IBM Migration Utility for z/OS

Installation, User's Guide and Reference

Version 4 Release 1

SC27-4553-00

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 469.

First Edition (September 2012)

This edition applies to IBM Migration Utility for z/S, Version 4 Release 1, Program Number 5655-MGU 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.

IBM welcomes your comments. For information on how to send comments, see “How to send your comments to IBM” on page x.

This publication is available on the Web at:
http://www.ibm.com/software/awdtools/ispfproductivitytool/

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

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 wants to convert them to COBOL programs, or who wants to write new programs using the Easytrieve Plus syntax. It can be used as a reference for writing programs using Migration Utility syntax by those users who do not have Easytrieve Plus.

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, “Using Migration Utility,” on page 5 gives an overview of running Migration Utility with both existing and new Easytrieve Plus programs, and describes how to use the one-step translating driver.

Chapter 3, “Conversion guidelines,” on page 13 describes some techniques and utilities you can use to test your converted programs. It also points out a few features of Easytrieve that you need to check carefully to ensure a smooth conversion.

Chapter 4, “Defining entities,” on page 51 tells you how to define entities such as files, tables, records, and working storage.

Chapter 5, “Program instruction reference,” on page 67 describes in detail each Easytrieve instruction that Migration Utility supports.

Chapter 6, “SQL/DB2 support,” on page 125 and Chapter 7, “SQL File I/O statement reference,” on page 137 describe SQL matters.

Chapter 8, “DLI/IMS support,” on page 143 describes the DLI/IMS™ interface, which is provided through the DLI, FILE, and RETRIEVE statements.

Chapter 9, “Creating HTML and spreadsheet files,” on page 155 describes how to create reports and files that are easily accessed by standard HTML browsers and spreadsheet software.

Chapter 10, “User exits,” on page 189 tells you how to use and write user exits. A user exit is a PEngiCCL macro that is invoked by the translator when a job ends, to extract information collected by the translator.

Chapter 11, "Installing Migration Utility," on page 203 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 13, "Dynamic I/O mode and PDS/PDSE support," on page 245 describes the options that are used in translation, and the effect that different option values will have.

Chapter 13, "Dynamic I/O mode and PDS/PDSE support," on page 245 describes Dynamic I/O mode and support for PDS and PDSE libraries.

Chapter 14, "Toolkit replacement macros," on page 251 describes Toolkit and date-handling replacement macros, and enhanced date threshold handling.

Chapter 15, "Easytrieve Classic translator," on page 287 describes the process of translating Easytrieve Classic programs to Easytrieve Plus or COBOL.

Chapter 16, "Debugging Migration Utility programs," on page 299 describes how to go about debugging Migration Utility programs.

Chapter 18, "Messages," on page 333 lists all the messages that Migration Utility provides through all the steps of creating COBOL programs from Easytrieve programs.

The appendix "Migration Utility JCL," on page 445 lists the supplied JCL for running Migration Utility and the FSYMIG00 control file.

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.

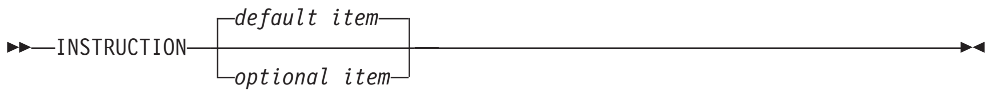
  The ►── symbol indicates that a statement is continued from the previous line.

  The ──►◄ indicates the end of a statement.
- **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).

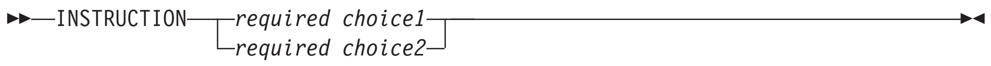
```
►►──INSTRUCTION──required item──────────────────────────────────────────►◄
```

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

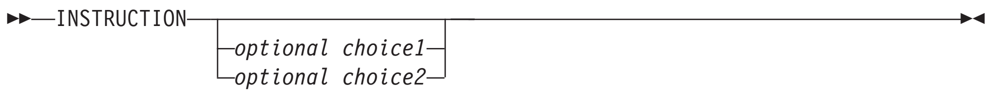


- When you can choose from two or more items, they appear vertically in a stack.

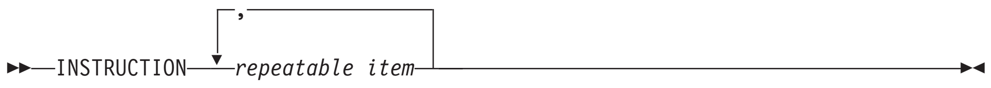
  If you **must** choose one of the items, one item of the stack appears on the main path.



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

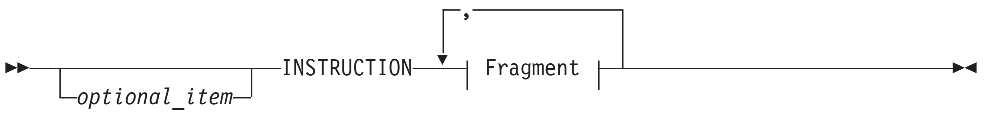


- An arrow returning to the left above the main line indicates an item that can be repeated. When the repeat arrow contains a separator character, such as a comma, you must separate items with the separator character.

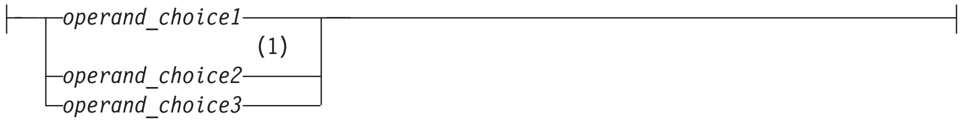


  A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

All of these elements can be combined together into one diagram. For example:



Fragment:



Notes:

1 *operand_choice2* and *operand_choice3* must not be specified together.

optional_item
: Is an optional item, and when you code the command, you may code the item or not.

INSTRUCTION
: This key word must be specified and coded as shown.

Fragment
: This item is a required operand. You can see the choices for the operand in the fragment of the syntax diagram shown below `Fragment:` at the bottom of the diagram. The return loop in the main diagram shows that 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.

For example, here are some acceptable commands:

INSTRUCTION *operand_choice1*
optional_item INSTRUCTION *operand_choice_1*, *operand_choice_2*

And some not acceptable commands:

optional_item operand_choice_2
  - It misses out INSTRUCTION
INSTRUCTION
  - It doesn't supply an operand after the key word
INSTRUCTION *operand_choice2*, *operand_choice3*
  - It breaks the restriction mentioned in the note

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of these methods to send us your comments:
- Use the form on the Web at www.ibm.com/software/ad/rcf/.
- Send an email to comments@us.ibm.com, or mail your comments to
  IBM Corporation
  H150/090
  555 Bailey Avenue
  San Jose, CA
  95141-1003
  U.S.A.

  Include this information:
  – Your name and address
  – Your email address
  – Your telephone or fax number
  – The publication title and order number:
    Migration Utility Installation, User's Guide and Reference
    SC27-4553-00
  – The topic and page number related to your comment
  – The text of your comment.

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.

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

If you have a technical problem

Do not use the comment feedback methods. Instead, do one of these:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM support portal at http://www.ibm.com/systems/z/support/

Summary of changes

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

First edition (September 2012)

Differences between IBM Migration Utility V3.2 and IBM Migration Utility V4.1

- The Report Modernization Utility (RMU) system is distributed with IMU V4.1 at no additional cost. For additional information, refer to the *Report Modernization Utility User's Guide and Reference* (publication number SC19-2726-00).
- Performance enhancements:
  - IMU translator has been changed to manage internal arrays in dynamic mode. For example, the number of supported fields, files, and reports is no longer fixed. The queues are dynamically expanded as needed. Therefore, IMU Version 4.1 uses much less memory to translate programs than previous versions that depended on a maximum number of fixed slots.
  - The default COBOL Compiler PROCESS values in the EZPARAMS table have been changed to `FASTSRT,NUMPROC(NOPFD),TRUNC(BIN),OPTIMIZE(FULL)`. Be aware of the COBOL compiler options that make a performance difference. For example, FASTSRT, NUMPROC(PFD),TRUNC(OPT), and OPTIMIZE(FULL) COBOL compiler options generate more efficient code. Global PROCESS options for IMU compiled COBOL can be placed in the EZPARAMS table.
- Support for 64 Control Breaks
- Support for the IMU Installation Verification feature.
- Support for additional Pan Audit macros.
- Support for Block Size on file definition in Easytrieve Plus programs.
- Option to write report lines with expanded control characters.
- Support for the IDMS IDD statement.
- Support for the Long Block Interface (LBI).
- Support for the Easytrieve Plus DEFER file option.
- Support for Easytrieve Plus version 11.00.
- Increased macro variables buffer capacity from 32k to 64k.
- Enhanced Parallel Testing Utility.
- Discovery utility for linked Easytrieve Plus programs.
- Support for LOW-VALUES in IF statement.
- Enhanced hex literal (constants) handling.
- Support for creating PDSE members.
- Support for "END" statement in Easytrieve Plus program source and logic for handling CARD file type.
- Support for FSIGN=YES/ALL in the SEARCH statement.
- Support for SQL/DB2 table alias name usage.
- Support for ALPHABETIC, ALPHABETIC-LOWER and ALPHABETIC-UPPER test (these are IMU only options).

- Support for the PROCESS SQL COBOL compiler option.
- Warning message when alphanumeric arguments in an IF statement are not of equal length.
- Support for SQL SET and SQL VALUES in dynamic SQL mode.
- Some miscellaneous syntax issues have been addressed.

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 overall 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, DLI/IMS, IDMS, tape files, and unit record devices. It also supports the Easytrieve Macro Language and COPY directive. In most instances no changes will be required to your existing Easytrieve programs.

Easytrieve Plus programs can be converted to COBOL/390 and z/OS Enterprise COBOL, and wth minor changes the generated COBOL source can be ported to non-z/OS platforms.

Translating concepts

Migration Utility translator reads in programs modeled (written) in Easytrieve Plus format and converts them to standard IBM 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.

```
             ┌─────────────┐            ┌─────────────┐            ┌───────────┐
Easytrieve   │             │ PEngiBAT   │             │ COBOL      │           │
Source       │ PEngiEZT    │ Source     │ PEngiBAT    │ Source     │ COBOL     │
   ─────────→│ Translator  │───────────→│ Translator  │──────────→│ Compiler  │
             │             │            │             │            │           │
             └─────────────┘            └─────────────┘            └───────────┘
```

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

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 128).

