiCCL message text includes a supplement text, up to 12 characters long, of the data string in error. The supplement text is separated from the message by a “:”.

```
DEFCOM-01,012  -TEXT-:INPUT DATA LENGTH IS ZERO
|          |        |
|          |        |
Error  Severity   Supplement and Message
Number   Code              Text
```

PEngiCCL (macro) and Function error messages are in the form of PEngiCCL Mnote (Macro Note). That is, the messages are preceded by the word **MNOTE**. These messages are described in “Migration Utility macro generated messages” on page 165 and “Migration Utility function generated messages” on page 188, in error number sequence. The word "**MNOTE**" and the condition code are not shown as part of the message since they do not change.

```
**MNOTE** 012 DCCL-01  MAXIMUM OF NN OBJECTS EXCEEDED
   |          |                        |
   |          |                        |
  Condition   Error                Message
       Code   Number                 Text
```

Error messages are displayed following the statement or macro in error. Use the index (in the back of this book) to locate the message.

Macro instructions and functions are imbedded in the program source with respective parameters. Errors can be detected during parameter collection or during the execution of the macro(s).

When collecting macro parameters, the PEngiCCL macro processor collects all macro parameters, bound by the Macro Start (_ or EXCCL) and the Macro End (;) delimiter, before it gives control to each macro for processing. The syntax errors are detected and displayed during the parameter collection process.

When collecting Function parameters, the PEngiCCL function processor collects all function parameters, bound by the paired parentheses following the function name, before it gives control to each function for processing. The syntax errors are detected and displayed during the parameter collection process.

All other errors are detected during the macro execution. Thus, errors are displayed following the last macro parameter of each macro invocation.

Note 1: All PEngiCCL preprocessor messages are included in "PEngiCCL generated messages" on page 199 for convenience. Since the PEngiCCL preprocessor is a macro interpreter, most messages are related to the interpretation of the macro directives imbedded in the PEngiCCL macros. Such messages are encountered during the development of the new PEngiCCL macros.

Note 2: It is possible to get PEngiCCL preprocessor messages due to the previously detected errors or MNOTES. You should resolve all obvious errors by elimination process first. PEngiCCL preprocessor errors are usually caused by fairly obvious mistakes such as long data strings, missing parameters, or null data strings. If you still cannot resolve a PEngiCCL preprocessor error after eliminating all MNOTES and obvious errors, please contact the IBM service center.

Note 3: The most frequent errors are caused by the misplaced macro-end-delimiter (:), or by data placed in column 72, or before column 12. Some errors can be caused by unpaired quotes or parentheses. To solve the problem please check:

1. Every macro instruction must be terminated by a ":".
2. Macro parameters can be coded following the macro instruction name on the same line, or starting in column 12 on subsequent lines. Column 72 is used as continuation byte. Do not code any data in CC 72 unless you are intending to continue a quoted string.
3. Quoted strings must contain paired quotes. If you need a quote as a data item, code double quotes.
4. Bracketed parameters must contain paired brackets. The translator searches all parameters until a paired bracket is found, which may cause parsing of unintended strings that follow macro parameters.

Migration Utility (macro) generated error messages

EZT000-00 &text

Explanation: This message is a generic message for errors detected while interpreting the EASYTRAN/EZPARAMS parameters.

User Response: The &text is self-explanatory. Make necessary changes as needed.

EZT000-00 MAXIMUM OF 256 TRANSLATE WORDS EXCEEDED

Explanation: The number of translate words exceeds maximum of 256.

User Response: Limit the number of translate words in EZPARAMS member to maximum of 256.

EZT000-01 *NN* **:ILLEGAL NUMBER OF DECIMAL PLACES**

Explanation: The specified number of decimal places is illegal as written.

User Response: Make sure that the number of decimal places is numeric and less than 18.

EZT000-02 &WORD :NOT SUPPORTED BY THE TRANSLATOR

Explanation: The displayed statement is not supported by PEngiEZT.

User Response: Correct or remove the erroneous statement.

EZT000-03 &WORD :STATEMENT ILLEGAL OR OUT OF SEQUENCE

Explanation: The displayed statement is illegal or out of sequence. Possible causes are:
- Missing PROC before Report Exits
- Illegal Report Exit Name
- ENDPROC not preceded by a PROC
- HEX display specified in Report Exits
- HEX mask followed by extraneous parameters
- DEFINE used inside a JOB (Define is not supported inside a job)
- Misplaced Field Qualifier in field definition
- Table entry contains too many arguments
- RESET specified for non-work field

User Response: Correct or remove the erroneous statement.

EZT000-04 &FIELD :UNSUPPORTED FIELD CLASS

Explanation: Field class is not A, N, P, B, K or U.

User Response: Enter the correct field class.

EZT000-05 &WORD :UNKNOWN OR INCOMPLETE STATEMENT

Explanation: Statement preceding the message is incomplete or not an Easytrieve statement.

User Response: Correct the erroneous statement.

EZT000-06 &WORD :ILLEGAL FIELD POSITION

Explanation: Field position is not numeric or name referenced is undefined.

User Response: Correct the erroneous statement.

EZT000-07 &WORD :ILLEGAL OCCURS OR INDEX STATEMENT

Explanation: Missing or non-numeric duplication factor, or missing index name.

User Response: Correct the erroneous statement.

EZT000-08 &WORD :ILLEGAL BINARY FIELD MEMORY SIZE

Explanation: Binary field size is not 1, 2, 3, or 4.

User Response: Correct the erroneous statement.

EZT000-09 &WORD :FILE WAS NOT DEFINED

Explanation: File to COPY was not defined.

User Response: Correct the erroneous statement.

EZT000-10 &WORD :MISPLACED OR UNSUPPORTED MASK

Explanation: Field MASK is not supplied or it is illegal.

User Response: Code the correct field mask.

EZT000-11 &WORD :MASK ID WAS PREVIOUSLY DEFINED

Explanation: Mask-id was previously defined.

User Response: Remove duplicate definition or assign a new Mask-ID.

EZT000-12 MAXIMUM OF *NN* MASK IDS EXCEEDED

Explanation: Translator supports maximum of *NN* Mask-IDS.

User Response: Resort to MASK usage to reduce the number of Mask-IDS.

EZT000-13 BWZ OPTION SPECIFIED FOR NON NUMERIC FIELD

Explanation: None (unused message).

User Response: None.

EZT000-14 &FIELD :FIELD IS OUTSIDE OF GROUP RANGE

Explanation: The field is a member of a group definition but its starting position plus the length would exceed the Group length.

User Response: Adjust the Group Field size to accommodate your field size.

EZT000-15 &GFIELDS FIELD NAMES EXCEEDED

Explanation: The number of program fields exceeds the number of fields specified by the FIELDS=NN of the EASYTRAN macro (see Chapter 8, “Installation and Migration Utility options”, on page 115).

User Response: Increase FIELDS=NN parameter on EASYTRAN macro to accommodate your needs.

EZT000-16 COPY IS NOT SUPPORTED FOR TABLE FILES

Explanation: A COPY was specified for an external table file.

User Response: Migration Utility does not support COPY for external tables. Replace COPY by an ARG and DESC fields.

EZT000-16 INDIRECT COPY OR INCONSISTENT FILE ATTRIBUTES

Explanation: The file referenced by the COPY was a copy file, or its attributes are not consistent with the attributes of the current file.

User Response: Correct the erroneous statement.

EZT000-17 MAXIMUM OF N'&PREFIX COPIES EXCEEDED

Explanation: Number of COPY files exceeded nn.

User Response: Resort to other methods of defining your files.

EZT000-18 &WORD FILE WAS PREVIOUSLY DEFINED

Explanation: Duplicate file name.

User Response: Choose a unique file name.

EZT000-18 &FILE :FILE CONFLICTS WITH FIELD NAME

Explanation: A field exists that conflicts with &FILE name (duplicate name).

User Response: File and field names must be unique in COBOL. Assign a unique file name and change all references in the program to the new name.

EZT000-19 &WORD :ILLEGAL OR NON NUMERIC STRING

Explanation: The displayed field is not numeric or the value is not allowed by the preceding statement.

User Response: Correct the erroneous statement.

EZT000-20 &WORD :ILLEGAL OR UNDEFINED NAME

Explanation: The displayed name is illegal or not defined.

User Response: Correct the erroneous statement.

EZT000-21 &WORD :INVALID NUMBER OF TABLE ROWS

Explanation: Number of external table rows is not numeric or not supplied.

User Response: Code the proper number of table rows.

EZT000-22 &WORD JOB STATEMENT IS NOT SUPPORTED

Explanation: Possible causes:
- SQL was coded on the job statement
- Incomplete ″START″ or ″FINISH″ or ″NAME″ or ″ENVIRONMENT″ statements
- Unknown or illegal statement

User Response: Correct the erroneous statement.

EZT000-23 &WORD :UNDEFINED FIELD/KEY NAME

Explanation: The specified key is undefined.

User Response: Correct the erroneous name.

EZT000-24 &WORD :ILLEGAL SORT STATEMENT

Explanation: The statement is not a legal SORT statement.

User Response: Correct the erroneous statement.

EZT000-25 &FILE :UNDEFINED OR ILLEGAL FILE NAME

Explanation: The displayed file is not defined or it is illegal as coded.

User Response: Correct the erroneous statement.

EZT000-25 FILE QUALIFIER FOR ″RECORD-LENGTH″ FIELD IS REQUIRED. EXAMPLE: FILEIN1:RECORD-LENGTH.

Explanation: RECORD-LENGTH was coded without a file qualifier.

User Response: Migration Utility requires a file qualifier for the RECORD-LENGTH reserved field. Add a file qualifier to the statement.

EZT000-25 FILE OR TABLE QUALIFIER FOR ″&FIELD″ FIELD IS REQUIRED

Explanation: The &FIELD is defined more than once in the program. The reference to &FIELD could not be resolved based on files found in the JOB statement.

User Response: Add a file or SQL table qualifier to the statement.

EZT000-25 FILE OR TABLE QUALIFIER FOR ″&FIELD″ HOST VARIABLE IS REQUIRED. EXAMPLE: FILEIN1.&FIELD

Explanation: The &FIELD used as a host variable is defined more than once in the program. The reference

to &FIELD could not be resolved based on files found in the JOB statement.

User Response: Add a file or SQL table qualifier to the statement. For example, `SQLTAB.&FIELD` or `FILEIN:&FIELD`.

EZT000-26 &WORD :NOT ALLOWED

Explanation: The displayed option is not a valid option for the preceding statement.

User Response: Code the correct option.

EZT000-26 &WORD :NOT ALLOWED. NUMERIC TYPE IS REQUIRED. COBOL STATUS IS ALPHA TYPE. CHANGE TARGET TO ALPHA OR USE MOVE INSTEAD OF ASSIGN.

Explanation: The &WORD is a FILE -STATUS field being assigned to a numeric field.

User Response: In the generated COBOL, status codes are alphanumeric 2 byte fields, while Easytrieve status code is numeric. You can change your target field to an alphanumeric field, or use the MOVE statement instead of the assign.

This message can be avoided by running Migration Utility with the IOCODE=EASYT option.

EZT000-27 &ZF2 :ILLEGAL ASSIGNMENT OR INSTRUCTION

Explanation: Assignment is not allowed as written.

User Response: Correct the erroneous statement.

EZT000-28 &WORD :IF STATEMENT IS INCOMPLETE

Explanation: More operands are expected in the IF statement.

User Response: Make sure that the IF statement is complete.

EZT000-29 &WORD :ILLEGAL RELATIONAL/LOGICAL OPERATOR

Explanation: The Relational/Logical Operator is not a valid Easytrieve Operator.

User Response: Code the correct Operator.

EZT000-30 &WORD :ILLEGAL COL/POS VALUE

Explanation: The coded value is not allowed.

User Response: Code the correct value.

EZT000-31 &WORD :EXPECTED ″KEY″ NOT LOCATED

Explanation: The file KEY is not provided following the DDNAME of synchronized processing definition

User Response: Code the required parameters.

EZT000-32 CANNOT RESOLVE REPORT NAME

Explanation: A PRINT statement was issued without a report name in a JOB that has multiple REPORT statements without a report name.

User Response: Correct the REPORT statements by adding a valid report name. Correct the PRINT statement to reference a valid report.

EZT000-33 UNPAIRED END-IF OR END-DO STATEMENT

Explanation: Too many or too few END-IF or END-DO terminators found.

User Response: Make sure that the terminators pair with the IF or DO statements.

EZT000-34 &WORD :ILLEGAL PUT OR GET FORMAT

Explanation: PUT or GET is incomplete or followed by illegal parameters.

User Response: Correct the erroneous statement.

EZT000-35 PERFORM PROCEDURE IS MISSING

Explanation: A procedure name was not found following PERFORM statement.

User Response: Code the required procedure name.

EZT000-36 ILLEGAL GO TO STATEMENT

Explanation: The statement is incomplete or improper.

User Response: Code the required parameters.

EZT000-37 &WORD :ILLEGAL POINT FORMAT

Explanation: The POINT is incomplete or followed by illegal parameters.

User Response: Correct the erroneous statement.

EZT000-38 &WORD :ILLEGAL READ FORMAT

Explanation: The READ is incomplete or followed by illegal parameters.

User Response: Correct the erroneous statement.

EZT000-39 &WORD :ILLEGAL WRITE FORMAT

Explanation: The WRITE is incomplete or followed by illegal parameters.

User Response: Correct the erroneous statement.

EZT000-40 &WORD :ILLEGAL DO WHILE FORMAT

Explanation: The DO is not followed by WHILE/UNTIL statement.

User Response: Code WHILE or UNTIL following the DO statement.

EZT000-41 &WORD :ILLEGAL ASSIGNMENT

Explanation: Improper assignment format.

User Response: Correct the erroneous parameter.

EZT000-42 &WORD :ILLEGAL MOVE EXPRESSION

Explanation: The MOVE is not followed by TO, or FILL not followed by the fill character in quotes.

User Response: Correct the erroneous statement.

EZT000-43 &WLABNAME :ILLEGAL OR DUPLICATE PARAGRAPH

Explanation: The paragraph or procedure name is not a valid name or it was previously defined.

User Response: Correct the erroneous statement.

EZT000-44 &LIT... :LITERAL IS ILLEGAL OR TOO LONG (OVER 58 BYTES EXCLUDING QUOTES)

Explanation: The HEADING literal is over 58 characters or not enclosed in quotes. &LIST is the first 20 characters of the literal.

User Response: Correct the erroneous statement.

EZT000-45 &WORD EXCEEDS nn CHARACTERS

Explanation: The field name exceeds 18 characters. This error occurs when translator is running in NATIVE mode.

User Response: Reduce the field name to maximum of 16 characters.

EZT000-46 INVALID LABELS PARAMETER COMBINATION

Explanation: Parameters combination for LABELS is improper.

User Response: Correct the erroneous parameters

EZT000-47 &WORD :UNSUPPORTED EASYTRIEVE STATEMENT

Explanation: Illegal FILE parameters or RETRIEVE WHILE was specified.

User Response: Remove RETRIEVE, it is not supported by the translator. Correct the erroneous parameters.

EZT000-48 CONFLICTING FILE I/O USAGE

Explanation: The file does not qualify for the specified I/O. Possible causes:
- PUT or WRITE issued to a file open for input only
- GET or READ or POINT issued to a file defined with CREATE option
- FILE parameters specify UPDATE and VSAM-SEQ

User Response: Correct the erroneous parameters/statements.

EZT000-49 TITLE LENGTH EXCEEDS MAXIMUM OF *NN*

Explanation: The combined length of all fields and literals on the TITLE line exceeds the total Print Line size.

User Response: Reduce literal and fields or increase the SIZE parameter.

EZT000-50 LITERAL IS TOO LONG

Explanation: Literal exceeds 130 characters.

User Response: Translator supports literal up to 130 characters long. Reduce the literal.

EZT000-51 ILLEGAL SEARCH FORMAT

Explanation: SEARCH is incomplete or contains extraneous parameters.

User Response: Correct the erroneous statement.

EZT000-52 SUMFILE &WORD IS NOT DEFINED

Explanation: The File Name specified following the SUMFILE is not defined.

User Response: Code the correct file name.

EZT000-53 &WORD IS ILLEGAL SUM FIELD

Explanation: Undefined or non-numeric field used in SUM.

User Response: Correct the erroneous statement.

EZT000-54 SUMFILE BUT NO CONTROL BREAKS

Explanation: SUMFILE specified for Report that has no Control Breaks.

User Response: Remove the SUMFILE or code at least one Control Break.

EZT000-55 RECURSIVE USE OF ″FINAL″

Explanation: ″FINAL″ is out of sequence or previously coded.

User Response: Correct the erroneous statement.

EZT000-56 MAXIMUM OF *NN* PARAGRAPHS EXCEEDED

Explanation: The number of program paragraphs exceeds the specified number by the MAXPROC=NN.

User Response: The number of paragraphs is controlled via the MAXPROC=NN translator option. See Chapter 8, “Installation and Migration Utility options”, on page 115.

EZT000-57 &NESTCTR OF *NN* BRACKET LEVELS EXCEEDED

Explanation: The translator supports maximum of 8 nested IF/DO statements.

User Response: Reduce the nest to 8 or less.

EZT000-58 ILLEGAL ARITHMETIC EXPRESSION

Explanation: The expression is incomplete.

User Response: Correct the erroneous expression.

EZT000-59 IMPROPER ″MOVE LIKE″ EXPRESSION

Explanation: Incomplete or improper MOVE LIKE statement.

User Response: Correct the erroneous statement.

EZT000-60 &WORD IS UNDEFINED

Explanation: The &WORD field is not defined, or a working storage (W) group field was referenced in a MOVE LIKE statement.

User Response: Correct the erroneous statement.

EZT000-60 ″&KEY″ :KEY FOR &FILE IS UNDEFINED

Explanation: The specified &KEY is undefined.

User Response: Correct the erroneous name.

EZT000-60 ″&KEY″ :KEY FOR &FILE1 AND &FILE2 IS IN CONFLICT. ASSIGN UNIQUE KEY NAMES.

Explanation: The specified &KEY is defined for two different files.

User Response: Correct the erroneous name.

EZT000-61 &WORD :ILLEGAL CALL EXPRESSION

Explanation: The CALL is incomplete or followed by unknown parameters.

User Response: Correct the erroneous statement.

EZT000-62 &FIELD :AMBIGUOUS VALUE

Explanation: A value is coded for a filed that contains REDEFINE statement, directly or indirectly.

User Response: Remove the erroneous value.

EZT000-63 LABEL ″&LABEL″ IS INSIDE AN IF/DO/CASE NEST

Explanation: Unpaired END-IF or END-DO, or procedure was coded inside an IF/DO logic.

User Response: See “Labels inside a DO and IF pair of statements” on page 18.

EZT000-64 &SORTEXIT PROC IS UNDEFINED OR MISPLACED

Explanation: Expecting procedure name for SORT Exit. None found.

User Response: Correct the extraneous statement.

EZT000-65 &WORD :″&SORTEXIT.. PROC″ NOT FOUND

Explanation: procedure name does not match the Proc Name specified by the SORT INPUT EXIT.

User Response: Correct the procedure name.

EZT000-66 SELECT IS NOT IN ″&SORTEXIT.. PROC″ RANGE

Explanation: SELECT statement was located outside of SORT EXIT Proc.

User Response: Remove or correct the statement.

EZT000-67 &WORD IS ILLEGAL ″STOP″ OPTION

Explanation: Unknown STOP option.

User Response: Remove the extraneous parameter.

EZT000-68 RECURSIVE USE OF REPORT EXIT

Explanation: The exit was previously specified for this Report.

User Response: Remove the extraneous exit.

EZT000-69 &WORD OVERLAPS PREVIOUS FIELD BY *XX*. MAX AVAILABLE OVERLAP IS *YY* POSITIONS.

Explanation: Absolute position for the field would cause it to overlap the previous field. This is allowed by Easytrieve, however PEngiEZT sometimes cannot allow the overlap due to COBOL restrictions.

User Response: *XX* is the number of positions that are overlapping. *YY* is the maximum number of positions that PEngiEZT was able to compensate. You can reduce the field size or shift its location to the right, or if possible change the mask.

The overlap can also be caused by a long field title. The starting position should be tuned as conditions permit.

Caution: Any reduced field mask can cause a loss of leading data digits. Use extreme care.

EZT000-70 &WORD ILLEGAL ADJUSTMENT

Explanation: A relative position was placed at the beginning of print line before any fields or literals.

User Response: Remove the incorrect statement.

EZT000-71 ″SUM″ DOES NOT FOLLOW ″CONTROL″ STATEMENT

Explanation: The SUM statement is out of sequence.

User Response: The SUM must be coded following the CONTROL statement.

EZT000-72 COBOL=&GCOBOL NOT ″COBOL390″ OR ″COBOLII″

Explanation: COBOL option is invalid.

User Response: PEngiEZT supports COBOL-II and COBOL S/390® only. Code `COBOL=COBOLII` or `COBOL=COBOL390`.

EZT000-73 ELSE IS OUT OF SEQUENCE

Explanation: ELSE was found without a previous IF statement.

User Response: Correct erroneous statement.

EZT000-74 ″&WORD″ IS ILLEGAL OR CONFLICTING ASSIGNMENT

Explanation: One of the following problems was detected:

- SPREAD and NOADJUST were detected in the same REPORT.
- Too many, or too few, arguments were coded for a logical operation.

User Response: Correct the problem.

EZT000-75 NUMBER PRINT/DISPLAY LINES EXCEEDS *NN*

Explanation: The number of PRINT/DISPLAY lines exceeds the number of lines specified by the LINES=NN of the EASYTRAN macro (see Chapter 8, “Installation and Migration Utility options”, on page 115).

User Response: Increase LINES=NN parameter on EASYTRAN macro to accommodate your needs.

EZT000-76 NUMBER PRINT/DISPLAY FIELDS EXCEEDS *NN*

Explanation: The number of PRINT/DISPLAY fields exceeds the number of fields specified by the RFIELDS=NN of the EASYTRAN macro (see Chapter 8, “Installation and Migration Utility options”, on page 115).

User Response: Increase RFIELDS=NN parameter on the EASYTRAN macro to accommodate your needs.

EZT000-77 &SYSPARM BAD EASYTRIEVE PROGRAM NAME

Explanation: The PARM=(EASYTRAN:*XXXXXXXX*) on the translator EXEC is improper, or your MEMBER= member name coded in the PROC is too long (over 8 digits).

User Response: Make sure that your member name is 1-8 characters long. The PARM=(EASYTRAN:&MEMBER) is located in the PROC (JCL). MAKE sure that the format of the PARM= is correct.

EZT000-78 EASYT000 DEMO MODE. LIMIT RECORD SIZE TO 80

Explanation: PEngiEZT is in DEMO mode.

User Response: No solution. DEMO mode allows you to experiment with files of record length of 80 and less.

EZT000-79 EXPRESSION IS TOO LONG

Explanation: The bracketed expression exceeds the total length allowed by the MAXSTR=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase MAXSTR=NN parameter on the EASYTRAN macro to accommodate your needs or reduce the length of your expression.

EZT000-80 NUMBER OF TITLES EXCEEDS *NN*

Explanation: The number of TITLE lines exceeds the number of lines allowed by the HEADERS=NN of the EASYTRAN macro.

User Response: Increase HEADERS=NN parameter on the EASYTRAN macro to accommodate your needs.

EZT000-81 NUMBER OF FILES EXCEEDS *NN*

Explanation: The number of defined files exceeds the number allowed by the FILES=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase FILES=NN parameter on the EASYTRAN macro to accommodate your needs.

EZT000-82 NUMBER OF "IF" NESTS EXCEEDS *NN*

Explanation: The number of nested IF statements exceeds the number allowed by the NESTS=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase NESTS=NN parameter on the EASYTRAN macro to accommodate your needs, or reduce the number of nested IF statements by making separate expressions.

EZT000-83 NUMBER OF MACRO PARAMETERS EXCEEDS *NN*

Explanation: The number of Easytrieve macro parameters supplied following the %NAME exceeds the number of parameters allowed by the MPARMS=NN of the EASYTRAN macro. This error can occur by improper continuation or termination of the string (a misplaced + or -). See Chapter 8, "Installation and Migration Utility options", on page 115.

User Response: Increase MPARMS=NN to accommodate your needs or remove unneeded parameters.

EZT000-84 NUMBER OF NESTED MACROS EXCEEDS *NN*

Explanation: The number of nested macros triggered by the current macro exceeds the maximum allowed by the MNESTS=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase MNESTS=NN to accommodate your needs or reduce the number of macro nests.

EZT000-85 NUMBER OF INDEX ENTRIES EXCEEDS *NN*

Explanation: The number of fields using OCCURS with INDEX exceeds the number allowed by the INDEXS=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase INDEXS=NN to accommodate your needs.

EZT000-86 NUMBER OF TITLE FIELDS EXCEEDS *NN*

Explanation: The number of TITLE fields exceeds the number allowed by the HFIELDS=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase HFIELDS=NN to accommodate your needs.

EZT000-87 NUMBER OF "IF" ARGUMENTS EXCEEDS *NN*

Explanation: The number of arguments in the IF statement exceeds the number of arguments allowed by the MAXARG=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase MAXARG=NN to accommodate your needs

EZT000-88 NUMBER OF PROCS EXCEEDS *NN*

Explanation: The number of PROC declarations exceeds the number allowed by the MAXPROC=NN of the EASYTRAN macro (see Chapter 8, "Installation and Migration Utility options", on page 115).

User Response: Increase MAXPROC=NN to accommodate your needs.

EZT000-89 NUMBER OF REPORTS EXCEEDS 99

Explanation: You have more than 99 reports in your program.

User Response: PEngiEZT supports maximum of 99 reports in a single program. Split your program into multiple smaller programs.

EZT000-90 &REPORT IS NOT DEFINED IN JOB *NN*

Explanation: The Report Name referenced within the previous JOB scope was not defined.

User Response: Correct the erroneous statement.

EZT000-90 &REPORT: CANNOT RESOLVE DEFAULT PRINT.

Explanation: A PRINT without a report name was processed but there were no reports coded without a PRINTER file. &REPORT is the default name assigned to that print statement.

User Response: Make sure that all PRINT and REPORT statements are properly coded.

EZT000-91 &FIELD, ILLEGAL FIELD VALUE. NULL LITERAL IS ILLEGAL

Explanation: The declared field value is not a proper literal, or it is inconsistent with the field type.

User Response: Correct the erroneous literal or value.

EZT000-92 &FILE :FILE KEY IS REQUIRED

Explanation: The KEY cannot be identified for an INDEXED file

User Response: You must provide the KEY-NAME as part of the file definition.

Note: The translator defaults to the first field name in the record definition. The field must be an alphanumeric field.

EZT000-93 &FILE:&KEY :KEY CANNOT BE NUMERIC

Explanation: The file key is not an alphanumeric item for an indexed file.

User Response: This is COBOL restriction. Define the key as an alphanumeric field.

EZT000-94 &WORD :ILLEGAL RECORD/BLOCK SIZE ASSIGNMENT

Explanation: The record size or the block size is not numeric.

User Response: Code correct numeric value.

EZT000-95 COMPUTED RECORD LENGTH *NN* **EXCEEDS** *NN*

Explanation: The sum of all field lengths exceeds the declared record length.

User Response: Verify field definitions and adjust appropriately. See "Translating guidelines" on page 2 for variable-length files.

EZT000-96 FILE &FILE HAS NO ALLOCATED STORAGE

Explanation: The computed record size for the specified file is zero.

User Response: You must specify record length on the FILE statement or provide a valid record layout.

EZT000-97 &FIELD :AMBIGUOUS FIELD POSITION OR INDEX USAGE

Explanation: The field position as specified cannot be translated, or the INDEX usage is improper.

User Response: This message can be eliminated by re-arranging field definitions. If the field in error is a numeric field that redefines an alphanumeric field, switch them around.

EZT000-97 &FIELD :DESTRUCTIVE OVERLAP FOR FIELD WITH OCCURS. ADD A GROUP FIELD DEFINITION FOR NN CHARACTERS AFTER &GROUP FIELD.

Explanation: The &FIELD is a part of complex group definition with OCCURS and INDEX that overlaps other fields in the manner that cannot be handled by COBOL.

User Response: Re-arrange or simplify the layout. For example, FIELDA below was changed to FIELDX, and FIELD was changed to be a 200 bytes alpha field.

```
FIELDA         1      200 A  OCCURS 100 INDEX (INDEX1)
FIELDB    FIELDA       10 A
FIELDC    FIELDA +10   20 A
FIELDD    FIELDA +05   10 A
```

When changed to format below, the problem is corrected.

```
FIELDX         1      200 A  OCCURS 100 INDEX (INDEX1)
FIELDA    FIELDX      200 A
FIELDB    FIELDA       10 A
FIELDC    FIELDA +10   20 A
FIELDD    FIELDA +05   10 A
```

EZT000-98 &FIELD :DUPLICATE WORKING STORAGE FIELD

Explanation: Field was previously defined.

User Response: Re-name the field in question.

EZT000-99 NN :ADJUSTMENT EXCEEDS MAXIMUM SPACE OF &SPACE

Explanation: The specified adjustment exceeds the specified maximum allowed by the SPACE NN report parameter.

User Response: Code the adjustment.

EZT000-9A &FIELD IN TITLE *NN* OVERLAPS PREVIOUS FIELD

Explanation: The field or literal shown cannot fit in the available space.

User Response: Code the field position or column.

EZT000-9B NN: TITLE/LINE *NN* IS OUT OF SEQUENCE OR ILLEGAL AS SPECIFIED

Explanation: The TITLE or LINE is out of position or the number is out of sequence.

User Response: Correct the problem.

EZT000-9C ″VARYING″ USED FOR NUMERIC FIELD OR A TABLE ITEM

Explanation: VARYING fields can be only alphanumeric fields and non-table item.

User Response: Correct the problem.

EZT000-9D ″VARYING″ FIELD LENGTH MUST BE GREATER THAN 2

Explanation: The length specified for a VARYING field is less than 3.

User Response: Correct the problem.

EZT000-9E FIELD LENGTHS ARE NOT EQUAL IN LOGICAL EXPRESSION

Explanation: A logical ″AND″, ″OR″, ″XOR″ operate on fields of equal length but the specified field arguments are of unequal length.

User Response: Correct the problem.

EZT000-9F COMPLEX ″ON″ EXPRESSION IS NOT SUPPORTED

Explanation: An IF Bit Test for ″ON″ was coded with multiple arguments/expressions.

User Response: The ″ON″ condition in IF must be coded as a single argument in expression.

EZT000-9G &FIELD: LENGTH OF PACKED UNSIGNED FIELD EXCEEDS 15

Explanation: The length of a PU field exceeds 15 bytes. COBOL cannot handle it.

User Response: Limit PU fields to maximum of 15 bytes.

EZT000-9H ″&WORD″ HEX NUMBERS FOUND IN ARITHMETIC

Explanation: A hex number was found in an arithmetic expression.

User Response: Correct the problem.

EZT000-9I &OBJECT: TABLE REQUIRES AT LEAST TWO FIELDS

Explanation: The number of fields defined for the table is less than two.

User Response: Easytrieve Plus tables must have two fields, ARG and DESC.

EZT000-9J &OBJECT: ″ENDTABLE″ IS MISSING

Explanation: The ″ENDTABLE″ could not be located following table data items. This can also be caused by unpaired quotes in a data string.

User Response: Check for ENDTABLE, make sure that quoted strings start and end with a quote.

EZT000-9K &OBJECT: ILLEGAL INPUT FILE (TABLES CANNOT BE SORTED)

Explanation: SORT was specified for a table file.

User Response: Easytrieve Plus tables cannot be sorted. Resort to external file techniques.

EZT000-9L &OBJECT: REPORT WAS PREVIOUSLY DEFINED

Explanation: Duplicate report name.

User Response: Make report names unique.

EZT000-9M TABLE DATA ITEM(S) ARE NOT A VALID COBOL LITERAL

Explanation: The value in the table is not a valid COBOL Literal.

User Response: Change the value to be a valid COBOL Literal.

EZT000-9N &FIELD: DUPLICATE FIELD NAME

Explanation: Duplicate field name in the SUM list.

User Response: Remove the duplicate field.

EZT000-9O &WORD: TABLE ARG IS OUT OF SEQUENCE

Explanation: Table data element is out of sequence.

User Response: Make sure that the data elements are in sequence.

EZT000-9P &JOBFILE: NOT A VALID FILE

Explanation: The file in error is a table.

User Response: Tables cannot be used in Synchronized File processing. Correct the statement.

EZT000-9Q COBOL LEVEL &FLEVL FOR &FIELD EXCEEDS *NNN*

Explanation: Number of nested groups (field levels) exceeds maximum allowed.