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 (Typically 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
```

Chapter 2. Using Migration Utility

Migration Utility can be used in two ways:

1. Compile and run your existing Easytrieve Plus programs.

   In this case, you must do a one-time conversion to make sure that programs run correctly. Refer to Chapter 3, "Conversion guidelines," on page 13 for more details on how to do conversions.
2. Compile and run your new programs written in Easytrieve Plus syntax.

In either case, the JCL and PROCs located in the SYS1.SFSYJCLS library, as described in this chapter, will get you started.

Migration Utility executes a few job steps when translating Easytrieve Plus programs to COBOL.

The current release of Migration Utility can be run in two ways:

1. Using the One-Step driver program FSYTPA00.

   The One-Step driver program translates programs in a single job step, and is quite compatible with Easytrieve Plus. The differences amount to changing EZTPA00 to FSYTPA00 on the EXEC statement, and providing access to Migration Utility instead of Easytrieve Plus load libraries.

   This method is the preferred method, and must be used for the Parallel Testing utilities.
2. Using JOBs and PROCs compatible with previous releases, as described in "Using JCL with multiple steps" on page 10.

   This mode of operation should be used to preserve compatibility with previous releases.

   Note: This method cannot be used with the Parallel Testing conversion utilities (see "The Automated Parallel Testing utility" on page 22).

   This mode of operation sharply differs from how the Easytrieve Plus EZTPA00 compiler works, thus substantial changes are required to your existing JCL.

Using the one-step translating driver

Program FSYTPA00 replaces the EZTPA00 Easytrieve Plus compiler exec.

FSYTPA00 reads the #EZTPROC proc from the installed libraries. This proc will have been customized by your system programmer or installer. It contains all JCL and steps necessary for FSYTPA00 to translate programs in a single step. This proc is not read by the operating system, but rather, it is read by FSYTPA00 as a regular text file and interpreted as needed.

Note: #EZTPROC is invisible to the users. Do not make your own copy and modify it unless absolutely needed.

The default proc is #EZTPROC, and is located in SYS.SFSYJCLS. This proc contains job steps and JCL that collectively describe the resources needed for the translation process. Although this proc is invisible to you, you should be aware of multiple job steps in the proc. The following job steps are in the proc:

FSCCL1
Translates Easytrieve Plus programs to PEngiBAT

FSCCL2
Translates PEngiBAT to COBOL

SQLTRAN
Runs generated COBOL through the DB2/SQL translator

COBOL
Compiles the generated COBOL program

LKED Links the compiled module

LKGO
Runs Link and Go programs

The sample JCLs supplied in SYS1.SFSYJCLS provide the basic statements needed to translate and run Migration Utility.

There are some files that you may want to redirect to your own output, or provide additional input. See “Overriding ddnames in your JCL” on page 8 for files that you can override.

Using Migration Utility with your existing Easytrieve Plus jobs

To use the One-Step translator driver with your existing Easytrieve jobs, make the following changes. This procedure applies to both Link and Go jobs and Compile and Link jobs.

1. Add this statement to your JCL:
   ```
   // JOBLIB DD DSN=SYS1.SFSYLOAD,DISP=SHR
   ```
2. Change the EZTPA00 program name on the EXEC statement (on the PARM= for DLI) to FSYTPA00.
3. Remove any unnecessary Easytrieve Plus libraries, such as Toolkit macros and load libraries.
4. Remove EZTVFM file from JCL. It is not used by Migration Utility.
5. Code REGION=0M on the JOB or the EXEC statement to allocate enough memory.
6. If you are running a DB2 program, make sure that DNSLOAD and DNSEXIT load libraries are concatenated to the JOBLIB/STEPLIB.
7. Submit your job.

Like Easytrieve Plus, the one-step driver program runs all steps as a single job step. Temporary files, virtual files, and work files are dynamically allocated at run time. There is no need to add these files to your JCL unless you need more space than the default amount specified by the WRKSPACE= parameter. For more information refer to the DYNALLOC= and WRKSPACE= parameters in Chapter 12, “Migration Utility translation options,” on page 217.

If "PARM LINK (&PROGRAM R)" is coded in your program, the translator will create a load module for future use, otherwise it will run (Link and Go). Note that this is how Easytrieve Plus works.

Using Migration Utility for new programs

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

To translate and run new programs, use one of the sample JCLs or PROCs provided in SYS1.SFSYJCLS (and listed in “Migration Utility JCL,” on page 445.) Each job has a documentation section to help you customize it for your needs. Here is the list and a brief description of each supplied JCL:

JCMUCLGJ
One-step translate, link and go JCL (no proc).

JCMUCLGP
One-step translate, link and go (using proc).

JCMUCL1J
One-step translate and link (no proc).

JCMUCL1P
One-step translate and link (using proc).

JCMUCL2J
Two-step translate and link (no proc).

JCMUCL2P
Two-step translate and link (using proc).

JCMUCNV1
Automated conversion initiation job.

JCMUIMSJ
Sample job for translating IMS/DLI programs.

JCMUMIG1
Automated conversion engine with self restart.

JCMUMIG2
Manual conversion engine with no restart.

JCMUSQLJ
Two-step translate, link and bind for SQL.

JCMUSQLP
Two-step translate, link and bind for SQL(using proc).

| **JCMUDYNP**
| Translate, compile, link and execute a DB2 program in Dynamic mode.
| (Link and Go in Dynamic SQL mode with instream source.) Replace SYSIN
| at the bottom by your own Easytrieve Plus program.

| **JCRSQL0P**
| CAF (STATIC) mode SQL install verification JCL to run the TESTCOL0 test
| program.

| **JCRSQL1P**
| Dynamic SQL install verification JCL to run the TESTCOL1 test program.

| **JCRSQL2P**
| TSO Loader STATIC mode install verification JCL to run the TESTCOL2
| test program.

Note that you can use these JOBS for existing programs as well.

Controlling Translator listings and messages

The translator prints a condensed listing of your Easytrieve Plus program, by default, to SYSLIST1. Errors are printed to the FJSYSER1 file.

COBOL or SQL/DB2 translator listings are printed only when the return code is greater than 4.

The Link Map is always printed.

A COBOL listing is automatically printed for abnormally terminated Link and Go jobs.

You can provide overrides in your program via the EASYTRAN DEBUG statement. The DEBUG must be coded on a single line. Continuation is not supported. For more details of the available options, refer to DEBUG= parameter.

Example:

```
PARM LINK (PROGNAM1 R)
* EASYTRAN: DEBUG(LIST BLIST COBOL)
* END-EASYTRAN
- program statements -
```

Depending on the supplied DEBUG= options, the following possible listings are produced:

SYSLIST1
FSCCL1 step listing of Easytrieve Plus program

FJSYSER1
FSCCL1 step error messages

SYSLIST2
FSCCL2 step listing of PEngiBAT statements

FJSYSER2
FSCCL2 error messages

SYSTLIST
SQL/DB2 translator Listing (DB2 only)

SYSTERM
SQL/DB2 translator messages

SYSTSPRT
SQL/DB2 bind step listing (DB2 only)

COBLIST
COBOL compiler listing

LKEDMAP
Link map

Overriding ddnames in your JCL

You can provide the following overrides in your JCL:

COBLIB
Optionally, provide one or more additional libraries for COBOL copy books. This library is normally not needed unless you are using COBOL copy books for field definitions.

DBRMLIB
Needed for DB2 programs. This file is required when translating DB2 programs. This is an output file passed to the DB2 bind step (see JCMUSQLJ and JCMUSQLP jobs). LRECL=80,RECFM=FB,DSORG=PO.

FJBIND0
Translator-generated SQL BIND file. This file is required when translating DB2 programs. LRECL=80,RECFM=FB, DSORG=PS. The default is a temporary file.

FJERLOG
Optional error log file. The FSYTPA00 program logs the failed translating step to this file. LRECL=80,RECFM=FB, DSORG=PS. This file is primarily used to track errors when using the Automated Conversion engine (JCMUMIG1) proc. Coding DISP=MOD for this file, results in a cumulative error log of all translated programs.

FJPROC0
Optionally, allows override of the default PROC. Use it to point to your own #EZTPROC, should you have to create one. For example:

```
//FJPROC0 DD  DSN=USER1.CONV.JCL(#EZTPROC),DISP=SHR
```

FJSYSPH
Optionally, the generated COBOL can be saved into this file. LRECL=80, RECFM=FB. The default is a temporary file.

FJSYSIN
Optionally, EZPARAMS library and the member name. For example:

```
//FJSYSIN DD DSN=SYS1.SFSYEZTS(EZPARAMS),DISP=SHR
```

The default is in DSN=SYS1.SFSYEZTYS

PANDD
Optional additional libraries where your Easytrieve Plus macros are located. Check with your installation standards. This ddname can be changed to a different name. If so, you must use the ddname assigned at your installation.

SYSLIB
Optional additional load libraries for your called submodules used by the link step.

SYSLMOD
Your program is linked into this library. This file is required in the JCL for programs that contain `PARM LINK (&PROGRAM R)` in the Easytrieve Plus source.

| A system allocated temporary data set name cannot be used for SYSLMOD. Make sure that your SYSLMOD (if specified) is a non-temporary data set.

| IMU allocates its own temporary SYSLMOD because the temporary DSN coded in the JCL is controlled by the operating system. The attempt to re-allocate SYSLMOD with the system-generated DSN fails.

Using JCL with multiple steps

These guidelines are valid for translating existing or newly developed Easytrieve programs to COBOL/390 and later versions of COBOL.

Migration Utility JCL library (distributed with the product) contains standard procedures for running the translator. See “System information” on page 196 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 207.

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 "Compatibility check" on page 30 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 11, "Installing Migration Utility," on page 203 for details.

   Change `EZPARMS= symbolic`, in the Proc, to point to the PDS where the 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 embedded 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.

Chapter 3. Conversion guidelines

Your existing Easytrieve Plus programs must be tested to make sure that no functionality is compromised. There are two ways of testing:

1. You can convert programs manually. This can be accomplished by converting each program and running it the Migration Utility way, the Easytrieve Plus way and comparing the outcomes visually or using a file compare tool. You would tweak your existing Easytrieve Plus JCL, or one of the supplied JCL, to test programs the Migration Utility way.

   This method is appropriate when Easytrieve Plus programs and macros are not centrally placed or cannot be centrally placed into a library for mass conversion, or when the overhead of learning and setting up the Automated Parallel testing utilities is not justified by the relatively small number of programs to do (30 or less).
2. You can convert and parallel test programs using the supplied Automated Parallel testing utilities. This method is preferred method. It is appropriate for large number of Easytrieve Plus programs that can be centrally accessed.

Discovering Easytrieve Plus programs

Before converting, you have to:

1. Locate all Easytrieve Plus programs.
2. Locate all Easytrieve Plus macros.
3. Locate Jobs and Procs that contain Easytrieve Plus steps.
4. Identify Link and Go runs.
5. Identify programs that run in linked mode.

Use FSYCNV70 (JCYCNV70), FSYCNV72 (JCYCNV72) and FSYCNV74 (JCYCNV74) to scan load libraries for linked Easytrieve Plus programs. JCYCNV74 produces a cross reference of program names vs PROC/JOB/EXEC names.

Use the FSYCNV50 (JCYCNV50) program to locate Easytrieve Plus programs source and JOBS and copy them into a common library for easy access.

Once you create a PDS of Easytrieve Plus programs and macros, you can run FSYCNV55 (JCYCNV55) to produce some valuable statistics about your Easytrieve Plus portfolio.

Do the following:

1. Read “Compatibility check” on page 30 to familiarize yourself with the differences between Easytrieve Plus and Migration Utility.
2. Run the FSYCNV50 (JCYCNV50) program to locate all Easytrieve Plus programs and instream (Link and Go jobs).
3. Run FSYCNV55 (JCYCNV55) to obtain program statistics.
4. Determine if you should use the manual process or the automated tools.
   - For the manual process, follow the hints in “A brief review of the manual Parallel Testing process” on page 22.
   - For the automated process, follow the instructions in “The Automated Parallel Testing utility” on page 22.

Using the FSYCNV50 (JCYCNV50) utility

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

This utility scans all PDS libraries in the supplied input PDS directory table. It identifies Easytrieve Plus JOBs and programs and copies them into a central location.

Use the supplied JCL JCYCNV50 to run this job. Embedded comments in the JCL describe all needed files.

FSYCNV50 reads members found in each PDS supplied in the USRDSN1 table and does the following steps:

1. Detects and copies all Easytrieve Plus programs to the NEWSRC output file.
2. Detects jobs that execute EZTPA00 and EZTPA00X programs and:
   - For //SYSIN DD *
     - copies inline Easytrieve Plus programs to NEWSRC
     - changes the SYSIN in the job to point to the new member
     - copies the updated job to the NEWJCL library
   - For SYSIN DD DSN=&DSN(&MEMBER)
     - copies the SYSIN Easytrieve Plus program to NEWSRC
     - copies updated job to the NEWJCL library

It is possible to have name overlaps (duplicate names) during the FSYCNV50 process, because the job reads multiple libraries as per the supplied USRDSN1 file.

The duplicate names are renamed to NEW*nnnnn* for Easytrieve Plus programs, and JCL*nnnnn* for JOBS, where *nnnnn* is the sequence number. That is, the first original name is copied as is and any additional duplicates are given a new name.

Programs are considered to be Easytrieve Plus programs if one or more of the following criteria are found in the first and second word of each line:

FILE &word
where &word is 8 characters or less

SORT &word
where &word is 8 characters or less

JOB INPUT &word
where &word is 8 characters or less

Jobs that invoke EZTPA00/EZTPA00X program via a symbolic SYSIN value and procs with a symbolic SYSIN value cannot be processed. Such jobs must be located manually or using other available tools, if any.

Data set names coded in the USRDSN1 table and not located on the system are bypassed.

FSYCNV50 runtime options

FSYCNV50 runtime options allow you some flexibility. They are specified as follows:

```
FSYCNV50 PARM=(&option1,&option2)
```

&option1 can be SYSIN or NOSYSIN:

SYSIN
: find programs pointed to by `//SYSIN DSN=&DSN(&MEMBER)`

NOSYSIN
: ignore programs pointed to by `//SYSIN DSN=&DSN(&MEMBER)`

&option2 can be SOURCE or NOSOURCE:

SOURCE
: find programs not embedded in the JCL

NOSOURCE
: ignore programs not embedded in the JCL

Examples:

```
//STEP010   EXEC PGM=FSYCNV50,PARM=(SYSIN,NOSOURCE)

//STEP010   EXEC PGM=FSYCNV50,PARM=(NOSYSIN,SOURCE)