User Response: Simplify the record layout.

EZT000-9R UNPAIRED END-CASE STATEMENT

Explanation: Extraneous END-CASE was detected.

User Response: Make sure that CASE - END-CASE are properly paired.

EZT000-9S ″CASE″ NOT IMMEDIATELY FOLLOWED BY ″WHEN″

Explanation: CASE statement syntax error.

User Response: Code WHEN statement immediately after the CASE.

EZT000-9T ″&WORD″ CANNOT BE USED IN THIS CONTEXT

Explanation: Syntax error was detected in CASE or WHEN statement.

User Response: CASE must be followed by a data field name. WHEN cannot be followed by a data field name or an arithmetic expression. See “CASE, WHEN, OTHERWISE and END-CASE statements” on page 58 or the Easytrieve Plus reference manual for more rules.

EZT000-9U ″&WORD″ TYPE IS INCOMPATIBLE WITH COMPARE ARG

Explanation: Syntax error was detected in WHEN statement or statement is not supported as written.

User Response: See “CASE, WHEN, OTHERWISE and END-CASE statements” on page 58 or the Easytrieve Plus reference manual for proper rules.

EZT000-9V &FIELD: NUMERIC GROUP WITH OCCURS IS NOT SUPPORTED

Explanation: A numeric field with OCCURS was coded as a group item.

User Response: Simplify the definition. For example, you can define an alpha field with occurs and make the numeric field subordinate to the alpha field.

EZT000-9X &WORD: NUMBER OF REPORT LITERALS EXCEEDS *NNN*

Explanation: Number of report constants (literals) exceeds maximum allowed.

User Response: Increase FIELDS=NNN value in the EZPARAMS.

EZT000-9Y :&FIELD SQL HOST VARIABLE IS UNDEFINED

Explanation: The &FIELD is undefined.

User Response: Code the correct field name.

EZT000-9Y :&WORD SQL HOST VARIABLE IS NOT IN &FILE RECORD

Explanation: The host variable is undefined.

User Response: Use a valid, defined field.

EZT000-9Z ″WHEN″ IS OUTSIDE OF ″CASE″ SCOPE

Explanation: ″WHEN″ statement was detected outside of CASE - END-CASE scope.

User Response: Correct the problem.

EZT000-A1 &FIELD: CANNOT SORT ON FIELD WITH OCCURS

Explanation: &FIELD was defined with OCCURS.

User Response: Fields defined with OCCURS cannot be sorted on. Correct the statement. If you must sort on a field with OCCURS, adjust the layout such that the same record segment can be accessed via a field without OCCURS.

EZT000-A2 &JOBID ″JOB INPUT SQL″ BUT NO VALID SELECT FOUND

Explanation: SELECT was not located for this JOB.

User Response: Code a SELECT as required by Easytrieve Plus.

EZT000-A3 IMPROPER NUMBER OF DCLINCL PARAMETERS

Explanation: Too few or extraneous parameters were detected in ″SQL DCLINCL″.

User Response: See Chapter 5, “SQL/DB2 support”, on page 87 for proper syntax.

EZT000-A4 TABLE NAME FOR SQL FILE IS MISSING

Explanation: The file was declared as an SQL file but there were not SQL Tables associated with it.

User Response: See Chapter 5, “SQL/DB2 support”, on page 87 for proper FILE syntax.

EZT000-A6 &WORD: TABLE NAME FOR SQL INCLUDE IS NOT CODED

Explanation: Easytrieve Plus ″SQL INCLUDE″ was coded without the proper ″FROM &TABLE″ statement.

User Response: Correct the problem.

EZT000-A7 &SQLTABL: UNDEFINED TABLE OR NOT IN DCLINCL

Explanation: &SQLTABLE column or field definitions cannot be resolved.

User Response: Make sure that the table is defined in one of the ″SQL DCLINCL &NAME″ declares and that there is an ″SQL INCLUDE ....″ coded in working storage or an SQL File. Note that &SQLTABLE field name must be 01 level COBOL definition coded in the DCLGEN copybook which is included via ″SQL DCLINCL″.

EZT000-A8 &FILE: SQL FILE IN SYNCHRONIZED PROCESS

Explanation: &FILE is an SQL file.

User Response: Files declared as SQL files cannot be used in synchronized file processing.

EZT000-A9 &FILE: EXCEEDS 26 TABLES OR STATEMENT DOES NOT SUPPORT MULTIPLE TABLES

Explanation: Maximum number of SQL tables in a single SQL statement has been exceeded, or multiple tables have been coded for SQL statement that does not operate on multiple tables.

User Response: Correct the problem.

EZT000-AA &SQLTABL: DUPLICATE SQL TABLE IN FILE DEFINITION

Explanation: Duplicate SQL Table name.

User Response: Remove the duplicate table.

EZT000-AB %COBOL CANNOT BE INSIDE DO/IF STATEMENT.

Explanation: Illegally placed %COBOL Statement.

User Response: Place %COBOL outside of IF/DO nest.

EZT000-AC &WORD :UNDEFINED FILE / NOT AN SQL FILE

Explanation: An SQL request was coded for a file that was not defined as an SQL file.

User Response: Correct the problem.

EZT000-AD &WORD :CONFLICT IN SQL FILE USAGE JOB/&EASYFUN

Explanation: A FETCH was coded for SQL file that has been used on the JOB statement.

User Response: Refer to Easytrieve Plus reference manual for proper SQL file usage.

EZT000-AE &WORD :FILE NOT CODED FOR UPDATE/NO UPDATE COLUMNS

Explanation: Update was specified for SQL File that has not been coded for UPDATE, or no update columns exist.

User Response: Correct the problem. Refer to Easytrieve Plus reference manual for SQL File Update rules.

EZT000-AF SELECT FOR ″JOB INPUT SQL″ WITHOUT ″INTO″

Explanation: No INTO specified for SELECT or INTO is out of place.

User Response: Select coded for ″JOB INPUT SQL″ requires ″INTO″ statement. Refer to Easytrieve Plus reference manual.

EZT000-AG ″UPDATE″ SYNTAX ERROR OR CONFLICTING WITH ″ORDER″

Explanation: UPDATE and ORDER specified on the same SQL Statement.

User Response: DB2 does not support ORDER and UPDATE concurrently. Refer to SQL reference for proper rules.

EZT000-AH ″&WORD″ CANNOT RESOLVE TABLE NAME

Explanation: ″FROM″ was coded without a table.

User Response: Provide a table name following the FROM statement.

EZT000-AI &WORD ″NULL″ USED FOR NON-NULLABLE FIELD

Explanation: IF NULL was specified for a non-nullable column or field.

User Response: Correct the problem.

EZT000-AJ MULTIPLE EZT STATEMENTS FOUND ON JOB LINE. CORRECT IT BY CODING ONE STATEMENT PER LINE.

Explanation: JOB statement was terminated with a period and followed by another Easytrieve statement on the same line.

User Response: Correct the problem.

EZT000-AK UNSUPPORTED ″SELECT″ EXPRESSION SYNTAX. FILE PARAMETERS CANNOT BE CODED ON THE SAME LINE.

Explanation: SELECT statement follows SQL file definition, but the SELECT line, or the last line belonging to the SELECT contains other file options.

User Response: Code other parameters on separate line(s).

EZT000-AL SUBSCRIPT/INDEX ″&WSUBWRD″ DISALLOWED BECAUSE OF PERFORMANCE REASONS.

Explanation: U, BL1 and BL3 fields are disallowed in INDEX for performance reasons.

User Response: Create a BL4 field, move the disallowed index into it and use it as subscript.

EZT000-AM &ESIZE: DBCS FIELD SIZE IS NOT MULTIPLES OF TWO.

Explanation: K type field length is not multiple of two.

User Response: Correct the problem.

EZT000-AN &MACELIA: MULTI COPYBOOK FOR OBJECT NOT UNIQUE.

Explanation: A macro was coded with the same prefix more than one time for the same object/file.

User Response: Correct the problem.

EZT000-AO &OBJECT: LAYOUT IS TOO COMPLEX FOR COPYBOOK=YES

Explanation: The layout is composed of one or more macros and hard-coded field definitions.

User Response: When COPYBOOK=YES is coded, the layouts must be fully defined within one or more macros, or hard-coded field definitions only. You cannot have a mixture of hard-coded fields and macros because the hard-coded definitions will not be found in the copybook. Either hard code all fields or defined all fields in the macros. Another way of solving this problem is to remove the macro from the EZTABLE0 list.

EZT000-AP &MACELIA: NUMBER OF COPYBOOKS EXCEEDS &GNCOPIES

Explanation: Number of allowed Easytrieve Plus macros has been exceeded.

User Response: Increase the number of allowed macros via NCOPIES= in EZPARAMS.

EZT000-AQ &FORIG: CANNOT REDUCE THE FIELD NAME TO 18 CHARS. SIMPLIFY ″&PFXELIA″ PREFIX.

Explanation: The field in question is located in a macro that was coded with a long prefix, or the macro was used multiple times, and the additional prefix assigned to it resulted in a long field name.

User Response: Code macro with a shorter prefix or a unique prefix. The prefix should be one character followed by a dash.

If you must, drop the dash. Remember to change all field names in your program to reflect the new prefix.

EZT000-AR &WPROCNAM PROC: NO MATCHING ″END-PROC″ FOUND

Explanation: Missing END-PROC.

User Response: Correct the problem.

EZT000-AS &arg1 .. &ARGN :INCOMPATIBLE CLASS.

Explanation: Compare arguments are not compatible, that is, you are comparing a numeric field with an alphanumeric field. Solution: Correct the problem.

EZT000-AT &FMASK: MASK DOES NOT MATCH FIELD SIZE OF &FSIZE

Explanation: The number of digits represented by &FMASK does not match the number of digits represented by the field.

User Response: Correct the problem.

EZT000-AU &SUBSCRIPT :SUBSCRIPT IS NOT ALLOWED

Explanation: A subscript was coded for a field without OCCURS.

User Response: Correct the problem.

EZT000-AV &FIELD :FIELD W/OCCURS - SUBSCRIPT IS REQUIRED

Explanation: &FIELD requires a subscript.

User Response: Correct the problem.

EZT000-AW &WORD :ILLEGAL SUBSCRIPT ARGUMENT

Explanation: &WORD is not a valid subscript.

User Response: Correct the problem. Subscript must be a numeric field or literal.

EZT000-AX &FIELD :FIELD REQUIRES N LEVEL(S) OF SUBSCRIPTS

Explanation: The number of coded subscripts does not match the number of required subscripts for this field.

User Response: Provide the correct number of subscripts. The number of required subscripts is the number of OCCURS statements for all groups that the field belongs to, including the OCCURS for the field in question, if coded.

EZT000-AY &FIELD :VALUE STRING LENGTH EXCEEDS 160 BYTES

Explanation: The VALUE string exceeds 160 bytes in length.

User Response: If your string contains repeating characters, consider defining the field using VALUE ALL |&VAL|. Otherwise, initialize the field in the Activity Section.

EZT000-AZ ″&WORD″ IS A COBOL RESERVED VERB. RENAME IT AND CHANGE ALL REFERENCES TO NEW NAME.

Explanation: The &WORD conflicts with COBOL Reserved Verbs.

User Response: The &WORD must be renamed in your Easytrieve Program to a non-conflicting name. All references in your program to &WORD must be changed too.

EZT000-B1 LENGTH OF ASSUMED KEY ″&KEY″ EXCEEDS COBOL LIMIT OF 255 BYTES.

Explanation: Wrong VSAM file key definition.

User Response: The key of VSAM files is assumed to be the first defined filed in record definition if not supplied in the FILE statement via the (KEY &KEY) definition. Make sure that you specify the correct key.

EZT000-B2 DECLARED KEY &KEY″ FOR RELATIVE &FILE FILE IS NOT DEFINED A ″W 4 B″ FIELD.

Explanation: Wrong RELATIVE VSAM file key definition or the declared key is not defined.

User Response: The key for relative VSAM files must be a 4 byte binary field defined in working storage. The key must be defined before the FILE statements and it must be a 4 byte binary field.

EZT000-B3 DBD-NAME/SUBSCHEMA-NAME IS MISSING.

Explanation: DLI DBD-NAME is not supplied.

User Response: Code DBD-NAME/SUBSCH EMA-NAME as required for DLI files.

EZT000-B4 &MACRO IS NOT ALLOWED FOR DLI/IDMS.

Explanation: The default I/O macro &MACRO does not support DLI/IDMS files.

User Response: Currently, Migration Utility does not support DLI and IDMS. The only way around it is to create a custom I/O macro.

EZT000-B5 DECLARED KEY ″&KEY″ FOR PDS &FILE FILE IS NOT AN ALPHA FIELD, OR ITS DECLARED SIZE IS LESS THAN 8 BYTES.

Explanation: Bad &KEY field name or definition.

User Response: PDS file key must be an alphanumeric field and at least 8 bytes long.

EZT000-B6 INCONSISTENT NUMBER OF MACRO PARAMETERS

Explanation: The number of supplied macro parameters is wrong.

User Response: Refer to specific macro coding conventions in this manual.

EZT000-B7 DATE MASK ″&MASK″ IS NOT SUPPORTED

Explanation: The supplied date mask is not supported by Migration Utility.

User Response: For supported masks, see “Available date masks” on page 137.

EZT000-B8 REFERENCE TO &FILE &FIELD UNAVAILABLE

Explanation: The &FIELD reference was found within the JOB Activity that belongs to &FILE file, but the &FILE file was not present within the same JOB activity.

User Response: Correct the erroneous statement.

EZT000-B9 NNN LRECL VALUE NOT 0 OR > 4

Explanation: The declared record length for a variable file is invalid.

User Response: A variable-length file record length must include 4 extra bytes. If you are running with IOMODE=DYNAM, set the record length to 0; otherwise, code the correct record length that includes 4 extra bytes.

EZT000-BA BIT OPERATION IN REPORT EXITS NOT SUPPORTED

Explanation: One of the following problems was detected:
- Logical ON/OFF was detected in report exit.
- Logical operation XOR, AND, OR was detected in report exit.
- HEX number was used in report exit on numeric field.

User Response: Correct the problem. For logical operation in report exit, use other means of conducting the same test.

EZT000-BC SQL/SELECT STATEMENT MUST BEGIN ON A SEPARATE LINE

Explanation: SQL or SELECT is preceded by another statement on the same line.

User Response: SQL and SELECT statements must begin on a separate line due to syntax differences. Make sure that SQL/SELECT is not preceded by any other statements on the same line.

EZT000-BD &FILE RECORD LENGTH OF &SIZE EXCEEDS 32767

Explanation: The computed record length is over the system limit.

User Response: Adjust record length to proper size.

EZT000-BE &FILE :INCONSISTENT NUMBER OF MATCH KEYS

Explanation: The number of keys for &FILE does not match the number of keys coded for the first synchronized file.

User Response: Code the proper number of match keys.

EZT000-BF REQUIRED ″CONTROL″ NOT CODED ON REPORT STATEMENT

Explanation: BEFORE-BREAK or AFTER-BREAK report exit was coded for a report without CONTROL statement.

User Response: Report exits can be used for reports with CONTROL statement only. Add a CONTROL statement or remove the exits.

EZT000-BG &FIELD :ILLEGAL FIELD CLASS

Explanation: The field &FIELD is not of the correct type/class for this instruction. For example, numeric vs alphanumeric.

User Response: Use a field of the correct type.

EZT000-BI MASK TARGET FIELD CANNOT BE NUMERIC

Explanation: Target field of MOVE with MASK option is not alphanumeric (A field), or the field cannot be a target of MOVE with MASK.

User Response: Use alphanumeric field as target in MOVE.

Migration Utility macro generated messages

BCPY-01 XXXXX IS AN INVALID AREA GROUP NAME

Explanation: The AREA= object name is not a valid COBOL field name.

User Response: Object names can be 1-16 characters long and must follow COBOL Field naming conventions.

BCPY-02 XXXXX IS ILLEGAL LEVEL NUMBER

Explanation: An invalid COBOL field level number has been detected.

User Response: Valid level numbers are 01-99.

BCPY-03 XXXXX IS AN ILLEGAL FIELD NAME

Explanation: An invalid COBOL field name has been detected.

User Response: Field names can be 1-30 characters long and must follow COBOL Field naming conventions.

BCPY-04 ILLEGAL COBOL PICTURE

Explanation: An invalid COBOL field picture has been detected.

User Response: Code a valid COBOL field picture. Note that edit COBOL pictures are not allowed in the record definitions.

BCPY-05 COPY IS IMPROPER AS WRITTEN

Explanation: An improperly coded COPY was detected in the DEFINE macro.

User Response: Refer to Appendix A of PEngiBAT/PEngiONL manual for allowed COPY statement formats.

BCPY-06 TOO MANY REPLACING IDENTIFIERS

Explanation: The maximum number of 256 ordered REPLACING statements was detected.

User Response: Reduce the number of REPLACING pairs to less than 256.

BCPY-07 FIELD DEFINITION IS INCOMPLETE

Explanation: The definition of the last field on the copybook is not complete.

User Response: Correct the problem.

BCPY-08 XXXXX FIELD IS UNDEFINED OR LEVELS ARE INCONSISTENT IN REDEFINES EXPRESSION

Explanation: The redefined field is undefined or the level number of the redefined and redefining fields are inconsistent.

User Response: Correct the problem.

BCPY-09 Text1 Text2 Text3; INVALID REPLACING OPTION

Explanation: An improperly coded COPY REPLACING was detected in the DEFINE macro.

User Response: Refer to Appendix A of PEngiBAT/PEngiONL manual for allowed COPY statement formats.

BCPY-10 Text1 Text2 Text3 Text4; RECURSIVE USE OF PSEUDO TEXT

Explanation: Multiple pairs of pseudo text has been detected.

User Response: Migration Utility allows only one ordered pair of pseudo text replacement. Delete extra statements.

BCPY-11 INCONSISTENT LEVEL FOLLOWING A GROUP ITEM

Explanation: A group field was not followed by a field of higher level number.

User Response: Correct the problem.

BCPY-12 SIZE= VALUE IS ILLEGAL OR NOT SUPPLIED

Explanation: The SIZE= was not provided for the COPY Member NOQUEUE option.

User Response: The NOQUEUE option requires the SIZE= parameter.

BCPY-13 RECURSIVE 01 LEVEL INSIDE COPY

Explanation: Multiple 01 levels were detected in the copybook.

User Response: Remove extraneous 01 levels.

BCPY-14 LENGTH OF REDEFINED FIELD XXXXX IS INCONSISTENT

Explanation: The length of Redefined field is not equal to the length of the Redefining field.

User Response: Correct the problem.

BCPY-15 XXXXX: IS AN ILLEGAL COBOL FIELD NAME

Explanation: An invalid COBOL field name has been detected.

User Response: Field names can be 1-30 characters long and must follow COBOL Field naming conventions.

BCPY-16 ″RENAMES″ IS NOT SUPPORTED, USE REDEFINES

Explanation: RENAMES was detected in the copybook.

User Response: Migration Utility does not support RENAMES. The alternative is to use REDEFINES.

BCPY-17 PARAMETER CONFLICT, NOQUEUE AND PREFIX

Explanation: NOQUEUE and PREFIX= options were coded.

User Response: NOQUEUE and PREFIX options are mutually exclusive. Refer to the reference manual.

CBAS-01 THE 'COBOLBAS' MACRO HAS BEEN IMPROPERLY PLACED WITHIN &SYSECT DIVISION/SECTION, MACRO IGNORED

Explanation: The COBOLBAS macro is misplaced in the PEngiBAT source.

User Response: The COBOLBAS macro must be placed before any divisions or sections.

CBAS-02 COPY=&COPY IS UNKNOWN COPY DIRECTIVE

Explanation: The COPY= option is not COPY, ++INCLUDE or -INC.

User Response: Correct the problem.

CICSBASE-01 THE 'CICSBASE'' MACRO HAS BEEN IMPROPERLY PLACED WITHIN &SYSECT DIVISION/SECTION, MACRO IGNORED

Explanation: The CICSBASE macro is misplaced in the PEngiONL source.

User Response: The CICSBASE macro must be placed before any divisions or sections.

CICSBASE-02 COPY=&COPY IS UNKNOWN COPY DIRECTIVE

Explanation: The COPY= option is not COPY, ++INCLUDE or -INC.

User Response: Correct the problem.

CICSBASE-03 MAPSET=XXXX IS AN INVALID MAPSET NAME

Explanation: XXXX is longer than 7 characters or it is not a valid program name.

User Response: Mapset names can be 1 to 7 characters long and start with an alpha character.

DCCL-01 MAXIMUM OF NN OBJECTS EXCEEDED

Explanation: The maximum number of supported Migration Utility objects have been exceeded.

User Response: Refer to "OBJECTS" on page 124. If you must, consolidate your Object Definitions or arrange them such that you issue fewer Define macros. If it is not possible to consolidate any Objects, reduce the number of Objects by coding them using native COBOL.

DCCL-02 XXXXX HAS BEEN PREVIOUSLY DEFINED

Explanation: The xxxxx Object has been previously defined via the DEFINE macro.

User Response: Change the Object name to a unique name.

DCCL-03 POS NN IS AN INVALID OR ILLEGAL POSITION VALUE, FIELD-NAME FIELD

Explanation: The coded position NN for the FIELD-NAME field in item NN is not numeric or it is preceded by a "-".

User Response: Code NN according to the Migration Utility standards.

DCCL-04 XXXXX: DUPLICATE OR ILLEGAL FIELD PICTURE, FIELD-NAME

Explanation: The field picture XXXXX coded for the FIELD-NAME is not a valid COBOL field picture, or the PIC was specified more than one time, or the PIC was not coded but the FIELD-NAME has never been previously defined with a valid picture.

User Response: Correct the invalid picture if it is invalid, remove the duplicate PIC if it is a duplicate, or code the picture if it has never been defined before.

DCCL-05 XXXXX HAS NO ALLOCATED STORAGE

Explanation: All fields coded for the Object XXXXX have been found in error, or no fields have been coded, or the Macro End Delimiter (;) has been misplaced.

User Response: Correct all fields that are in error and make sure that the Macro End Delimiter is placed properly.

DCCL-06 VALUE: ILLEGAL USE OF VAL KEYWORD IN FIELD-NAME

Explanation: The value specified for the FIELD-NAME field is either illegal or in the wrong format.

User Response: The Value can be specified only for the fields defined in the AREA Objects. The Value can be coded following the VAL or the EQ Positional Parameter Identifiers only. The Value cannot be coded for literal or group fields.

DCCL-07 XXXXX: ILLEGAL KEYWORD IN FIELD DEFINITION, FIELD-NAME

Explanation: None (This is an unused message)

DCCL-08 INCONSISTENT MACRO KEYWORDS USAGE OR NO KEYWORDS SPECIFIED

Explanation: The supplied DEFINE macro keywords are in conflict, that is, two or more mutually exclusive keywords have been coded in the same macro, or no valid keywords have been coded.

User Response: Remove the unneeded keywords. The keywords that may be in conflict are: AREA=, HEADER=, LINE=, FRAME=, FILE=, PAGEFOOT= and CTLFOOT=. For example, the FILE= and the AREA= keywords can coexist, however, the LINE= and the HEADER= keywords are mutually exclusive and cannot be specified in the same macro definition.

DCCL-09 EXCESSIVE PARAMETERS IN DEFINITION

Explanation: A null Positional Parameter has been detected. A null parameter can be caused by placing two commas in succession in the macro parameter string.

User Response: Remove/correct the null string.

DCCL-10 ILLEGAL/CONFLICTING USE OF PIC, FIELD-NAME

Explanation: The PIC has been coded for a literal, or PIC was specified more than one time for a field.

User Response: The PIC Positional Identifier is not allowed for literal, therefore it must be removed.

DCCL-11 VAL KEYWORD IS ILLEGAL FOR LITERAL, XXXXX

Explanation: The VAL has been coded for a literal.

User Response: The VAL Positional Identifier is not allowed for literals, therefore it must be removed.

DCCL-12 XXXXX IS AN INVALID LITERAL

Explanation: The XXXXX is not a valid COBOL literal.

User Response: Code a valid COBOL literal.

DCCL-13 VALUE IS INCONSISTENT WITH PIC TYPE, FIELD-NAME

Explanation: For alphanumeric fields, the coded value is not enclosed in quotes or equal to 'SPACE' or 'SPACES'. For numeric fields, the coded value is enclosed in quotes when it should not be.

User Response: Correct the value to be consistent with the field type.

DCCL-14 POS NN WOULD CAUSE AN OVERLAP ON PREVIOUS FIELD. THE NEXT AVAILABLE POSITION IS POS NN.

Explanation: The computed start position of the FIELD-NAME field would overlap the end position of the field before it. This can occur only for the Header, Line, Ctlfoot and Pagefoot Objects.

User Response: Adjust the POS value of the field such that the position less the field length is equal to or greater than the calculated next available position displayed in the message.

DCCL-15 THE ABOVE OBJECT MUST BE DEFINED IN WORKING STORAGE AREA

Explanation: The DEFINE macro for the Object is not within the Working Storage Section, or the "Working-Storage Section" declarative has been misplaced or misspelled.

User Response: Move the DEFINE macro and Parameters to reside within Working-Storage Section.

DCCL-16 THE CALCULATED STORAGE OF NNN EXCEEDS THE SPECIFIED SIZE OF NNN

Explanation: The storage needed to house all defined fields is greater than the specified Object Size. This error can be caused in two ways:

- The SIZE=NNN parameter does not properly reflect the object length.
- The length of one or more fields within the object is inaccurate.

User Response: Increase the SIZE=NNN value to reflect the calculated size, or adjust the field lengths to give the correct size.

DCCL-17 SIZE=NNN IS AN ILLEGAL OR INSUFFICIENT SIZE

Explanation: For the AREA Objects, the SIZE=NNN is either not numeric or it exceeds 32767. For the FRAME Objects, the SIZE=NNN is either not numeric or greater than 32767, or the computed frame size is less than one. Note that the frame size is computed by dividing the SIZE=nnn by the NUP value if the SIZE= was specified, or by dividing 132 by the NUP value if the size was not specified.

User Response: Correct the size to comply with Migration Utility SIZE= requirements.

DCCL-18 XXXXX USAGE IS ILLEGAL FOR LINES

Explanation: The XXXXX usage has been specified for a field within a Header, Line, Ctlfoot or Pagefoot Object.

User Response: The field usage is allowed within the AREA objects only. Remove the usage clause.

DCCL-19 XXXXX USAGE IS ILLEGAL

Explanation: The XXXXX usage has been coded for an alphanumeric field, or the picture for the FIELD-NAME field has been improperly coded as an alphanumeric picture.

User Response: The COMP, COMP-2, COMP-3 and COMP-4 usage can be coded for numeric fields only. If the FIELD-NAME field is a numeric field then change the field picture to reflect a numeric field, otherwise remove the XXXXX usage clause.

DCCL-20 MAX OF NN FIELD DEFINITIONS EXCEEDED, FIELD-NAME FIELD

Explanation: The number of items that can be held in the internal Migration Utility fields queue has been exceeded.

User Response: Refer to FIELDS= keyword of COBOLBAS/CICSBASE macro.

DCCL-21 THE SUM/FORMULA/ACCUM IS ILLEGAL IN THE AREA/HEADER/PAGEFOOT DEFINITION, FIELD-NAME.

Explanation: A SUM, FORMULA or ACCUM field qualifier has been improperly coded within an AREA, HEADER or PAGEFOOT object.

User Response: For proper usage refer to the description of the field qualifier in error, in the ″Define Macro″ section of this document.

DCCL-22 DUPLICATE OR ILLEGAL USE OF SUM/ACCUM OR EXIT/LIT KEYWORD IN FIELD-NAME FIELD

Explanation: A SUM, ACCUM or EXIT field qualifier has been specified more than one time for the FIELD-NAME field, or the SUM, ACCUM or EXIT field qualifiers have been inconsistently used for the same field, or a SUM, ACCUM or EXIT field qualifier has been coded for a field within an AREA object.

User Response: For proper usage refer to the description of the field qualifier in error, in the ″Define Macro″ section of this document.

DCCL-23 ALPHANUMERIC PICTURE/FIELD USED ILLEGALLY IN ARITHMETIC EXPRESSION IN FIELD-NAME FIELD

Explanation: A SUM or an ACCUM field qualifier has been coded for an alphanumeric field, that is-the field to be summed/accumulated has an alphanumeric picture.

User Response: If the field picture has been improperly coded as an alphanumeric picture, correct it, or else remove the field qualifier.

DCCL-24 RIGHT PAREN IS MISSING IN ARITHMETIC EXPRESSION, FIELD-NAME FIELD

Explanation: The arithmetic expression following the FIELD-NAME field is not properly enclosed in parentheses.

User Response: Every arithmetic expression must be enclosed in parentheses, with an equal number of left and right parentheses.

DCCL-25 MAX OF NN FORMULA DEFINITIONS EXCEEDED, FIELD-NAME FIELD

Explanation: The maximum number of arithmetic expressions (formulas) that Migration Utility can accommodate has been exceeded.

User Response: Refer to FORMULAS= keyword of COBOLBAS/CICSBASE macro.

DCCL-26 THE AREA=XXXXX IS IMPROPERLY CODED FOR FILE OPTION, ONE AREA IS REQUIRED IN FILE DEFINITION

Explanation: Multiple Object names have been coded in the AREA keyword in attempt to define file(s).

User Response: When defining files, only one object name can be supplied in the AREA= parameter. Remove extraneous object names from the AREA= sublist.

DCCL-27 FILE=XXXXX HAS BEEN PREVIOUSLY DEFINED

Explanation: A file of the same name has been previously defined.

User Response: Alter the file name to a unique name.

DCCL-28 MAX OF NN FILE DEFINITIONS EXCEEDED

Explanation: The maximum number of file definitions that can be generated by Migration Utility has been exceeded.

User Response: Refer to FILES= keyword of COBOLBAS/CICSBASE macro.

DCCL-29 XXXXX IS AN INVALID AREA GROUP NAME

Explanation: The XXXXX object name exceeds 16 characters or it contains no characters.

User Response: Code the object name within the limits of Migration Utility AREA object naming conventions.

DCCL-30 SIZE=XXX IS ILLEGAL AS WRITTEN

Explanation: The SIZE= parameter has been improperly coded or the size value is not numeric.

User Response: Code the size parameter according to Migration Utility keyword parameter conventions.

DCCL-31 LABEL=XXXXX NOT STANDARD OR OMITTED

Explanation: The specified label is not supported by Migration Utility or it is improperly coded.

User Response: Refer to the LABEL= keyword description in the ″Define Macro″.

DCCL-32 RECFM=X NOT F,V,S, OR U

Explanation: The specified record format is not supported by Migration Utility or it is improperly coded.

User Response: Refer to the RECFM= keyword description in the ″Define Macro″.

DCCL-33 BLKSIZE=NNN IS ILLEGAL AS WRITTEN

Explanation: The BLKSIZE= parameter has been improperly coded or it is not numeric.

User Response: Refer to the BLKSIZE= keyword description in the ″Define Macro″.

DCCL-34 XXXXX, GROUP ITEM IS ILLEGALLY WRITTEN AS THE LAST ENTRY IN DEFINITION

Explanation: A Group item has been declared but it was not terminated with an ENDG directive.

User Response: Insert an ENDG directive after the last field definition which is a part of the declared Group.

DCCL-35 UNBALANCED ENDG NEAR XXXXX

Explanation: An extraneous ENDG directive has been detected. There are more ENDG directives coded than declared GROUP items. The extraneous ENDG is located near item XXXXX as displayed before the message.

User Response: Remove the extraneous ENDG directive.

DCCL-36 XXXXX, GROUP DEFINITION IS ILLEGAL

Explanation: A duplicate Group field qualifier or a duplicate Redefines directive for a Group field has been detected, or a Group or Redefines has been coded within Line, Header, Ctlfoot or Pagefoot object.

User Response: Remove the unneeded Group qualifier or Redefines directive. Note that the Group and Redefines can be used within AREA objects only.

DCCL-37 XXXXX IS AN INVALID OCCURS CLAUSE

Explanation: The OCCURS directive has been improperly coded.

User Response: Refer to the OCCURS directive description in the ″Define Macro″ section of this document for proper format.

DCCL-38 ALIGN=XX, IS AN INVALID/ILLEGAL ALIGN

Explanation: The ALIGN= keyword parameter has been improperly coded.

User Response: Refer to the ALIGN= keyword description in the ″Define Macro″ section of this document for the proper format.

DCCL-39 XXXXX IS ILLEGAL AS WRITTEN

Explanation: The object shadowed is either not coded or it equals the current object name, or the SHADOW directive was previously processed.

User Response: Refer to the SHADOW directive description in the ″Define Macro″ section of this document for the proper format.

DCCL-40 XXXXX IN SHADOW DEFINITION HAS NOT BEEN PREVIOUSLY DEFINED OR IT IS NOT A LINE/CTLFOOT OBJECT

Explanation: The object shadowed has not been previously defined or it is not a Line or Ctlfoot type of object.

User Response: The SHADOW directive requires that

the object shadowed is previously defined as a Line or Ctlfoot object. Rearrange DEFINE macro definitions such that the object to be shadowed is defined.

DCCL-41 XXXXX IS AN ILLEGAL FIELD NAME, ASSIGNING BAD-NAME-NN

Explanation: The displayed field is an illegal COBOL field name. This error can be caused by other errors that might have been detected, thus causing Migration Utility to expect a field name prematurely.

User Response: If you truly coded a bad field name, correct it. If the error was caused due to other errors then correct other errors.

DCCL-42 XXXXX STATEMENT IS OUT OF SEQUENCE, FIELD-NAME FIELD

Explanation: Two field qualifiers, directives or positional identifiers were coded in succession. This problem can be created when the PIC or POS positional identifiers or the EQ or REDEFINES directives are not followed by the respective picture/values/fields.

User Response: Add the necessary picture/values/fields.

DCCL-43 XXXXX: FORMULA FIELD NAME IS NOT UNIQUE

Explanation: The XXXXX formula field name has been previously defined.

User Response: Choose a unique formula field name not previously defined.

DCCL-44 XXXXX: IS AN ILLEGAL COBOL FIELD NAME

Explanation: The XXXXX field name is an illegal COBOL field name. It contains illegal characters or it is too long.

User Response: Correct the field name in error.

DCCL-45 XXXXX, YYYYY HAS NOT BEEN PREVIOUSLY DEFINED IN THE ZZZZZ DEFINITION

Explanation: The YYYYY field to shadow from has not been defined in the ZZZZZ object definition.

User Response: Change shadow expression to use the correct field name.

DCCL-46 XXXXX OBJECT NAME IS TOO LONG. TRUNCATING TO MAXIMUM OF 08 CHARACTERS

Explanation: The object name is too long.

User Response: Assign a new name, up to a maximum of 08 characters. Note that the object names for Line, Header, Ctlfoot and Pagefoot objects are limited to 08 maximum characters in length.

DCCL-47 NUMBER OF NN FRAMES EXCEEDED

Explanation: The maximum number of Frames that can be handled by Migration Utility has been exceeded.

User Response: Refer to FRAMES= keyword of COBOLBAS macro.

DCCL-48 FRAME=XXXXX CONTAINS ILLEGAL DIMENSION

Explanation: The NUP (dimension) parameter has been improperly coded.

User Response: Refer to the FRAME= keyword description in the ″Define Macro″ section of this document for proper frame coding conventions.

DCCL-49 EXPECTING A KEYWORD, FOUND XXXXX IN THE FRAME. DEFAULTING TO ″LINE″

Explanation: A Line, Header, Ctlfoot or Pagefoot keyword is expected.

User Response: Code the proper keyword.

DCCL-50 XX PRINT CONTROL IS OUT OF SEQUENCE

Explanation: The XX print carriage control characters have been detected inside a frame object definition where it does not belong.

User Response: Refer to the FRAME= keyword description in the ″Define Macro″.

DCCL-51 XXX KEY FOR &FILE IS ILLEGAL OR NOT DEFINED IN &FILE-&AREA RECORD

Explanation: XXX VSAM file key field is not defined in the file record layout.

DCCL-52 VCHAN OPTION IS IMPROPERLY USED WITH FRAME BOX FORMAT

Explanation: The VCHAN was coded for a frame which does not contain any Headers, Ctlfoots or Pagefoots.

User Response: Remove the VCHAN parameters. Note the VCHAN option can be coded only for frames which contain nothing but detail lines.

DCCL-53 MAXIMUM OF NN FIELD GROUP LEVELS EXCEEDED

Explanation: The maximum number of field levels supported by Migration Utility has been exceeded.

User Response: Limit the field levels to the maximum that can be supported by Migration Utility.

DCCL-54 XXXXX FIELD IN YYYYY IS NOT UNIQUE

Explanation: The XXXXX field name in the YYYYY object has been previously defined.

User Response: Assign a unique field name. Note that all fields defined within the AREA objects must be unique.

DCCL-55 XX AND YY FIELD LEVELS ARE INCONSISTENT IN REDEFINE EXPRESSION

Explanation: The field level of the redefining field is not equal to the field level of the redefined field.

User Response: COBOL requires that the redefined and the redefining fields are at the same level. Correct your definitions.

DCCL-56 XXXXX FIELD EXCEEDS 30 POSITIONS

Explanation: The field name exceeds 30 positions.

User Response: Migration Utility allows field names up to 30 characters in length. Assign a proper name.

DCCL-57 OCCURS CLAUSE IS ILLEGAL FOR XXXXX. THE CLAUSE IS VALID ONLY FOR GROUP FIELD DEFINITIONS

Explanation: The OCCURS directive has been coded for an elementary item (field).

User Response: The OCCURS directive can be coded only for Group items, therefore you must change your field to a Group field in order to use the OCCURS.

DCCL-58 XXXXX IS ILLEGAL OR DUPLICATE FIELD HEADER FOR YYYYY FIELD

Explanation: The XXXXX header exceeds 30 characters, or the HDR has been previously coded, or the HDR has been coded for a field defined in a HEADER or PAGEFOOT object.

User Response: Limit the header string to maximum of 30 characters if it is too long, or delete the unneeded entry.

DCCL-59 XXXXX: ILLEGAL YYYY OPTION.

Explanation: The YYYYY parameter is unknown to define macro.

User Response: Remove bad parameter.

DCCL-60 XXXXX FILE DEFINITION IS OUTSIDE OF FILE SECTION

Explanation: An attempt was made to define a file outside of FILE SECTION.

User Response: Make sure that the ″FILE SECTION″ was declared before attempting to define any files.

DCCL-61 XXXXX INVALID FUNCTION OPTION

Explanation: XXXX option is unknown to Define Macro.

User Response: Correct it.

DCCL-62 XXXXX UNDEFINED OBJECT

Explanation: XXXX Object/Field is not defined.

User Response: Correct it.

DCCL-63 MAXIMUM OF NN FUNCTIONS EXCEEDED.

Explanation: Number of Migration Utility functions has been exceeded.

User Response: Refer to FUNCTIONS= keyword of DEFINE macro.

DCCL-64 XXXXX: ILLEGAL FUNCTION PARAMETER LIST

Explanation: Invalid or null function parameters.

User Response: Correct it.

DCCL-65 XXXXX: ILLEGAL USE OF FUNCTION

Explanation: Function format is not supported as coded.

User Response: Correct it.

DCCL-66 XXXXX: INVALID USE OF OBJECT IDENTIFIER

Explanation: XXXX is not a valid parameter for the Object in question.

User Response: Correct it.

DCCL-67 NUMBER OF HDR COLUMNS EXCEEDS 3

Explanation: More than 3 strings have been coded for the HDR.

User Response: Correct it. Note that strings composed of multiple words must be enclosed in quotes.

DCCL-68 XXXXX DEMO MODE. YOU MUST USE TEST FILES.

Explanation: You are licensed for the Migration Utility Product DEMO only.

User Response: Demo accepts 80 byte records for PEngiBAT/PEngiEZT products, and Migration UtilityADR file for PEngiONL.

DCCL-69 ″VARCHAR″ CAN BE USED WITH GROUP FIELDS ONLY

Explanation: The VARCHAR qualifier was coded for an elementary item.

User Response: This option is valid for group fields only.

DCCL-70 RECURSIVE USE OF ″VARCHAR″ WITHIN THE GROUP

Explanation: The VARCHAR qualifier was coded for group field and for one of its subordinate fields.

User Response: This option is valid for group fields only. It can be specified once for each group.

DCPY-01 AREA=XXXXX NAME IS ILLEGAL

Explanation: The AREA= object name is not a valid COBOL field name.

User Response: Object names can be 1-16 characters long and must follow COBOL Field naming conventions.

DCPY-02 PARM1 PARM2 (ILLEGAL LEVEL NUMBER)

Explanation: Expecting a level number in the copybook near items PARM1 PARM2.

User Response: Correct it.

DCPY-03 XXXXXX IS AN ILLEGAL FIELD NAME

Explanation: Expecting a field name, found XXXXXX.

User Response: Correct it.

DCPY-04 AREA= PARAMETER IS MISSING

Explanation: The AREA= parameter is not supplied.

User Response: Correct it.

DPNL-01 MAXIMUM OF NN OBJECTS EXCEEDED

Explanation: The maximum number of supported Migration Utility objects have been exceeded.

User Response: Refer to OBJECTS= keyword of COBOLBAS/CICSBASE macro. If must, consolidate your Object Definitions or arrange them such that you issue fewer Define macros. If it is not possible to consolidate any Objects, reduce the number of Objects by coding them using native COBOL.

DPNL-02 XXXXX WAS PREVIOUSLY DEFINED

Explanation: The xxxxx Object has been previously defined.

User Response: Change the Object name to a unique name.

DPNL-03 POS NN: INVALID/ILLEGAL POSITION

Explanation: The coded position NN is not numeric or it is preceded by a “-”.

User Response: Code NN according to the Migration Utility standards.

DPNL-04 XXXXX: ILLEGAL FIELD PICTURE

Explanation: The field picture XXXXX coded a valid COBOL field picture, or the PIC was specified more than one time, or the PIC was not coded but the FIELD-NAME has never been previously defined with a valid picture.

User Response: Correct the invalid picture if it is invalid, remove the duplicate PIC if it is a duplicate, or code the picture if it has never been defined before.

DPNL-05 XXXXX HAS NO ALLOCATED STORAGE

Explanation: All fields coded for the Object XXXXX have been found in error, or no fields have been coded, or the Macro End Delimiter (;) has been misplaced.

User Response: Correct all fields that are in error and make sure that the Macro End Delimiter is placed properly.

DPNL-06 VAL XXXX: RECURSIVE USE OF VALUE

Explanation: The value specified is either illegal or in the wrong format.

User Response: The Value can be coded following the VAL Positional Parameter Identifier. The Value cannot be coded for literal.

DPNL-10 PICTURE IS ILLEGAL FOR LITERALS

Explanation: The PIC has been coded for literal.

User Response: The PIC Positional Identifier is not allowed for literal, therefore it must be removed.

DPNL-11 VAL IS ILLEGAL FOR LITERALS

Explanation: The VAL has been coded for literal.

User Response: The VAL Positional Identifier is not allowed for literals, therefore it must be removed.

DPNL-12 XXXXX: END QUOTE IS MISSING

Explanation: The XXXXX literal for alphanumeric field is not enclosed in quotes.

User Response: Code a valid COBOL literal enclosed in quotes.

DPNL-14 POS NN OVERLAPS PREVIOUS FIELD

Explanation: The start position of the field would overlap the end position of the previous field.

User Response: Adjust the POS value of the field such that the position less the field length is equal to or greater than the calculated next available position displayed in the message.

DPNL-15 DEFPANEL IS OUTSIDE OF WORKING STORAGE

Explanation: The DEFPANEL macro for the Object is not within the Working Storage Section, or the "Working-Storage Section" declarative has been misplaced or misspelled.

User Response: Move the macro and parameters to reside within Working-Storage Section.

DPNL-16 THE CALCULATED STORAGE OF NNN EXCEEDS THE SPECIFIED SIZE OF NNN

Explanation: The storage needed to house all defined fields is greater than the specified Object Size. This error can be caused in two ways:

1. The SIZE= parameter does not properly reflect the object length.
2. The length of one or more fields within the object is inaccurate.

User Response: Increase the SIZE= value to reflect the calculated size, or adjust the field lengths to give the correct size.

DPNL-17 SIZE=NNN: ILLEGAL OR INSUFFICIENT SIZE

Explanation: Size is illegal as written.

User Response: Refer to SIZE= of DEFPANEL macro for conventions.

DPNL-20 MAX OF NN FIELDS EXCEEDED

Explanation: The number of items that can be held in the internal Migration Utility fields queue has been exceeded.

User Response: Refer to FIELDS= keyword of COBOLBAS/CICSBASE macro.

DPNL-30 SIZE=XXX IS ILLEGAL AS WRITTEN

Explanation: The SIZE= parameter has been improperly coded or the size value is not numeric.

User Response: Code the size according to SIZE= keyword of the " DEFPANEL" macro.

DPNL-37 XXXXX: NO VALID PICTURE FOUND

Explanation: The XXXX field is coded without a valid picture.

User Response: Refer to the PIC directive rules of the "DEFPANEL" macro. Note that a valid field picture must be supplied, or the field must have been previously defined with a valid picture.

DPNL-38 ALIGN=XX: INVALID/ILLEGAL ALIGN

Explanation: The ALIGN= keyword parameter has been improperly coded.

User Response: Refer to the ALIGN= keyword description of the DEFPANEL macro.

DPNL-39 XXXXX IS ILLEGAL AS WRITTEN

Explanation: The object shadowed is either not coded or it equals the current object name.

User Response: Refer to the SHADOW directive description in the DEFPANEL macro.

DPNL-40 XXXXX IN SHADOW DEFINITION NOT DEFINED

Explanation: The object shadowed has not been previously defined.

User Response: The SHADOW directive requires that the object shadowed is previously defined.

DPNL-41 XXXXX ILLEGAL VALUE OR NOT IN QUOTES

Explanation: The specified value is invalid.

User Response: The value must be a valid numeric literal for numeric fields, alphanumeric literal for alphanumeric fields, or a field name.

DPNL-42 XXXXX STATEMENT IS OUT OF SEQUENCE

Explanation: Two field qualifiers, directives or positional identifiers were coded in succession. This problem can be created when the PIC or POS positional identifiers or the EQ or REDEFINES directives are not followed by the respective picture/values/fields.

User Response: Add the necessary picture/values/fields.

DPNL-43 XXXXX: EXTENDED ATTRIBUTE SUPPORT IS REQUIRED

Explanation: An extended attribute such as `COLOR, BLINK. . .` was detected, but the map was not defined with the extended attribute support.

User Response: Refer to: EXTATT=, DSATTS= AND MAPATTS= keyword of the ″DEFPANEL″ macro.

DPNL-44 XXXXX: ILLEGAL COBOL FIELD NAME

Explanation: The XXXXX field name is an illegal COBOL field name. It contains illegal characters or it is too long.

User Response: Correct the field name in error.

DPNL-45 XXXXX, YYYYY HAS NOT BEEN PREVIOUSLY DEFINED IN THE ZZZZZ DEFINITION

Explanation: The YYYYY field to shadow from has not been defined in the ZZZZZ object definition.

User Response: Change shadow expression to use the correct field name.

DPNL-50 XX ROW IS OUT OF SEQUENCE

Explanation: The ROW XX value is not numeric or it is out of sequence.

User Response: Row number must be numeric and defined in sequence in respect to the previous row.

DPNL-51 SCROLL XXXX IS ILLEGAL

Explanation: XXX is not numeric or it exceeds the maximum number of rows supported by the map.

User Response: Number of scroll rows must be numeric. It also cannot exceed the maximum number of rows declared by the SIZE=(rows,cols) keyword of the ″DEFPANEL″ macro.

DPNL-52 XXXX: UNPAIRED SCROLL/END-SCROLL

Explanation: SCROLL / END-SCROLL statements are not paired

User Response: For each SCROLL NN there must be one END-SCROLL statement in the map definition. Code statements respectively.

DPNL-53 XXXX: MAPFRM/MAP NAME IS OVER 7 CHARACTERS IN LENGTH

Explanation: The XXXX name is too long.

User Response: Limit the name to maximum of 7 characters.

DPNL-54 XXXXX ILLEGAL PARAMETER IN &KEYWORD=

Explanation: The XXXXX parameter is illegal for the displayed keyword.

User Response: Refer to the ″DEFPANEL″ macro for specific keyword parameters.

DPNL-55 XXXX: ILLEGAL USE OF FUNCTION

Explanation: A function was specified for a filler or in the MAPFRM object.

User Response: Remove the function statement.

DPNL-56 XXXXX FIELD EXCEEDS 20 POSITIONS

Explanation: The field name exceeds 20 positions.

User Response: DEFPANEL allows field names up to 20 characters in length. Assign a name of no more than 20 characters.

DPNL-57 XXXX: INCONSISTENT MAPFRM USAGE

Explanation: XXXX is not a valid MAPFRM (TYPE=MAPFRM), or MAPFRM= was coded for TYPE=MAPFRM.

User Response: Refer to the ″DEFPANEL″ macro for map format usage.

DPNL-58 XXXXX IS ILLEGAL OR RECURSIVE ATTR

Explanation: Illegal or recursive use of attribute was detected.

User Response: Refer to the ″DEFPANEL″ macro for valid attribute combinations.

DPNL-59 XXXXX: VALUE/LITERAL EXCEEDS 120 CHARACTERS

Explanation: The XXXXX value is too long.

User Response: Reduce the value to maximum of 120 characters.

DPNL-60 NN: ROW NOT LOCATED OR INVALID

Explanation: ROW NN is not numeric, or ROW NN was expected but not found.

User Response: Code proper row number.

DPNL-61 @PFAID NOT FOLLOWED BY BRACKETED AID LIST

Explanation: @PFAID parameters are not enclosed in parentheses.

User Response: Put parentheses around parameters.

DPNL-62 XXXXX: UNDEFINED OBJECT

Explanation: XXXX Object/Field specified in the function is not defined or it is not a valid CON or SEL function.

User Response: Correct it.

DPNL-63 MAXIMUM OF NN FUNCTIONS EXCEEDED.

Explanation: Number of Migration Utility functions has been exceeded.

User Response: Refer to FUNCTIONS= keyword of the DEFPANEL/DEFINE macro.

DPNL-64 XXXXX: ILLEGAL FUNCTION PARAMETER LIST

Explanation: Function parameters are not enclosed in parentheses.

User Response: Every function must be followed by a parameter list enclosed in parentheses. If there are no parameters then an empty () list must be specified.

DPNL-65 &NAME: CURSOR LOCATION NOT SPECIFIED

Explanation: Insert Cursor (IC) was not located during the decoding of the map fields.

User Response: IC must be specified for a field within the map.

DPNL-66 XXXXX: RECURSIVE USE OF LIT

Explanation: LIT was coded more then one time for the same field.

User Response: Remove the duplicate LIT.

ECCL-01 MAXIMUM OF NN ELEMENTS EXCEEDED IN THE XXXXX/CONTROL DEFINITION

Explanation: The PPL of the XXXXX PPLI contains too many objects.

User Response: Reduce the number of objects to an acceptable level.

ECCL-02 ILLEGAL USE OF CH1 OR NEWPAGE ON DETAIL LINE DEFINITION

Explanation: CH1 or NEWPAGE print carriage control has been coded for a detail object (line).

User Response: Migration Utility restricts the CH1 and NEWPAGE usage to the CTLFOOT, PAGEFOOT and HEADER type of objects. Remove the illegal print carriage control.

ECCL-03 XXXXX IS AN UNDECLARED OR NULL ENTRY IN THE YYYYY/FRAME/CONTROL DEFINITION

Explanation: The XXXXX object has not been defined via the Define macro, or two commas have been coded in succession, or no entries have been supplied for the YYYYY PPLI.

User Response: Determine the cause and correct it.

ECCL-04 XXX YYYYY IS AN ILLEGAL PRINT CONTROL FOR DETAIL LINE OBJECTS OF VERTICAL REPORTS (REFER TO FRAME LINE OBJECTS)

Explanation: The XXX print carriage control specified for the YYYYY LINE object is not allowed by Migration Utility. The YYYYY LINE is the internal Frame name assigned by Migration Utility to the detail line.

User Response: Migration Utility allows only SP1 print carriage control for the detail lines of Vertical Reports. If you need double or triple space, you can use ″SP1 NEXT″ technique to insert blank lines where

needed. Note that the overstrike (SP0) is not allowed for vertical reports either.

EXTF-01 FILE KEYWORD IS MISSING NEAR, PARM1, PARM2

Explanation: The FILE keyword is expected.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-02 DDNAME, UNDEFINED/CONFLICTING OUTPUT FILE

Explanation: The DDNAME for the output file is undefined or it was used as input file.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-03 XXXXXX, IS AN UNKNOWN PARAMETER

Explanation: The XXXXXX parameter is not a valid EXTRACT macro parameter/keyword.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-04 DDNAME, INPUT FILE IS NOT DEFINED

Explanation: The DDNAME for input file is not defined via the DEFINE macro.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-05 FIELD IS RECURSIVELY USED IN MATCH LIST

Explanation: The FIELD name was specified twice for the same file.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-06 FIELD IS ILLEGAL OR UNDEFINED

Explanation: The FIELD name specified is not defined.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-07 FIELD CONTAINS ILLEGAL SEQUENCE ATTRIBUTE

Explanation: The sequence attribute is not (A) or (D).

User Response: Verify and correct.

EXTF-08 XXXXXX IS AN ILLEGAL RELATIONAL OPERATOR

Explanation: The XXXXXX is not a valid KEEP operator in EXTRACT macro.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-09 FIELD, EXCEEDS MAXIMUM OF NN FIELDS

Explanation: The FIELD would exceed the fields queue capacity of NN.

User Response: Decrease the number of fields in output record.

EXTF-10 MATCH KEYS ARE MISSING FOR DDNAME

Explanation: No valid ″BY″ fields were detected for DDNAME

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-11 DDNAME EXCEEDS MAXIMUM OF NN FILES

Explanation: Maximum number of input files that can be handled by the EXTRACT macro was exceeded.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-12 DDNAME IS MISSING RECORD DEFINITION

Explanation: File DDNAME was defined in error by the DEFINE macro, or the work area specified via the USING statement was not defined.

User Response: This can be caused by other improper parameters. Verify and correct.

EXTF-13 OBJECT RECORD IS NOT DEFINED WITH A PREFIX

Explanation: The output file record or specified work area is not defined with a prefix.

User Response: Verify and correct.

EXTF-14 FIELD IS NOT COMPONENT OF INPUT FILES

Explanation: The ″BY″ field (key) is not defined in the input file record.

User Response: Verify and correct.

EXTF-15 FIELD1 AND FIELD2 ARE INCOMPATIBLE

Explanation: The FIELD1 and FIELD2 are not of the same format. One is numeric, the other is alphanumeric.

User Response: Verify and correct.

EXTF-16 FILE DDNAME, INCORRECT NUMBER OF MATCH KEYS

Explanation: The number of ″BY″ fields (keys) is not equal to the number of keys for the first file.

User Response: Verify and correct.

EXTF-17 FIELD IS NOT COMPONENT OF DDNAME FILE

Explanation: The ″BY″ field (key) is not defined in the input file record.

User Response: Verify and correct.

EXTF-18 XXXXXX DUPLICATE OR ILLEGAL PARAGRAPH NAME

Explanation: The XXXXXX paragraph name is invalid or it was used for some other file.

User Response: Verify and correct.

EXTF-19 DDNAME, DUPLICATE OR CONFLICTING INPUT FILE

Explanation: The DDNAME was previously processed in the same EXTRACT/MATCH, or a user I/O macro fake DDNAME coincides with one of the files defined via the DEFINE macro or other facilities.

User Response: Verify and correct.

EXTF-20 USING XXXXXX WORK AREA IS NOT DEFINED OR IT IS ILLEGAL AS USED

Explanation: The XXXXXX work area was not defined via the DEFINE macro.

User Response: Verify and correct.

EXTF-21 DDNAME, XXXXXX WORK AREA IS NOT DEFINED

Explanation: See EXTF-20 message.

EXTF-22 DDNAME, XXXXXX WORK AREA IS NOT DEFINED

Explanation: See EXTF-20 message.

EXTF-23 DDNAME BLKSIZE NNN IS INVALID

Explanation: The NNN value is not numeric or it exceeds maximum

User Response: Verify and correct.

EXTF-24 FILE-SCAN DOES NOT SUPPORT ″SORT″ ON INPUT

Explanation: Files cannot be sorted with FILE-SCAN.

User Response: Remove SORT option. If file(s) must be sorted, sort them before invoking EXTRACT.

FCCL-01 XXXXX IS AN ILLEGAL FILE/ASSIGN NAME

Explanation: The XXXXX is an invalid file ddname or longer than 8 characters.

User Response: Change the file name or the assign name to comply with the system file naming standards.

FCCL-02 (This message is unused at this time)

FCCL-03 ORG=XXXXX IS AN UNKNOWN FILE ORGANIZATION

Explanation: The XXXXX is an unknown Migration Utility file organization.

User Response: Code one of Migration Utility supported file organizations. Refer to the ORG= keyword description in the ″Define Macro″ section of this document for valid parameters.

FCCL-04 BUFFERS=XXXXX IS ILLEGAL AS WRITTEN

Explanation: The Buffers= value is either not numeric or it exceeds 256.

User Response: Code the proper Buffers= value.

FCCL-05 CONFLICTING OR ILLEGAL FILE ACCESS ORG=XXXXX AND ACCESS=YYYYY

Explanation: For ORG=SEQ, ORG=SEQUENTIAL, ORG=PUNCH, ORG=READER, ORG=PRINTER, ORG=VSAM-SEQ, or ORG=VSAM-SEQUENTIAL, the ACCESS=YYYYY is not ACCESS=SEQUENTIAL. For ORG=INDEXED or ORG=RELATIVE the ACCESS=YYYYY is not ACCESS=DYNAMIC or ACCESS=SEQUENTIAL or ACCESS=RANDOM.

User Response: Correct the ORG= and ACCESS= to be consistent according to the Migration Utility conventions.

FCCL-06 KEY=XXXXX NULL OR ILLEGAL FILE KEY

Explanation: The VSAM file key field name is either not supplied or it is longer than 16 characters.

User Response: The KEY= is a required parameter for VSAM files. Correct it as needed.

FCCL-07 ALTKEY=XXXXX IS ILLEGAL AS WRITTEN

Explanation: The VSAM Alternate key field name is longer than 16 characters.

User Response: Migration Utility supports field names up to 16 character long. Correct it as needed.

FCCL-08 FILEIO OPTION HAS BEEN PREVIOUSLY ISSUED, PEngiCCL PROVIDES FOR ONE FILE I/O ONLY

Explanation: FILEIO=YES has been previously processed for another file.

User Response: Migration Utility can generate file read procedure logic for ONLY one file. If you need to read more than one file, you can code the read/write logic using native COBOL, in the Procedure Division. However, FILEIO=YES must be removed for this file.

GCCL-01 MAX OF NN OBJECTS EXCEEDED

Explanation: The maximum number of generated reports that can be handled by Migration Utility has been exceeded.

User Response: You are limited to the number of reports that can be generated by Migration Utility in a single program. If you need more reports, create a subprogram with additional reports in it.

GCCL-02 REPORT=XXXXXXX HAS BEEN PREVIOUSLY DEFINED

Explanation: The XXXXXXX report has been previously defined.

User Response: Change the report name to a unique name.

GCCL-03 GENERATE MACRO IMPROPERLY USED OUTSIDE OF PROCEDURE DIVISION

Explanation: The GENERATE macro has been issued outside of Procedure Division.

User Response: Code the macro within the Procedure Division boundaries.

GCCL-04 XXXXX: UNDECLARED OR NULL ENTRY IN THE LINE DEFINITION

Explanation: An undefined or null object has been detected in the Line Positional Parameter List.

User Response: All objects in the Line PPL must have been previously defined via the Define macro. A null entry can be caused by two commas in succession or the absence of PPL parameters.

GCCL-05 ILLEGAL PRINT CONTROL ON THE 1ST HEADER LINE

Explanation: A print carriage control other than CH1 or NEWPAGE has been coded before the first Header line.

User Response: CH1 and NEWPAGE print carriage control are the only ones allowed on the first Header line.

GCCL-06 XXXXX: UNDECLARED OR NULL ENTRY IN THE CTLFOOT DEFINITION

Explanation: An undefined or null object has been detected in the Ctlfoot Positional Parameter List.

User Response: All objects in the Ctlfoot PPL must have been previously defined via the Define macro. A null entry can be caused by two commas in succession or the absence of PPL parameters.

GCCL-07 REPORT OBJECTS HAVE NOT BEEN DEFINED VIA THE DEFINE MACRO

Explanation: A GENERATE macro has been coded but no valid objects have been defined via the Define macro. This also could be caused if all Defined Objects have been found in error.

User Response: Define your report objects (Lines, Headers, Pagefoots and Ctlfoots) before you issue the Generate macro.

GCCL-08 FIELD OBJECTS HAVE NOT BEEN DEFINED VIA THE DEFINE MACRO

Explanation: A GENERATE macro has been coded, but no valid fields have been defined via the Define macro. Also caused when all Defined Objects have been found in error.

User Response: Define your report objects (Lines, Headers, Pagefoots and Ctlfoots) before you issue the Generate macro.

GCCL-09 XXXXX HAS NOT BEEN PREVIOUSLY DEFINED

Explanation: The displayed field name has not been previously defined in any Defined Objects or it was misspelled.

User Response: The fields used in the Control Break (Control PPLI) must be defined within an object via the Define macro. If it was misspelled then correct the field name. If the field is not within a defined object, but is needed, define it in a work AREA object in working storage.

GCCL-10 UNPAIRED PARENS IN XXXXX LINE DEFINITION

Explanation: The Line PPL is coded in sublisted form, but the expression contains an uneven number of left and right parentheses.

User Response: Make sure that the expression contains an even number of left and right parentheses. Also make sure that the expression begins with a left parenthesis "(" and ends with a right parenthesis ")".

GCCL-11 THE WORD 'FINAL' IS IMPROPERLY PLACED IN THE CONTROL DEFINITION (IT MUST FOLLOW THE 'CONTROL' WORD IF SPECIFIED)

Explanation: The word "FINAL" has been misplaced in the CONTROL PPL.

User Response: The word "FINAL" in the Control PPL represents the Final Control Break. When coded, it is treated as any other control break fields. However, since the control breaks follow a hierarchy, the final control break must be first in the list, if specified.

GCCL-12 MAX OF NN OBJECTS EXCEEDED, ITEM XXXXX

Explanation: The maximum number of supported Migration Utility objects have been exceeded. This error is caused in the Generate macro while trying to add internally generated objects to the objects queue. The NN is the maximum number of objects that can be handled by Migration Utility.

User Response: Refer to the OBJECTS= of the COBOLBAS/CICSBASE macro. If you still have problems consolidate your Object Definitions or arrange them such that you issue fewer Define macros. If it is not possible to consolidate any Objects, reduce the number of Objects by coding them using native COBOL.

GCCL-13 XX IS AN ILLEGAL CONTROL SEQUENCE OPTION

Explanation: The XX field sequence attribute coded for one of the control fields in the Control PPL is not (A) or (D).

User Response: A field can be in ascending (A) sequence or descending (D) sequence. Correct the attribute in error.

GCCL-14 XXXXX IN YYYYY FORMULA IS ILLEGAL AS WRITTEN WITH SUM/ACCUM/EXIT OPTION

Explanation: The XXXXX field qualifier was used for a reserved field (such as @COUNT) in the formula YYYYY, or the SUM field qualifier was used in the formula YYYYY which resides in a non-Ctlfoot object.

User Response: Remove the field qualifier in error.

GCCL-15 XXXXX IN YYYYY FORMULA IS NOT A NUMERIC FIELD

Explanation: The field XXXXX has not been defined as a numeric field.

User Response: Only numeric fields can be used in arithmetic formulas. Either change the field definition to a numeric usage or correct the formula expression to include the correct fields.

GCCL-16 XXXXX IN YYYYY FORMULA IS UNDEFINED OR ILLEGAL AS WRITTEN

Explanation: The field XXXXX in the YYYYY formula is either an invalid COBOL field name or it has not been defined via the Define macro.

User Response: All fields which are coded in a formula expression must be defined within an object via the Define macro. Fields which are not defined must be moved into a field which has been defined in order to be used in the formula.

GCCL-17 NUMBER OF NN SUMS EXCEEDED

Explanation: The maximum number of SUM fields that can be handled by Migration Utility in a single report has been exceeded. The fields included in this count are the SUM fields within the CTLFOOT objects and CTLFOOT formulas.

User Response: Reduce the number of SUM fields.

GCCL-18 XXXXX IN YYYYY FORMULA IS UNDEFINED OR ILLEGAL AS WRITTEN

Explanation: The field XXXXX in the YYYYY formula is either an invalid COBOL field name or it has not

been defined via the Define macro.