//STEP010   EXEC PGM=FSYCNV50,PARM=(SYSIN,SOURCE)
```

Input files

USRDSN1
: A PDS member that contains data set names to process. The data sets must be PDS or PDSE libraries. Use the supplied #DSNAMES located in the SYS1.SFSYEZTS library as a skeleton. Follow the instructions in #DSNAMES.

MACTABL
: This is a conversion table for macro names that are longer than 8 characters. If long macro names can be found in the Easytrieve Plus programs, they must be converted to names of 8 characters or less. Code the old name in positions 1–16, including the % sign. Code the new name in position 18–26. Use the supplied #MACNAME located in SYS1.SFSYEZTS as a skeleton.

IFILE00
: This is a dummy file used by the dynamic allocation program.

: **Attention:** Do not change this file.

Output files

NEWJCL
: This file contains all jobs that were found with one or more "EXEC PGM=EZTPA00/EZTPA00X" steps. Make sure you allocate enough space and directory blocks to hold all selected jobs.

NEWSRC
: This file contains all located Easytrieve Plus programs. Long macro names are replaced by the short macro names found in the MACTABL file. Make sure that you allocate enough space and directory blocks to hold all selected programs. A good way is to allocate a large secondary space, should you need it.

TMPXRF3
: This file contains the list of searched PDSs and the origin of the processed members.

TMPXRF4
: This file contains the list of JOBs copied into NEWJCL.

TMPXRF5
: This file contains the list of Easytrieve Plus programs copied into NEWSRC.

TMPXRF6
: This job contains the list of duplicate names in the NEWSRC.

TMPJCL
: Temporary staging area for all selected members. Make sure to allocate enough space and directory blocks to hold all jobs from all selected data sets.

TMPJCL2
: This is a work file. It contains one record per selected job.

TMPDSN2
: This is a work file. It contains one record per selected program.

TMPXRF1, TMPXRF2
: These are work files. They contain one record per selected program.

SORTWK1-SORTWK4
: These are work files. They contain one record per selected program.

SORTWK01, SORTWK02
: These are sort program work files.

Using the FSYCNV55 (JCYCNV55) utility

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

This utility scans Easytrieve Plus programs centrally located in a PDS (or multiple concatenated PDSs), and produces reports containing valuable statistics about programs. Programs are classified as DB2, DLI/IMS or regular programs.

Input files

FILEIN
: This file must point to one or more PDSs where Easytrieve Plus programs are located.

IMACRO1–MACRO8
: Libraries where your application Easytrieve Plus macros are located.

EXCLUDE
: This is a table of members to be excluded from the search (see the #EXCLUDE member in SYS1.SFSYEZTS for the layout).

Output files

REPORT0
: A list of programs sorted by number of reports in each program.

REPORT1
: A list of programs by type and number of lines of code, including a summary recap by type, number of sorts, number of jobs, number of SQL statements, and so on. Make sure that you look at the bottom of this report. All summary matrices are located at the bottom of this report.

REPORT2
: A list of located macros.

REPORT3
: Programs found in input without a JOB statement (which implies that they are probably not Easytrieve Plus programs).

Running the JCYCNV70, JCYCNV72 and JCYCNV74 Utilities (discovering Easytrieve Plus load modules)

- Run the JCYCNV70 utility to create #MEMTAB of Easytrieve Plus load modules (see JCL for details).
- Run the JCYCNV72 utility to locate JCL and expand PROCs (see JCL for details).
- Run the JCYCNV74 utility to locate the EXEC statement in the expanded JCL (see JCL for details).

Using the FSYCNV70 (JCYCNV70) utility

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

This utility scans load libraries for a specific string in the load modules.

Input files

USRDSN1
: A PDS member that contains data set names to process. The data sets must be PDS or PDSE libraries. Use the supplied #LOADLIB located in the SYS1.SFSYEZTS library as a skeleton. Follow the instructions in #LOADLIB.

USRCHAR
: A PDS member that contains the search strings. The search strings are strings that identify the Easytrieve Plus load modules. Currently EZTPA00 is the only string used to identify the Easytrieve Plus programs. Other strings can be added.

Output files

SYSPRINT
: A list of load modules that contain a located string.

MEMTABLE
: A table of program names that contain a located string. Presumably, these are Easytrieve Plus programs. Used as input to JCYCNV74.

MIG00TAB
: These are EXEC statements that can be used to run JCYMIG00.

Using the FSYCNV72 (JCYCNV72) utility

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

This utility scans for jobs in the libraries supplied by the USERDSN1 table and expands PROCs inside the jobs giving statements that are expanded by JES2/JES3 runs. The purpose is to create expanded PROCs so that the JCYCNV74 run can detect all Easytrieve Plus steps (steps that run as Link and Go and steps that run as load modules).

Input files

USRDSN1
A PDS member that contains data set names to process. The data sets must be PDS or PDSE libraries. Use the supplied #DSNAMES located in the SYS1.SFSYEZTS library as a skeleton. Follow the instructions in #DSNAMES.

FJIPROCS
One or more PDS libraries that contain PROCs. PROCs are pulled in from this library.

FJIINCL
One or more PDS libraries that contain INCLUDEs for JCL (if any). INCLUDE statements found in the JCL or PROCS are pulled in from this library.

Output files

FJOTEMP
Temporary staging area for all selected members. Make sure to allocate enough space and directory blocks to hold all jobs from all selected data sets.

FJOJOB0
Output PDS contains Jobs/JCL expanded by FSYCNV72. Note that FSYCNV71 is called by FSYCNV72 to expand all Jobs/JCL. This PDS becomes an input to FSYCNV74 (JCYCNV74) or other search tools.

FJOJORG
A PDS that contain JOBs selected from libraries in USRDSN1. This file is optional. Remove DD from the JCL if not desired.

FJOJCHG
A PDS that contains all JOBs that contained one or more EXEC statements without step name. Step names are changed to //G to make it a valid step name for JCYMIG00 run. The adjusted JCL should be used as input to JCYMIG00 (FSYMIG00 program).

This file is optional. Remove DD from the JCL if not desired.

ORIGRPT
This file contains the blueprint of searched libraries.

DUPSRPT
Report of duplicate Jobs/JCL member names.

All other files are temporary files used for the duration of the run.

Using the FSYCNV74 (JCYCNV74) utility

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

This utility scans for jobs in the supplied library and creates a cross reference of JOB/EXEC names found in the JCL.

Input files

FJEJOB0
A pds created by JCYCNV72 (FJOJOB0 file). It contains JCL/JOBs with expanded PROCs.

MEMTABL
A sequential file that contains a list of programs to look for. Normally, this is the MEMTABLE created by JCYCNV70 run. This table should be massaged to make sure that only programs of interest are included.

IFILE00
This is a work file internally used by the program. Leave as is.

Output files

FJXRFFL
A cross reference file of Job/PROC and EXEC names that match those in the MEMTABLE.

All other files are temporary files used for the duration of the run.

Using FSYMIG21 (JCYMIG21) stand-alone compare JCL generator for tape files

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

This utility creates compare and purge JCL from the data set names generated by the parallel test. The utility must be run when the compare step in the parallel test run is deferred due to tape files.

This utility creates:
- Stand-alone compare (JCYMIG20) job using data sets located in FJCOMP0 file.
- Purge JCL that purges data sets cretated by the parallel test run.

Input files

FJCOMP0
Input file generated by the parallel test run.

This is the //SYSUT2 DD DSN=&SYSUID.CONV.FJCOMP0.&JOBNAME file defined in #FJICNT2/FJECNT2 control files ("//COMPARE-STEP"). This file contains a list of files to be compared.

FJICNTL
DD name for the Control file used in JCYMIG00 (dataset should be #FJICNT2 or FJECNT2).

FJJOB00
Optional JOB statement(s) for the created JCYMIG20 and purge JCL.
- If present, statements in this file are used.
- If omitted, a default JOB statement is used.

PARM PARM options on the exec (for details, see "Using FSYMIG20 (JCYMIG20) stand-alone compare utility") as these values are passed on the EXEC statement of the created JCYMIG20 job.

```
PARM=(&P1,&P2,&P3,&P4,&P5,&P6,&P7,&P8,&P9)
```

Where:

&P1=COMPARE
Compare option. Leave this parameter as is.

&P2 =NOLET
Condition Code option:

NOLET
Return error code.

LET Return RC=00 even if compare errors exist.

&P3 =&program
User exit program. Code 1-8 character exit program name. The default is NOEXIT.

&P4 =&mode
Compare mode for printer files:

BYTE Compares byte by byte.

WORD
Compares word by word.

&P5 = Printer file CC handling:

EXPCC
Replicate print control characters.

NOEXPCC
Compare line by line (ignore print control characters).

&P6 =64
Compare error limit. Compare program terminates when this limit is reached.

&P7= Job statement accounting information. Must be enclosed in parentheses.

Example: `'(1234,N0000000)'`

&P8= Job statement user name.

Example: `'JOHN DOE'`

&P9= JOB class.

Example: `A` for CLASS=A.

Output files

FJPURGE
A JOB for purging datasets created by the parallel test run.

FJCMP20 JCYMIG20 job (Compare JCL)
By default, the job is punched into the internal reader, however JCL can be changed to write this job to a file (See JCL).

Using FSYMIG20 (JCYMIG20) stand-alone compare utility

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

This utility lets you compare two files at the same time. The utility considers print control characters and has the ability to call a user exit for massaging the input buffers before comparison. A full description of its capabilities can be found in the comments imbedded in the JCYMIG20 JCL located in the SYS1.SFSYJCLS PDS.

Input files

FJCOMP1
: Input file #1 to be compared.

FJCOMP2
: Input file #2 to be compared.

PARM
: PARM options on the exec (for details, see "Using FSYMIG20 (JCYMIG20) stand-alone compare utility" on page 20).