User Response: All fields which are coded in a formula expression must be defined within an object via the Define macro. Fields which are not defined must be moved into a field which has been defined to be used in the formula.

GCCL-19 NUMBER OF NN SAVE FIELDS EXCEEDED

Explanation: The maximum number of Saved fields that can be handled by Migration Utility in a single report has been exceeded. The fields included in this count are the fields which are printed on the report headers, control break totals and pagefoots. Note that the literal and formulas are not included in this count.

User Response: Reduce the number of fields that must be saved. Such fields are defined within the Header, Pagefoot and Ctlfoot objects, and included on the report.

GCCL-20 XXXXX: UNDECLARED OR NULL ENTRY IN THE CONTROL/CTLFOOT DEFINITION

Explanation: The displayed Object has not been defined via the Define macro or it was misspelled.

User Response: The objects used in the Control Break (Control PPL or CTLFOOT PPL) must be defined as CTLFOOT objects via the Define macro. If it was misspelled correct the object name, else remove it.

GCCL-21 SEQUENCE=XXXXX IS INVALID, VALID OPTIONS ARE SEQUENCE=YES OR SEQUENCE=NO

Explanation: An invalid option has been coded for the report sequence check.

User Response: Use the correct sequence option, Sequence=Yes or Sequence=No.

GCCL-22 XXXXX IS AN INVALID REPORT FILE DDNAME

Explanation: The REPORT= report DDNAME is not a valid system ddname or it exceeds 8 characters in length or it is less than 6 characters in length or it is equal ″REPORTS″ literal.

User Response: Correct the report name to comply with Migration Utility coding conventions.

GCCL-23 XXXXX IS AN UNKNOWN GENERATE MACRO KEYWORD

Explanation: The XXXXX is an unknown GENERATE macro positional parameter list identifier (PPLI).

User Response: Correct or remove the PPLI in error.

GCCL-24 DUPLICATE CTLFOOT PPL DEFINITION

Explanation: The CTLFOOT PPLI has been coded more than one time within a single Generate macro.

User Response: The CTLFOOT PPLI can appear only once for each Generate macro invocation. Correct the extraneous CTLFOOT PPLI.

GCCL-25 DUPLICATE PAGEFOOT PPL DEFINITION

Explanation: The PAGEFOOT PPLI has been coded more than one time within a single Generate macro.

User Response: The PAGEFOOT PPLI can appear only once for each Generate macro invocation. Correct the extraneous PAGEFOOT PPLI.

GCCL-26 DUPLICATE HEADER PPL DEFINITION

Explanation: The HEADER PPLI has been coded more than one time within a single Generate macro.

User Response: The HEADER PPLI can appear only once for each Generate macro invocation. Correct the extraneous HEADER PPLI.

GCCL-27 DUPLICATE CONTROL PPL DEFINITION

Explanation: The CONTROL PPLI has been coded more than one time within a single Generate macro.

User Response: The CONTROL PPLI can appear only once for each Generate macro invocation. Correct the extraneous CONTROL PPLI.

GCCL-28 DUPLICATE LINE PPL DEFINITION

Explanation: The LINE PPLI has been coded more than one time within a single Generate macro.

User Response: The LINE PPLI can appear only once for each Generate macro invocation. Correct the extraneous LINE PPLI.

GCCL-29 MAXIMUM ALLOWED CONTROL BREAKS EXCEEDED

Explanation: Maximum of 16 Control Breaks has been exceeded. The count also includes the ″Final″ control break.

User Response: None. You have reached the maximum number of control breaks that can be supported by Migration Utility.

GCCL-30 XXXXX LITERAL IN YYYYY OVERLAPS THE PREVIOUS LITERAL BY NN POSITIONS

Explanation: A LINE or A CTLFOOT object has been coded in the Header PPL. The Generate macro attempted to construct a Header line from the field names defined in such object. The Generate macro tries to align the field name literal with the actual position of the edited field contents as defined in the object. This error is caused when the field name is longer than the field and there are not enough fillers between the fields to compensate for the difference in length, thus causing an overlap on the previous field's name literal.

User Response: Allow more spaces between the current field and the previous field in the object in error. The NN displayed in the message indicates the number of needed positions. Also, if appropriate, consider using the Define macro option ALIGN=(YES,NN). The Align option will automatically allocate enough fillers between the fields to accommodate a header constructed of field names.

GCCL-31 RECURSIVE USE OF THE XXXXX FIELD IN THE CONTROL DEFINITION

Explanation: The XXXXX field has been used more than once in the Control PPL.

User Response: Remove the extraneous field.

GCCL-32 XXXXX FRAME HAS NOT BEEN PREVIOUSLY DEFINED

Explanation: The frame XXXXX has not been defined via the Define macro or the frame name was misspelled.

User Response: Correct the frame name if it has been misspelled, or else define it via the Define macro.

GCCL-33 XXXXX FRAME CAUSED AN OVERFLOW ON THE FRAME DATA QUEUE OF 512 CHARACTERS

Explanation: When generating code for one or more frames, the Generate macro collects all internally assigned object names and print carriage control in a buffer of 512 characters. This error is caused when the length of all collected object names and print carriage control characters exceeds 512.

User Response: The number of objects that can be included in a single report with one or more frames is limited to the number of object names that can fit into a 512 characters buffer. Therefore, there is no solution to this problem unless you can reduce the number of objects within frames.

GCCL-34 FRAMES XXXXX AND YYYYY CONTAIN UNEQUAL SIZE OR DIMENSION

Explanation: Frame XXXXX and frame YYYYY are inconsistent, that is, the frame size (SIZE=) and dimension (NUP) are not equal.

User Response: Migration Utility handles multiple frames in a single report only of equal size and dimension. Change the frame parameters for frame XXXXX and frame YYYYY so that they are equal in size and dimension.

GCCL-35 RECURSIVE OR ILLEGAL USE OF FRAME XXXXX

Explanation: The frame XXXXX has been specified more than one time in the Frame PPL.

User Response: Remove the extraneous parameter.

GCCL-36 XXXXX IS AN UNKNOWN HANDLE OPTION

Explanation: The XXXXX Handle Option is an unrecognized option.

User Response: Refer to the HANDLE description in the ″Generate Macro″ description of this document.

GCCL-37 XXXXX: UNDECLARED/NULL OR ILLEGAL OBJECT IN THE GENERATE OBJECT LIST

Explanation: An undefined, null or illegal object has been detected in the OBJECT= Parameter List of Generate macro.

User Response: All objects in the OBJECT= list must have been previously defined via the AREA= and PREFIX= option of Define macro. A null entry can be caused by two commas in succession.

GCCL-38 XXXXX: ILLEGAL USE OF OBJECT. XXXXX IS NOT DEFINED WITH A PREFIX

Explanation: An illegal object has been detected in the OBJECT= Parameter List of Generate macro.

User Response: All objects in the OBJECT= list must be defined with the PREFIX= option of Define macro.

GCCL-39 XXXXX: NNN OPTIONAL LINES EXCEEDED

Explanation: The maximum number of optional lines that can be handled by Migration Utility in a single report has been exceeded.

User Response: Reduce the number of optional lines.

GCCL-40 PAGE OR SIZE OR ORPHAN VALUE IS NOT NUMERIC

Explanation: The value specified for PAGE= or SIZE= or ORPHAN= keyword parameters of Generate/Generatm macro is not numeric.

User Response: Correct the erroneous value.

GCCL-41 INCONSISTENT USE OF CTLFOOT AND PAGEFOOT OPTIONS

Explanation: The CTLFOOT and PAGEFOOT objects listed in the Generate macro are not of the same origin. This is caused when the FRAME PPL is coded in the Generate macro in combination with free objects.

User Response: The objects listed in the CTLFOOT and PAGEFOOT positional parameter list must all be defined either in the frame or as free objects. That is, if the CTLFOOT objects are defined in the frame then the PAGEFOOT objects must also be defined in the frame and vice versa.

GCCL-42 DUPLICATE SORT OR READ PPL DEFINITION

Explanation: The SORT or READ PPLI has been issued more than one time for the same GENERATE macro.

User Response: Remove the duplicate PPLI.

GCCL-43 XXXXXXXX PPL PARAMETERS EXCEED 512 CHARACTERS

Explanation: The total length of the SORT or READ PPL strings exceeds 512 characters.

User Response: Reduce the length of the field or paragraph names to bring the total length below 512 characters. XXXXXXXX is the string which caused the overflow.

GCCL-44 ″HANDLE″ IS ILLEGAL WITH READ/SORT OPTION

Explanation: The HANDLE PPLI has been coded in the GENERATE macro that uses SORT or READ option.

User Response: The ″HANDLE″ cannot be used with SORT or READ option of the GENERATE macro. If you must handle a Header record, do so via a SORT or READ Input Exit. The header record information can be saved in working storage and accessed as such.

GCCL-45 &REPORT XXX PREFIX IS NOT AVAILABLE

Explanation: There is a conflict in prefix usage for GENERATE/GENERATM macros.

User Response: Choose a different prefix. Refer to the GENERATE macro PREFIX= keyword description.

GCCL-46 &REPORT READ/SORT USING IS NOT ALLOWED OR PREVIOUSLY DEFINED NAME

Explanation: READ or SORT with USING &REPORT was specified with a shared printer DDname (DDNAME=).

User Response: Sharing of I/O such as READ or SORT results in concurrent printing of the subject Reports. Refer to the GENERATE macro for DDNAME= keyword description.

GCCL-47 &TBname TABLE LEVELS IS LESS THAN #CTL BREAKS

Explanation: HANDLE TABLE was specified for a table containing fewer table levels than the number of report control breaks.

User Response: Synchronize table levels with the control breaks. Refer to the GENERATE macro HANDLE TABLE rules.

GCCL-48 PLOT (X Y) CODED FOR NON-FRAME &REPORT, OR IT WAS PLACED AFTER THE ″FRAME″ PPL

Explanation: PLOT (X,Y) option was specified for report/section that does not use FRAMEs, or the PLOT (X Y) was placed after the FRAME PPL in the &REPORT section.

User Response: PLOT can be specified in the GENERATM/GENERATE macro only for sections that use FRAMEs. The statement PLOT (X Y) must be placed before the FRAME PPL (it is the rule).

GCCL-49 CHANNEL STOPS EXCEED NN LINES PER PAGE

Explanation: A VCHAN stop for one of the channels exceeds the number of lines per page specified by the PAGE=nn.

User Response: If the PAGE=nn is correct refer to the default VCHAN values listed in Appendix A. You can override the defaults by providing a VCHAN list as part of the Frame definition.

GCLR-01 XXXXXX IS NOT DEFINED.

Explanation: The CCGCLEAR Macro object was not defined by the DEFINE macro.

User Response: Only objects defined via the DEFINE macros can be used with CCGCLEAR.

GENM-01 REPORT SECTIONS ARE OUT OF ORDER

Explanation: In GENERATM macro, SECTION and END-SECTION are not specified in the correct sequence, that is, the first section does not begin with a SECTION identifier, or an END-SECTION is not followed by a SECTION identifier.

User Response: Refer to the GENERATM macro sections coding standards.

GENM-02 SECTION &SECTION EXCEEDS PARAMETER CAPACITY

Explanation: The length of all macro parameters (data strings) for the named section exceeds 6,144 characters.

User Response: Refer to the GENERATM macro sections coding standards.

GSNP-01 OBJECT=XXXXXX IS NOT DEFINED

Explanation: The CCGSNAP macro object was not defined by the DEFINE macro.

User Response: Only objects defined via the DEFINE macros can be used with CCGSNAP.

SCCL-01 XXXXXXXX REPORT IN THE SORT/READ IS UNDEFINED OR FOLLOWED BY EXTRANEOUS FIELDS

Explanation: The Report in the SORT USING &REPORT or READ USING &REPORT was not previously defined, or extraneous parameters have been detected in the definition.

User Response: Remove extraneous parameters, or refer to the correct &REPORT name.

SCCL-02 SORT REQUESTED BUT NO SORT FIELDS SPECIFIED

Explanation: The SORT or READ option used in the GENERATE macro definition is incomplete.

User Response: Correct the erroneous value.

SCCL-03 FILE &SORTDDN ALREADY EXISTS OR BAD DDNAME

Explanation: The internally generated sort DDNAME overlaps a previously defined file DDNAME.

User Response: Alter the externally defined DDNAME so that it does not interfere with the internally Migration Utility generated names.

SCCL-04 UNKNOWN SORT TYPE IN &FIELD DEFINITION

Explanation: The sort type attribute is not ″(A)″ or ″(D)″.

User Response: Correct the erroneous type.

SCCL-05 SORT FIELD &FIELD IS UNDEFINED OR ILLEGAL

Explanation: The &FIELD field in the SORT PPL is either undefined or illegal as written.

User Response: Correct the erroneous field.

SCCL-06 SORT/READ FILE(S) NAME IS MISSING OR UNDEFINED

Explanation: The &FILE in the SORT FILE &FILE or READ FILE &FILE is undefined or not coded.

User Response: Code the correct file name for as per SORT/READ requirements.

SCCL-07 &AREA IN THE SORT/READ PPL IS NOT DEFINED IN WORKING STORAGE OR LINKAGE SECTION

Explanation: The &AREA work area in the SORT/READ PPL was not defined via the Define macro in working storage or linkage section.

User Response: Make sure that the &AREA definition resides in working storage or linkage section.

SCCL-08 XXXXXXXX IS AN ILLEGAL EXIT PARAGRAPH NAME

Explanation: The XXXXXXXX is an illegal COBOL/Migration Utility paragraph name.

User Response: Correct the paragraph name. Note that Migration Utility paragraph names must start with an alphabetic character an can contain only hyphens an alphanumeric characters.

SCCL-09 &EXIT OUTPUT EXIT IS ILLEGAL IN READ PPL

Explanation: An output exit was coded as part of the READ PPL.

User Response: The READ PPL does not support an output exit. Remove extraneous exit.

SCCL-10 RECURSIVE USE OF &FIELD IN SORT/READ OPTION

Explanation: The &FIELD field was specified more than one time in the SORT PPL.

User Response: Remove extraneous field definition.

SCCL-11 XXXXXX IS ILLEGAL AS WRITTEN

Explanation: The user I/O macro exceeds 9 characters or is less than 2 characters long.

User Response: Correct it.

SCCL-12 MAXIMUM OF NN INPUT FILES EXCEEDED

Explanation: The maximum number OF READ/SORT files has been exceeded.

User Response: Decrease the number of files.

SCCL-13 XXXXXX WORK AREA IS NOT DEFINED

Explanation: The XXXXXX work area was not defined by the DEFINE macro.

User Response: Define it or use the correct work area.

TBDF-01 DATA STRING EXCEEDS 16384 CHARACTERS

Explanation: The length of table CREATE definitions (data strings between CREATE and DATA parameters) is over 512 characters or the length of all field definitions (data strings) for a table in a nested macro is over 16384 characters.

User Response: Parameters such as the field names and the pictures are counted in the size. You can shrink the field names to fit more fields in the buffer.

TBDF-02 &AREA TABLE DEFINITION IS INCOMPLETE

Explanation: The DEFTABLE macro was coded without CREATE and/or DATA options.

User Response: Add the required options/parameters.

TBDF-03 UNEVEN NUMBER OF FIELDS OR NO DATA FOUND

Explanation: The number of data strings following the DATA is not an integral number of expected data COLUMNS. Either COLUMNS NN is improper or the supplied data fields are out of synchronization.

User Response: Add the required data fields, correct COLUMNS parameter.

TBDF-04 XXXX IS NOT NUMERIC

Explanation: The XXXX data is not numeric but it is being assigned to a numeric field.

User Response: Correct the data.

TBDF-05 &AREA HAS BEEN PREVIOUSLY DEFINED

Explanation: A duplicate OBJECT name has been detected.

User Response: Assign an new Name.

TBDF-06 &AREA, EXCEEDS MAX OF NN AREAS

Explanation: You have reached the maximum number of objects that can be defined by the DEFTABLE macro.

User Response: Consider combining two or more tables into a single table.

TBDF-08 COLUMNS/LEVELS NN IS ILLEGAL AS SPECIFIED

Explanation: LEVELS NN or COLUMNS NN is not numeric, or COLUMNS NN is greater than the number of fields contained in the table record.

User Response: Correct the problem.

TBDF-09 CREATE/DATA PRECEDED BY OTHER INFORMATION

Explanation: Extraneous parameters have been detected before CREATE or DATA statements.

User Response: Remove extraneous parameters.

TBDF-10 XXXXX: DATA STRING IS TOO LONG

Explanation: The string of alphanumeric field exceeds the length allowed by the field definition.

User Response: Correct the problem.

TBDF-11 CONFLICT: MEMORY DYNAMIC AND HARD CODED DATA

Explanation: An attempt to define a table with ″MEMORY DYNAMIC″ and Hard Coded Data.

User Response: Tables with ″MEMORY DYNAMIC″ option are allocated at program run time. Hard coded data cannot be defined for such tables. Refer to DEFTABLE macro for proper usage of ″MEMORY DYNAMIC″ option.

TSRV-01 TBSERV USED OUTSIDE OF PROCEDURE DIVISION

Explanation: The TBSERV macro was issued outside of Procedure Division.

User Response: Correct the problem. If Procedure Division line was specified, make sure that there were no errors on the macro before Procedure Division statement.

TSRV-02 &TBname, USING &OBJECT IS INVALID OR MISSING

Explanation: The TBSERV OPEN or TBSERV CREATE function was coded for a table without the USING option.

User Response: You must provide a USING &OBJECT parameters to identify an area for the table.

TSRV-03 &TBname, &TBKEY IS NOT DEFINED

Explanation: The specified table key is not a part of the table record definition.

User Response: All table keys must be an integral part of the table record.

TSRV-05 &FUN FOR &TBname IS AN UNKNOWN FUNCTION

Explanation: The specified function is not supported.

User Response: Refer to the TBSERV macro reference for valid functions.

TSRV-06 &TBname WAS NOT PREVIOUSLY DEFINED

Explanation: A TBSERV function was requested for a table which was not previously opened or created.

User Response: Refer to the TBSERV macro reference for valid function sequences.

TSRV-07 &TBname WAS PREVIOUSLY OPENED

Explanation: A TBSERV CREATE function was requested for a table that was previously created/opened.

User Response: Use a different TBname.

TSRV-08 &TBname IS DEFINED AS A FILE DDNAME

Explanation: A TBSERV CREATE/OPEN function was requested for a table, but the TBname used is already assigned to a file.

User Response: Use a different TBname.

TSRV-09 &TBname, ROWS NN IS INVALID

Explanation: The ROWS value is not numeric.

User Response: Specify a numeric value.

TSRV-10 &TBname, KEY IS NOT DEFINED IN &OBJECT

Explanation: The specified table key is not a part of the table record definition.

User Response: All table keys must be an integral part of the table record.

TSRV-11 &TBname, -TEXT- IS AN UNKNOWN PARAMETER

Explanation: The displayed text is not a valid TBSERV option.

User Response: Correct the problem.

TSRV-12 &TBname, COLUMNS/LEVELS NN IS INVALID

Explanation: LEVELS NN or COLUMNS NN is not numeric, or COLUMNS NN is greater than the number of fields contained in the table record.

User Response: Correct the problem.

TSRV-13 &TBname ROWS IS INVALID OR NOT SPECIFIED

Explanation: ROWS value is not numeric or ROWS 0 was specified.

User Response: Correct the problem.

TSRV-14 &TBname -PARAGRAPH- IS AN INVALID EXIT NAME

Explanation: The paragraph name for error handling is not a valid COBOL paragraph name.

User Response: Correct the problem. Note that the paragraph name must begin with a letter.

TSRV-15 &TBname &TBKEY EXCEEDS KEY QUEUE CAPACITY

Explanation: There are too many table keys. Maximum of 256 table keys can be specified in a single Migration Utility program which is an average of 8 keys per table.

User Response: Decrease the number of table keys. Consider combining two or more tables into a single table if possible.

TSRV-16 &TBname, ″NOT″ IS NOT FOLLOWED BY AN OPERATOR

Explanation: An improper logical term was specified.

User Response: Refer to the TBSERV ″KEY″ coding standards.

TSRV-17 COLUMNS NN EXCEEDS THE NUMBER OF TABLE FIELDS

Explanation: The number of specified columns exceeds the number of fields contained in the table record.

User Response: Correct the problem.

TSRV-18 &TBname ″DIRECT″ OPTION REQUIRES A NUMERIC KEY AND A SINGLE EQUAL RELATION.

Explanation: The DIRECT access was specified but the table key is not a numeric field with EQUAL relation.

User Response: Refer to the TBSERV ″KEY″ coding standards.

TSRV-19 &TBname, &TBbind IS AN INVALID TABLE FOR BIND

Explanation: The BIND was coded with a table name greater than 8 characters long.

User Response: Correct the problem.

TSRV-20 &TBname, BINARY SEARCH REQUIRES ″EQUAL″ IN KEY RELATION

Explanation: The BINARY search option was coded without the EQUAL in key relational operator.

User Response: Correct the problem.

TSRV-21 &TBname, ″-TEXT-″ NOT ALLOWED IN BINARY SEARCH

Explanation: The BINARY search option was coded with the -text- in key relational operator.

User Response: Refer to TBSERV macro reference for proper syntax.

TSRV-22 &TBname, MULTI-LEVEL TABLE SPECIFIED FOR BINARY SEARCH

Explanation: The BINARY search option was coded for a multi-level table.

User Response: Refer to TBSERV macro reference for proper syntax.

TSR1-01 &TBname, B&FUN UNSUPPORTED TBSERV1 FUNCTION

Explanation: A non-supported function was requested for one level table.

User Response: Correct the problem.

TSR1-02 &TBname, &FIELD FORMULA FOUND IN DEFINITION

Explanation: A formula was coded in the table record definition.

User Response: Formulas are not valid in table records. Remove the formula.

TSR1-03 &TBname, &TBbind IS UNDEFINED FOR BIND

Explanation: The &TBname was coded with a BIND for &TBbind, but &TBbind table was not defined.

User Response: Provide the proper table name in the BIND list.

TSR1-04 &TBname, ROWS/COLUMNS OF &TBbind ARE INCONSISTENT

Explanation: The &TBname is a single-level table and &TBbind is a multi-level table.

User Response: None. A multi-level table cannot be bound to a single-level table.

TSR1-05 &TBname, NULLSON/NULLSOFF REQUIRES QMFLAG

Explanation: Handling of null rows was requested but the table was not created/opened with the QMFLAG option.

User Response: Refer to the TBSERV NULLSON/NULLSOFF coding standards.

TSR1-07 &TBname &Fun, CONFLICTS WITH MEMORY STATIC

Explanation: The ALLOC/DEALLOC was specified for a static table.

User Response: Refer to TBSERV macro reference for proper syntax.

TSRM-01 &TBname, &FUN UNSUPPORTED TBSERVM FUNCTION

Explanation: A non-supported function was requested for multi-level table.

User Response: Correct the problem.

TSRM-02 &TBname, &FIELD FORMULA FOUND IN DEFINITION

Explanation: A formula was coded in the table record definition.

User Response: Formulas are not valid in table records. Remove the formula.

TSRM-03 &TBname, &TBbind IS UNDEFINED FOR BIND

Explanation: The &TBname was coded with a BIND for &TBbind, but &TBbind table was not defined.

User Response: Provide the proper table name in the BIND list.

TSRM-04 &TBname, ROWS/COLUMNS OF &TBbind ARE INCONSISTENT

Explanation: The &TBname is a multi-level table and &TBbind is a multi-level table but the number of table levels are not equal.

User Response: None. A multi-level table can be bound to a multi-level table only if the table levels are equal. A single-level table can be bound to a multi-level table, however.

TSRM-05 &TBname, NULLSON/NULLSOFF REQUIRES QMFLAG

Explanation: Handling of null rows was requested but the table was not created/opened with the QMFLAG option.

User Response: Refer to the TBSERV NULLSON/NULLSOFF coding standards.

VCCL-01 XXX IS AN UNKNOWN CHANNEL

Explanation: XXX is an unsupported/unknown Migration Utility print carriage control.

User Response: Refer to APPENDIX A in the Migration Utility manual for valid print carriage control characters.

VCCL-02 XXX IS AN ILLEGAL CHANNEL STOP

Explanation: The line number which represents the channel is not numeric or it exceeds 66.

User Response: All Virtual Channels must be assigned a line number from 1 to 66, because the line number represents that line on the page. For example, if CH2 is assigned to line 20, then every time when CH2 is used before a print object Migration Utility will skip to line 20. Correct the line number to comply with Migration Utility standards.

VCCL-03 NNN EXCEEDS PAGE OF 66 LINES

Explanation: See ″VCCL-02″ message above.

VCCL-04 XXXXX EXTRANEOUS POSITIONAL PARAMETERS, IGNORED

Explanation: An odd number of parameters has been supplied in the VCHAN sublist.

User Response: The VCHAN sublist parameters must be coded in pairs. For each Virtual Channel there must follow a line number for that channel. Correct the VCHAN sublist such that it has an even number of parameters.

VCCL-05 INCONSISTENT CH1 STOP, LINE 1 IS FORCED

Explanation: A line number other than line 1 has been coded for CH1 or NEWPAGE.

User Response: Migration Utility always forces line 1 for CH1 or NEWPAGE print controls. Code 1 for the line number to avoid error.

VCCL-06 CHANNEL XXXXX STOP IS OUT OF SEQUENCE

Explanation: The line number for the channel XXXXX is lower than the line number of the previous channel.

User Response: The line numbers represent the stops on the page. All stops must be in sequence by channel number. That is, stops for higher channels must be higher than the stops for the lower channels. Put your VCHAN sublist into proper sequence.

XCNV-01 BOOK &NAME NAME IS ILLEGAL

Explanation: The copybook name is invalid.

XCNV-02 &WORD IS ILLEGAL LEVEL NUMBER

Explanation: Expecting a level number. This problem can be caused by missing periods in the COBOL copybook.

XCNV-03 &WORD IS AN ILLEGAL FIELD NAME

Explanation: The name is not a legal COBOL field name allowed by Migration Utility.

XCNV-04 NAME= PARAMETER IS MISSING

Explanation: Copybook name was not supplied (NAME= is missing).

XCNV-06 TOO MANY TRANSLATE WORDS OR END-TRANS MISSING

Explanation: 256 translate pairs of words exceeded.

XCNV-08 ″RENAMES″ IS NOT SUPPORTED, USE REDEFINES

Explanation: RENAMES cannot be interpreted by this facility. Resort to REDEFINES statement.

Migration Utility function generated messages

ABEND00-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

ABEND00-02 &PROGRAM: PROGRAM NAME IS TOO LONG

User Response: Code 1 - 8 characters valid program name.

ABEND00-03 &PROGRAM: NNN PROGRAMS EXCEEDED

User Response: Use fewer number of ABEND programs if possible, otherwise resort to your own ABEND handling in Native COBOL.

ABEND00-04 UNKNOWN FUNCTION PARAMETERS

Explanation: Parameter is not supported by the function.

ADDSIGN-01 OBJECT LENGTH IS INVALID

Explanation: The specified object length is not numeric or it is greater than 9 or it is less than 1.

ADDSIGN-02 TARGET IS INVALID OR NOT SUPPLIED

Explanation: The target object name is either invalid or not coded.

ADDSIGN-03 OBJECT IS INVALID OR NOT SUPPLIED

Explanation: The source object name is either invalid or not coded.

AUTOHELP-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

AUTOHELP-02 &MAPNAM: INVALID MAP NAME

Explanation: &MAPNAM is invalid or not defined.

AUTOHELP-03 &WORD: UNDEFINED PARAMETER

Explanation: Parameter is not supported by the function.

BROWSE0-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

BROWSE0-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

BROWSE0-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

BROWSE0-04 &FILKEY: INVALID OR UNDEFINED FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

BROWSE0-05 XXXXX FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-File () )`.

BROWSE1-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

BROWSE1-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

BROWSE1-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

BROWSE1-04 &FILKEY: INVALID OR UNDEFINED FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

BROWSE1-05 XXXXX FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

BROWSE2-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

BROWSE2-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

BROWSE2-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

BROWSE2-04 &FILKEY: INVALID FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

BROWSE2-05 LINKMOD IS NOT PROVIDED

User Response: A default program name (Link to Module) must be coded as per BROWSE2 function standards.

BROWSE2-06 XXXXX FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

BROWSE3-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

BROWSE3-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

BROWSE3-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

BROWSE3-04 &FILKEY: INVALID OR UNDEFINED FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

BROWSE3-05 XXXXX FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

BUILDJCL-01 MAXIMUM JCL ENTRIES EXCEEDS NNN

User Response: Increase QSIZE=NNN in the BUILDJCL Macro Prototype.

CNVBIN0-01 OBJECT LENGTH OF &WFLENT EXCEEDS 3 CHRS

Explanation: The length of conversion object is invalid.

CNVBIN0-02 OBJECT &OBJECT IS INVALID OR NOT SUPPLIED

Explanation: &OBJECT is not a valid COBOL field name.

CNVBIN0-03 TARGET &TARGET IS INVALID OR NOT SUPPLIED

Explanation: &TARGET is not a valid COBOL field name.

CNVBIN1-01 OBJECT LENGTH OF &WFLENT EXCEEDS 3 CHRS

Explanation: The length of conversion object is invalid.

CNVBIN1-02 OBJECT &OBJECT IS INVALID OR NOT SUPPLIED

Explanation: &OBJECT is not a valid COBOL field name.

CNVBIN1-03 TARGET &TARGET IS INVALID OR NOT SUPPLIED

Explanation: &OBJECT is not a valid COBOL field name.

COMMAND-01 &WORD IS OUT OF SEQUENCE

Explanation: The &WORD is unknown to COMMAND function or it is logically placed out of sequence.

COMMKEY-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

COMMKEY-02 &WORD IS ILLEGAL OR TOO LONG Cause; The buffer name (first parameter) or the index name specified by the CINDEX= is too long or not a valid COBOL field name. Note also that the buffer name can be up to 7 characters long. The index name can be up to 8 characters long.

CONSTRUC-01 &OBJECT IS UNDEFINED

Explanation: The specified object &OBJECT is not defined via Migration Utility facilities.

CONSTRUC-02 &OBJECT DOES NOT CONFORM TO FUNCTION RULES

Explanation: The type of object is not supported for the requested option. Refer to the ″CONSTRUC″ function in the reference manual for valid choices.

CONSTRUC-03 &RCODE: INVALID RETURN CODE NAME

Explanation: Return code field name is less than 4 characters or it does not start with ″RC-″.

CONSTRUC-04 &OBJECT: TABLE AREA WAS NOT DEFINED

Explanation: A table service was requested but the table was not created.

CONSTRUC-05 &OBJECT: SEED FUNCTION WAS NOT DEFINED

Explanation: SEED option was requested for an object that does not have any defined seed functions.

CONSTRUC-06 &OBJECT: ″&WOPTION″ UNSUPPORTED OPTION

Explanation: The specified option is not supported for the requested object.

CONSTRUC-07 &OBJECT: CICS MODE UNSUPPORTED OPTION

Explanation: The specified option is not supported for CICS® programs.

CONTROL-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

CONTROL-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

CONTROL-03 &WORD: UNKNOWN SENDMAP PARAMETER

Explanation: Parameter is not supported by the function.

CVBOOL0-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

CVBOOL0-02 OBJECT &OBJECT IS ILLEGAL OR NOT DEFINED

Explanation: The specified object name is illegal or not defined via Migration Utility facilities.

CVBOOL1-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

CVBOOL1-02 OBJECT &OBJECT IS ILLEGAL OR NOT DEFINED

Explanation: The specified object name is illegal or not defined via Migration Utility facilities.

DATEADJ-01 &DATE: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATEADJ-02 &MASK: UNKNOWN DATE MASK

Explanation: Mask &MASK is not supported by the function.

DATEADJ-03 BASE &BASE: NOT ALLOWED WITH &MASK

Explanation: The specified base cannot be used with the supplied mask.

DATEADJ-04 (nnn) IS NOT A VALID NUMERIC LITERAL

Explanation: Numeric literal is expected, none found.

DATEADJ-05 &MASK DOES NOT QUALIFY FOR MONTHS ADJUSTMENT

Explanation: The date format specified by the mask does not qualify for the month adjustment.

DATEADJ-06 OPTIONS CONFLICT: DAYS/MONTHS/YEARS, USE ONE ONLY

Explanation: Options are in conflict.

DATEDAY-01 &DATE: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATEDAY-02 &MASK: UNKNOWN DATE MASK

Explanation: Mask &MASK is not supported by the function.

DATEDAY-03 &BASE: BASE IS NOT 360 OR 365

Explanation: The specified base is not supported.

DATEDIF-01 &DATE: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATEDIF-02, XXXXX ILLEGAL PARAMETER(S)

Explanation: The parameter is not supported by the function.

DATEMAX-01 &DATE: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATEMAX-02 &MASK: UNKNOWN DATE MASK

Explanation: Mask &MASK is not supported by the function.

DATEMAX-03 BASE &BASE: NOT SUPPORTED BY DATEMAX

Explanation: The specified base cannot be used with DATEMASK function.

DATEREG-01 &DATE: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATEREG-02 &MASK: UNKNOWN DATE MASK

Explanation: Mask &MASK is not supported by the function.

DATEREG-03 &BASE: BASE IS NOT 360 OR 365

Explanation: The specified base is not supported.

DATESRV-01 &DATE: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATESRV-04 &FRMASK: MASK NOT SUPPORTED BY DATEMAX

Explanation: The specified mask is not supported by the DATEMAX function.

DATESRV-03 &FUNCT: ILLEGAL DATE FUNCTION

Explanation: The function is not a valid date function.

DATESRV-04, XXXXX ILLEGAL PARAMETER(S)

Explanation: Parameter is not supported by the function.

DATESRV-05 DATEDIF MASKS ARE NOT EQUAL

Explanation: The ″FROMmask″ and the ″TOmask″ are not compatible.

DATESRV-06 OPTIONS CONFLICT: DAYS/MONTHS/YEARS, USE ONE ONLY

Explanation: Options are in conflict.

DATESWP-01 &DATE IS AN ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATESWP-02 &MASK IS AN UNKNOWN DATE FORMAT

Explanation: Mask &MASK is not supported by the function.

DATEVAL-01 &DATE: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DATEVAL-02 &MASK: UNKNOWN DATE MASK

Explanation: Mask &MASK is not supported by the function.

DATEVAL-03 BASE &BASE: NOT ALLOWED WITH &MASK

Explanation: The specified base cannot be used with the supplied mask.

DEFERTAB-01 RECURSIVE USE OF DEFERTAB

Explanation: DEFERTAB was coded more than one time in the program.

DEFERTAB-02 &NAME MULTIPLE NAMES NOT SUPPORTED

Explanation: The NAME= is improperly coded.

DEFERTAB-03 &NAME: NO MESSAGES SUPPLIED

Explanation: DEFERTAB definition was not followed by valid messages.

DEFERTAB-04 &VAL IS NOT NUMERIC

Explanation: The specified value is expected to be numeric.

DEFERTAB-05 &QAREA HAS BEEN PREVIOUSLY DEFINED

Explanation: Conflict in naming conventions. Other objects have the same name.

DEFERTAB-06 &QAREA EXCEEDS MAX OF N'&GTABLQNAM AREAS

Explanation: Too many table entries. Reduce the number of tables if possible.

DEFERTAB-07 UNEVEN NUMBER OF DATA FIELDS SUPPLIED

Explanation: The data strings for generating errors are not paired.

DEFERTAB-10 *XXXXX*: DATA STRING IS TOO LONG

Explanation: The literal is too long (exceeds 40 bytes).

DELSIGN-01 OBJECT LENGTH IS INVALID

Explanation: The specified object length is not numeric or it is greater than 9 or it is less than 1.

DELSIGN-02 TARGET IS INVALID OR NOT SUPPLIED

Explanation: The target object name is either invalid or not coded.

DELSIGN-03 OBJECT IS INVALID OR NOT SUPPLIED

Explanation: The source object name is either invalid or not coded.

DIMAGE-01 &OPTION: ILLEGAL OPTION

Explanation: The specified option is not supported by the function.

DIMAGE-02 &FIELD: ILLEGAL FIELD NAME

Explanation: The name is not a valid COBOL field name.

DIMAGE-03 &RCODE: INVALID RETURN CODE NAME

Explanation: Return code field name is less than 4 characters or it does not start with ″RC-″.

DSTRING-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

DSTRING-02 &WORD IS ILLEGAL OR TOO LONG

Explanation: Illegal parameter.

DSTRING-03 &BUFNAME IS INCONSISTENTLY USED

Explanation: The specified buffer name was previously used in DSTRING function with different options. DSTRING buffer and options must be consistent to its first declaration.

FILESRV-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

FILESRV-02 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

FILESRV-03 &FILKEY: INVALID FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

FILESRV-04 &IOFUN: UNKNOWN I/O REQUEST

Explanation: I/O &IOFUN is not supported by the function.

FILESRV-05 &WORD: UNKNOWN FILESRV PARAMETER

Explanation: Parameter is not supported by the function.

FILESRV-06 EXIT FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

FUNCTION-001 XXXXX: UNKNOWN PARAMETER

Explanation: Parameter is not supported by the function.

FUNCTION-002 &NAME :IMPROPER FUNCTION NAME

Explanation: For local functions, the function name exceeds 23 characters. For all other functions, the function name exceeds 8 characters.

FUNCTION-003 &NAME: DUPLICATE OR ILLEGAL FUNCTION

Explanation: Duplicate function name or function was found in error.

FUNCTION-004 &ELIAS : IMPROPER ELIAS NAME

Explanation: Elias name is more than 23 characters long or it is not a valid COBOL paragraph name.

FUNCTION-005 PARM= EXCEEDS 40 CHARACTERS

Explanation: PARM=&PARM exceeds 40 characters. Reduce PARM+ string.

FUNCTION-006 PARM= :NOT CON OR SEL OPTION

Explanation: The function type described by the PARM= is not CON or SEL. Note that the type must be the first argument in the PARM= list, that is, `PARM=(CON . .)`.

FUNCTION-007 &MEMBER: IMPROPER USING &MEMBER NAME

Explanation: Function USING &MEMBER. The supplied function library name (&MEMBER) is more than 8 characters in length.

FUNCTION-009 TOO MANY PARAMETERS IN USING LIST

Explanation: The USING option was coded with too many parameters.

FUNCTION-010 &ELIAS IS NOT DEFINED IN &MEMBER

Explanation: Improper use of Elias name.

GRETURN-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

GRETURN-02 &WORD: UNKNOWN GRETURN PARAMETER

Explanation: Parameter is not supported by the function.

GVALUES-01 &WORD IS NOT A VALID FIELD NAME

Explanation: The name is not a valid COBOL field name.

GVALUES-02 ″&WORD″ IS OUT OF SEQUENCE

Explanation: Parameter is not supported by the function as written.

HEXSTR0-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

HEXSTR0-02 OBJECT &OBJECT IS ILLEGAL OR NOT DEFINED

Explanation: The specified object name is illegal or not defined via Migration Utility facilities.

HEXSTR0-03 DDNAME IS INVALID OR NOT SUPPLIED

Explanation: The specified DDname is invalid, or it has not been coded.

HEXSTR0-04 &WLENGTH IS INVALID LENGTH VALUE

Explanation: The length is not numeric or it exceeds the maximum allowed size. Note that in CICS environment the length cannot exceed 100.

HEXSTR0-05 &DDNAME NOT ALLOWED IN CICS MODE

User Response: In CICS mode, HEXSTR0 function can be used to format data into a buffer only.

HEXSTR1-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

HEXSTR1-02 OBJECT &OBJECT IS NOT DEFINED

Explanation: The specified object name is illegal or not defined via Migration Utility facilities.

HEXSTR1-03 DDNAME IS INVALID OR NOT SUPPLIED

Explanation: The specified DDname is invalid, or it has not been coded.

HEXSTR1-04 &OBJECT LENGTH EXCEEDS &BUFSIZE

Explanation: The length is too long or not numeric. If too long and you truly must convert it to hex, use HEXSTR1 multiple times, each time doing a section of the object.

HEXSTR1-05 &OBJECT NOT ALLOWED IN CICS MODE

User Response: In CICS mode, HEXSTR1 function can be used to format data into a buffer only.

INITKEY-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

LINKMOD-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

LINKMOD-02 &PROGRAM: INVALID PROGRAM NAME

User Response: Code 1 - 8 characters valid program name.

LINKMOD-03 &WORD: UNDEFINED PARAMETER

Explanation: Parameter is not supported by the function.

LINKMOD-04 EXIT FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

MANGMAP-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

MANGMAP-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

MANGMAP-03 &WORD: UNKNOWN MANGMAP PARAMETER

Explanation: Parameter is not supported by the function.

MAPTKEY-01 &OPTION: ILLEGAL FUNCTION OPTIONS

Explanation: Parameter is not supported by the function.

MAPTKEY-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

MAPTKEY-03 &WORD: GROUP FIELD IS NOT DECLARED

Explanation: MAPTKEY function was coded to handle a non-group key. Refer to the MAPTKEY description in the reference manual.

MAPTKEY-04 &WORD: KEY IS NOT DEFINED IN &MAPNAM

Explanation: The specified field is not defined in the map &MAPNAME.

MSGTXT0-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

MSGTXT1-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

RECVMAP-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

RECVMAP-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

RECVMAP-03 &WORD: UNKNOWN RECVMAP PARAMETER

Explanation: Parameter is not supported by the function.

RECVMAP-04 EXIT FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

REPCHR0-01 &WORD: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

REPCHR0-02 &WORD: INVALID NUMBER OF MASK DIGITS

Explanation: The number of significant mask characters is not numeric.

SENDMAP-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

SENDMAP-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

SENDMAP-03 &WORD: UNKNOWN SENDMAP PARAMETER

Explanation: Parameter is not supported by the function.

SENDMAP-04 EXIT FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

SENDMAP-05 CURSOR AND CURSORLOC USAGE CONFLICT

Explanation: CURSOR and CURSORLOC have been both coded. These options are mutually exclusive.

SENDMSG-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

SENDMSG-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

SENDMSG-03 &WORD: UNKNOWN SENDMSG PARAMETER

Explanation: Parameter is not supported by the function.

SENDMSG-04 CODE &CODE: ILLEGAL COMBINATION

Explanation: CODE (&MSGID &MSGCODE) is improperly coded.

SENDMSG-05 EXIT FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

SETATTR-01 &WORD IS OUT OF SEQUENCE

Explanation: The &WORD cannot be understood by the function.

SETATTR-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

SETATTR-03 &WORD: UNKNOWN ATTRIBUTE

Explanation: Parameter is not supported by the function.

SETATTR-04 &FIELD NOT IN SCROLL AREA, ATTR IGNORED.

Explanation: The &FIELD is not in the map scroll area.

SETATTR-04 &FIELD: NOT IN &MAPNAM

Explanation: The &FIELD is not defined the referenced map.

SETATTR-06 &WORD: EXCEEDS MAXIMUM ATTRIBUTE ELEMENTS

Explanation: Too many attributes are coded. If you need to code all attributes use multiple SETATTR functions, each one with fewer attributes.

TSQSRV-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

TSQSRV-02 &WORD: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

TSQSRV-03 &WORD: INVALID FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

TSQSRV-04 &IOFUN: UNKNOWN I/O REQUEST

Explanation: I/O request is not supported by the function.

TSQSRV-05 &WORD: UNKNOWN TSQSRV PARAMETER

Explanation: Parameter is not supported by the function.

TSQSRV-06 EXIT FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

TSQSRV-07 &QNAME: INVALID QUEUE NAME

Explanation: The TSQ name is invalid as written.

TSQSRV-08 TSQSRV &FILKEY NOT IN WORKING STORAGE

Explanation: The file key for TSQ must be defined in working storage. It does not seem to be so.

TSQSRV-09 SYNTAX ERROR. THE ″FROM″ INFORMATION IS INCOMPLETE

Explanation: Improper ″FROM″ parameters.

TSQSRV-10 FILE KEY OF &FLNAME2 IS NOT UNIQUE

Explanation: The FROM &FILE key is equal to the key assigned to the TSQ file key. Note that the keys used by the TSQSRV function must be unique.

TSQSRV-11 SYNTAX ERROR. THE ″FROM″ IS ILLEGAL FOR ″&IOFUN″

Explanation: The ″FROM″ was coded but it is not supported by the I/O function.

UPDATE0-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

UPDATE0-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

UPDATE0-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

UPDATE0-04 &FILKEY: INVALID FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

UPDATE0-05 A VALID ADD/DEL/UPD MUST BE PROVIDED

Explanation: No valid action was selected. At least one must be specified.

UPDATE0-06 *XXXXX* FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

UPDATE0-07 INITIALIZE PARAMETERS ARE IMPROPER AS CODED

Explanation: INITIALIZE is improper as coded. Refer to the reference manual.

UPDATE1-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

UPDATE1-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

UPDATE1-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

UPDATE1-04 &FILKEY: INVALID FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

UPDATE1-05 A VALID ADD/DEL/UPD MUST BE PROVIDED

Explanation: No valid action was selected. At least one must be specified.

UPDATE1-06 *XXXXX* FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

UPDATE1-07 INITIALIZE PARAMETERS ARE IMPROPER AS CODED

Explanation: INITIALIZE is improper as coded. Refer to the reference manual.

UPDATE2-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

UPDATE2-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

UPDATE2-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

UPDATE2-04 &FILKEY: INVALID OR UNDEFINED FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

UPDATE2-05 *XXXXX* FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

UPDATE2-06 INITIALIZE PARAMETERS ARE IMPROPER AS CODED

Explanation: INITIALIZE is improper as coded. Refer to the reference manual.

UPDATE3-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

UPDATE3-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

UPDATE3-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

UPDATE3-04 &FILKEY: INVALID FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

UPDATE3-05 A VALID ADD/DEL/UPD MUST BE PROVIDED

Explanation: No valid action was selected. At least one must be specified.

UPDATE3-06 *XXXXX* FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

UPDATE3-07 INITIALIZE PARAMETERS ARE IMPROPER AS CODED

Explanation: INITIALIZE is improper as coded. Refer to the reference manual.

UPDATE4-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

UPDATE4-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

UPDATE4-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

UPDATE4-04 &FILKEY: INVALID FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

UPDATE4-05 A VALID ADD/DEL/UPD MUST BE PROVIDED

Explanation: No valid action was selected. At least one must be specified.

UPDATE4-06 XXXXX FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

UPDATE4-07 INITIALIZE PARAMETERS ARE IMPROPER AS CODED

Explanation: INITIALIZE is improper as coded. Refer to the reference manual.

UPDATE5-01 &PARM: NOT CONSTRUCTOR OPTION

Explanation: Function is a Constructor but it was used as Selector.

UPDATE5-02 &MAPNAM: UNDEFINED MAP NAME

Explanation: &MAPNAM is invalid or not defined.

UPDATE5-03 &DDNAME: UNDEFINED FILE NAME

Explanation: &DDNAME is invalid or not defined.

UPDATE5-04 &FILKEY: INVALID OR UNDEFINED FILE KEY NAME

Explanation: &FILKEY is invalid or not defined.

UPDATE5-05 XXXXX FUNCTION NOT ENCLOSED IN PARENTHESES

User Response: All file I/O and exit functions must be coded enclosed in parentheses. Example: `READEXIT (SEL_READ-FILE () )`.

XCTLMOD-01 &PARM: NOT SELECTOR OPTION

Explanation: Function is a Selector but it was used as Constructor.

XCTLMOD-02 &PROGRAM: INVALID PROGRAM NAME

User Response: Code 1 - 8 characters valid program name.

XCTLMOD-03 &WORD: UNDEFINED PARAMETER

Explanation: Parameter is not supported by the function.

PEngiCCL generated messages

ACCL00-001 12 MACNAME :<FUNCTION> EXPECTED VARIABLE NOT PROVIDED

Explanation: The format of the ACCL function is wrong.

User Response: Refer to the coding standards of the ACCL Directive for the function in error. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-002 12 MACNAME :<FUNCTION> UNKNOWN ACCL FUNCTION

Explanation: The displayed function is not an ACCL Directive Function.

User Response: Correct the function.

ACCL00-003 12 MACNAME :<FUNCTION> EXPECTED TEXT NOT PROVIDED

Explanation: The format of the ACCL function is wrong.

User Response: Refer to the coding standards of the ACCL Directive for the function in error. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-004 12 VARNAME :<FUNCTION> INVALID SUBSCRIPT

Explanation: The value contained in the subscript variable is not a valid subscript.

User Response: Variables used as subscripts must numeric and one dimensional. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-005 12 VARNAME :<FUNCTION> IMPROPER DATA OR EXCEEDS MAX LENGTH

Explanation: The input data string is longer than the allocated memory for the target Variable.

User Response: Limit input data string to the maximum allowed by the target variable. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-006 12 VARNAME :<FUNCTION> INCONSISTENT DATA FOR VARIABLE TYPE

Explanation: The input data format is not compatible with the target Variable data type.

User Response: This can happen if an attempt is made to set a non numeric value into a SETA or SETB Variable. Correct the ACCL Directive to use consistent data. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-007 12 -TEXT- :<FUNCTION> UNDEFINED INTERNAL REFERENCE LABEL

Explanation: The macro reference label in the ACCL SETVB function is undefined.

User Response: Either provided the required label or remove the statement from the SETVB list. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-008 12 VARNAME :<FUNCTION> VECTOR DIRECTIVE DOES NOT FOLLOW

Explanation: An ACCL SELECT or an ACCL INDEX directive is not properly followed by a Vector format directive.

User Response: Refer to the ACCL SELECT and ACCL INDEX coding standards. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-009 12 PGMNAME :<FUNCTION> PROGRAM CANNOT BE LOADED

Explanation: The program cannot be loaded or it was not located in the load/core library.

User Response: Make sure that you are pointing to the correct load/core library and that the program exists. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-010 12 VARNAME :<FUNCTION> VECTOR VARIABLE SLOTS ARE < 24 BYTES

Explanation: The declared variable-length used in ACCL SETVB is less than 24 bytes.

User Response: Refer to the coding standards of the ACCL SETVB directive. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-011 12 VARNAME :<FUNCTION> NOT ENOUGH SLOTS IN VECTOR VARIABLE

Explanation: The number of reference labels provided in the ACCL SETVB list exceeds the dimension of the specified vector variable.

User Response: Increase the dimension of the vector variable. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-012 12 MACNAME :<FUNCTION> ONE OF VECTOR ARGUMENT EXCEEDS 16 CHR

Explanation: An argument/word in the ACCL SETVB list exceeds 16 characters.

User Response: The ACCL SETVB arguments/words can be maximum of 16 characters. Reduce the size of the argument in error. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-013 12 VARNAME :<FUNCTION> VARIABLE IS NOT A SETC SYMBOL

Explanation: The specified variable is not a SETC symbol.

User Response: Refer to the coding standards of the ACCL Function in error. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-014 12 VARNAME :<FUNCTION> variable-length IS < 256 BYTES

Explanation: The specified variable allocated memory is less than 256.

User Response: Refer to the coding standards of the ACCL Function in error. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-015 12 MACNAME :<FUNCTION> INVALID PUNCH FILE NAME

Explanation: The punch file name specified is not a valid DDname.

User Response: Punch file name must start with an alpha character, it cannot contain special characters, and it cannot exceed 7 positions. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-016 12 PGMNAME :<FUNCTION> INVALID USER PROGRAM NAME

Explanation: The program name specified in the ACCL CALL directive is invalid.

User Response: A program name must start with an alpha character, it cannot contain special characters, and it cannot exceed 8 positions. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-017 12 PGMNAME :<FUNCTION> MAXIMUM USER PROGRAMS EXCEEDED

Explanation: The maximum number of user loaded programs was exceeded.

User Response: PEngiCCL will handle maximum of 16 user loaded programs. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-018 12 -TEXT- :<FUNCTION> ILLEGAL OR NULL VARIABLE IN CALL LIST

Explanation: The ACCL CALL directive requires exactly one parameter in the call list.

User Response: Correct the problem. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-019 12 -TEXT- :<FUNCTION> VARIABLE IS NOT &SYSLIST

Explanation: The variable coded with ACCL BOX directive is not the &SYSLIST variable.

User Response: The ACCL BOX directive requires the &SYSLIST variable. Refer to the ACCL BOX directive coding standards. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-020 12 -TEXT- :<FUNCTION> FILE/MEMBER NAME IS INVALID

Explanation: The file/member name specified in the ACCL OPEN directive is invalid.

User Response: The member name must start with an alpha character, it cannot contain special characters, and it cannot exceed 8 positions. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-021 12 -TEXT- :<FUNCTION> MAXIMUM USER FILES EXCEEDED

Explanation: The maximum number of punch files has been exceeded.

User Response: PEngiCCL can handle maximum of 8 punch files. Reduce the number of punch files. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-022 12 -TEXT- :<FUNCTION> START - END COLUMNS ARE INVALID

Explanation: The values specified in the ACCL OPEN directive for Start-End columns and/or start of continuation and the comment column are inconsistent.

User Response: Refer to the coding standards of the ACCL OPEN directive. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

ACCL00-023 12 -TEXT- :<FUNCTION> OPTION IS NOT TOKEN/NOTOKEN

Explanation: The supplied option in the ACCL OPEN or ACCL READ directive is invalid.

User Response: An option can be TOKEN,

NOTOKEN, or left out. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ACCL00-024 12 -TEXT- :<FUNCTION> FILE/MEMBER ALREADY EXISTS

Explanation: The new member name in the ACCL RENAME already exists.

User Response: Choose a unique name. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ACCL00-025 12 -TEXT- :<FUNCTION> FILE/MEMBER DOES NOT EXIST

Explanation: The member name to be renamed by ACCL RENAME does not exist.

User Response: Provide the correct name. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ACCL00-026 12 -TEXT- :<FUNCTION> FILE/MEMBER NOT CLOSED

Explanation: ACCL OPEN was attempted without closing the previous OPEN.

User Response: Issue ACCL CLOSE before attempting this open. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ACCL00-001 12 MACNAME :<FUNCTION> ARGUMENTS ARE IMPROPER AS WRITTEN

Explanation: The format of the ACCL function is wrong.

User Response: Refer to the coding standards of the ACCL Directive for the function in error. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ACTR01-001 12 MACNAME :ACTR COUNTER EXCEEDED

Explanation: The number of PEngiCCL internal macro branch instructions has been exceeded. The MACNAME is the macro in error. This was probably caused by an infinite loop, or the ACTR counter is not sufficient enough to accommodate macro needs.

User Response: Check for possible loops or increase the ACTR counter. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADOIF0-001 12 LABEL :UNDEFINED INTERNAL REFERENCE LABEL

Explanation: The internal macro reference label is undefined.

User Response: Add the necessary internal reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADOIF0-002 12 LABEL :CANNOT BRANCH TO ITSELF, WOULD CAUSE LOOP

Explanation: An ADOIF directive target reference label refers to the directive itself.

User Response: Correct the erroneous branch. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADOIF0-003 12 LABEL :INTERNAL REFERENCE LABEL LENGTH ERROR

Explanation: The internal macro reference label exceeds 12 characters or it is less than 2 characters.

User Response: Correct the erroneous reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADOIF0-004 12 LABEL :THE SUBROUTINE WAS ALREADY USED IN NEST

Explanation: A recursive use of the ADOIF directive for the same macro subroutine has been detected. That is, the routine labeled with the LABEL internal macro reference name was invoked for second time from the ADO nest.

User Response: Correct the erroneous reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADOIF0-005 12 LABEL :EXCEEDS MAXIMUM NUMBER OF ALLOWED NESTS

Explanation: Maximum number of PEngiCCL subroutine nests has been exceeded.

User Response: Reduce the number of subroutine nests by reorganizing macro code. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADOIF0-006 12 :LOOP COUNTER OF ZERO IS ILLEGAL

Explanation: The ADOIF directive loop counter expression resulted in zero or a negative number after it had been evaluated.

User Response: Make sure that the loop counter expression results in a positive number. If the error

occurred on a Migration Utility macro, see “Note 2” on page 151.

ADO000-001 12 LABEL :UNDEFINED INTERNAL REFERENCE LABEL

Explanation: The internal macro reference label is undefined.

User Response: Add the necessary internal reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADO000-002 12 LABEL :CANNOT BRANCH TO ITSELF, WOULD CAUSE LOOP

Explanation: An ADO directive target reference label refers to the directive itself.

User Response: Correct the erroneous branch. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADO000-003 12 LABEL :INTERNAL REFERENCE LABEL LENGTH ERROR

Explanation: The internal macro reference label exceeds 12 characters or it is less than 2 characters.

User Response: Correct the erroneous reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADO000-004 12 LABEL :THE SUBROUTINE WAS ALREADY USED IN NEST

Explanation: A recursive use of the ADO directive for the same macro subroutine has been detected. That is, the routine labeled with the LABEL internal macro reference name was invoked for second time from the ADO nest.

User Response: Correct the erroneous reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADO000-005 12 LABEL :EXCEEDS MAXIMUM NUMBER OF ALLOWED NESTS

Explanation: Maximum number of PEngiCCL subroutine nests has been exceeded.

User Response: Reduce the number of subroutine nests by reorganizing macro code. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ADO000-006 12 :LOOP COUNTER OF ZERO IS ILLEGAL

Explanation: The ADOIF directive loop counter expression resulted in zero or a negative number after it had been evaluated.

User Response: Make sure that the loop counter expression results in a positive number. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AGO000-001 12 LABEL :UNDEFINED INTERNAL REFERENCE LABEL

Explanation: The internal macro reference label is undefined.

User Response: Add the necessary internal reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AGO000-002 12 LABEL :ILLEGAL TARGET REFERENCE LABEL, WOULD CAUSE LOOP

Explanation: An Ago directive target reference label refers to the directive itself.

User Response: Correct the erroneous branch. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AGO000-003 12 LABEL :INTERNAL REFERENCE LABEL LENGTH ERROR

Explanation: The internal macro reference label exceeds 12 characters or it is less than 2 characters.

User Response: Correct the erroneous reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AGO000-004 12 MACNAME :ACTR COUNTER EXCEEDED

Explanation: The number of PEngiCCL internal macro branch instructions has been exceeded. The MACNAME is the macro in error. This was probably caused by an infinite loop, or the ACTR counter is not sufficient enough to accommodate macro needs.

User Response: Check for possible loops or increase the ACTR counter. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AGO001-001 12 LABEL :UNDEFINED INTERNAL REFERENCE SYMBOL

Explanation: The internal macro reference label is undefined.

User Response: Add the necessary internal reference label. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AGO001-002 12 LABEL :CANNOT BRANCH TO ITSELF, WOULD CAUSE LOOP

Explanation: An Ago directive target reference label refers to the directive itself.

User Response: Correct the erroneous branch. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AGO001-003 12 LABEL :INTERNAL REFERENCE SYMBOL LENGTH ERROR

Explanation: The internal macro reference label exceeds 12 characters or it is less than 2 characters.

User Response: Correct the erroneous reference label. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AGO001-004 12 LABEL :INTERNAL REFERENCE SYMBOL IS MISSING

Explanation: The internal macro reference label is undefined.

User Response: Add the necessary internal reference label. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AGO001-005 12 MACNAME :ACTR COUNTER EXCEEDED

Explanation: The number of PEngiCCL internal macro branch instructions has been exceeded. The MACNAME is the macro in error. This was probably caused by an infinite loop, or the ACTR counter is not sufficient enough to accommodate macro needs.

User Response: Check for possible loops or increase the ACTR counter. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-001 12 :LOGICAL/RELATIONAL TERM IS EXPECTED

Explanation: An SLE is expected in the PEngiCCL internal protocol but it cannot be found. This indicates a problem with PEngiCCL Macro Preprocessor.

User Response: Contact PEngiCCL software support center.

AIF000-001 12 :PEngiCCL LOGIC ERROR, SLE IS MISSING

Explanation: An SLE is expected in the PEngiCCL internal protocol but it cannot be found. This indicates a problem with PEngiCCL Macro Preprocessor.

User Response: Contact PEngiCCL software support center.

AIF000-002 12 :INCONSISTENT DATA TYPE IN RELATION

Explanation: An SLC, SAE or ELE is expected in the PEngiCCL internal protocol but it cannot be found. This indicates a problem with PEngiCCL Macro Preprocessor.

User Response: Contact PEngiCCL software support center.

AIF000-002 12 :PEngiCCL LOGIC ERROR, SLC, SAE OR ELE IS MISSING

Explanation: An SLC, SAE or ELE is expected in the PEngiCCL internal protocol but it cannot be found. This indicates a problem with PEngiCCL Macro Preprocessor.

User Response: Contact PEngiCCL software support center.

AIF000-003 12 :INCONSISTENT DATA TYPE IN RELATION

Explanation: The work buffer cannot accommodate the requirements of the conditional expression.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-003 12 :INTERMEDIATE WORK BUFFER IS TOO SMALL

Explanation: The work buffer cannot accommodate the requirements of the conditional expression.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-004 12 :UNKNOWN RELATIONAL OPERATOR

Explanation: The maximum number of bracketed expressions that can be supported by PEngiCCL has been exceeded.

User Response: Your are limited to the maximum of 64 bracketed expressions in a single conditional request. Limit the number of bracketed expressions to the maximum of 64. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-004 12 :THE NUMBER OF 64 BRACKETED EXPRESSIONS EXCEEDED

Explanation: The maximum number of bracketed expressions that can be supported by PEngiCCL has been exceeded.

User Response: Your are limited to the maximum of 64 bracketed expressions in a single conditional request. Limit the number of bracketed expressions to the maximum of 64. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-005 12 :UNKNOWN RELATIONAL OPERATOR AIF000-005 12 :EXPRESSION PROTOCOL CHAIN IS BROKEN

Explanation: The logical expression protocol chain is broken. This indicates a problem with PEngiCCL Macro Preprocessor.

User Response: Contact PEngiCCL software support center.

AIF000-006 12 :MISSING OPERAND IN EXPRESSION

Explanation: The PEngiCCL NUL protocol is outside of the answer slot range. The logical expression protocol chain is broken. This indicates a problem with PEngiCCL Macro Preprocessor.

User Response: Contact PEngiCCL software support center.

AIF000-006 12 :NUL PROTOCOL IS OUT OF RANGE

Explanation: The PEngiCCL NUL protocol is outside of the answer slot range. The logical expression protocol chain is broken. This indicates a problem with PEngiCCL Macro Preprocessor.

User Response: Contact PEngiCCL software support center.

AIF000-007 12 :LOGICAL/RELATIONAL TERM IS EXPECTED

Explanation: The work buffer cannot accommodate the requirements of the conditional expression.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-007 12 :INTERMEDIATE WORK BUFFER IS TOO SMALL

Explanation: The work buffer cannot accommodate the requirements of the conditional expression.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-008 12 :DATA VALUE IS EXPECTED IN RELATION

Explanation: The work buffer cannot accommodate the requirements of the conditional expression.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-008 12 :INTERMEDIATE WORK BUFFER IS TOO SMALL

Explanation: The work buffer cannot accommodate the requirements of the conditional expression.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-009 12 :INCONSISTENT DATA TYPE IN RELATION

Explanation: The maximum number of bracketed expressions that can be supported by PEngiCCL has been exceeded.

User Response: Your are limited to the maximum of 64 bracketed expressions in a single conditional request. Limit the number of bracketed expressions to the maximum of 64. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

AIF000-009 12 :THE NUMBER OF 64 BRACKETED EXPRESSIONS EXCEEDED

Explanation: The maximum number of bracketed expressions that can be supported by PEngiCCL has been exceeded.

User Response: Your are limited to the maximum of 64 bracketed expressions in a single conditional request. Limit the number of bracketed expressions to the maximum of 64. If the error occurred on a Migration

Utility macro, see “Note 2” on page 151.

AIF000-014 12 -TEXT- :LOGICAL/RELATIONAL TERM IS EXPECTED

Explanation: A data item or a bracketed expression is not followed by a logical or relational operator.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AIF000-015 12 -TEXT- :DATA VALUE IS EXPECTED IN RELATION

Explanation: Two or more logical or relational operators have been coded in succession.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AIF000-016 12 -TEXT- :INCONSISTENT DATA TYPE IN RELATION

Explanation: A logical or relational operation has been coded for data items of different format, that is, numeric data and alphanumeric data.

User Response: Make sure that the data items in relation are of the same type. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AIF000-018 12 -TEXT- :LOGICAL OPERATOR IS EXPECTED

Explanation: A logical operator or a Boolean is expected in the conditional expression but none found. This error is caused while evaluating the logical ″NOT″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AIF000-019 12 -TEXT- :BOOLEAN IS EXPECTED IN EXPRESSION

Explanation: A Boolean is expected in the conditional expression but none found. This error is caused while evaluating the logical ″NOT″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AIF000-020 12 -TEXT- :BOOLEAN IS EXPECTED IN EXPRESSION

Explanation: A Boolean is expected in conditional expression but none found. This error is caused while evaluating the logical ″OR″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

AIF000-021 12 -TEXT- :EXPECTING A LOGICAL OPERATOR

Explanation: A logical operator is expected in conditional expression but none found. This error is caused while evaluating the logical ″OR″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

AIF000-022 12 -TEXT- :EXPECTING A BOOLEAN IN 2ND OPERAND

Explanation: A Boolean is expected in second operand of conditional expression but none found. This error is caused while evaluating the logical ″OR″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

AIF000-023 12 -TEXT- :EXPECTING LOGICAL OR IN EXPRESSION

Explanation: A logical operator is expected in conditional expression but none found. This error is caused while evaluating the logical ″OR″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

AIF000-024 12 -TEXT- :ILLEGAL LOGICAL EXPRESSION

Explanation: The outcome of the conditional expression did not result in a valid Boolean. This is probably a PEngiCCL preprocessor error.