```
PARM=(&P1,&P2,&P3,&P4,&P5,&P6)
```

Where:

&P1=COMPARE
: Compare option. Leave this parameter as is.

&P2 =NOLET
: Condition Code option:

  NOLET
  : Return error code.

  LET
  : Return RC=00 even if compare errors exist.

&P3 =&program
: User exit program. Code 1-8 character exit program name. The default is NOEXIT.

&P4 =&mode
: Compare mode for printer files:

  BYTE
  : Compares byte by byte.

  WORD
  : Compares word by word.

&P5 =
: Printer file CC handling:

  EXPCC
  : Replicate print control characters.

  NOEXPCC
  : Compare line by line (ignore print control characters).

&P6 =64
: Compare error limit. Compare program terminates when this limit is reached.

Exit program FSYXIT00 is supplied with the product. It can be located in the SYS1.SFSYEZTS PDS.

For more detail on exit capabilities, see "Tailoring FSYXIT00 compare exit program" on page 29.

Output files

FJERLOG
: Log of detected errors is printed in hexadecimal in segments of 100 bytes.

The hexadecimal print shows 100 bytes from FJCOMP1, then 100 bytes from the FJCOMP2 file, below each other for easier comparison, until all bytes are shown.

FJSTATS
A cumulative statistics file for each run.

SYSOUT
Additional compare statistics

A brief review of the manual Parallel Testing process

1. Create conversion libraries where you will keep your converted Easytrieve Plus program source, Easytrieve Plus macros, JCL and PROCS.
2. Locate JCL and make sure that the JOBs run cleanly the Easytrieve Plus way. This establishes your test environment for that specific JOBs.
3. Run your Easytrieve Plus program(s) through the Migration Utility translator and:
   - Make the necessary changes to any flagged statements until you get clean output.
   - Create load modules (link programs) for programs that run in compiled mode. Note that Link and Go programs do not have to be linked.

     Note: Consider using one of the automated conversion engines JCMUMIG1 or JCMUMIG2 for this step, as described in "The Automated Parallel Testing utility."
4. Change the Easytrieve Plus JOBs to save print and output files to disk for the comparison utility.
5. Run your tailored JOBs the Easytrieve Plus way.
6. Tailor your working Easytrieve Plus JOBs to include all necessary JCL statements to run with COBOL generated by Migration Utility. Note that the printer and output files must be saved to a new set of files for comparison.
7. Run your JOBs the Migration Utility (COBOL) way.
8. If all goes well, compare the output files using a comparison utility.
9. If the comparison fails, you must determine the cause, make program changes if necessary, and rerun the Migration Utility JOB. Rerun the Easytrieve Plus JOB only if you made changes to the program. Remember to delete the output files from the previous run so that you can run the program again.
10. If all the files compare successfully, you must do one or more tests using different input criteria or input files to make sure that the special conditions (if any) work properly. This requires repeating steps 4, 6, 7, and 8.
11. When you are satisfied with the outcome, you can tailor your JCL for production use and promote your tested programs to production.

The Automated Parallel Testing utility

The goal of this utility is to have your computer do the work with as little human intervention as possible. Your JOB may contain multiple steps comprising both Easytrieve Plus and non–Easytrieve Plus programs. Ideally, you would want to run the Easytrieve Plus steps, the Migration Utility steps, the comparison steps and all non–Easytrieve Plus steps in a single job. The Automated Parallel Testing Utility comes very close to achieving this goal.

Provided utility programs

The following utility programs are provided:

FSYCNV50
(JCYCNV50) Easytrieve Plus programs Discovery utility.

FSYCNV55
(JCYCNV55) Easytrieve Plus programs Analysis utility.

FSYMIG00
(JCYMIG00) JCL Adjuster utility program.

FSYMIG05
Migration Utility step initiator for Link and Go programs (replaces EZTPA00).

FSYMIG10
Migration Utility one-step driver program (runs Migration Utility Link and Go and application programs). This program is initiated by FSYMIG05 or the boot strap of linked program.

FSYMIG20
Compare Program driver (runs the compare program for each pair of files). A step to run this program is generated in your JCL by the FSYMIG00 program.

FSYCNV20
Compare Program. This program is invoked by FSYMIG20 for each pair of files to compare.

FSYXIT00
Compare program user exit. This is an optional exit. See PARM= for the
| FSYCNV20 step located in the #FJICNTL/#FJICNT2 file.

The following JOBS are provided in the &HQUAL.SFSYJCLS library:

JCYCNV50
Easytrieve Plus programs Discovery utility

JCYCNV55
Easytrieve Plus programs Analysis utility

| **JCYCNV70**
| Scans user load libratries for Easytrieve Plus load modules and creates a
| table of located modules.

| **JCYCNV72**
| Locate JCL in PDS libraries and expand PROCs for JCYCNV74 for the
| modules in the member table created by JCYCNV70. Internal programs
| used are FSYCNV73 and FSYCNV71.

| **JCYCNV74**
| Locate EXEC statements in the JCL library created by JCYCNV72.

JCYMIG00
JCL Adjuster program (creates Production or Parallel Test JCL)

JCMUMIG1
Automated Conversion Engine

JCMUMIG2
Manual Conversion Proc

JCMUCNV1
Automated Conversion Initiator (initiates JCMUMIG1)

Preparing Jobs for the Automated Parallel Testing utility

Note: All jobs are located in the SYS1.SFSYJCLS library.

Create Parallel Testing/Conversion Environment

The first step is to create the Parallel Testing/Conversion environment.

1. Create the PROCS/JCL and PDS libraries needed for the conversion. You can use the JCGENFIL job to do this. Make sure that you tailor data set names so that they are unique to your own environment.

   Modified Easytrieve Plus MACRO libraries can be shared with others. Sharing program, JCL, load and statistics libraries can only complicate things. The following files or libraries are needed:

DSN=&HQUAL.MUCONV.BOOTSTRP.LOADLIB	Linked programs boot straps for parallel testing
DSN=&HQUAL.MUCONV.COBSRC	Generated COBOL (see FJSYSPH)
DSN=&HQUAL.MUCONV.DBRMLIB	DBRM library needed for DB2 only
DSN=&HQUAL.MUCONV.ERRORS	Automated conversion error log
DSN=&HQUAL.MUCONV.EZTEST.LOADLIB	Easytrieve Plus linked modules. May use production instead.
DSN=&HQUAL.MUCONV.FJOJOB0	Output JOBS of JCYMIG00
DSN=&HQUAL.MUCONV.FJOPROC	Adjusted PROCS - output of JCYMIG00
DSN=&HQUAL.MUCONV.FJSYSP0	FJSYSP0 saved by conversion JCMUMIG1/JCMUMIG2 PROCS
DSN=&HQUAL.MUCONV.LOADLIB	Load library for converted linked program
DSN=&HQUAL.MUCONV.EZPLUS.MACROS	Easytrieve Plus application modified macros
DSN=&HQUAL.MUCONV.EZPLUS.SOURCE	Modified Easytrieve Plus programs

2. Tailor conversion control files used by the JCYMIG00 job:
   - #FJICNTL for Easytrieve Plus jobs that do not contain tape files.
   - #FJICNT2 for Easytrieve Plus jobs that contain tape files.
   - #FJECNTL for Easytrieve Classic jobs that do not contain tape files.
   - #FJECNT2 for Easytrieve Classic jobs that contain tape files.

   A master copy of these files is located in &HQUAL.SFSYEZTS. Copy these files into your own PDS and tailor them to your requirements. Follow imbedded comments in each file to get you started. Note that some files created in step 1 may have to be specified in these control files.
3. If you are converting Easytrieve Classic, continue with "Prepare JCL/JOBs to run all steps for the comparison" on page 25. If you are converting Easytrieve Plus, tailor the #CNVPROC file. This file is used by the jobs JCMUMIG1 and JCMUMIG2. A master copy of this file is located in &HQUAL.SFSYJCLS. Copy this file into your own PDS and tailor it to your requirements. There are enough comments embedded in the file to get you started. Do not forget to point the FJPROC0 ddname in your conversion proc JCMUMIG1 or JCMUMIG2 to this file.

   Non-SMS users: You must make dynamically-allocated files available to subsequent steps. To do this, globally change the DISP= values:
   - The ",PASS," option to ",CATLG,"

- The “,KEEP,” option to “,CATLG,”

4. Tailor your conversion engine. You can use the JCMUMIG2 proc to run your programs manually through Migration Utility one by one, or you can tailor the JCMUMIG1 proc and JCMUCNV1 JCL to initiate the automatic conversion engine.

   The automatic conversion engine submits programs for conversion automatically, one by one, by reading a directory table of programs to do. For more details, read the comments embedded in the procs.

Run Easytrieve Plus programs through Migration Utility

The next step is to run the Easytrieve Plus programs through Migration Utility using the JCMUCNV1/JCMUMIG1 or JCMUMIG2 utilities.

1. Programs that run in compiled mode will be linked:
   - A load module is created to SYSLMOD library
   - A bootstrap module for parallel test is created to the BOOTSTRP library
   - File information file FJSYSP0 is saved for use by the JCYMIG00 JCL Adjuster utility
2. Link and Go programs are translated but not linked. These programs are translated "in-flight" during the JCYMIG00 and Parallel Test run as well.

   Note: You must submit JCMUCNV1 to initiate JCMUMIG1. The JCMUMIG1 has a restart step to restart itself for the next program.

Prepare JCL/JOBs to run all steps for the comparison

The next step is to prepare your existing JCL/JOBs to run the Migration Utility, Easytrieve Plus/Classic, and Compare steps back-to-back.

Note: It is essential to tailor the #FJICNTL/#FJICNT2 and #FJECNTL/#FJECTN2 control files before doing this task, as most information inserted or deleted from your JOB comes from these files.

To do this, use the supplied JCL JCYMIG00. JCYMIG00 runs the FSYMIG00 program. FSYMIG00 detects Easytrieve Plus programs that run in compiled mode, creating the necessary steps to run the Migration Utility steps, the Easytrieve Plus steps, and comparison step for each Easytrieve Plus step found in the JCL.

Note: Easytrieve Classic does not run in compiled mode.

Note: REGION=0M on the job or the EXEC statement must be coded to allocate enough memory for your JOB.

FSYMIG00 detects Link and Go Easytrieve Plus/Classic programs and:

- Runs the Migration Utility translator for SYSINs found in the job to make sure that the program translates cleanly, and obtains the necessary file information for generating the Easytrieve Plus/Classic (EZT) step JCL.
- Creates a modified JOB to run FJOJOB0 with the necessary steps to run the Migration Utility steps, the Easytrieve Plus/Classic steps, and the comparison step for each Easytrieve Plus/Classic step found in the JCL, including any other non-Easytrieve steps. Note that all external PROCs are converted into in-stream PROCs to reduce fragmentation.

Rules observed when preparing your job for parallel testing

The JCYMIG00 (FSYMIG00) program observes the following rules when making changes to your JCL. "Migration Utility Step" refers to the Migration Utility step. "EZT Step" refers to the Easytrieve Plus/Classic step.

1. SYSOUT files (mostly PRINTER files)

   Migration Utility Step:
   : Files are redirected to a sequential disk file. `DISP=(NEW,CATLG,CATLG)` is used.

   EZT Step:
   : Files are redirected to a sequential disk file. `DISP=(NEW,CATLG,CATLG)` is used.

   Compare:
   : Files are compared using data set names as generated above.

2. Output DISP=NEW, Disk and Tape files (sequential files)

   Migration Utility Step:
   : Files are used as defined in the original JCL.

   EZT Step:
   : New files are generated to a disk file using the same attributes. `DISP=NEW` is preserved in the EZT step for compatibility.

   Compare:
   : Files are compared using data set names as generated above.

3. Output DISP=MOD, Disk and Tape files (sequential files)

   Migration Utility Step:
   : Files are used as defined in the original JCL.

   EZT Step:
   : New files are generated to a disk file using the same attributes. However, the original input files are copied to these new files in the Migration Utility step to preserve input contents so that the EZT step can add records (MOD on) to it. `DISP=MOD` is preserved in the EZT step for compatibility.

   Compare:
   : Files are compared using data set names as generated above.

4. Output DISP=OLD, Disk and Tape files (sequential files)

   Migration Utility Step:
   : Files are used as defined in the original JCL.

   EZT Step:
   : New files are generated to a disk file using the same attributes, however, these files are created in the Migration Utility step with the same attributes as the original file defined in the JCL. `DISP=OLD` is preserved in the EZT step for compatibility.

   Compare:
   : Files are compared using data set names as generated above.

5. New VSAM files (opened for output only, KSDS, RRN, and ESDS)

   Migration Utility Step:
   : Files are used as defined in the original JCL.

   EZT Step:
   : New files are generated to a disk file using the same attributes.

Note: VSAM files are allocated via SMS JCL statements.

Compare:
Files are compared using data set names as generated above.

6. Update VSAM files (I-O mode KSDS and RRN)

   REPRO Step:
   A utility IDCAMS REPRO step is generated before the Migration Utility step that copies the original VSAM files to another set, with the same attributes as the original files.

   Note: VSAM files are allocated via SMS JCL statements.

   Migration Utility Step:
   Files are used as defined in the original JCL.

   EZT Step:
   Files are used as generated by the IDCAMS REPRO step.

   Compare:
   Files are compared using data set names as generated above.

7. The original JOB step executing Easytrieve Plus/Classic Link and Go (or compiled program) is converted to run COBOL generated by Migration Utility, followed by the customized EZT step, followed by the compare step.

   The output files created by Migration Utility are passed on to the compare program and other JOB steps, while the files created by Easytrieve Plus/Classic are passed on to the compare program only.

Parallel testing restrictions

1. Updates to DB2 tables and the IMS/DLI data base must be verified manually. Migration Utility does not have the ability to clone DB2 tables and IMS/DLI data bases for parallel testing and comparison. Programs that use DB2 tables or IMS/DLI as input are not a problem, as input files need not be compared.
2. The input files must not change while the utility is running, as any change to the input files might cause a mismatch. For example, if you point to a file that is updated by CICS while running, that file may contain different information when running the Migration Utility and EZT steps.
3. Files that are SORTED in place may not produce the correct results. For example, if your program sorts FILEIN to FILEIN, but it uses FILEIN, before sort, to produce output (report or output file), the order of records will not be the same when the Easytrieve Plus/Classic step runs. Sorting files in place is not a standard practice, so this restriction should have minimum impact on the overall task. All such programs and jobs cannot be validated by this utility.
4. Be aware of time and date fields, and any time-dependent logic. If your program makes decisions depending on a cut-off date or time, or outputs the current date and time to output files, the comparison will fail. If you depend on the cut-off date or time, make sure that you run the test within the same window. If your output files contain date or time stamps, you can choose to massage records before comparison through a user exit (see the FSYXIT00 sample program located in SYS1.SFSYEZYS).
5. Before attempting to run a parallel test, test your job to make sure that it runs cleanly with Easytrieve Plus/Classic and is in a good working condition. This is a required task, since encountering environmental problems, or attempting to run a job without knowing that it runs with Easytrieve Plus/Classic, may further complicate the process.

6. When parallel testing DB2 programs, make sure that the DB2 run mode is compatible with the Easytrieve Plus/Classic run mode. That is, if Easytrieve Plus runs DB2 programs in dynamic mode, Migration Utility must run in Dynamic mode too. Also, make sure that the correct SSID and PLAN are used for both steps. To make testing easier, enable Migration Utility to run with the same PLAN names used to run Easytrieve Plus/Classic programs.
7. Link and Go programs that DISPLAY to SYSPRINT may produce compare errors due to DISPLAY lines being written by Easytrieve Plus/Classic immediately after the Easytrieve Plus/Classic program listing. To cope with the problem, FSYMIG10 (parallel testing driver program) inserts a JOB with a "DISPLAY NEWPAGE '<REPORTS>' " before the first JOB procedure into every Link and Go Easytrieve Plus/Classic program. When comparing, the compare program positions the Easytrieve Plus reports beyond the <REPORTS> tag, thus properly positioning SYSPRINT to the first report line. The change is made to the temporary copy (created by Migration Utility) of your program source, thus the change has no impact on the production version of your program.
8. Extraneous blank lines found in reports are bypassed with a warning message. The return code is not affected.
9. Some programs may produce output file records padded with binary zeros by Easytrieve Plus/Classic and spaces by COBOL due to partial data being moved directly into the I/O area. To avoid compare errors, you can add a routine to the compare exit program, to replace the trailing binary zeros with spaces.

Running the Parallel Test

1. Inspect your JOB generated by the FSYMIG00 (JCYMIG00) program. The job is located in the FJOJOB0 PDS. Make sure everything looks correct.
2. Submit the job and let it run.

Note: Do not change the job name in your job after you run it through JCYMU00. The reason is that the data set names for the output files contain the job name that was in the JCL before it was run through JCYMIG00. Changing the job name means the comparison would not work.

Do not worry about runtime statistics at this time. The Migration Utility step translates Link and Go programs twice and makes copies of certain files for the EZT and compare steps. Depending on the file types, this may add a substantial amount of processing to the Migration Utility step.

The compare program runs after the Migration Utility and EZT steps are completed. The records found in error are printed in hex under the original DD name, in segments of 100 bytes. For example, REPORT1 errors would be printed to REPORT1 ddname, FILEOUT to FILEOUT ddname, and so on. The input files are dynamically allocated by the FSYMIG20 program: FJCOMP1 for the Migration Utility file, and FJCOMP2 for the EZT file.

The outcome is recorded to the FJSTATS and SYSOUT files.

The hex print is in increments of 100 bytes. The records in error are printed below each other. For example, a 150-byte record would print:

```
100 bytes of record from file 1
100 bytes of record from file 2
50 bytes of record from file 1
50 bytes of record from file 2
...
```

In this way, records are shown above each other for easier comparison.

After the compare step completes, files created by the EZT step are deleted (depending on the PARM= option of the compare step). Files are deleted only when no differences are found.

The file defined by the FJPURGE ddname contains an IEFBR14 EXEC step with DISP=(MOD,DELETE,DELETE) for each generated data set. You can submit this job to purge all generated data sets should you have to rerun your test or should they no longer be needed. The output data sets found in the original JCL are not deleted.

Inspect your output. If you find errors, you must determine the cause, correct your Easytrieve Plus/Classic program, and rerun the parallel test. Otherwise, you can assume that results are compatible.

A good practice is to rerun the parallel test using different input and different criteria (if the logic is parameter driven or time dependent) to make sure that all conditions are tested.

When you are satisfied with the outcome, proceed with the steps in "Preparing JCL for Production use."

Tailoring FSYXIT00 compare exit program

The FSYXIT00 COBOL program allows you to massage input records before comparison. FSYXIT00 is located in SYS1.SAFSEZTS. For tailoring information, read the comments embedded in the program.

In short, two parameters, LS-REQUEST-LIST and LS-RECORD, are received as defined in the Linkage Section. The program is entered initially for each pair of files to be compared, to establish the comparison parameters (A00000-GETPARM-LOGIC paragraph). Comparison options can be changed in this paragraph.

The program is then entered one time for each record in each file (B00000-MASSAGE-LOGIC paragraph). Records can be changed in this paragraph. For example, time stamps, dates, and other known problem fields can be spaced out to avoid compare failure.

Preparing JCL for Production use

The FSYMIG00 (JCYMIG00) job tailors JCL based on the PARM=TEST or PARM=PROD in the JCL.

To change your job for production use, create a separate #FJICNTL/#FJECNTL member (you can call it a different name) and make changes to it for production environment. Run JCYMIG00 with the PARAM=PROD option.

JCL for production use must have the Migration Utility runtime library &HQUAL.SFSYLOAD allocated to JOBLIB or STEPLIB. The Easytrieve Plus/Classic libraries must be removed from the JCL. The EZTPA00 program for Link and Go steps must be changed to FSYTPA00. JCYMIG00 makes all these changes for you based on the information in the #FJICNTL/#FJECNTL file.

FSYMIG00 (JCYMIG00) required files

FJIJOB0
: This is your input PDS that contains Easytrieve Plus/Classic jobs. Multiple PDSs can be concatenated.

FJIPROC
This is your PROCs and INCLUDEs library that Easytrieve Plus/Classic JOBS use. Multiple PDSs can be concatenated.

FJICNTL
This file must point to #FJICNTL/#FJECNTL parameters file:
- #FJICNTL for Easytrieve Plus jobs that do not contain tape files.
- #FJICNT2 for Easytrieve Plus jobs that contain tape files.
- #FJECNTL for Easytrieve Classic jobs that do not contain tape files.
- #FJECNT2 for Easytrieve Classic jobs that contain tape files.

FJIBOOT
This is the output bootstrap load library. Bootstraps for converted programs must reside in this PDS. Bootstraps are created by JSMUMIG1 and JCMUMIG2 procs.

FJOJOB0
This PDS contains the tailored jobs.

FJOPROC
This PDS contains the tailored PROCS for production use (PARM=PROD).

FJTEMP0
This is a temporary file used for instream Easytrieve Plus/Classic programs.

Dynamic Allocation option

There are five options in EZPARAMS that control the runtime dynamic allocation feature.

Note that these options are required for parallel testing.

```
DYNALLOC=YES
PRINTER=SYSPRINT
SORTWORK=3
SYSOUT='*,REFDD=SYSPRINT'
WRKSPACE='(CYL,(10,50),RLSE),UNIT=SYSDA'
```

When using these options, there is no need to add TEMPWK*, SORTFL*, SORTWK*, and virtual files to your JCL. In addition, PRINTER files are allocated dynamically at run time for all missing PRINTER files, including the FJSYABE (the abend file).

You can change SYSOUT and WRKSPACE to accommodate your needs. Be careful with space. A good practice is to allocate a small primary space with some larger secondary space. This way, if your file is too large for primary space, the secondary space will cover for it.

Should you need more space for any given file, you can add it to your JCL. Files are dynamically allocated only if they are not found in the JCL.

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 supports VSAM, QSAM, SAM, SQL/DB2, DLI/IMS, IDMS, tape files, and unit record devices.

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 207. 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).

All VSAM files

Migration Utility requires that VSAM files are defined as VSAM files in your Easytrieve Plus program while Easytrieve Plus does not. This holds true for input and output VSAM files. See "Defining VSAM files" on page 52 for available parameters.

All other files (flat files and unit record devices)

If you are using IOMODE=NODYNAM, all input and output files must be defined with the proper RECFM, that is V, VB, VBS, F, FB, U and PRINTER. COBOL returns the I/O error code 39 when the record format defined in the program does not match the JCL or file catalog.

If you are using IOMODE=DYNAM or IOMODE=DYNAM24, all output files must be defined with the proper RECFM, that is V, VB, VBS, F, FB, U and PRINTER. The record format for input files is determined at run time by Migration Utility. Migration Utility I/O modules return I/O error code 39 when the record format for output files defined in the program does not match the JCL or file catalog.

Record length handling

If you are using IOMODE=DYNAM or IOMODE=DYNAM24, output files must be defined with the proper RECFM, that is V, VB, VBS, F, FB, U and PRINTER.

If you are using IOMODE=DYNAM or IOMODE=DYNAM24, the record format for input files is determined at run time by Migration Utility.

If you are using IOMODE=DYNAM or IOMODE=DYNAM24, the record length for both input and output files is not a concern.

If you are using IOMODE=NODYNAM (native COBOL I/O), make sure that the record size of each file is fully defined. This can be done by coding the record length on the file statement, or by defining the record layout in full. COBOL does not retrieve record size dynamically during the run time.

SBCS and DBCS character support

Migration Utility supports single-byte character set (SBCS) and K (DBCS) field types.

- 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).

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 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 4-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:

```
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
```

Restriction:

Packed Unsigned (U) fields, 1-byte binary fields and 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: SSMODE=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 unintended 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 the `NAMETAB` parameter of EZPARAMS to change any special characters found in Easytrieve field names to make field names COBOL compliant.

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

Migration Utility Options are described in Chapter 11, “Installing Migration Utility,” on page 203.

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, rearrange 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 Migration Utility.
- 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*. The subordinate fields of length greater than 1 redefine the group and are treated as fields without occurs.

  Example

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

  is converted to

  ```
     02 WFIELD1-G1.
        03 FILLER OCCURS 50 TIMES INDEXED BY INDEXY-001.
           04 WFIELD1.
              05 WFIELD2               PIC X(1) VALUE SPACES.
     02 FILLER REDEFINES WFIELD1-G1.
        03 FILLER                PIC X(1).
        03 WFIELD3               PIC X(9).
        03 FILLER                PIC X(40).
  ```

Binary field handling

2-byte and 4-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 62.) COBOL Compiler option TRUNC(OPT) is recommended.

1-byte and 3-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 62.)

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-values or high-values to the fields, such as binary zeros or all X"FF". For such cases, replace the hex value by either the "LOW-VALUE" or "HIGH-VALUE" statement.

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.

Supporting VS COBOL and other incompatible COBOL subroutines

The Easytrieve programs are translated to be compatible with z/OS Enterprise COBOL. Called subprograms that are written in other COBOL dialects such as VS COBOL, and which contain VSAM file I/O routines, may experience a problem. This is strictly a COBOL compatibility problem. Such subroutines must be compiled with a COBOL/390 or z/OS Enterprise COBOL 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 quotation marks.
2. It generates a dynamic call for program names that are not enclosed in quotation marks.

Easytrieve does not accept quotation marks 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 quotation marks.

Addressing mode considerations

The addressing mode of called programs must be compatible with the addressing mode of your Migration Utility-generated COBOL program.

- When you want to use dynamic I/O mode:
  - Use IOMODE=DYNAM if your sub-program was compiled and linked with AMODE(31) or AMODE(ANY).
  - Use IOMODE=DYNAM24 if your sub-program was compiled and linked with AMODE(24). Parameters can be locally provided in the Easytrieve Plus program after the PARM statement as follows:

    Example:

    ```
    * EASYTRAN: IOMODE=DYNAM24
    * END-EASYTRAN
    ```
- When you want to use static I/O mode:
  - If your sub-program requires AMODE(31) or AMODE(ANY), Migration Utility's default settings should be sufficient.

– If your sub-program requires AMODE(24), translate the Easytrieve Plus program with IOMODE=NODYNAM and PROCESS DATA(24). Parameters can be locally provided in the Easytrieve Plus program after the PARM statement as follows:

  Example:

```
* EASYTRAN: PROCESS DATA(24)
* EASYTRAN: IOMODE=NODYNAM
* END-EASYTRAN
```

  The above options compile COBOL below the line and use standard COBOL file definitions and I/O statements. Ensure that the file record length is fully specified and that, for VSAM files, the first defined field is the file key (as described in "VSAM key usage" on page 33).

RETURN-CODE considerations

Migration Utility considers RETURN-CODE after each CALL according to the standard calling conventions. This normally results in a good run, but potentially a bad return code can be returned at End of Job if a called sub-program does not set the return code properly.

Easytrieve does not save RETURN-CODE unless specifically requested on the CALL statement. Use the RETURNC=EASYT EASYTRAN/EASYPARAMS option to make RETURN-CODE handling compatible with Easytrieve Plus.

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 differences at run time 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.

Sign of numeric 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).

COBOL forces a “C” sign in positive zoned decimal numeric fields that are defined with a sign. Refer to the FSIGN= option in Chapter 12, “Migration Utility translation options,” on page 217 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 2-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 2-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 2-byte binary field can accommodate up to 9,999 in value (unless TRUNC(OPT) 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 MOVENUM=NATIVE is in effect and 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 an incorrect 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.

When MOVENUM=EASYT is in effect, Migration Utility expands moves according to Easytrieve Plus rules. This option gives you optimum compatibility. Refer to Chapter 12, “Migration Utility translation options,” on page 217 for MOVENUM= option pros and cons.

When MOVENUM=NATIVE is in effect, 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 98.

- FILE-STATUS code in the generated COBOL program is a 2-byte alphanumeric field while in Easytrieve it is a fullword numeric field. Instructions in an Easytrieve program that assign FILE-STATUS to a numeric field are flagged as errors.

  Example:

  ```
  RETURN-CODE = FILEIN: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 2-digit constant (literal) enclosed in quotation marks.

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 55 for syntax rules.

JCL for converted program

JCL needed to run Link and Go programs, or linked programs of current Migration Utility Version 2 Release 1 or later, is compatible with Easytrieve Plus as described in Chapter 2, "Using Migration Utility," on page 5. If you elect to use the multi-step JCL compatible with the previous releases, or if you need to generate JCL for a new program, you can create skeleton JCL for the COBOL programs as described below.