User Response: Contact PEngiCCL software support center.

AIF000-025 12 -TEXT- :EXPECTING BOOLEAN IN 1ST OPERAND

Explanation: A Boolean is expected in first operand of conditional expression but none found. This error is caused while evaluating the logical ″AND″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

AIF000-026 12 -TEXT- :EXPECTING UPCODE OR BOOLEAN

Explanation: A logical operator is expected in conditional expression but none found. This error is caused while evaluating the logical ″AND″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

AIF000-027 12 -TEXT- :BOOLEAN IS EXPECTED IN EXPRESSION

Explanation: A Boolean is expected in second operand of conditional expression but none found. This error is caused while evaluating the logical ″AND″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

AIF000-028 12 -TEXT- :EXPECTING LOGICAL AND

Explanation: A logical ″AND″ operator is expected in conditional expression but none found. This error is caused while evaluating the logical ″AND″.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

ALOC00-001 12 :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

ANAC00-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

ANAC00-002 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript value exceeds the declared variable dimension.

User Response: If you are trying to write your own PEngiCCL macro, you must make sure that the subscript does not exceed the declared variable dimension. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

ANUC00-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

ANUC00-002 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript value exceeds the declared variable dimension.

User Response: You must make sure that the subscript does not exceed the declared variable dimension. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

APIC00-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

APIC00-002 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript value exceeds the declared variable dimension.

User Response: You must make sure that the subscript does not exceed the declared variable dimension. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

APIC00-003 12 VARNAME :DATA STRING EXCEEDS MAXIMUM VARIABLE SIZE

Explanation: The data string (COBOL field picture) is longer than the target variable VARNAME can accommodate.

User Response: You must make sure that the target variable can accommodate your data strings. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

APIC00-004 12 PICTURE :PICTURE IS TOO LONG OR BAD DUPLICATION FACTOR

Explanation: The COBOL field picture exceeds 30 characters or it is improperly coded.

User Response: Correct the picture.

APIC00-005 12 PICTURE :ILLEGAL PICTURE FORMAT OR NO DATA INCLUDED

Explanation: The displayed picture contains illegal COBOL picture characters.

User Response: Correct the picture.

APIC00-006 12 PICTURE :RECURSIVE USE OF DECIMAL POINT

Explanation: Two or more decimal points have been detected in the COBOL picture.

User Response: Remove the extraneous decimal points.

APIC00-008 12 PICTURE :PICTURE CONTAINS ILLEGAL CHARACTERS

Explanation: The displayed picture contains illegal COBOL picture characters.

User Response: Correct the picture.

APIC00-009 12 PICTURE :PICTURE CONTAINS NUMERIC AND ALPHANUM SYMBOLS

Explanation: The displayed picture contains a mixture of numeric and alphanumeric picture characters.

User Response: Correct the picture.

APIC00-010 12 PICTURE :PICTURE EXCEEDS NUMERIC LIMIT OF 31 CHARACTERS

Explanation: The picture represents a number of more than 31 digits long.

User Response: Correct the picture.

APUNCH-001 12 :DATA MUST BE IN QUOTES FOR PUNCH DIRECTIVE

Explanation: A PUNCH directive has been attempted to punch non-quoted data.

User Response: The punch directive accepts only quoted data strings. Enclose data in quotes. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

APUNCH-002 12 :UNPAIRED/ILLEGAL QUOTES IN QUOTED STRING

Explanation: An unpaired number of quotes has been detected in a quoted data string.

User Response: Correct the data string to contain an even number of quotes. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

AREPRO-001 12 :REPRO IS ILLEGALLY FOLLOWED BY A DIRECTIVE

Explanation: A REPRO directive was followed by another directive.

User Response: The Repro directive can be used to reproduce text cards only.

ASMPUN-001 12 -TEXT- :EXPANDED PARAMETERS EXCEED 1 LINE

Explanation: A text line inside ASM macro type exceeds 1 line.

User Response: Adjust the text so that it is less than 72 bytes long.

ASOC00-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

ASORT0-001 12 :PREPROCESSOR ERROR, ASORT EXPRESSION IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

ASORT0-002 12 VARNAME :FSASORT1 - SORT ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

ATRC00-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

ATRC00-002 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript value exceeds the declared variable dimension.

User Response: You must make sure that the subscript does not exceed the declared variable dimension. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

COBMNL-001 12 MACNAME :NUMBER OF NESTED MACROS EXCEEDS MAXIMUM

Explanation: The number of supported nested macros has been exceeded.

User Response: Check to make sure that you are not invoking macros recursively from a nested macro. If you absolutely need additional macro nesting capacity, contact your PEngiCCL software administrator. The support for nested macros is generated at PEngiCCL installation time. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

COBMNL-002 12 MACNAME :FSCOBMNL LOGIC ERROR, CANNOT LOCATE MACRO NAME

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

COBMNL-003 12 MACNAME :MACRO NAME IS TOO LONG

Explanation: The macro name exceeds 12 characters.

User Response: Code the correct macro name. Note that the macro name can be up to 8 characters long on MVS/XA™ and VM/CMS operating systems, because of the PDS and CMS member naming conventions. However, temporary macro names can be up to 12 characters long. If the error occurred inside a Migration Utility macro, contact Migration Utility software support center.

COBMNL-004 12 MACNAME :ILLEGAL DECLARATION OF MACRO NAME

Explanation: The _ macro delimiter was specified without a macro name following it.

User Response: Supply the macro name.

COBRUN-001 12 VARNAME :COMPUTED SUBSCRIPT IS ZERO, IT IS ILLEGAL

Explanation: The computed subscript value is zero.

User Response: You must make sure that the subscript is not zero. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

COBRUN-002 12 DIRECTIVE :UNDEFINED PROGRAM IN FSDIRTAB

Explanation: PEngiCCL was improperly installed. A program used by the displayed DIRECTIVE was not properly resolved by the link edit program.

User Response: Contact your PEngiCCL software administrator.

CPYRUN-001 12 :COPY DIRECTIVE BUT NO MEMBER SPECIFIED

Explanation: An FSCOPY directive was coded without a copy member name.

User Response: Specify the member name to be copied following the FSCOPY directive.

CPYRUN-002 12 :IMPROPER SPECIFICATION OF MEMBER NAME

Explanation: An FSCOPY directive was coded without a copy member name.

User Response: Specify the member name to be copied following the FSCOPY directive.

CPYRUN-003 12 COPYNAME :COPY MEMBER NAME IS TOO LONG

Explanation: The FSCOPY member name exceeds 12 characters.

User Response: Code the correct FSCOPY member name. Note that the copy name can be up to 8 characters long on MVS/XA and VM/CMS operating systems, because of PDS and CMS member naming conventions.

CPYRUN-004 12 COPYNAME :NUMBER OF NESTED COPY EXCEEDS MAXIMUM

Explanation: The number of supported nested FSCOPY directives has been exceeded.

User Response: If you absolutely need additional FSCOPY nesting capacity, contact your PEngiCCL software administrator. The support for nested FSCOPY directives is generated at PEngiCCL installation time.

CPYRUN-005 12 COPYNAME :COPY ALREADY USED IN NEST (THIS NEST IS ILLEGAL)

Explanation: The COPYNAME copy member has been previously copied in this FSCOPY nest.

User Response: Only unique member names can be included in a FSCOPY directive nest, since duplicate names could cause an infinite FSCOPY loop. If you are in a need of multiple copies of the same member, consider issuing separate FSCOPY directives for each one, or write a PEngiCCL macro instead.

CPYUSR-001 12 COPYNAME :NUMBER OF NESTED COPY EXCEEDS MAXIMUM

Explanation: The number of supported nested FSCOPY/COPY directives has been exceeded.

User Response: If you absolutely need additional FSCOPY/COPY nesting capacity, contact your PEngiCCL software administrator. The support for nested FSCOPY/COPY directives is generated at PEngiCCL installation time.

CPYUSR-002 12 COPYNAME :COPY ALREADY USED IN NEST (THIS NEST IS ILLEGAL)

Explanation: The COPYNAME copy member has been previously copied in this FSCOPY nest.

User Response: Only unique member names can be included in a FSCOPY directive nest, since duplicate names could cause an infinite FSCOPY loop. If you are in a need of multiple copies of the same member, consider issuing a separate FSCOPY directives for each one, or write a PEngiCCL macro instead.

DEFADO-001 12 -TEXT- :INTERMEDIATE OUTPUT EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the ADO expression in the preprocessed format. The -TEXT- is the data string which caused the overflow.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. However, the preferred way would be to shrink the ADO expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFADO-002 12 -TEXT- :UNPAIRED LEFT PAREN IN EXPRESSION

Explanation: The internal target reference label expression in the ADO directive exceeds 256 characters.

User Response: Reduce the expression to below 256 characters in length. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFADO-003 12 :ILLEGAL INTERNAL REFERENCE LABEL

Explanation: The internal target reference label is not supplied.

User Response: The internal target reference labels must start with a "." (period) and contain at least one character. Correct the label. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFADO-004 12 -TEXT- :UNPAIRED RIGHT PAREN IN EXPRESSION

Explanation: There are more right parentheses than left parentheses in the internal target reference label or loop counter expression.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFADO-005 12 :INTERMEDIATE INPUT EXPRESSION IS TOO LONG

Explanation: Refer to the DEFADO-002 message.

DEFADO-006 12 -TEXT- :INTERNAL REFERENCE LABEL IS MISSING

Explanation: The internal target reference label is not supplied.

User Response: The internal target reference labels must start with a "." (period) and contain at least one character. Correct the label. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFADO-007 12 -TEXT- :PERIOD IS MISSING IN REFERENCE LABEL EXPRESSION

Explanation: The internal target reference label is not supplied.

User Response: The internal target reference labels must start with a "." (period) and contain at least one

character. Correct the label. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFADO-008 12 LABEL :INTERNAL REFERENCE LABEL IS TOO LONG

Explanation: The internal target reference label exceeds 12 characters.

User Response: Limit your label to maximum of 12 characters. Note that this does not include the loop counter expression, if supplied. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFADO-009 12 -TEXT- :ILLEGAL ADO/ADOIF LOOP COUNTER EXPRESSION

Explanation: The tail-end of the internal target reference label expression is illegal as written.

User Response: Correct or truncate the unneeded data string. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFCCL-001 12 MACNAME :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The ACCL directive parameters are too long.

User Response: Reduce the size of ACCL directive list/parameters.

DEFCCL-002 12 -TEXT- :STRING EXCEEDS 256 CHARACTERS

Explanation: A single data string exceeds 256 characters.

User Response: Reduce the string in error to less than 256 characters.

DEFCCL-003 12 -TEXT- :INVALID ACCL SERVICE CODE

Explanation: An invalid/unknown ACCL function has been detected.

User Response: Refer to the PEngiCCL Manual for supported ACCL functions.

DEFCCL-004 12 -TEXT- :EXPECTING ACCL DIRECTIVE

Explanation: The directive is not an ACCL directive.

User Response: None. The FSDEFCCL program supports only ACCL directive.

DEFCOM-001 12 MACNAME :INPUT DATA LENGTH IS ZERO

Explanation: The internal macro parameters work buffer has been corrupted.

User Response: Contact PEngiCCL software support center.

DEFCOM-002 12 -TEXT- :MACRO LABEL IS TOO LONG

Explanation: The macro label (paragraph name) exceeds 12 characters.

User Response: Reduce the label size to maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFCOM-003 12 MACNAME :MACRO NAME IS MISSING

Explanation: The internal macro parameters work buffer has been corrupted.

User Response: Contact PEngiCCL software support center.

DEFCOM-005 12 MACNAME :MACRO NAME IS TOO LONG

Explanation: The macro name exceeds 12 characters

User Response: Code the correct macro name. Note that the macro name can be up to 8 characters long on MVS/XA and VM/CMS operating systems, because of the PDS and CMS member naming conventions. However, the temporary macro names can be up to 12 characters long.

DEFCOM-006 12 VARNAME :MAXIMUM NUMBER OF POSITIONAL VARIABLES EXCEEDED

Explanation: The maximum number of positional parameters that can be supported by PEngiCCL has been exceeded.

User Response: The maximum number of positional parameters supported by PEngiCCL is 32,767. It is unlikely that anyone would intentionally code more than 32,767 positional parameters for a single macro invocation. The number of parameters is further limited by the work buffer size. Check to make sure that the macro end delimiter (;) is properly placed at the end of macro parameters, as this could cause extraneous data to be included as part of the macro parameters.

DEFCOM-008 12 VARNAME :UNDEFINED KEYWORD PARAMETER FOR THIS MACRO

Explanation: The VARNAME is an undefined or undeclared keyword parameter, so the keyword is not supported by the macro.

User Response: You are allowed to use only those keyword parameters which have been declared in the macro model. If this is a Migration Utility macro, refer to the appropriate section in this document for valid keywords.

DEFCOM-009 12 -TEXT- :END QUOTE IS MISSING IN QUOTED STRING

Explanation: An uneven number of quotes has been detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

DEFCOM-010 12 -TEXT- :UNPAIRED QUOTES IN QUOTED STRING

Explanation: An uneven number of quotes has been detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

DEFCOM-011 12 -TEXT- :RIGHT PAREN IS MISSING IN SUBLISTED STRING

Explanation: There are more left than right parentheses in the sublist, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses.

DEFCOM-012 12 -TEXT- :RIGHT PAREN IS MISSING IN SUBLISTED STRING

Explanation: There are more left than right parentheses in the sublist, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses.

DEFCOM-013 12 -TEXT- :IMPROPER TERMINATION OF SUBLISTED STRING

Explanation: There are more left than right parentheses in the sublist, which are not a part of a quoted string, or the last character of the sublist is not a right parenthesis. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses and that the sublist ends with a right parenthesis.

DEFCOM-014 12 -TEXT- :UNPAIRED LEFT PAREN IN SUBLISTED STRING

Explanation: There are more left than right parentheses in the sublist, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses.

DEFCOM-015 12 -TEXT- :UNPAIRED RIGHT PAREN IN SUBLISTED STRING

Explanation: There are more right than left parentheses in the sublist, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have the same number of left parentheses and right parentheses.

DEFCOM-016 12 -TEXT- :UNPAIRED PARENS IN SUBLISTED STRING

Explanation: The number of left parentheses is not equal to the number of right parentheses in the sublist, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses.

DEFCOM-017 12 -TEXT- :UNPAIRED QUOTES IN QUOTED STRING

Explanation: An uneven number of quotes has been detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

DEFCOM-019 12 -TEXT- :SUBLISTED STRING IS TOO LONG

Explanation: The data string exceeds maximum allowable string size.

User Response: Limit your sublisted string to the allowable size. The maximum string size is set at PEngiCCL installation time. The default size is 256 characters.

DEFCOM-020 12 -TEXT- :ILLEGAL CHARACTERS IN MACRO LABEL

Explanation: The macro label (paragraph name) contains illegal characters.

User Response: The macro label (paragraph name) can contain alphanumeric characters A-I, J-R, S-Z, 0-9, "#", ".", and "-". You may be further limited to the characters allowed for the language in use. Delete illegal characters.

DEFCOM-021 12 MACNAME :INSUFFICIENT VIRTUAL STORAGE, CANNOT CONTINUE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

DEFCOM-022 12 VARNAME :RECURSIVE USE OF KEYWORD PARAMETER

Explanation: The VARNAME keyword has been coded more than one time for a single macro invocation.

User Response: Remove the duplicate.

DEFCOM-023 12 VARNAME :UNDEFINED KEYWORD PARAMETER FOR THIS MACRO

Explanation: The VARNAME is an undefined or undeclared keyword parameter, so the keyword is not supported by the macro.

User Response: You are allowed to use only those keyword parameters which have been declared in the macro model. If this is a Migration Utility macro, refer to the appropriate section in this document for valid keywords.

DEFCOM-024 12 MACNAME :INSUFFICIENT VIRTUAL STORAGE, CANNOT CONTINUE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

DEFCOM-025 12 MACNAME :INPUT MACRO PARAMETERS STRING IS TOO LONG

Explanation: The macro parameters exceed the work buffer capacity or the macro end delimiter is missing. The -TEXT- is the data string which caused the overflow.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

DEFCOM-026 12 -TEXT- :ILLEGAL/INVALID FORM OF EXPRESSION

Explanation: A character following a sublisted string has been detected that is not a comma, space, or macro end delimiter.

User Response: Remove the unneeded character(s).

DEFKIK-001 12 :CONDITIONAL INTERPRETER LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

DEFKIK-002 12 :DIVISION IS IMPROPERLY DECLARED

Explanation: One of the COBOL division declarations is not followed by the word "DIVISION". That is, the Division declarative is either incomplete or misspelled.

User Response: Correct the statement in error.

DEFKIK-003 12 :CONTROL SECTION IS IMPROPERLY DECLARED

Explanation: One of the COBOL section declaratives is not followed by the word "SECTION". That is, the Section declaration is either incomplete or misspelled.

User Response: Correct the statement in error.

DEFKIK-004 12 :DECLARATION OF DIVISION/SECTION IS INCOMPLETE

Explanation: One of the COBOL section or division declaratives is followed by all spaces.

User Response: Correct the statement in error.

DEFKIK-005 12 -TEXT- :VERB/STATEMENT DISALLOWED DUE TO KICKS OPTION

Explanation: The displayed COBOL VERB/STATEMENT is disallowed because of

KICKS=YES in the COPTION PEngiCCL preprocessor options.

User Response: The KICKS VERBS/STATEMENTS disallowed are located in the FSKIKTAB table. This table has been distributed with the VERBS/STATEMENTS, as per FSKIKTAB description in this document or it has been customized by your PEngiCCL software administrator. In either case, if the KIKS=YES option is selected for PEngiCCL preprocess, you cannot use any VERBS/STATEMENTS in your program that exist in the FSKIKTAB.

DEFKIK-006 12 -TEXT- :V.S.M ERROR ALLOCATING CSECT CB QUEUE

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

DEFLEX-001 12 -TEXT- :UNPAIRED PARENS IN EXPRESSION

Explanation: The number of left parentheses is not equal to the number of right parentheses, which are not a part of a quoted string, in the expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-002 12 -TEXT- :UNPAIRED LEFT PAREN IN EXPRESSION

Explanation: The number of left parentheses is not equal to the number of right parentheses, which are not a part of a quoted string, in the expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-003 12 :INTERMEDIATE OUTPUT EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the decoded format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

DEFLEX-004 12 -TEXT- :UNPAIRED RIGHT PAREN IN EXPRESSION

Explanation: The number of right parentheses is not equal to the number of left parentheses, which are not a part of a quoted string, in the expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-005 12 :INTERMEDIATE INPUT EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the requirements of the conditional expression.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-006 12 -TEXT- :VARIABLE NAME IS TOO LONG

Explanation: The variable name in attribute T' expression is missing or it is too long.

User Response: Code a variable name following the attribute T'. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-007 12 -TEXT- :ILLEGAL FORM OF ATTRIBUTE T EXPRESSION

Explanation: The variable name in attribute T' expression does not begin with a "&" or it begins with a "&&"

User Response: Code a variable name properly following the attribute T'. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-008 12 -TEXT- :UNPAIRED QUOTES IN QUOTED STRING

Explanation: The string contains an uneven number of quotes. A quoted string must contain an even number of quotes. Double quotes inside a quoted string can be coded for quotes which need to be a part of the data string.

User Response: Code expression according to the PEngiCCL coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-009 12 -TEXT- :INCONSISTENT EXPRESSION, LOGIC/REL TERM EXPECTED

Explanation: A data item or a bracketed expression is not followed by a logical or relational operator.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-010 12 -TEXT- :ILLEGAL AMP SIGN IN QUOTED EXPRESSION

Explanation: A single "&" has been detected at the end of a quoted data string.

User Response: A single "&" indicates the beginning of a variable. Code the variable name as needed. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-011 12 :EXPRESSION EXCEEDS MAXIMUM OF 64 NESTS

Explanation: The maximum number of bracketed expressions supported by PEngiCCL has been exceeded.

User Response: Your are limited to the maximum of 64 bracketed expressions in a single conditional request. Limit the number of bracketed expressions to the maximum of 64. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-012 12 -TEXT- :ILLEGAL FORM OF EXPRESSION,)(IS ILLEGAL

Explanation: A left and a right parenthesis have been coded back-to-back outside a quoted string.

User Response: Insert the appropriate logical or relational operator between the parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-014 12 -TEXT- :ILLEGAL FORM OF EXPRESSION, LOGIC/REL TERM EXPECTED

Explanation: A data item or a bracketed expression is not followed by a logical or relational operator.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-015 12 -TEXT- :ILLEGAL FORM OF EXPRESSION, LOGICAL TERM EXPECTED

Explanation: A data item or a bracketed expression is not followed by a logical or relational operator.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-016 12 -TEXT- :ILLEGAL FORM OF EXPRESSION, AND / OR EXPECTED

Explanation: A data item or a bracketed expression is not followed by a logical operator.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-017 12 -TEXT- :ILLEGAL FORM OF EXPRESSION,)(IS ILLEGAL

Explanation: A left and a right parenthesis have been coded back-to-back outside a quoted string.

User Response: Insert the appropriate logical or relational operator between the parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-019 12 -TEXT- :ILLEGAL FORM OF EXPRESSION, RELATION EXPECTED

Explanation: A logical or relational operator was followed by a right parenthesis ")".

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-021 12 -TEXT- :ILLEGAL FORM OF EXPRESSION, LOGICAL TERM EXPECTED

Explanation: A data item or a bracketed expression is not followed by a logical or relational operator.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-022 12 -TEXT- :ILLEGAL FORM OF EXPRESSION, AND / OR EXPECTED

Explanation: A data item or a bracketed expression is not followed by a logical operator.

User Response: Make sure that your conditional expression complies with PEngiCCL conditional expression coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-023 12 -TEXT- :ALPHANUMERIC EXPRESSION EXCEEDS 256 CHARACTERS

Explanation: A single quoted/data string in conditional expression exceeds 256 characters.

User Response: Reduce the string size to below 256. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-024 12 -TEXT- :ILLEGAL FORM OF LOGICAL EXPRESSION

Explanation: The expression is invalid as written. The -TEXT- is the tail-end of the expression in error.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-025 12 -TEXT- :ARITHMETIC EXPRESSION EXCEEDS 256 CHARACTERS

Explanation: A single arithmetic expression, in the conditional expression, exceeds 256 characters.

User Response: Reduce the expression size to below 256. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-026 12 -TEXT- :ILLEGAL EXPRESSION, ALPHA TERM IS UNEXPECTED

Explanation: A quoted data string has been coded following a relational or logical operator that was preceded by a numeric term or expression.

User Response: Make sure that the data type in the relation or expression is of the same type, that is, all numeric or all alphanumeric, but not a mixture of both. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-027 12 -TEXT- :ILLEGAL EXPRESSION, NUMERIC TERM IS UNEXPECTED

Explanation: A numeric term/expression has been coded following a relational or logical operator that was preceded by an alphanumeric string.

User Response: Make sure that the data type in the relation or expression is of the same type, that is, all numeric or all alphanumeric, but not a mixture of both. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-028 12 -TEXT- :ILLEGAL EXPRESSION, THE NOT IS UNEXPECTED

Explanation: The logical operator ″NOT″ is illegal as written or out of sequence. The logical ″NOT″ can be used before a Boolean or a logical expression and in conjunction with the logical operators: AND OR, AND NOT, OR NOT.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-029 12 -TEXT- :ILLEGAL EXPRESSION, RELATIONAL TERM IS UNEXPECTED

Explanation: The relational operator is illegal as written or out of sequence. A relational operator must be preceded and followed by a data string or an arithmetic expression.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-030 12 -TEXT- :ILLEGAL EXPRESSION, LOGICAL TERM IS UNEXPECTED

Explanation: The logical operator is illegal as written or out of sequence. A logical operator must be preceded and followed by a Boolean or a relational expression.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-031 12 -TEXT- :ILLEGAL EXPRESSION, NUMERIC TERM IS UNEXPECTED

Explanation: A numeric term/expression has been coded following a relational or logical operator that was preceded by an alphanumeric string.

User Response: Make sure that the data type in the relation or expression is of the same type, that is, all numeric or all alphanumeric, but not a mixture of both. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-033 12 -TEXT- :NULL EXPRESSION IS NOT ALLOWED

Explanation: A bracketed expression has been coded with no data, so it is simply “()”.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-034 12 -TEXT- :INCOMPLETE/ILLEGAL EXPRESSION

Explanation: There are more left than right parentheses in expression, or expression was prematurely terminated.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFLEX-035 12 -TEXT- :ILLEGAL FORM OF SUBSTRING/CONCATENATION

Explanation: The substring expression is illegal as written. The -TEXT- is the tail-end of the expression in error.

User Response: Correct the expression in error. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMAC-001 12 MACNAME :INTERMEDIATE OUTPUT EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the nested macro parameters in the decoded format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

DEFMOD-001 12 VARNAME :PROTOTYPE MODEL VARIABLE SYMBOL IS TOO LONG

Explanation: The variable symbol exceeds 12 characters.

User Response: Reduce the variable symbol to maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-002 12 VARNAME :IMPROPER VARIABLE SYMBOL SPECIFICATION

Explanation: The VARNAME has been coded as a keyword variable (with "="), but a keyword variable is not allowed in the macro label.

User Response: Change the variable to a non-keyword format (drop the "="). If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-003 12 MACNAME :MACRO NAME IS NOT FOUND IN PROTOTYPE DEFINITION

Explanation: The prototype model macro name is missing.

User Response: Add the macro name to the model statements. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-004 12 -TEXT- :PROTOTYPE MODEL MACRO NAME IS TOO LONG

Explanation: The prototype model macro name exceeds 12 characters.

User Response: Code the correct macro name. Note that the macro name can be up to 8 characters long on MVS/XA and VM/CMS operating systems, because of the PDS and CMS member naming conventions. However, temporary macro names can be up to 12 characters long. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-005 12 -TEXT- :PROTOTYPE MODEL MACRO NAME IS INCONSISTENT

Explanation: The macro name does not equal to the member name that houses the macro source.

User Response: Make your macro name in the prototype model equal to the PDS/CMS member name that houses this macro. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-006 12 -TEXT- :PROTOTYPE MODEL VARIABLE SYMBOL IS TOO LONG

Explanation: The variable symbol exceeds 12 characters.

User Response: Reduce the variable symbol to maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-007 12 -TEXT- :NO VARIABLE FOUND IN PROTOTYPE DEFINITION

Explanation: A local or global directive has been coded without the variable name.

User Response: Add the required variable or remove the unneeded directive. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-008 12 -TEXT- :MISSING END QUOTE, PROTOTYPE MODEL IS INCOMPLETE

Explanation: Unpaired quotes have been detected in the macro prototype model definition.

User Response: Add quotes as needed. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-011 12 -TEXT- :INCOMPLETE QUOTED STRING IN PROTOTYPE DEFINITION

Explanation: Unpaired quotes have been detected in the macro prototype model definition.

User Response: Add quotes as needed. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-012 12 -TEXT- :RIGHT PAREN IS MISSING IN PROTOTYPE DEFINITION

Explanation: The number of right parentheses doesn't equal the number of left parentheses in the expression, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-017 12 -TEXT- :UNPAIRED PARENS IN PROTOTYPE DEFINITION

Explanation: The number of right parentheses doesn't equal the number of left parentheses in the expression, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-019 12 VARNAME :EXCEEDS VARIABLES BUFFER CAPACITY

Explanation: The acquired buffer during the variable decoding in PASS1 cannot accommodate the variable data string. This is probably PEngiCCL preprocessor error.

User Response: Contact PEngiCCL software support center.

DEFMOD-020 12 VARNAME :SUBLISTED STRING IS TOO LONG

Explanation: The data string exceeds maximum allowable string size.

User Response: Limit your sublisted string to the allowable size. The maximum string size is set at PEngiCCL installation time. The default size is 256 characters. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

DEFMOD-021 12 VARNAME :ILLEGAL PROTOTYPE VARIABLE SYMBOL

Explanation: The variable name contains illegal characters. The variable name can contain the alphanumeric characters A-I, J-R, S-Z, 0-9, "#", ".", and "-".

User Response: Assign a name that contains the allowed characters only. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-022 12 VARNAME :ILLEGAL CHARACTERS IN PROTOTYPE VARIABLE SYMBOL

Explanation: The variable name contains illegal characters. The variable name can contain the alphanumeric characters A-I, J-R, S-Z, 0-9, "#", ".", and "-".

User Response: Assign a name that contains the allowed characters only. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-023 12 VARNAME :ILLEGAL CHARACTERS IN PROTOTYPE VARIABLE SYMBOL

Explanation: The variable name contains illegal characters. The variable name can contain the alphanumeric characters A-I, J-R, S-Z, 0-9, "#", ".", and "-".

User Response: Assign a name that contains the allowed characters only. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-024 12 -TEXT- :MISSING RIGHT PAREN IN GBL/LCL SET DEFINITION

Explanation: A right parenthesis is missing in the dimension of a local or global set symbol.

User Response: Add the necessary right parenthesis. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-025 12 VARNAME :ILLEGAL USE OF RESERVED SYSTEM VARIABLE

Explanation: The VARNAME is a reserved PEngiCCL system variable symbol. System variable symbol cannot be declared inside a macro prototype or macro set symbols.

User Response: Use a non-system variable name. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-026 12 -TEXT- :ILLEGAL VALUE IN SUBLIST DIMENSION

Explanation: A null entry has been coded for the local/global set symbol dimension.

User Response: Code a numeric dimension. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-027 12 -TEXT- :DIMENSION EXCEEDS 5 DIGITS IN GBL/LCL DEFINITION

Explanation: The dimension of the local/global set symbol exceeds 5 characters.

User Response: Limit the dimension to 5 characters in length. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-028 12 -TEXT- :DIMENSION IS NOT NUMERIC IN GBL/LCL DEFINITION

Explanation: The dimension value of the local/global set symbol is not numeric.

User Response: Code a numeric dimension. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-029 12 -TEXT- :ILLEGAL DIMENSION IN GBL/LCL SET DEFINITION

Explanation: The dimension value of the local/global set symbol is zero.

User Response: The allowed dimension can be 1 to 32767. Code a valid dimension. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-030 12 -TEXT- :DIMENSION EXCEEDS MAXIMUM IN GBL/LCL DEFINITION

Explanation: The dimension value of the local/global set symbol is greater than 32767.

User Response: The allowed dimension can be 1 to 32767. Code a valid dimension. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-031 12 -TEXT- :ILLEGAL LCL/GBL DECLARATIVE

Explanation: The local/global set symbol dimension is illegal as written.

User Response: Code dimension according to the PEngiCCL coding standards. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-032 12 -TEXT- :ILLEGAL OR UNDECLARED SYMBOL IN SET DEFINITION

Explanation: The Symbol used in the local/global set dimension is undefined.

User Response: Code dimension according to the PEngiCCL coding standards. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-033 12 VARNAME :DUPLICATE OR ILLEGAL PROTOTYPE VARIABLE SYMBOL

Explanation: The VARNAME variable has been previously declared either in the prototype model or as a local/global set symbol.

User Response: Delete the duplicate variable definition. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-035 12 MACNAME :NO VIRTUAL STORAGE AVAILABLE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

DEFMOD-036 12 VARNAME :INCONSISTENT GLOBAL VARIABLE DEFINITION

Explanation: The VARNAME global set symbol/variable is not consistent with the definition of the same global set symbol/variable defined in another macro. The items that can cause inconsistency are the dimension, the set symbol type and the variable-length.

User Response: Identify the differences and code your variable to comply. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-037 12 VARNAME :INCONSISTENT GLOBAL VARIABLE DEFINITION

Explanation: The VARNAME global set symbol/variable is not consistent with the definition of the same global set symbol/variable defined in another macro. The items that can cause inconsistency are the dimension, the set symbol type and the variable-length.

User Response: Identify the differences and code your variable to comply. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-038 12 VARNAME :MAXIMUM NUMBER OF LOCAL VARIABLES EXCEEDED

Explanation: The maximum number of macro variables (prototype model and set symbols) that can be coded for this macro has been exceeded.