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 "Using JCL with multiple steps" on page 10.

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, and SYSOUT 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.

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 on your report do not match exactly to those printed by Easytrieve Plus, consider using the MOVERPT=EASYT option as described in Chapter 12, “Migration Utility translation options,” on page 217.

If the MOVERPT=EASYT option does not resolve the problem, make manual adjustment to the Easytrieve Plus 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.

Packed unsigned 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.

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.

To preserve compatibility with previous releases, Migration Utility provides for the following EASYTRAN/EZPARAMS options:

OCCURS1=0
: Accepts OCCURS as is (compatible with Easytrieve Plus)

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 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

Migration Utility flags referenced file fields of unused files within the same job in compliance with Easytrieve Plus.

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.

Reports are normally printed to SYSPRINT by Easytrieve Plus when no PRINTER is specified on the REPORT statements. Make sure that you specify the correct PRINTER= option in the EZPARAMS/EASYTRAN. When a printer file is not specified, Migration Utility writes to a file as specified by the PRINTER= option. For example, when PRINTER= SYSPRINT is in effect, the report is written to SYSPRINT. When PRINTER= AUTOGEN is in effect, an internal printer file is generated. When PRINTER=REPORT0 is in effect, the report is written to REPORT0 file.

Note: Migration Utility Version 1 always defaulted to PRINTER=AUTOGEN. There was no override.

PRINTER=SYSPRINT is required when using the Automated Parallel testing utilities.

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.

VSE operating system issues

Migration Utility can generate programs that can be compiled and run on a VSE operating system. Programs must be generated to run in static I/O mode, as dynamic I/O modules are not available for VSE.

To generate programs for a VSE system, in the EZPARAMS on the last statement, change `OPSYS=MVS;` to `OPSYS=VSE;`. This causes Migration Utility to generate file SELECT statements compatible with VSE.

Note that the SYS numbers coded on the FILE statements in Easytrieve programs are bypassed by Migration Utility. The SYS numbers can be removed, should they cause translator errors.

As there is no runtime library for a VSE system, you must generate programs to run in static I/O mode. Also, you must generate programs that do not use any dynamically loaded Migration Utility subroutines. See "Generating standalone COBOL."

Generating standalone COBOL

To generate standalone COBOL programs that do not use the Migration Utility's runtime load library, do the following:

1. Run Migration Utility translator with the following EZPARAMS options:

   ```
   IOMODE=NODYNAM
   DYNALLOC=NO
   TBMEMORY=STATIC
   SQLMODE=BIND
   PRINTER=AUTOGEN
   ```
2. Make sure that file definitions fully define and specify the correct record length.
3. Make sure that VSAM file keys are properly defined.
4. Depending on your program, you may need to obtain source code from IBM for one or more of the following programs:

   FSABEC16
   Abend handler

   EZTPX01
   PARM extractor

   FSDATEZ0
   Date routines interface

   FSDATSRV
   Date service routines

   FSLOPER0
   Exclusive OR support

   FSLOPER1
   Exclusive AND support

   FSVLNT00
   Variable-length file support

   FSVLNT03
   Variable-length file support

FSVLNT90
: Variable-length file support

FSYGPCB0
: DLI PCB resolver

FSYGDBD0
: DLI SSA resolver

The source for the above programs can be obtained by contacting the IBM Support Center.

Incompatible field masks

Easytrieve Plus allows Z's after the decimal point. COBOL does not.

Migration Utility adjusts all masks that contain Z's after the decimal point to 9's. For example, MASK ('ZZZ,ZZZ.ZZ-') is changed to MASK ('ZZZ,ZZZ.99-').

To print spaces when a field value is zero, use the BWZ mask option. For example, MASK ('ZZZ,ZZZ.99-' BWZ).

Also, you can use FORCEBWZ=YES EZPARAMS/EASYTRAN option to generate BWZ automatically.

Note that in general, Z's are changed by Migration Utility to 9's when coded out of sequence to prevent COBOL errors.

For example, MASK ('ZZZ,999,ZZ9') would be changed by Migration Utility to MASK ('ZZZ,999,999').

Page overflow considerations

When DISPLAY statements are used in report exits, the page over flow detected by Migration Utility could potentially be different from the page overflow detected by Easytrieve Plus. The symptom is usually missing titles, i.e.,overflow never takes place. To combat this situation, use REXOVCK macro as described in “Report exits” on page 111.

IF statement considerations

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.

Example:

```
FIELDA   W     4  A  VALUE '1234'
FIELDB   W     6  A  VALUE '123456'

IF FIELDA = FIELDB
    DISPLAY 'FIELDS MATCH'
END-IF
```

In Easytrieve Plus, the IF statement results in a true outcome while in COBOL it is false as “1234” is not equal to “123456”.

IMU issues an MNOTE message to alert the user of such potential problem.

If the output produced by IMU does not match the output produced by Easytrieve Plus, check the IMU translator listing (SYSLIST1) for potential MNOTEs and change program to comply with the COBOL rules.

JOB I/O statistics differences

- The I/O count produced by IMU shows the number of File I/O requests. The I/O count produced by IMU should not be taken as the number of successfully processed records.
- The I/O count produced by Easytrieve Plus shows the number of successfully retrieved/processed records and it does not reflect correctly the number of attempted I/O operations.
- The IMU I/O statistics do not always match the counts produced by Easytrieve Plus. This will be especially visible when a JOB flow ends prematurely with a STOP, or for VSAM files that retrieve records in random mode.

Example 1: If a STOP statement is encountered, the I/O count for a JOB file shows one extra record read on IMU statistics. If the job reaches EOF, the I/O counts match.

Example 2: If random VSAM records are retrieved 1000 times, and only 50 records are found, IMU show 1000 in the I/O count, while Easytrieve Plus shows only 50.

Example 3: Report print control characters are combined differently by IMU, thus the I/O count for report files might be different.

Example 4: When doing a Synchronized file JOB, records are read in advance. IMU's logic is different from that of Easytrieve Plus, thus the I/O count may be different.

The number of successfully processed records should be counted by the program.

SORT record sequence considerations

IMU generates standard COBOL SORT statements from the SORT statements found in Easytrieve Plus programs. The SORT in use is the default SORT used by your system.

When sort records contain identical keys, the sequence of such records do not always sort the same way as with the Easytrieve Plus sort.

Use the EQUALS sort option to make sure that records are sorted in the correct sequence. If EQUALS is not your installation default, add this to your application program JCL:

```
//DFSPARM DD *
  OPTION EQUALS
/*
```

File naming (DDname) restrictions

The file names defined in the Easytrieve Plus programs might conflict with the file names used by the translator.

The conflict exists for Link and Go programs only.

Use the DDFILTER=YES EZPARAMS option to filter out the conflicting DD names.

The FSYDDTAB located in the SYS1.SFSYCCLC library contains a complete list of conflicting DD names. The table can be modified to accommodate individual requirements. The following names are distributed in FSYDDTAB:

```
CEEOPTS     FJSYSER2     SYSLMOD
DBRMLIB     FJSYSIN      SYSPUNCH
FJBATPH     FJSYSJC      SYSTERM
FJBIND0     FJSYSP0      SYSTLIST
FJBLUEP     FJSYSPH      SYSUT1
FJCARD0     FJSYSPW      SYSUT2
FJCCLLB     HEXLIST      SYSUT3
FJCPYLB     SYSCIN       SYSUT4
FJMACLB     SYSLIB       SYSUT5
FJNAMES     SYSLIN       SYSUT6
FJSYS01     SYSLIST1     SYSUT7
FJSYSER1    SYSLIST2
```

Packed unsigned fields

Chapter 4. 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
DLI	IMS/DLI
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
IDMS	IDMS 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

IS	ISAM Files are flagged

Non-supported file attributes (these attributes are bypassed)

ASA	Option is ignored

WORKAREA
This option is supported for the EXIT option only. Otherwise it is ignored.

EXTENDED
Option is not supported

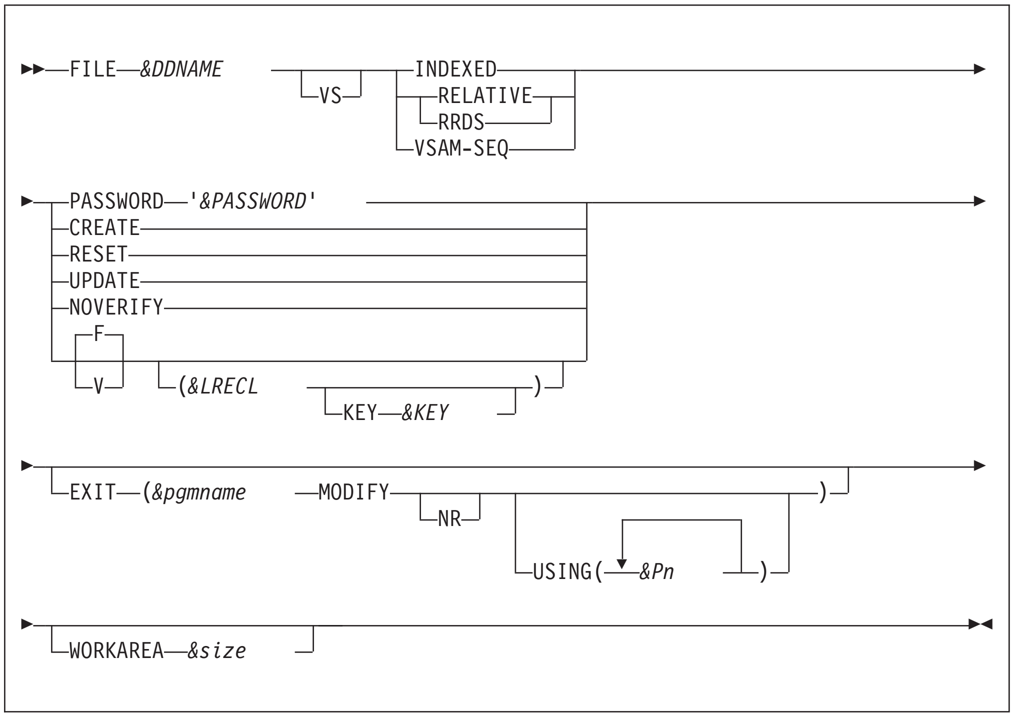
DBSCODE
Option is not supported

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 1- to 8-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.

EXIT Identifies the exit to be taken before and after file I/O. Exit allows you to prescreen or modify input and output records. Refer to "File I/O Exits" on page 189 for details of I/O exits. EXIT is supported for VSAM and sequential files only. Easytrieve Plus does not support EXIT for DLI/IMS, IDMS, or DB2.

&pgmname
: The exit program to be invoked.

MODIFY
: This is a required parameter for VSAM files. When MODIFY is specified, I/O operations are performed by the generated COBOL. The exit program is called after the input operations such as GET and READ, and before the output operations such as PUT or WRITE.

NR This is a VSE-related parameter. It is ignored by Migration Utility.

USING
: Identifies additional fields (parameters) to be passed to the exit program. &P1 ... &Pn are the field names to be used.

Important notes:

1. Your exit program receives at least two parameters. The first parameter points to file record, the second parameter points to a WORKAREA, followed by additional parameters as specified by the USING option (if any). WORKAREA is required for the MODIFY option.
2. The file record is the input file record. You must move this file record to the WORKAREA in your exit program. You can modify the

WORKAREA as needed. Upon return from your exit, the WORKAREA is moved to the file record and made available to your generated COBOL program.

3. Files with EXIT are always generated with IOMODE=DYNAM.

WORKAREA
Identifies the amount of storage to allocate for a work buffer. This parameter is optional. It is used only for files with the EXIT option, otherwise it is ignored.

&size is the number of bytes to allocate. The maximum is 32768.