User Response: The number of variables is limited as declared in the macro statement via the VARQ=NN option, where NN = the number of allowed variables. The default support for the number of macro local and global variables is established at PEngiCCL installation time by the PEngiCCL software administrator.

User Response: Increase the NN value of the VARQ=NN option on the macro statement to support additional entries. If the error occurred on a Migration Utility macro, contact Migration Utility software support center. Caution: Do not grossly over estimate the NN value, as it could cause the use of unnecessary virtual storage.

DEFMOD-039 12 VARNAME :MAXIMUM NUMBER OF GLOBAL VARIABLES EXCEEDED

Explanation: The maximum number of the global set symbols/variables has been exceeded.

User Response: The default support for the number of global variables is established at PEngiCCL installation time by the PEngiCCL software administrator. If you are in a need of more variables, have the PEngiCCL software administrator change the default value. The value can be changed via the GBLGRP=NN keyword in the FSCOBNUC program. However, the PEngiCCL nucleus must be re-linked. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-040 12 VARNAME :ILLEGAL FORM OF SUBSCRIPT EXPRESSION

Explanation: An element (slot) length has been coded for a sublisted local/global SETA or SETB symbol.

User Response: The element size is supported for the SETC symbols only. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-041 12 VARNAME :LENGTH DEFINITION IN SUBSCRIPT IS TOO SHORT/LONG

Explanation: The length for the VARNAME sublisted SETC symbol is either zero or it exceeds 15 digits.

User Response: Code a proper numeric length. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-042 12 VARNAME :LENGTH VALUE IN SUBSCRIPT EXPRESSION IS NOT NUMERIC

Explanation: The length value for the VARNAME sublisted SETC symbol is not numeric.

User Response: Code a proper numeric length. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-043 12 VARNAME :LENGTH VALUE IN SUBSCRIPT EXPRESSION IS ILLEGAL

Explanation: The length for the VARNAME sublisted SETC symbol is either zero or it exceeds 15 digits.

User Response: Code a proper numeric length. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-044 12 -TEXT- :INVALID/ILLEGAL PROTOTYPE MODEL EXPRESSION

Explanation: Sublisted prototype model expression is illegally terminated. That is, the expression is not followed by a space or comma.

User Response: Remove extraneous data following the expression or insert a comma or space. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFMOD-045 12 MACNAME :COBOLBAS OR CICSBASE MACRO IS REQUIRED

Explanation: The COBOLBAS or the CICSBASE macro was not coded before the macro in error.

DEFMOD-046 12 MACNAME :REQUIRES CICSBASE MACRO

Explanation: The macro can be used with CICSBASE macro only.

DEFMOD-047 12 MACNAME :REQUIRES COBOLBAS MACRO

Explanation: The macro can be used with COBOLBAS macro only.

DEFOPT-001 04 -TEXT- :UNKNOWN OR IMPROPER COPTION PARAMETER

Explanation: The -TEXT- is an unsupported/unknown PEngiCCL option.

User Response: Correct by using one of the allowed COPTION parameters.

DEFOPT-002 04 -TEXT- :IMPROPER COPTION PARAMETER VALUE

Explanation: An illegal value or no data has been coded for one of the COPTION parameters.

User Response: Code a proper value for the keyword in error.

DEFSQE-001 12 MACNAME :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSQE-002 12 -TEXT- :STRING EXCEEDS 256 CHARACTERS

Explanation: A quoted data string exceeds maximum allowable string size.

User Response: Limit your quoted data strings to the allowable size of 256 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSQE-003 12 -TEXT- :ILLEGAL FORM OF EXPRESSION

Explanation: The quoted string contains either illegal attributes or improper concatenation. The -TEXT- is the tail-end of the string in error.

User Response: Correct the string to comply with PEngiCCL quoted string coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSQE-004 12 -TEXT- :UNPAIRED PARENS IN SUBSTRING EXPRESSION

Explanation: The number of right parentheses is not equal to the number of left parentheses in the substring expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSQE-005 12 -TEXT- :SUBSTRING EXCEEDS 256 CHARACTERS

Explanation: The substring expression exceeds 256 characters in length.

User Response: Limit your expression to maximum of 256 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSQE-006 12 -TEXT- :ILLEGAL SUBSTRING EXPRESSION

Explanation: The subscript expression is illegal as written.

User Response: Remove the unneeded expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-001 12 :VARIABLE TO BE SORTED IS NOT SUPPLIED

Explanation: The ASORT directive has been coded with no variable to sort.

User Response: Supply the variable to be sorted. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-002 12 VARNAME :VARIABLE SYMBOL IS TOO LONG

Explanation: The variable name exceeded 12 characters.

User Response: Limit the variable name to maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-003 12 -TEXT- :ILLEGAL SPECIFICATION OF VARIABLE SYMBOL

Explanation: The variable name to be sorted does not start with a “&”.

User Response: Prefix variable name with a “&”. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-004 12 VARNAME :UNDEFINED VARIABLE SYMBOL

Explanation: The VARNAME variable to be sorted is undefined in this macro.

User Response: Supply a correct variable that has

been defined in this macro. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-006 12 -TEXT- :ILLEGAL SORT TYPE, VALID OPTIONS ARE A OR D

Explanation: The sort option is invalid.

User Response: Correct the bad option. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-007 12 VARNAME :ILLEGAL USE OF PROTOTYPE MODEL VARIABLE

Explanation: The variable to be sorted is a prototype model variable. The prototype model variables cannot be sorted.

User Response: Use the correct variable that has been defined as a local or global set symbol. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-008 12 -TEXT- :ILLEGAL SORT SUPPRESS INDICATOR, MUST BE Y OR N

Explanation: The sort suppress option indicator is invalid. The suppress indicator can be Y or N only.

User Response: Correct the bad option indicator. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-009 12 VARNAME :ILLEGAL USE OF NON-DIMENSIONAL VARIABLE

Explanation: An attempt to sort a non-dimensional variable has been detected.

User Response: Only multi-dimensional variables can be sorted. Supply a multi-dimensional variable. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSRT-010 12 -TEXT- :ILLEGAL ASORT OPTIONS

Explanation: The ASORT options are illegal as written.

User Response: Correct the bad options. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-001 12 -TEXT- :UNPAIRED PARENS IN EXPRESSION

Explanation: The number of right parentheses is not equal to the number of left parentheses in the expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-002 12 -TEXT- :MISSING RIGHT PAREN IN EXPRESSION

Explanation: The number of right parentheses is not equal to the number of left parentheses in the expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-003 12 :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-004 12 -TEXT- :ILLEGAL FORM OF EXPRESSION (SUBSCRIPT EXPECTED)

Explanation: The subscript expression is not preceded by a valid variable symbol.

User Response: Correct the expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-005 12 -TEXT- :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-006 12 :ILLEGAL USE OF SUBSTRING IN ARITHMETIC EXPRESSION

Explanation: The double subscript (X,Y) was used for a variable that was not a &SYSLIST variable.

User Response: Correct the expression. The subscript can be of (X,Y) format only for the &SYSLIST system variable. If the error occurred on a Migration Utility

macro, contact Migration Utility software support center.

DEFSUB-007 12 -TEXT- :ILLEGAL FORM OF EXPRESSION

Explanation: A bracketed expression was coded with no data inside of it, as "()". This expression is illegal.

User Response: Correct the expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-008 12 -TEXT- :ILLEGAL BEGINNING/END OF EXPRESSION

Explanation: An arithmetic operator was followed by a right parenthesis, or a right parenthesis was not followed by an arithmetic operator.

User Response: Correct the expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFSUB-009 12 :EXPRESSION EXCEEDS MAXIMUM OF 64 BRACKETED TERMS

Explanation: The maximum number of bracketed expressions supported by PEngiCCL was exceeded.

User Response: Your are limited to a maximum of 64 bracketed expressions in a single arithmetical expression. Limit the number of bracketed expressions to the maximum of 64. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFTXT-001 12 MACNAME :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

DEFTXT-002 12 -TEXT- :STRING EXCEEDS 256 CHARACTERS

Explanation: The data string exceeds 256 characters.

User Response: Limit your data string to maximum of 256 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-001 12 MACNAME :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-002 12 VARNAME :ILLEGAL USE OF SUBSCRIPT FOR THIS VARIABLE

Explanation: The VARNAME variable is not a sublisted variable.

User Response: The non-sublisted variables cannot be subscripted. Delete the subscript. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-003 12 VARNAME :UNDEFINED VARIABLE SYMBOL

Explanation: The VARNAME variable is undefined in this macro.

User Response: Supply a correct variable that has been defined in this macro. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-004 12 VARNAME :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-005 12 VARNAME :ILLEGAL USE OF T ATTRIBUTE IN ARITHMETIC

Explanation: The use of T′ attribute has been detected in arithmetic expression.

User Response: The attribute T′ is an alphanumeric type. Delete the erroneous attribute. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-006 12 VARNAME :ILLEGAL USE OF SUBSTRING EXPRESSION

Explanation: The subscript expression is illegal as written or it is not allowed in expression.

User Response: Correct the erroneous expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-007 12 VARNAME :SUBLIST IS NOT ALLOWED

Explanation: The double subscript (X,Y) has been used for a variable that is not a &SYSLIST variable.

User Response: Correct the expression. The subscript can be of (X,Y) format only for the &SYSLIST system variable. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-008 12 VARNAME :ILLEGAL USE OF SUBSTRING IN ARITHMETIC

Explanation: A substring notation was detected following an attribute expression.

User Response: Substring usage is not allowed in an arithmetic expression, as it deals with alphanumeric data. Correct the erroneous expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-009 12 -TEXT- :UNPAIRED PARENS IN SUBSCRIPT EXPRESSION

Explanation: The number of right parentheses is not equal to the number of left parentheses in the expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-010 12 -TEXT- :ILLEGAL VALUE IN ARITHMETIC EXPRESSION

Explanation: Illegal data has been coded in arithmetic expression. The -TEXT- is the tail-end of the last data examined.

User Response: Remove or correct the illegal/invalid data string. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-011 12 :PREPROCESSOR LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

GENSUB-012 12 VARNAME :THE USE OF VARIABLE REQUIRES A SUBSCRIPT

Explanation: The VARNAME sublisted variable was coded without a subscript.

User Response: Code the required subscript. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-013 12 -TEXT- :UNPAIRED QUOTES IN HEX EXPRESSION

Explanation: Hex expression was not properly coded in quotes.

User Response: Code quotes around the hex value. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-014 12 -TEXT- :ILLEGAL HEX EXPRESSION

Explanation: Hex expression was either too long or too short.

User Response: Adjust the hex value within the limits of PEngiCCL rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENSUB-015 12 -TEXT- :ILLEGAL FORM OF EXPRESSION

Explanation: Hex expression contains illegal (non-hex) characters.

User Response: Correct the erroneous characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-001 12 MACNAME :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-002 12 VARNAME :VARIABLE SYMBOL IS TOO LONG

Explanation: The variable name exceeded 12 characters.

User Response: Limit the variable name to maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-003 12 VARNAME :UNDEFINED VARIABLE SYMBOL

Explanation: The VARNAME variable to be sorted was undefined to this macro.

User Response: Supply a correct variable that is defined in this macro. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-004 12 VARNAME :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-005 12 VARNAME :SUBLIST/SUBSTRING EXPRESSION IS ILLEGAL

Explanation: Attribute T′ expression for the VARNAME variable was followed by a quote and a left parenthesis, which implies a substring usage. Substrings are not allowed for attribute expressions.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-006 12 VARNAME :ATTRIBUTES ARE NOT ALLOWED

Explanation: Attribute T′ expression for the VARNAME variable was followed by a quote and a left parenthesis, which implies a substring usage. Substrings are not allowed for attribute expressions.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-007 12 VARNAME :SUBLIST/SUBSTRING EXPRESSION IS ILLEGAL

Explanation: Attribute K′ expression for the VARNAME variable was followed by a quote and a left parenthesis, which implies a substring usage. Substrings are not allowed for attribute expressions.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-008 12 VARNAME :SUBLIST EXPRESSION IS ILLEGAL

Explanation: A subscript was coded for an attribute N′ expression. The subscripts are allowed in the attribute N′ expression only for the &SYSLIST variable.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-009 12 VARNAME :SUBSTRING EXPRESSION IS ILLEGAL

Explanation: Attribute N′ expression for the VARNAME variable was followed by a quote and a left parenthesis, which implies a substring usage. Substrings are not allowed for attribute expressions.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-010 12 VARNAME :ILLEGAL USE OF SUBSCRIPT FOR VARIABLE TYPE

Explanation: The VARNAME variable was not a sublisted variable.

User Response: The non-sublisted variables cannot be subscripted. Delete the subscript. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-011 12 -TEXT- :STRING EXCEEDS MAXIMUM OF 256 CHARACTERS

Explanation: A continuous data string was detected that was longer than 256 characters.

User Response: Limit the string to maximum of 256 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-012 12 -TEXT- :INCOMPLETE SUBSCRIPT EXPRESSION

Explanation: The expression was not properly enclosed in parentheses or no data was supplied in subscript.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-013 12 -TEXT- :UNPAIRED RIGHT PAREN IN SUBSCRIPT EXPRESSION

Explanation: The number of right parentheses doesn't equal the number of left parentheses in the expression. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-014 12 -TEXT- :SUBSCRIPT EXPRESSION IS TOO LONG

Explanation: The expression exceeds 256 characters. The -Text- is the tail-end of the last data examined.

User Response: Limit expression to maximum of 256 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-015 12 -TEXT- :ILLEGAL USE OF SUBSTRING FOR VARIABLE TYPE

Explanation: The double subscript (X,Y) was used for a variable that was not a &SYSLIST variable.

User Response: Correct the expression. The subscript can be of (X,Y) format only for the &SYSLIST system variable. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-016 12 VARNAME :THE USE OF VARIABLE REQUIRES A SUBSCRIPT

Explanation: The VARNAME sublisted variable was coded without a subscript.

User Response: Code the required subscript. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GENTXT-017 12 -TEXT- :SINGLE QUOTE IS ILLEGALLY USED IN QUOTED STRING

Explanation: A single quote was detected in a quoted string.

User Response: Quotes inside a quoted string must be coded in pairs, that is, as double quotes. Correct the erroneous expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GETCOM-001 12 -TEXT- :ILLEGAL VALUE IN COBOL CC 7

Explanation: The character in position seven was not a "*", "/", "-" or a space.

User Response: Remove the invalid character.

GETCOM-002 12 :NON BLANK FOUND BEFORE CONTINUATION COLUMN CC16

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, however, data was located before continuation column 16 of this statement.

User Response: Continuation starts in position 16 for Assembler macros and in position 12 for COBOL macros. Correct the statement in error.

GETCOM-004 12 -TEXT- :OPCODE OR MACRO IS TOO LONG

Explanation: The macro name exceeded 12 characters.

User Response: Limit macro name to maximum of 12 characters.

GETCOM-005 12 MACNAME :FS-PEngiCCL MACRO IS ILLEGALLY USED IN FSCOPY

Explanation: The special PEngiCCL feature of FSCOPY directive to support CICS macros in COBOL was activated, but the MACNAME macro was not in FSUSRTAB macro table.

User Response: The FSUSRTAB contains all macro names allowed for use with the FSCOPY directive. Contact your PEngiCCL software administrator for the valid macros.

GETCPY-001 12 COPYNAME :COPY MEMBER NOT FOUND IN COPY LIBRARY

Explanation: The COPYNAME member did not exist in the copy library. This was caused by either an erroneous COPYNAME or the library which contained the member was not properly accessed/concatenated.

User Response: Correct the COPYNAME or concatenate/access the correct library.

GETCPY-002 12 COPYNAME :ERRORS DETECTED IN MACRO PROTOTYPE

Explanation: An error has been detected in Easytrieve Plus macro model.

User Response: Refer to PEngiEZT Reference Manual for syntax rules.

GETCPY-003 12 SUPPLIED PARAMETERS EXCEED BUFFER SIZE

Explanation: Easytrieve Plus macro input parameters exceed the reserved buffer space.

User Response: The buffer space is allocated based on input macro prototype. Verify input parameters for proper values and/or size.

GETCPY-004 12 KEYWORD :UNDEFINED KEYWORD

Explanation: Easytrieve Plus macro input keyword is not defined in macro model.

User Response: Verify input parameters for proper keywords.

GETCPY-005 12 KEYWORD :DUPLICATE USER KEYWORD

Explanation: There is a duplicate keyword in the supplied Easytrieve Plus macro parameters.

User Response: Remove the duplicate keyword.

GETCPY-006 12 -TEXT- :EXTRANEOUS USER PARAMETERS

Explanation: Too many parameters has been supplied for Easytrieve Plus macro.

User Response: Remove the extraneous parameters.

GETMAC-001 12 MACNAME :INVALID OP CODE OR MACRO NOT FOUND IN LIBRARY

Explanation: The MACNAME member did not exist in the macro library. This was caused by either an erroneous MACNAME or the library which contained the member was not properly accessed/concatenated.

User Response: Correct the MACNAME or concatenate/access the correct library.

GETMAC-002 12 MACNAME :INVALID OP CODE OR MACRO WAS FOUND IN ERROR

Explanation: The MACNAME member did not exist in the macro library or the macro was previously found in error. This could be caused by either an erroneous MACNAME or the library which contained the member was not properly accessed/concatenated.

User Response: Correct the MACNAME or concatenate/access the correct library.

GETPGM-001 12 PROGRAM :PREMATURE END OF INPUT OR MEND IS MISSING

Explanation: A premature end of input program source was detected. This could be caused by an improperly coded MEND directive in one of the temporary macros included before the program, or the program was not located in the FJSYSIN library.

User Response: Correct the erroneous temporary macro, or supply the proper program name on the FJSYSIN if on MVS/XA, or CMS member name and type if on VM/CMS.

GETPGM-002 12 -TEXT- :IMPROPER DECLARATION OF MACRO LABEL IN PROTOTYPE

Explanation: The prototype model macro label variable did not start with a "&" in the temporary macro definition. This error can happen only if you code temporary macros before your program.

User Response: Add a "&" to the macro label.

GETPGM-003 12 -TEXT- :EXPECTING A MACRO NAME, NONE FOUND

Explanation: The macro name was expected in the prototype model in the first 32 positions of the first model statement, or in the first 32 positions after the macro label, if the label was coded. This error can happen only if you code temporary macros before your program.

User Response: Make sure that the macro name starts within the first 32 positions of the first macro model statement.

GETPGM-004 12 -TEXT- :MACRO NAME EXCEEDS MAXIMUM LENGTH

Explanation: The macro name exceeded 12 characters. This error can happen only if you code temporary macros before your program.

User Response: Limit the macro name to maximum of 12 characters.

GETPGM-005 12 MACNAME :FSVSMADD CALL, NO VIRTUAL STORAGE AVAILABLE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA systems increase the REGION size on the EXEC statement, on VM/CMS

systems increase the virtual storage of your CMS machine.

GETPGM-006 12 MACNAME :FSVSMDSA CALL, NO VIRTUAL STORAGE AVAILABLE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA systems increase the REGION size on the EXEC statement, on VM/CMS systems increase the virtual storage of your CMS machine.

GETPGM-007 12 MACNAME :FSVSMDSA CALL, NO VIRTUAL STORAGE AVAILABLE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA systems increase the REGION size on the EXEC statement, on VM/CMS systems increase the virtual storage of your CMS machine.

GETPGM-008 12 MACNAME :PREMATURE END OF INPUT OR MEND IS MISSING

Explanation: A premature end of input program source was. This could be caused by an improperly coded MEND directive in one of the temporary macros included before the program, or the program was not located in the FJSYSIN library.

User Response: Correct the erroneous temporary macro or supply the proper program name on the FJSYSIN if on MVS/XA, or CMS member name ad type if on VM/CMS.

GETPGM-009 12 MACNAME :FSVSMDSA CALL, NO VIRTUAL STORAGE AVAILABLE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

GETPGM-010 12 VARNAME :PROTOTYPE MODEL VARIABLE SYMBOL IS TOO LONG

Explanation: The macro label (variable symbol) exceeded 12 characters. This error can happen only if you code temporary macros before your program. Solution. Limit the label symbol to the maximum of 12 characters.

GETSYS-001 12 MACNAME :V.S.M FAILED ON LOADING SYSTEM VARIABLES

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

GETSYS-002 12 MACNAME :SYSPARM DATA LENGTH EXCEEDS 54 BYTES

Explanation: The COPTION parameters coded in the SYSPARM of MVS/XA JCL exceeded 54 characters.

User Response: The MVS/XA SYSPARM can contain maximum of 54 characters. Reduce the COPTION parameters to the maximum of 54 bytes.

GSECT0-001 12 GSECT :CONTROL SECTION NAME EXCEEDS MAXIMUM LENGTH

Explanation: The GSECT expression exceeded 12 characters after all string substitutions have been made.

User Response: Limit the string to maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GSECT0-002 12 GSECT :CONTROL SECTION NAME IS ILLEGAL AS WRITTEN

Explanation: The GSECT expression was less than 2 characters after all string substitutions were made.

User Response: Limit the string to minimum of 2 and a maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GSECT0-003 12 GSECT :UNKNOWN CONTROL SECTION

Explanation: The GSECT generating section was unknown to PEngiCCL.

User Response: Code a proper GESCT identification. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GSECT0-004 12 -TEXT- :CONTROL SECTION NAME NOT ENCLOSED IN PARENS

Explanation: The GSECT name was not properly enclosed in parentheses.

User Response: Enclose the name in parentheses. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GSECT0-005 12 -TEXT- :ILLEGAL CHARACTERS IN CONTROL SECTION NAME

Explanation: The GSECT name contains illegal characters.

User Response: The allowed characters are: #, A-I, J-R, S-Z and 0-9. Change the GSECT name to contain proper characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

GSECT0-006 12 GSECT :ILLEGAL GSECT,LANG=ASM ALLOWS GP GSECT ONLY

Explanation: An illegal GSECT for LANG=ASM PEngiCCL preprocessor option was detected.

User Response: Only General purpose (GP) GSECT can be used when preprocessing assembler programs. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

IOR000-001 12 :ILLEGAL VALUE IN COBOL SEQUENCE (CC 0-5)

Explanation: Non-numeric characters was detected in position 1-6. This error appears only when COBLSEQ=YES COPTION is in effect.

User Response: Remove erroneous sequence number.

IOR003-001 12 PREMATURE END OF FUNCTION

Explanation: End of input source has been detected while decoding function. This can occur when input parameters contain unpaired quotes or parentheses.

User Response: Correct the problem.

IOR003-002 12 *XXXXX* UNPAIRED PARENS IN EXPRESSION

Explanation: Data has been detected in cc 8 - 12 before a paired parenthesis could be reached.

User Response: Correct the problem.

IOR003-003 12 *XXXXX* EXPRESSION IS TOO LONG

Explanation: Function, with its parameters, exceeds the buffer size specified by the BUFSIZE=nnn in COPTION. Also, this can occur when input parameters contain unpaired quotes or parentheses.

User Response: Correct the problem.

IOR003-004 12 *XXXXX* PARAMETER LIST IS MISSING

Explanation: Function is not followed by a parameter list enclosed in parentheses. The error can also occur when there are more right parentheses "(" than left parentheses ")" in the parameter list.

User Response: Correct the problem. Note that if you do not have any function parameters, you must code an empty list, that is `()`.

IOR003-005 12 CCL1 LOGIC ERROR

Explanation: A serious error has occurred during function decoding.

User Response: Contact Support Center.

IOR003-006 12 *XXXXX* FUN/OBJECT UNDEFINED OR TOO LONG

Explanation: Function is undefined or the function name is too long.

User Response: For function naming conventions refer to VSMF00-08 message, otherwise, functions must be declared in order to use them.

IOR003-007 12 *XXXXX* OVERLY FRAGMENTED FUNCTION

Explanation: The function design cannot be handled by Migration Utility preprocessor. This can happen when a function contains too many imbedded functions, causing the management of the generated code impossible.

User Response: Simplify function design.

IOR003-008 12 *XXXXX* IMPROPER USE OF FUNCTION

Explanation: The use of function is improper as coded.

User Response: Refer to PEngiCCL Reference Manual for function usage. In general, if a function is used with a COBOL instruction, then it must be coded with a leading "%".

For example:

```
IF SEL_OBJECT (OBJECT OPTION) = ZERO
```

is syntactically incorrect. It should be coded with "%" as:

```
IF %SEL_OBJECT (OBJECT OPTION) = ZERO
```

IOR003-009 12 XXXXX INCONSISTENT NUMBER OF PARAMETERS

Explanation: The number of coded function parameters does not match to the number declared in the function model.

User Response: Correct the problem.

IOR003-010 12 MAXIMUM OF 999 FUNCTIONS EXCEEDED

Explanation: The number of selector functions in the program exceeds 999.

User Response: None. The only recourse is to split your program into multiple modules or use fewer selector functions.

IOR003-011 12 XXXXX NON-BLANK BEFORE COLUMN 12

Explanation: A non-blank was detected in cc 8 - 12 during function decoding. This can occur when input parameters contain unpaired quotes or parentheses.

User Response: Correct the problem. Also see "Note 3" on page 151.

IOR003-012 12 XXXXX INVALID OR MISSING OBJECT NAME

Explanation: Non-inline function has a null first parameter, or the first parameter is a quoted string or not a valid COBOL field name.

User Response: Correct the problem.

IOR003-013 12 XXXXX INLINE FUNCTION LOGIC ERROR

Explanation: Inline function did not generate any statements.

User Response: This is function designer error. Function must be corrected to generate at least one COBOL line (even if it is a blank or comment.

IOR003-014 12 XXXXX INLINE FUNCTION AREA "A" NON-BLANK

Explanation: A non-blank was detected in cc 8 - 12 in the code generated by the inline function.

User Response: This is function designer error. Function must be corrected to generate inline statements starting in cc 12 and after.

MEXIT0-001 12 MACNAME :ERROR FREEING MACRO POINTERS

Explanation: An error occurred, which freed macro working set virtual storage pointers. This is Probably PEngiCCL preprocessor logic error.

User Response: Contact PEngiCCL software support center.

MNOTE0-001 12 -TEXT- :ILLEGAL EXPRESSION FORMAT

Explanation: The MNOTE condition code and text are illegal as coded. The -TEXT- is the last data analyzed.

User Response: Code the mnote according to the PEngiCCL MNOTE directive coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-002 12 -TEXT- :ILLEGAL EXPRESSION, CC AND TEXT REQUIRED

Explanation: The MNOTE condition code and text are illegal as coded. The -TEXT- is the last data analyzed.

User Response: Code the MNOTE according to the PEngiCCL MNOTE directive coding rules. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-003 12 -TEXT- :CONDITION CODE LENGTH ERROR

Explanation: The condition code was more than 10 characters long.

User Response: Limit condition code to a maximum of 10 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-004 12 -TEXT- :CONDITION CODE IS NOT NUMERIC

Explanation: The condition code was not numeric.

User Response: Code a numeric value for condition code from 0 through 999. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-005 12 -TEXT- :CONDITION CODE EXCEEDS 999

Explanation: The condition code exceeded maximum of 999.

User Response: Reduce condition code to maximum of 999. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-007 12 -TEXT- :SYNTAX ERROR, ILLEGAL TEXT FORMAT

Explanation: A null or illegal text was coded for the MNOTE message.

User Response: Correct the necessary message text. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-008 12 -TEXT- :SYNTAX ERROR, TEXT MUST BE IN QUOTES

Explanation: The MNOTE message was not enclosed in quotes.

User Response: Enclose the message text in quotes. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-009 12 -TEXT- :SYNTAX ERROR, END QUOTE IS MISSING

Explanation: The MNOTE message was not terminated with a quote.

User Response: Enclose the message in quotes. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

MNOTE0-010 12 -TEXT- :SYNTAX ERROR, IMPROPER ERROR NUMBER

Explanation: The MNOTE message is improper as coded.

User Response: Refer to PEngiCCL Reference manual for proper MNOTE syntax. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-001 12 VARNAME :VARIABLE SYMBOL IS TOO LONG

Explanation: The variable name exceeded 12 characters.

User Response: Limit the variable name to a maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-002 12 -TEXT- :SUBSCRIPT EXPRESSION EXCEEDS 256 CHARACTERS

Explanation: The subscript expression exceeded 256 characters. The -text- is the tail-end of the last data examined.

User Response: Limit expression to a maximum of 256 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-003 12 LABEL :INTERNAL MACRO REFERENCE LABEL IS TOO LONG

Explanation: The internal macro reference label was too long.

User Response: Limit the reference label symbol to a maximum of 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-004 12 LABEL :INVALID INTERNAL MACRO REFERENCE LABEL

Explanation: An internal macro reference label was coded without a directive. A directive could not be located in the first 32 bytes following the internal label.

User Response: An internal macro reference label must be followed by a valid assembler instruction or a PEngiCCL directive. Furthermore, the instruction/directive must be located within the first 32 bytes following the internal reference label. Correct erroneous statement. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-005 12 LABEL :INTERNAL MACRO REFERENCE REQUIRES A DIRECTIVE

Explanation: The directive following the internal macro reference label was too long or invalid as written.

User Response: Code the proper directive or assembler instruction following the internal reference label. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-006 12 -TEXT- :SET DIRECTIVE IS NOT PRECEDED BY A VARIABLE

Explanation: A PEngiCCL set directive was coded, but it was not preceded by a target variable symbol starting in position 1.

User Response: Provide a target variable for the SET directive. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-007 12 LABEL :ILLEGAL INTERNAL MACRO REFERENCE LABEL

Explanation: The internal reference label was illegal as written.

User Response: Code the internal reference label according to PEngiCCL conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-009 12 :ERROR, MACRO STATEMENT IS MISSING

Explanation: The ″MACRO″ statement was missing. A non-comment statement was encountered while searching for the MACRO statement.

User Response: All PEngiCCL macros must contain a MACRO statement before the prototype model statements. Add a ″MACRO″ statement to the macro. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-010 12 -TEXT- :PREPROCESSOR PROGRAM WAS IMPROPERLY INSTALLED

Explanation: PEngiCCL was improperly installed. A program used by the displayed DIRECTIVE was not properly resolved by the link edit program.

User Response: Contact your PEngiCCL software administrator.

PREGEN-011 12 -TEXT- :ILLEGAL FORM OF EXPRESSION FOR DIRECTIVE

Explanation: The variable coded starting in position 1 of the statement is not supported for this directive. That is, the directive does not support coding conventions of this type.

User Response: Remove the variable or correct the erroneous statement as needed. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-012 04 MACNAME :MEND STATEMENT IS MISSING, ASSUMED PRESENT

Explanation: The ″MEND″ statement is missing. The end of macro input statements was reached but no ″MEND″ statement was located.

User Response: All PEngiCCL macros must contain an MEND statement at the end of macro source. Add an MEND statement as required. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-013 12 :MACRO PROTOTYPE MODEL DEFINITION IS MISSING

Explanation: All statements following the ″MACRO″ statement are either comments or spaces, or no statements exist following the ″MACRO″ statement.

User Response: A PEngiCCL macro requires at least a MACRO, Prototype model and an MEND statement. Change your macro to comply with the PEngiCCL conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-014 12 :V.S.M ERROR OR INSUFFICIENT VIRTUAL STORAGE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

PREGEN-016 12 LABEL :INTERNAL REFERENCE LABEL IS ILLEGAL AS SPECIFIED

Explanation: The internal macro reference label is too short.

User Response: The internal reference label must start with a period (.) and contain 1 to 12 characters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-017 12 LABEL :DUPLICATE INTERNAL REFERENCE LABEL

Explanation: The internal reference label has been previously declared.

User Response: Choose a unique name to avoid duplicates. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-018 12 LABEL :MAXIMUM NUMBER OF REFERENCE LABELS EXCEEDED

Explanation: The number of internal reference labels exceeds the number of allocated slots. This is probably a PEngiCCL logic error.

User Response: Contact PEngiCCL software support center.

PREGEN-019 12 :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-020 12 :PREPROCESSOR PROGRAM 1 IS NOT RESOLVED

Explanation: PEngiCCL was improperly installed. A program used by the displayed DIRECTIVE, in the statement before this message, was not properly resolved by the link edit program.

User Response: Contact your PEngiCCL software administrator.

PREGEN-021 12 :PREPROCESSOR PROGRAM 2 IS NOT RESOLVED

Explanation: PEngiCCL was improperly installed. A program used by the displayed DIRECTIVE, in the statement before this message, was not properly resolved by the link edit program.

User Response: Contact your PEngiCCL software administrator.

PREGEN-022 12 :PREPROCESSOR PROGRAM 3 IS NOT RESOLVED

Explanation: PEngiCCL was improperly installed. A program used by the displayed DIRECTIVE, in the statement before this message, was not properly resolved by the link edit program.

User Response: Contact your PEngiCCL software administrator.

PREGEN-023 12 :PREPROCESSOR PROGRAM 4 IS NOT RESOLVED

Explanation: PEngiCCL was improperly installed. A program used by the displayed DIRECTIVE, in the statement before this message, was not properly resolved by the link edit program.

User Response: Contact your PEngiCCL software administrator.

PREGEN-024 12 VARNAME :ILLEGAL OR UNDEFINED VARIABLE SYMBOL

Explanation: The VARNAME variable is undefined in this macro.

User Response: Supply a correct variable that has been defined in this macro. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-025 12 VARNAME :ILLEGAL USE OF PROTOTYPE OR READONLY VARIABLE

Explanation: A set directive was coded using a read-only or a prototype model variable as the target.

User Response: The read-only and prototype model variables cannot be altered. Correct your expression to use a correct variable. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-026 12 VARNAME :VARIABLE IS INCONSISTENTLY USED WITH ITS DECLARATION

Explanation: A SET directive was attempted using the VARNAME variable as the target; however, the declared variable type was not consistent with the attempted set directive. That is, a SETC was attempted on a SETA or SETB variable type.

User Response: Use the SET directive which is consistent with the declared variable type. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-027 12 VARNAME :ILLEGAL USE OF SUBSCRIPT FOR NON-SUBSCRIPTED VARIABLE

Explanation: A subscript was coded for the VARNAME non-subscripted variable.

User Response: Correct the erroneous expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-028 12 VARNAME :VARIABLE REQUIRES THE USE OF SUBSCRIPT

Explanation: A subscript was not been coded for the VARNAME sublisted variable.

User Response: Code a subscript as required. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-029 12 : MEND WAS PROCESSED, THIS LINE IS ILLEGAL

Explanation: The displayed statement was located in the macro source after the MEND statement.

User Response: Place your MEND as the last entry of macro source. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-030 12 -TEXT- :VARQ= OPTION IS INVALID

Explanation: The NN value in the VARQ=NN was either not numeric, exceeded 10 digits or it exceeded 1024.

User Response: Correct the erroneous value. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

PREGEN-031 12 -TEXT- :MODE= OPTION IS INVALID

Explanation: The MODE= options of MACRO statement are invalid.

User Response: Refer to the PEngiCCL Reference Manual for the allowed Macro Modes.

PREGEN-032 12 -TEXT- :UNBALANCED TERMINATOR, MACRO STMT=NNNNN

Explanation: ENDAIF or ENDADO is missing.

User Response: Each AIFINL and ADOWHL must be paired with the respective terminator.

PREGEN-033 12 -TEXT- :NO TERMINATOR, MACRO STMT=NNNNN

Explanation: ENDAIF or ENDADO is missing

User Response: Each AIFINL and ADOWHL must be paired with the respective terminator.

READ00-001 12 VARNAME :VARIABLE IS NOT DEFINED

Explanation: The displayed Varname is undefined.

User Response: Contact Migration Utility Software Support Center.

READ00-002 12 -TEXT- :UNPAIRED OR IMPROPER USE OF QUOTES

Explanation: A data string with no ending quote was detected on the line displayed before the message.

User Response: Provide the end quote. Also see "Note 3" on page 151.

READ00-003 12 -TEXT- :RIGHT PAREN IS MISSING IN BRACKETED STRING

Explanation: Unpaired brackets were detected in the bracketed expression.

User Response: The displayed -TEXT- shows the beginning of the bracketed expression. Provide additional brackets as needed. Note that the bracketed expressions can span over multiple input lines.

READ00-004 12 -TEXT- :IMPROPER TERMINATION OF BRACKETED STRING

Explanation: Refer to the READ00-003 message.

READ00-005 12 -TEXT- :DATA ELEMENT IS TOO LONG

Explanation: The data string is longer than the maximum allowed by PEngiCCL (usually 256 bytes).

User Response: Reduce the data string in error.

READ00-006 12 -TEXT- :UNPAIRED PAREN OR EXPRESSION IS TOO LONG

Explanation: The data string enclosed in parentheses is too long or parentheses are not paired.

User Response: Reduce/correct the data string in error.

READ00-007 12 VARNAME :VARIABLE BUFFER LESS THAN MINIMUM REQUIRED

Explanation: The &SYSTOKEN system variable is not properly defined.

User Response: Contact Migration Utility software support center.

READ00-008 12 VARNAME :OVER 2046 WORDS IN INPUT. CANNOT TOKENIZE.

Explanation: The input text contains more than 2046 data elements.

User Response: Reduce the number of data elements in the input.

READ00-009 12 VARNAME :INPUT STRING EXCEEDS SYSTOKEN SIZE

Explanation: The data contained in the Variable of ACCL TOKEN directive exceeds 16,000 bytes

User Response: Reduce the variable contents to maximum of 16,000 bytes.

READ00-010 12 MACNAME :FILE IS NOT OPENED FOR READ

Explanation: An attempt was made to use the ACCL READ directive before an ACCL OPEN.

User Response: Code an ACCL OPEN directive before the ACCL READ.

READ00-011 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: Area before continuation column is not spaces.

User Response: This can occur on improperly coded bracketed expressions. i.e, an unpaired bracketed expression on a single line causes the read of the next statement/record that does not belong to the bracketed expression. Also see "Note 3" on page 151.

READ00-012 12 :DATA STRING IS TOO LONG

Explanation: The input bracketed data string exceeds the BUFSIZE= value in the COPTION statement

User Response: Increase BUFSIZE=NNNN value (refer to INSTALLATION Manual). Also see "Note 3" on page 151.

READ00-013 12 :EXPECTED CONTINUATION NOT FOUND

Explanation: The last Easytrieve Plus line was coded with a '+' or a '-' but here is no more input.

User Response: Correct the erroneous statement.

READ00-014 12 VARNAME :UNDECLARED VARIABLE SYMBOL

Explanation: VARNAME variable was located in Easytrieve Plus macro but there is no corresponding declared variable on the MACRO statement.

User Response: Add the VARNAME to the macro statement.

READ00-015 12 MACNAME :DUPLICATE TEMPORARY MACRO NAME

Explanation: The macro name already exists (it was previously processed)

User Response: Make sure that you code unique macro names. Macros coded at the beginning of an Easytrieve Plus program must be unique.

READ00-016 12 :MACRO NAME IS MISSING

Explanation: Easytrieve Plus macro statement "MSTART" was coded without the macro name.

User Response: Provide a valid macro name following the MSTART statement.

READ00-017 12 :LENGTH OF MACRO NAME EXCEEDS 12 CHARACTERS

Explanation: Easytrieve Plus macro name following the "MSTART" is too long.

User Response: Provide a valid 1-12 characters macro name.

READ00-018 12 : - TEXT -

Explanation: Error managing macro queue (most probably short on memory).

User Response: This message was probably preceded by a GETMAIN error. Try to increase REGION size on your JOB statement. Please ignore the text part of this message.

READ00-019 12 MACNAME :TEMPORARY MACRO "MEND" IS MISSING

Explanation: MEND was not located for the temporary macro.

User Response: Code MEND as required.

READ00-020 12 -TEXT- :IMPROPER TERMINATION OF STRING

Explanation: Expected space is not found following the expression.

User Response: Check expression syntax.

REFLAB-001 12 -TEXT- :UNDEFINED INTERNAL REFERENCE LABEL

Explanation: The displayed macro reference label is undefined

User Response: Provide the label inside the macro.

REFLAB-002 12 -TEXT- :INTERNAL REFERENCE LABEL LENGTH ERROR

Explanation: The reference label exceeds 12 characters.

User Response: Reduce the label size.

REPDIR-001 12 :PREPROCESSOR ERROR, STX,TXT,SPC IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPDIR-007 12 :PREPROCESSOR ERROR, ETX/SAE/SQE IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPDIR-008 12 :INTERMEDIATE BUFFER CAPACITY EXCEEDED

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

REPDIR-009 12 :PREPROCESSOR ERROR, CHAIN/NUL ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPDIR-010 12 VARNAME :INCONSISTENT/ILLEGAL SUBSTRING USAGE (N,M)

Explanation: A subscript was coded for an attribute N' expression. The subscripts are allowed in the attribute N' expression, only for the &SYSLIST variable.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

REPDIR-011 12 VARNAME :SUBSCRIPT VALUE EXCEEDS 32,767

Explanation: The computed subscript value exceeds 32,767.

User Response: Make sure that the subscript does not result in a number greater than 32,767. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

REPDIR-012 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript exceeds the declared variable dimension.

User Response: Make sure that the subscript expression does not result in a number greater than the variable dimension. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

REPSQC-001 12 :PREPROCESSOR ERROR, SQE/SQC IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPSQC-002 12 :PREPROCESSOR ERROR, SXE/EXC IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPSQC-003 12 :PREPROCESSOR ERROR, SXE/SXC IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPSQC-004 12 :ILLEGAL SUBSTRING DISPLACEMENT (MUST BE > 0)

Explanation: The computed displacement X of substring expression (X,Y) is less than 1 or negative.

User Response: Make sure that the substring expression results in a substring displacement position greater than zero. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

REPSQC-005 12 :SUBSTRING DISPLACEMENT EXCEEDS THE STRING LENGTH

Explanation: The computed displacement X of substring expression (X,Y) is greater than the string length.

User Response: Make sure that the substring expression results in a substring displacement within the range of the string size. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

REPSQC-007 12 :PREPROCESSOR ERROR, EQE/SQE IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPSQC-008 12 :INTERMEDIATE EXPRESSION IS TOO LONG

Explanation: The work buffer cannot accommodate the expression in the preprocessed format.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

REPSQC-009 12 :PREPROCESSOR ERROR, CHAIN/NUL ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

REPSQC-010 12 VARNAME :INCONSISTENT/ILLEGAL SUBSTRING USAGE (N,M)

Explanation: A subscript has been coded for an attribute N' expression. The subscripts are allowed in the attribute N' expression, only for the &SYSLIST variable.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

REPSQC-011 12 VARNAME :SUBSCRIPT VALUE EXCEEDS 32,767

Explanation: The computed subscript value exceeds 32,767.

User Response: Make sure that the subscript does not result in a number greater than 32,767. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

REPSQC-012 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript exceeds the declared variable dimension.

User Response: Make sure that the subscript expression does not result in a number greater than the variable dimension. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETA00-001 12 :PREPROCESSOR LOGIC ERROR, HCPSAE IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETA00-002 12 :PREPROCESSOR LOGIC ERROR, HCPEAE IS MISSING

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETA00-003 12 :PREPROCESSOR LOGIC ERROR, BAD SAC..EAC CHAIN

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETA00-004 12 :ILLEGAL/INCONSISTENT MATH OPERATIONS

Explanation: Two high order math operations (* OR /) were coded in succession. Or a variable between these operation codes contained a null value.

User Response: Correct the erroneous expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

SETA00-005 12 :INCOMPLETE/INCONSISTENT EXPRESSION

Explanation: The math expression is illegally terminated with a math operation symbol. This can be caused by a null value in the last variable within the expression.

User Response: Correct the erroneous expression. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

SETA00-006 12 :ILLEGAL FORM OF EXPRESSION

Explanation: Two data fields are detected in succession. This would indicate a problem with PEngiCCL preprocessor logic.

User Response: Contact PEngiCCL software support center.

SETA00-007 12 :PREPROCESSOR LOGIC ERROR, CHAIN/NUL ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETA00-009 12 VARNAME :NON-NUMERIC ENTRY USED IN ARITHMETIC

Explanation: The VARNAME variable does not contain numeric data.

User Response: Before you use a SETC symbol in an arithmetic expression, make sure that the data it contains is numeric. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETA00-010 12 VARNAME :PREPROCESSOR LOGIC ERROR, ILLEGAL HEX VALUE

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETA00-011 12 VARNAME :INCONSISTENT/ILLEGAL SUBSTRING USAGE (N,M)

Explanation: A subscript has been coded for an attribute N' expression. The subscripts are allowed in the attribute N' expression only for the &SYSLIST variable.

User Response: Code the expression to comply with PEngiCCL coding conventions. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

SETA00-012 12 VARNAME :PREPROCESSOR ERROR, SUBSCRIPT/CHAIN/NUL ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETA00-013 12 VARNAME :SUBSCRIPT VALUE EXCEEDS 32,767

Explanation: The computed subscript value exceeds 32,767.

User Response: Make sure that the subscript does not result in a number greater than 32,767. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETA00-014 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript exceeds the declared variable dimension.

User Response: Make sure that the subscript expression does not result in a number greater than the variable dimension. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETA00-015 12 :INTERMEDIATE ARITHMETIC VALUE EXCEEDS MAXIMUM

Explanation: A numeric overflow resulted while trying to carry out an arithmetic operation in expression. Note that in the intermediate operations, the high order (* or /) operations internal product or quotient can be up to 15 significant digits, and the low order (+ or -) operations internal sum or difference can be up to 31 digits. The final outcome of each individual expression cannot exceed 2147483647.

User Response: Make sure that the operands in math operations are not used improperly by validating each operand before it is used in an arithmetic operation. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETA00-016 12 :DIVISOR IS ZERO, CANNOT DIVIDE BY ZERO

Explanation: The computed divisor is zero.

User Response: Make sure that the operands in math operations are not used improperly by validating each operand before it is used in an arithmetic operation. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETA00-017 12 :MAXIMUM OF 64 TERMS/EXPRESSIONS EXCEEDED

Explanation: The maximum number of bracketed expressions that can be supported by PEngiCCL has been exceeded.

User Response: Your are limited to the maximum of 64 bracketed expressions in a single conditional request. Limit the number of bracketed expressions to the maximum of 64. If the error occurred on a Migration Utility macro, contact Migration Utility software support center.

SETA00-018 12 VARNAME :VARIABLE USED IN ARITHMETIC HAS A NULL STRING

Explanation: The VARNAME used in arithmetic expression contains no data.

User Response: Make sure that the operands in math operations are not used improperly by validating each operand before it is used in an arithmetic expression. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETA01-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETA01-002 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript exceeds the declared variable dimension.

User Response: Make sure that the subscript expression does not result in a number greater than the variable dimension. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETB00-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETB00-002 12 VARNAME :SUBSCRIPT EXCEEDS VARIABLE DIMENSION

Explanation: The computed subscript exceeds the declared variable dimension.

User Response: Make sure that the subscript expression does not result in a number greater than the variable dimension. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETB00-003 -TEXT- :VALUE IS NOT A BOOLEAN 0 OR 1

Explanation: The SETB value is not a valid Boolean.

User Response: Make sure that you use a valid Boolean in SETB directive.

SETCPY-001 12 :COPY DIRECTIVE BUT NO MEMBER SPECIFIED

Explanation: An FSCOPY directive was coded without a copy member name.

User Response: Specify the member name to be copied following the FSCOPY directive.

SETCPY-002 12 COPYNAME :IMPROPER SPECIFICATION OF MEMBER NAME

Explanation: An FSCOPY directive was coded without a copy member name.

User Response: Specify the member name to be copied following the FSCOPY directive.

SETCPY-003 12 COPYNAME :COPY MEMBER NAME IS TOO LONG

Explanation: The FSCOPY member name exceeds 12 characters.

User Response: Code the correct FSCOPY member name. Note: Because of the PDS and CMS member naming conventions, the copy name can be up to 8 characters long on MVS/XA and VM/CMS operating systems.

SETCPY-005 12 -TEXT- :ILLEGAL VALUE IN CC 7

Explanation: The character in position seven is not a "*", "/", "-" or a space.

User Response: Remove the invalid character.

SETCPY-006 12 -TEXT- :EXTRANEOUS DATA IN COPY STATEMENT

Explanation: The FSCOPY statement contains extraneous data following the copy member name.

User Response: Remove the unnecessary extraneous data from the statement.

SETCPY-007 12 -TEXT- :SPECIAL COPY IS ILLEGAL INSIDE MACRO DEFINITION

Explanation: The special "FSCOPY member (ASM)" FSCOPY expression was coded inside a macro.

User Response: The special FSCOPY for CICS macro support is not allowed inside macros. Remove it.

SETC00-001 12 MACNAME :PREPROCESSOR PROGRAM LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETC00-002 12 VARNAME :SUBSCRIPT EXCEEDS DECLARED VARIABLE DIMENSION

Explanation: The computed subscript exceeds the declared variable dimension.

User Response: Make sure that the subscript expression does not result in a number greater than the variable dimension. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETC00-003 12 VARNAME :DATA STRING EXCEEDS MAXIMUM VARIABLE SIZE

Explanation: The data string is longer than the variable size.

User Response: Limit the data string to the variable size. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETW00-001 12 MACNAME :NO DATA IN INPUT

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW00-002 12 MACNAME :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is

generated at PEngiCCL installation time. Increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters. See "Note 3" on page 151.

SETW00-003 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation, therefore, continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72. Also see "Note 3" on page 151.

SETW00-004 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error. Also see "Note 3" on page 151.

SETW01-001 12 :REQUIRED STRING NOT IN QUOTES

Explanation: The data string in the statement displayed before this message does not start with a quote.

User Response: Enclose the data string in quotes as required.

SETW01-003 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message. Also see "Note 3" on page 151.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW01-004 12 -TEXT- :UNPAIRED QUOTES IN QUOTED STRING

Explanation: An uneven number of quotes has been detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote. Also see "Note 3" on page 151.

SETW01-005 12 :UNPAIRED QUOTES, NO CONTINUATION

Explanation: The end of data string which starts with a quote was reached before the end quote could be located.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote. Also see "Note 3" on page 151.

SETW01-006 12 :UNPAIRED QUOTES IN QUOTED STRING

Explanation: An uneven number of quotes has been detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW01-007 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message. Also see "Note 3" on page 151.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW01-008 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message. Also see "Note 3" on page 151.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW01-009 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation, therefore, continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72. Also see "Note 3" on page 151.

SETW01-010 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW02-001 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW02-002 12 -TEXT- :END QUOTE MISSING

Explanation: The end of data string which starts with a quote was reached before the end quote could be located.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW02-003 12 :UNPAIRED QUOTES, BUT NO CONTINUATION

Explanation: The end of data string which starts with a quote was reached before the end quote could be located.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW02-004 12 :UNPAIRED QUOTES IN INPUT STRING

Explanation: An uneven number of quotes has been detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW02-005 12 :INPUT STRING IS TOO LONG
SETW02-006 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW02-007 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW02-008 12 -TEXT- :NON BLANK DATA BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW02-009 12 :DIRECTIVE EXPRESSION IS MISSING

Explanation: A directive was coded without the necessary expression.

User Response: Code the necessary expression as required by the directive.

SETW02-011 12 :UNPAIRED PARENS IN INPUT STRING

Explanation: The number of left parentheses is not equal to the number of right parentheses in the expression, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, see “Note 2” on page 151.

SETW02-012 12 -TEXT- :EXPRESSION MUST BE ENCLOSED IN PARENS

Explanation: The data string following the directive is not enclosed in parentheses.

User Response: Enclose expression in parentheses as required.

SETW02-013 12 -TEXT- :ILLEGAL TERMINATION OF SETB EXPRESSION

Explanation: The SETB expression is illegally terminated. A non-blank was coded following the last bracket of expression.

User Response: Remove the extraneous characters.

SETW02-014 12 -TEXT- :UNPAIRED PARENS OR ILLEGAL TERMINATION OF STRING

Explanation: The number of left parentheses is not equal to the number of right parentheses in the expression, which are not a part of a quoted string. The -TEXT- is the tail-end of the last data examined.

User Response: Make sure that you have an even number of left and right parentheses. If the error occurred on a Migration Utility macro, see "Note 2" on page 151.

SETW02-015 12 -TEXT- :EXPRESSION EXCEEDS MAXIMUM ALLOWABLE SIZE

Explanation: The target macro reference label expression exceeds 256 characters.

User Response: Limit your expression to maximum of 256 characters.

SETW02-017 12 -TEXT- :ILLEGAL TARGET REFERENCE LABEL

Explanation: The target macro reference label expression is too short.

User Response: Limit your expression to maximum of 2 characters.

SETW02-018 12 -TEXT- :ILLEGAL FORM OF EXPRESSION FOR SETB DIRECTIVE

Explanation: The expression following the SETB directive is not enclosed in parentheses, or it is not a valid Boolean variable (which is either 0 or 1).

User Response: The SETB directive requires that an explicit non-boolean expression be enclosed in parentheses. Correct it as required.

SETW03-001 12 :FS-PEngiCCL LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW03-002 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW03-003 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW03-004 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW04-001 12 -TEXT- :FS-PEngiCCL LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW04-002 12 -TEXT- :ILLEGAL OPCODE OR ILLEGAL EXPRESSION

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW04-004 12 -TEXT- :REQUIRED DATA IS NOT LOCATED

Explanation: The macro has been illegally placed near position 72.

User Response: Code the macro instruction within the first 32 positions following the macro label if any.

SETW04-005 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW04-006 12 :UNPAIRED QUOTES IN EXPRESSION

Explanation: An uneven number of quotes was detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW04-007 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW04-008 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW04-009 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW04-010 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW05-001 12 -TEXT- :PREPROCESSOR LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW05-005 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message. Also see “Note 3” on page 151.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW05-006 12 :UNPAIRED QUOTES IN EXPRESSION

Explanation: an uneven number of quotes was detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote. Also see “Note 3” on page 151.

SETW05-007 12 -TEXT- :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72. Also see “Note 3” on page 151.

SETW05-008 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error. Also see “Note 3” on page 151.

SETW05-009 12 -TEXT- :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72. Also see “Note 3” on page 151.

SETW05-010 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error. Also see “Note 3” on page 151.

SETW05-012 12 -TEXT- :ILLEGAL VALUE IN CC 7

Explanation: The character in position seven is not a "*", "/", "-" or a space.

User Response: Remove the invalid character.

SETW05-013 12 :EXPECTING CONTINUATION IN CC 72

Explanation: A comma was detected following the last data string in the macro prototype model, but the continuation position 72 does not contain a non-blank character. This error occurs only when IBM macro conventions are being used. That is, for macros that do not start with a macro delimiter.

User Response: Code a non-blank in position 72 to indicate continuation.

SETW06-001 12 :SYNTAX ERROR, MNOTE REQUIRES CONDITION CODE

Explanation: An MNOTE directive was coded without the message.

User Response: Code a condition code and a message enclosed in quotes following the directive.

SETW06-003 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW06-004 12 -TEXT- :REQUIRED STRING IS NOT IN QUOTES

Explanation: The data string in the statement displayed before this message does not start with a quote.

User Response: Enclose the data string in quotes as required.

SETW06-005 12 :UNPAIRED QUOTES, NO CONTINUATION

Explanation: The end of data string which starts with a quote was reached before the end quote could be located.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW06-006 12 :UNPAIRED QUOTES IN QUOTED STRING

Explanation: An uneven number of quotes was detected in a data string which started with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW06-007 12 :INPUT STRING IS TOO LONG SETW06-008 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW06-009 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW06-010 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW07-001 12 MACNAME :NO DATA IN INPUT

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW07-002 12 MACNAME :EXCEEDS BUFFER CAPACITY

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW07-003 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW07-004 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW08-001 12 :NO DATA IN INPUT

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW08-002 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW08-003 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW08-004 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW09-001 12 -TEXT- :IMPROPER VARIABLE SYMBOL IN PROTOTYPE NAME

Explanation: The macro label variable symbol did not start with a “&” or it is too long.

User Response: Correct the variable symbol as required.

SETW09-002 12 -TEXT- :FS-PEngiCCL LOGIC ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

SETW09-004 12 -TEXT- :MACRO NAME IS TOO LONG

Explanation: The macro name exceeds 12 characters.

User Response: Code the correct macro name. Note that the macro name can be up to 8 characters long on MVS/XA and VM/CMS operating systems, because of the PDS and CMS member naming conventions. However, temporary macro names can be up to 12 characters long.

SETW09-005 12 :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can

increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW09-006 12 :UNPAIRED QUOTES IN EXPRESSION

Explanation: An uneven number of quotes was detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW09-007 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW09-008 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW09-009 12 :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements in input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW09-010 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW09-011 12 :UNEXPECTED CONTINUATION IN CC72

Explanation: A blank was detected following the last data string in the macro prototype model, but the continuation position 72 contained a non-blank character. This error can be caused by an erroneous character in position 72 due to overlapping comments.

User Response: The macro prototype model coding standards require that a comma (,) is placed after the last data string on each line whenever continuation lines follow. The comma must not be placed on the last line. Correct the statement as required.

SETW10-004 12 -TEXT- :UNKNOWN ACCL FUNCTION

Explanation: The supported function is not supported by ACCL directive.

User Response: None. Use the correct function.

SETW10-005 12 -TEXT- :INPUT STRING IS TOO LONG

Explanation: The work buffer cannot accommodate the input data string. The string in error is displayed before the message.

User Response: The default work buffer size is generated at PEngiCCL installation time. You can increase the buffer size by coding the BUFSIZE=NNN in the PEngiCCL COPTION parameters.

SETW10-006 12 -TEXT- :UNPAIRED QUOTES IN EXPRESSION

Explanation: An uneven number of quotes was detected in a data string which starts with a quote.

User Response: A data string which is enclosed in quotes must contain an even number of quotes. Add the necessary quote.

SETW10-007 12 -TEXT- :INCOMPLETE CONTINUATION LINE

Explanation: A non-blank character was found in position 72 of the last statement, but there were no more statements to input. A non-blank character in position 72 denotes continuation and therefore continuation statements are expected.

User Response: Add the necessary continuation statements or remove the non-blank character from position 72.

SETW10-008 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: A non-blank was coded in position 72 of the previous statement, implying continuation, but data was located before continuation column 16 of this statement.

User Response: All continuation lines inside PEngiCCL macros must start in position 16. Correct the statement in error.

SETW10-009 12 -TEXT- :INCOMPLETE CONTINUATION LINE

Explanation: The same as the message SETW10-007.

SETW10-010 12 -TEXT- :NON-BLANK FOUND BEFORE CONTINUATION COLUMN

Explanation: The same as the message SETW10-008.

SRTMOD-001 12 MACNAME :FSBUSORT ERROR SORTING REFERENCE LABELS

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSMF00-001 12 INPUT STRING IS TOO LONG

Explanation: Input string exceeds 2048 characters. This can be caused by unpaired quotes or parentheses.

User Response: Correct the problem.

VSMF00-002 12 UNPAIRED PARENS/QUOTES (CHECK CC 72)

Explanation: Unpaired parens or quotes in the user parameters.

User Response: Make sure that you have paired parentheses or quotes.

VSMF00-003 12 TOO MANY FUNCTION PARAMETERS

Explanation: Inline function number of user parameters exceeds that allowed by the function model.

User Response: Code the correct number of parameters.

VSMF00-004 12 KEYWORD PARAMETERS ARE NOT ALLOWED

Explanation: A keyword parameter was coded in the function parameters.

User Response: Keywords are not allowed in the function parameter list. Remove it.

VSMF00-005 12 EXTRANEOUS DATA AFTER BRACKETED STRING

Explanation: The character past the last paired parenthesis is not a right parenthesis, a space, a comma or a period.

User Response: Remove extraneous character.

VSMF00-006 12 EXTRANEOUS DATA AFTER QUOTED STRING

Explanation: The character post last paired quote is not a right parenthesis, a left parenthesis, a space or a comma.

User Response: Remove extraneous character.

VSMF00-007 12 END QUOTE IS MISSING IN QUOTED STRING

Explanation: COBOL continuation line was detected but the continued line does not start with a quote.

User Response: Place end quote where it belongs.

VSMF00-008 12 RESULTING FUNCTION NAME IS TOO LONG

Explanation: Derived function paragraph name is more than 30 characters long.

User Response: For each function, Migration Utility generates a COBOL paragraph name under which it expands COBOL logic associated with the function. Subsequently, the generated paragraph is performed whenever such function is encountered in the COBOL source.

Local function paragraph name is composed of the function name, and a 6 digit sequence number. For example `SEL_READ-FILE (..):` function paragraph name would be `F00000-READ-FILE:`. Thus the function name can be maximum 23 characters.

CON_OBJECT (&OBJECT &OPTION) and SEL_OBJECT (&OBJECT &OPTION) function paragraph name is composed of the function type, the object name, and the option, preceded by the sequence number.

For example `SEL_OBJECT (FILEIN1 READ):` function paragraph name would be `F00000-SEL-FILEIN1-READ:`. Thus the object name plus the option can be maximum of 19 characters (including the dash).

VSM000-001 00 MEMBER :PROGRAM ERROR, FSVSMADD FOR EXISTING MEMBER

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSM000-002 00 MEMBER :INSUFFICIENT VIRTUAL STORAGE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

VSM000-003 00 MEMBER :PROGRAM ERROR, FSVSMDSA FOR NON EXISTING MEMBER

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSM000-004 00 MEMBER :INSUFFICIENT VIRTUAL STORAGE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system increase the virtual storage of your CMS machine.

VSM000-005 00 MEMBER :MEMBER NOT LOCATED, FSVSMLOC CALL

Explanation: A request to locate a member in the virtual storage manager pool resulted in "Not Found" condition. This message is for the internal PEngiCCL use only.

User Response: NONE

VSM000-006 00 MEMBER :PROGRAM ERROR, NO C/B CHAIN POINTERS

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSM000-007 00 MEMBER :INSUFFICIENT VIRTUAL STORAGE

Explanation: PEngiCCL preprocessor ran out of virtual storage.

User Response: On MVS/XA system increase the REGION size on the EXEC statement, on VM/CMS system

---

VSM000-008 00 MEMBER :PROGRAM ERROR, NO C/B CHAIN POINTERS

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSM000-009 00 MEMBER :DATA EXCEEDS V.S.M. BUFFER CAPACITY

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSM001-001 00 MEMBER :MEMBER NOT LOCATED, FSVSMLOC CALL

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSM001-002 00 MEMBER :PROGRAM ERROR, NO C/B CHAIN POINTERS

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

VSM001-003 00 MEMBER :PROGRAM ERROR, FREEMAIN/FREEVIS ERROR

Explanation: An internal PEngiCCL Macro Preprocessor logic error.

User Response: Contact PEngiCCL software support center.

Runtime I/O error messages

FSY001E *message text*

Explanation: This message is issued by Migration Utility when:

- A serious error is detected while loading Migration Utility macros byte code.

  In general, this indicates a corrupt SYS1.SFSYFJCC library pointed to by the FJCCLLB DDname. Make sure that your library has not been corrupted and that you are pointing to the correct byte code PDS.
- A serious error is detected in sub-modules while simulating logical operations during application run time. This is a user error.

User Response: The *message text* printed is self-explanatory.

FSY001I *message text*

Explanation: This is an informational message, usually printed after an E-level message is encountered.

FSY002E **&MODNAME:** *message text*

Explanation: This message is issued by Migration Utility whenever a serious error is detected in the dynamic I/O modules during the application run time.

User Response: The *message text* printed is self-explanatory.

Many unrecoverable I/O error conditions are intercepted by IBM standard I/O error routines. Always check console for messages not included in this manual.

VSAM I/O error supplemental RPL information

When running in dynamic mode, Migration Utility runtime VSAM I/O modules return COBOL-compliant STATUS-CODE to the application program, along with the RPL information as saved at the time of error.

When an application program abnormally terminates via the FSABECOB program, the RPL information is displayed on SYSOUT and FJSYABE files as a supplement to the STATUS-CODE error as follows:

```
FSDYNKSO: VSAM ERROR: &DNAME,&FUNCTION,RPLRTNCD=NN,RPLERRCD=NN,RPLCMPON=NN,
RPLFUNCD=NN
```

The displayed codes RPLRTNCD, RPLERRCD, and RPLCMPON are the values found in the RPL. These meaning of these codes can be found in the *VTAM/VSAM Messages and Codes* manual.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie New York 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR

PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

CICS	MVS/XA
DB2	OS/390
IBM	S/390
MVS	SQL/DS
MVS/ESA	z/OS

Other company, product, and service names may be trademarks or service marks of others.

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Y

Readers' Comments — We'd Like to Hear from You

IBM Migration Utility for z/OS and OS/390
User's Guide and Reference
Release 1

Publication No. SC27-1685-04

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

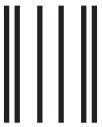
Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SC27-1685-04

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
H150/090
555 Bailey Avenue
San Jose, CA 95141-9989

Fold and Tape | **Please do not staple** | Fold and Tape

SC27-1685-04

Cut or Fold
Along Line

Program Number: 5655-I18

Printed in U.S.A.

SC27-1685-04