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 compatibility with Easytrieve, 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 embedded 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.

```
►►──FILE──&DDNAME    ──TABLE──┬─INSTREAM─────────────────────┬──&ARG     ─────►
                              └┤ External table parameters ├─┘

►─&POS    ──&LENGTH    ──&TYPE    ──&DESC    ──&POS    ──&LENGTH    ─────────►

►─&TYPE    ──┬──────────────────────────┬──────────────────────────────────►◄
             └┤ Instream table values ├─┘
```

Instream table values:

```
   ┌──────────────────────────┐
   ▼                          │
├────&ARGn    ──&DESCn    ────┴──ENDTABLE────────────────────────────────────┤
```

External table coding:

```
├──(──&ROWS    ──)──┬─────┬──┬──────────────────────────────┬──────────────┤
                    ├─F───┤  └─(──&LRECL    ──&BLKSIZE    ──)┘
                    ├─V───┤
                    ├─U───┤
                    ├─FB──┤
                    ├─VBS─┤
                    └─VB──┘
```

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 defines 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:

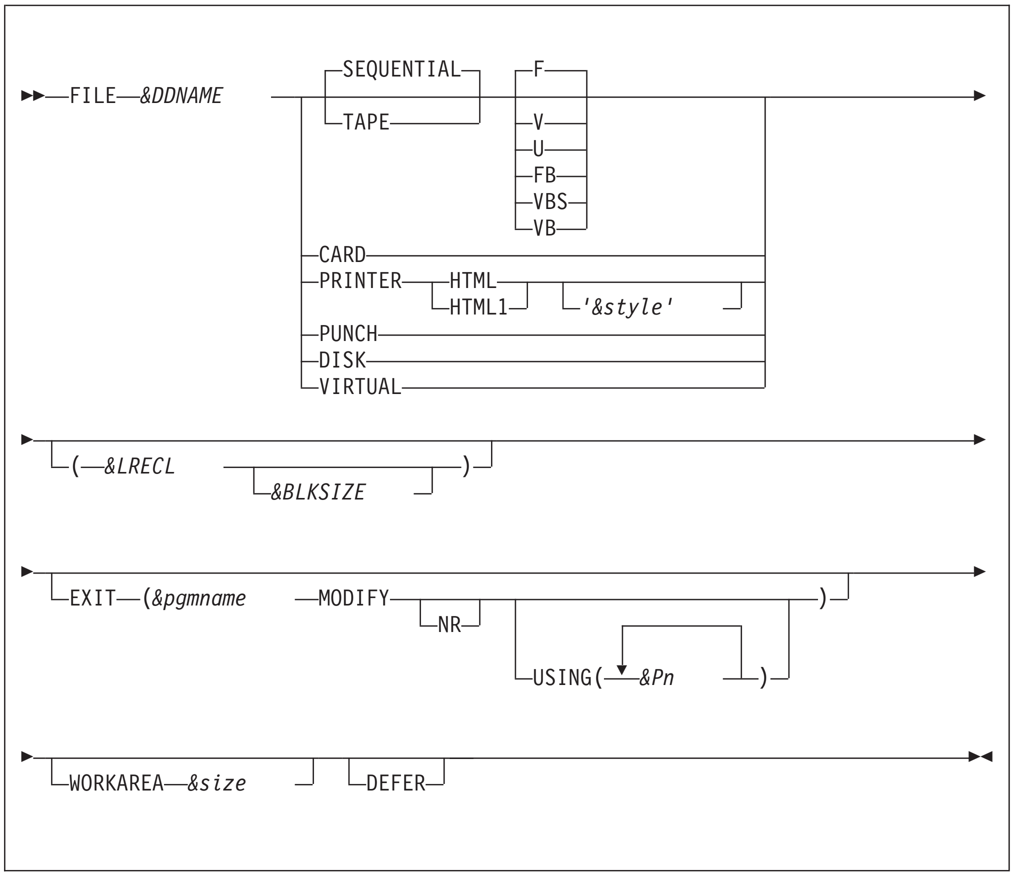
```
FILE BRTABLE TABLE (150) V(64)
             BRANCH         1 2  A
             DESCRIPTION    4 15 A
```

In the following example, the DESC field is a signed numeric field with two decimal places. To code a negative value for the DESC field, the last character of the DESC field can be coded as a character "}" through "R", such as 1234J. Migration Utility converts such characters to a minus sign and inserts a decimal point into the value if applicable.

```
FILE TABLE01 TABLE INSTREAM
ARG  1 4   N
DESC 6 5   N 2
1234 12345
1235 1234J
1236 12347
ENDTABLE
```

The 1234J is converted to a negative decimal literal as VALUE -123.41.

Defining unit record devices and sequential files



Parameters

&DDNAME
: One to eight character file name. This must be a valid file name.

  The DD name cannot be one of ddnames used by IMU/COBOL/Linker and SQL translator. When DDFILTER=YES is specified in the EZPARAMS, the name is validated against the FSYDDTAB located in SYS1.SFSYCCLC library.

Device:

CARD
: IMU detects the END statement in the Easytrieve Plus program source and creates a temporary file of records that follow the "END" statement. The CARD file is allocated to this temporary file if it was defined in the Easytrieve Plus program.

PRINTER
: Printer

  HTML
  : Produce HTML report

  HTML1
  : Produce HTML report with a tree of control breaks on the left side for easy selection

('&style')

Optional HTML CSS expression for fonts and decorating of report body. Example:

```
FILE REPORT1 PRINTER HTML1 +
     '</style> body {color: #00ff00; background-color: +
       black; font-size: 100%;} </style>'
```

When HTML document is printed to a disk file, the disk file must be downloaded to a PC or a UNIX platformserver in TEXT format as HTML file type. There is no parsing needed. The file then can be viewed via a browser.

When HTML1 document is printed to a disk file, Migration Utility automatically converts it to ASCII file ready te be downloaded to a PC or a UNIX platform server. The file must be downloaded in BINARY format and parsed on the PC/Server via the fsybpars provided java class. For parsing instructions, refer to “Running the HTML1 document parser - fsybpars” on page 170.

Note: HTML reports can be published direct to z/OS server for browsing. You do this by changing JCL as described in the "HFS (Unix file) requirements" section in Chapter 9.

Note: HTML and HTML1 are written unconditionally as variable length files with LRECL of 4096. For example, 1 byte for print control, 4091 for text and 4 bytes for variable length. The report data is mapped over the actual LRECL specified on PRINTER file definition, or LINESIZE stated on the REPORT statement.

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 for output files. Must be a multiple of record size. Add 4 for variable or spanned records. Block size for input files is not needed but it is accepted if supplied.

The maximum value is 32760. For devices that support large block interface (LBI), the maximum value is as per device limit.

Handling of this parameter is controlled by the BLKSIZE= in the EZPARAMS table as follows:

BLKSIZE=NATIVE

The block size is ignored by Migration Utility. Zero is used instead.

BLKSIZE=EASYT

The value is used as supplied.

Zero can be coded to use the blocksize from the JCL or the operating system default. When the value is zero, the block size is allocated as per DCB in the JCL if coded, otherwise it is the default value allocated by the operating system.

When the &LRECL and &BLKSIZE are not provided on file definition, allocation is:

- For disk files, as per DCB in the JCL if coded, otherwise the system default is used.
- For tape files, the &LRECL is from the DCB in the JCL and block size is as per system default (that is, the BLCKSIZE= in the JCL is ignored).

Note: Use zero for block size to get best performance. Inefficiently blocked files consume more I/O and cause longer running times.

EXIT Identifies the exit to be taken for file I/O. EXIT for sequential files allows you to:

1. Do file I/O in the exit program. This method is generated when the MODIFY option is not specified.
2. Do file I/O by the generated COBOL, and prescreen or modify input and output records. This method is generated when the MODIFY option is specified.

Refer to “File I/O Exits” on page 189 for details of I/O exits. EXIT is supported for VSAM, sequential and PRINTER files. Easytrieve Plus does not support EXIT for DLI/IMS, IDMS, or DB2.

&pgmname

The exit program to be invoked.

The default exit program name for PRINTER files is FSYPXIT1. For all other files, &pgmname is required following the EXIT identifier.

MODIFY

This parameter is optional for sequential files. It is a required parameter for VSAM files.

When MODIFY is specified, I/O operations are performed by the generated COBOL. The exit program is called after input operations such as GET and READ, and before output operations such as PUT or WRITE.

When MODIFY is not specified, your exit program is responsible for I/O operations, including file open and close.

Migration Utility does not support the MODIFY option for PRINTER files.

NR This is a VSE-related parameter. It is ignored by Migration Utility.

USING
Identifies additional fields (parameters) to be passed to the exit program. &P1 ... &Pn are the field names to be used.

Important notes:

1. When MODIFY is specified, your exit program receives at least two parameters. The first parameter points to the file record. The second parameter points to a WORKAREA, followed by additional parameters as specified by the USING option (if any).

   WORKAREA is required for the MODIFY option. The file record is the input file record. You must move this file record to the WORKAREA in your exit program. You can modify the WORKAREA as needed. Upon return from your exit, the WORKAREA is moved to the file record and is made available to your generated COBOL program.
2. When MODIFY is not specified, your exit program receives at least two parameters. The first parameter points to file record. The second parameter points to a BL4 I/O request code, followed by additional parameters as specified by the USING option (if any). Refer to "File I/O Exits" on page 189 for more details.
3. Files with the EXIT and MODIFY options are always generated with IOMODE=DYNAM.

WORKAREA
Identifies the amount of storage to allocate for a work buffer. This parameter is optional. It is used only for files with the EXIT option, otherwise it is ignored.

&size is the number of bytes to allocate. The maximum is 32768.

| **DEFER**
| Use this parameter to defer file open until it is accessed for input or
| output. Without this option, file is opened when JOB is initiated.

Note: Each file definition can be optionally followed by the record layout. For additional information refer to "Defining Records and Working Storage" on page 62.

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

```
►►──┬────────┬──┬─────────┬──&FIELD    ──┬─&POS ──────────────────────┬──────────►
    └─DEFINE─┘  └─&FILE:  ┘              └─ * ──┬──────────────────┬──┘
                                                └─( + &OFFSET1)  ──┘

►──┬─W─┬──┬─────────────┬──&OVERLAY    ──┬────────────────┬──────────────────────►
   └─S─┘  └─&FILE    ─:─┘                └─ + &OFFSET2  ──┘

►─&LENGTH    ──┬─A──────┬──&NDEC    ──┬──────┬──────────────────────────────────►
               ├─COMP-1─┤             └─EVEN─┘
               ├─COMP-2─┤
               ├─N──────┤
               ├─B──────┤
               ├─P──────┤
               ├─K──────┤
               └─U──────┘

►──┤ Heading information ├──┬──────────────────────────┬────────────────────────►
                            └─INDEX──(──&INDEX    ──)──┘

►──┬────────────────────────────────────────────────────────────────┬───────────►
   └─MASK──(──┬───────────┬──┬─────┬──'&MASK'    ──┬─────┬──)───────┘
              └─&MASKID  ─┘  └─BWZ─┘               └─HEX─┘

►──┬──────────────────┬──┬────────────────────────────┬──┬───────┬──────────────►
   └─OCCURS──&OCCUR  ─┘  └─VALUE──┬─────┬──&VALUE   ──┘  └─RESET─┘
                                  └─ALL─┘

►──┬─────────┬──────────────────────────────────────────────────────────────────►◄
   └─VARYING─┘
```

Heading information:

```
├──HEADING──(──'&HEAD1'   ──┬──────────────────────────────────┬──)────────────►
                            └─'&HEAD2'   ──┬─────────────────┬─┘
                                           └─'&HEAD3'      ──┘

►──┬─────────────────────────────────────────────────────────────────────────┬──┤
   └─(──FONT#──'&HEAD1'   ──┬──────────────────────────────────┬──)───────────┘
                            └─'&HEAD2'   ──┬─────────────────┬─┘
                                           └─'&HEAD3'      ──┘
```

Parameters

DEFINE
This keyword denotes the beginning of a field. It is typically 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-byte 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 quotation marks. 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.

  The behavior of this option depends on the RESET=EASYT/NATIVE EASYTRAN/EZPARAMS parameter:

  RESET=NATIVE
  : Resets fields the first time through the job logic.

  RESET=EASYT
  : Resets fields every time the JOB cycle is entered.

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(OPT) option will increase the maximum value of 2-byte and 4-byte fields to their maximum capacity. For the exact values, refer to the IBM COBOL Compiler manual for your operating system.

```
    Memory          Easytrieve             COBOL
      Size            Max-Value            Max-Value

     4 bytes       2,147,483,647+        999,999,999+
                   2,147,483,647-        999,999,999-

     3 bytes           8,388,607+          9,999,999+
                       8,388,607-                n/a

     2 bytes              32,767+              9,999+
                          32,767-              9,999-

     1 byte                  127+                 99+
                             127-                n/a
```

COBOL does not support 1-byte and 3-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 5. 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.

```
►►──COPY──&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 supports a maximum of 260 COPY statements for each program. It alters the field naming conventions of the newly created file by prefixing each field with a letter assigned to the file being defined.

The COPY statement results in duplicate field names. A file qualifier is required when accessing fields in files defined with a COPY statement.

Examples

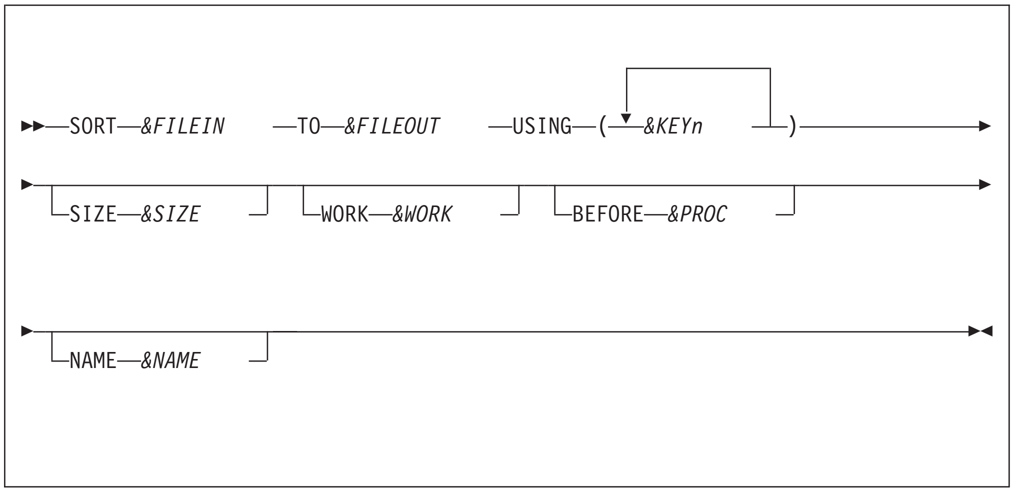
The examples show some variations of possible field definitions.

```
FILE FILEIN1 DISK (107)
   IN-ACCOUNT     01  10 N     MASK '99-99999999'   +                  |
                               HEADING ('ACCOUNT' 'NUMBER')            |    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')     |

FILE FILEIN2 DISK (107)                                                     FILEIN2
 COPY FILEIN1                                                                 COPY
                                                                            FILEIN1
```

```
JOB INPUT FILEIN1
                                                                   |
GET FILEIN2 STATUS                                                 |
IF EOF FILEIN2                                                     | Some
   STOP                                                            |
END-IF                                                             | References
                                                                   |
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)

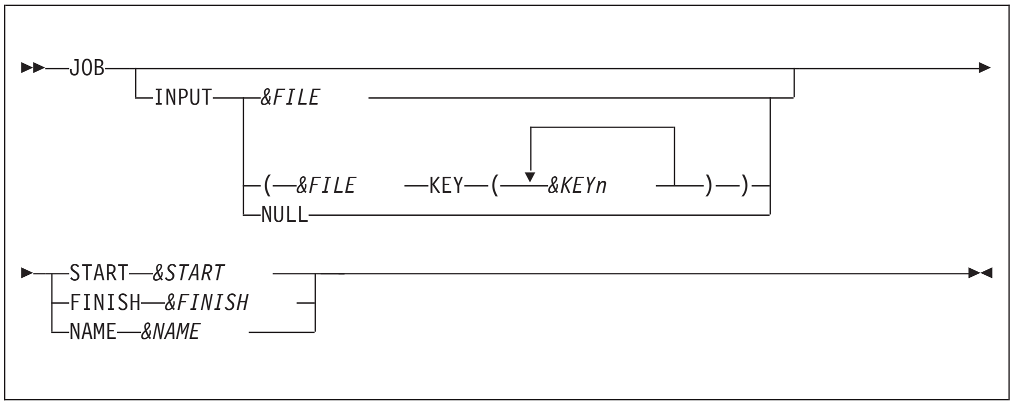
Example

This example shows sorting input file FILEIN to output file FILEOUT. 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 FILEOUT  (80)                               |
   OCUS-ACCT      01  15 N                       |
   OCUS-NAME      16  15 A                       |
   OCUS-ADDRESS1  32  15 A                       | Output file


SORT FILEIN TO FILEOUT  +
     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 71.

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 often 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 typically 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 72).

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.

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.

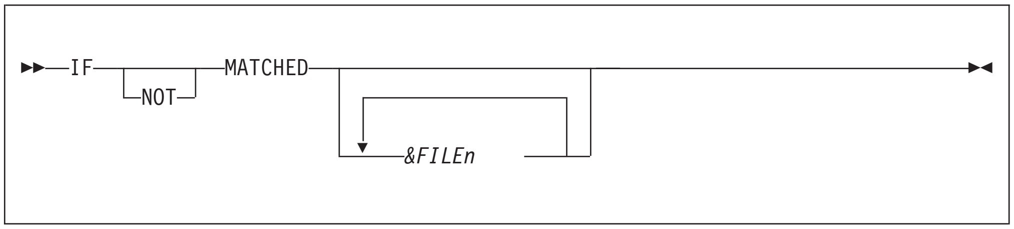
In JOB Cycle #5, the second CCCC record of FILE1 is available only. A FILE2 and FILE3 records are not available at all.

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.



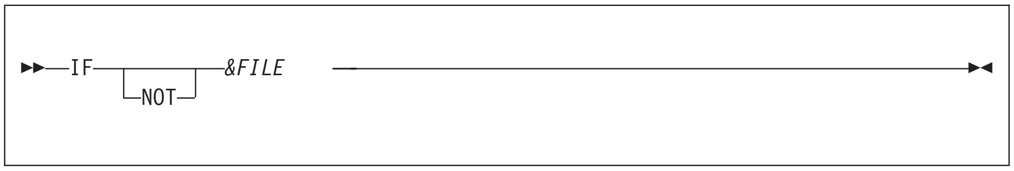
Parameter

&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.



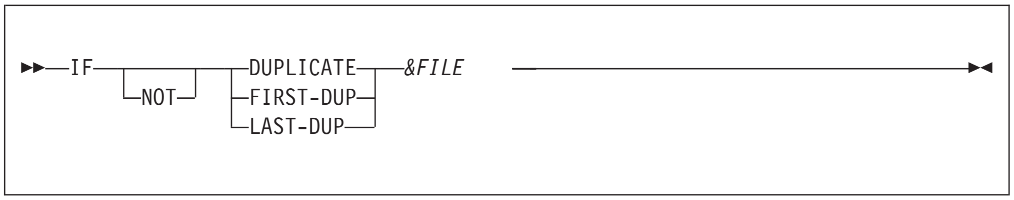
Parameter

&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 processing.

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.

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

Special IF statements in synchronized process

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                             |
                                                         |
```

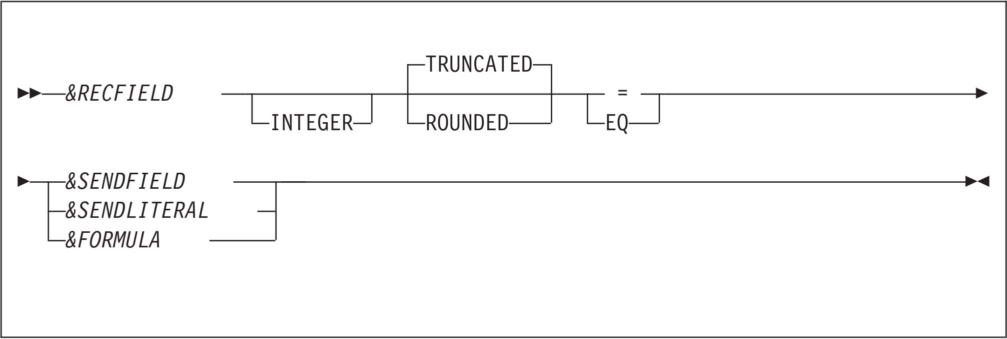
Assignment statement

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 quotation marks.

&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 field's data type. That is, numeric fields require a numeric source and alphanumeric fields require an alphanumeric source. Alphanumeric literals must be enclosed in quotation marks. Numeric literals 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

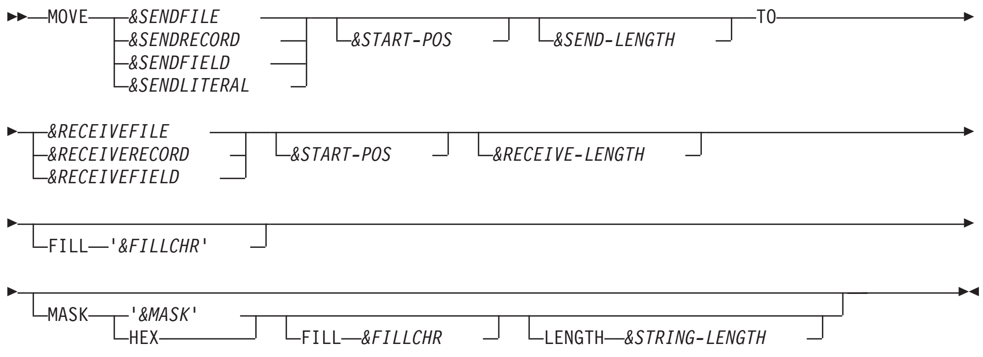
The MOVE statement is supported by Migration Utility in two ways. The type of move is controlled via the EASYTRAN/EZPARAMS MOVENUM= option.

1. When MOVENUM=NATIVE is in effect, the MOVE is generated to use native COBOL rules.
2. When MOVENUM=EASYT is in effect, the MOVE is generated according to Easytrieve Plus rules.

The COBOL MOVE statement does not directly correspond to the Easytrieve Plus MOVE. The COBOL MOVE behaves like the Easytrieve Plus ASSIGN statement. That is, the field types are considered and converted during the move, while the Easytrieve Plus MOVE statement moves data as is (without conversion).

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 values:

&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 literals must be enclosed in quotation marks.

&START-POS
: Start position within the sending field. This is a special Migration Utility feature not supported by Easytrieve Plus.

&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 values:

&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

&START-POS
: Start position within the receiving field. This is a special Migration Utility feature not supported by Easytrieve Plus.

&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. *&FILLCHR* can be a field name or a 1-character literal enclosed in quotes.

&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:

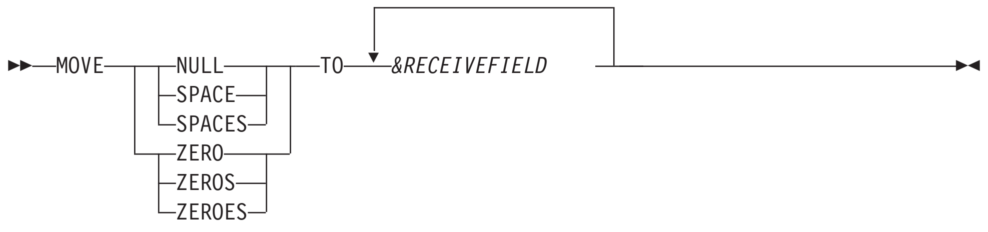
FIELD-2
-123

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.

When MOVENUM=NATIVE is in effect, 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.

When MOVENUM=EASYT is in effect, Migration Utility generates an internal ASSIGN statement for each elementary field. This method is compatible with Easytrieve Plus.

When MOVENUM=NATIVE is in effect, Migration Utility generates a standard COBOL move. This method may exhibit different behavior than Easytrieve Plus when a 1-byte or 3-byte binary field or a packed unsigned field is moved into an incompatible field type. This method is compatible with Migration Utility Version 1.

```
►►──MOVE──LIKE──┬─&SENDFILE───┬──TO──┬─&RECEIVEFILE───┬──────────────►◄
                └─&SENDRECORD─┘      └─&RECEIVERECORD─┘
```

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.

```
►►──PUT──&OUTFILE──┬─FROM──&FILE─┬──┬────────┬─────────────────►◄
                   └─FROM──&AREA─┘  └─STATUS─┘
```

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.

```
►►──WRITE──&OUTFILE    ──┬─┬─UPDATE─┬──┬─FROM──&FILE  ─┬─┬──┬────────┬──────────►◄
                         │ └─ADD────┘  └─FROM──&AREA  ─┘ │  └─STATUS─┘
                         └─DELETE────────────────────────┘
```

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 default is UPDATE.

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.

```
►►──GET──&FILE :   ──┬───────┬──┬─HOLD───┬──┬────────┬──────────────────────►◄
                     └─PRIOR─┘  └─NOHOLD─┘  └─STATUS─┘
```

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.

```
►►──READ──&FILE    ──KEY──┬─&FIELD─────┬──┬─HOLD───┬──┬────────┬──────────►◄
                          └─'&LITERAL'─┘  └─NOHOLD─┘  └─STATUS─┘
```

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.

```
►►──POINT──&FILE ──┬───────┬──┬─EQ─┬──┬─&FIELD─────┬──┬────────┬────────►◄
                   └─PRIOR─┘  ├─=──┤  └─'&LITERAL'─┘  └─STATUS─┘
                              ├─GE─┤
                              ├─GQ─┤
                              └─>=─┘
```

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 81 for 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.

```
►►──SEARCH──&TBNAME  ──WITH──&SEARCHARG  ──GIVING──&RESULT  ──────────►◄
```

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.

```
►►──PERFORM──&PROCNAME  ──────────────────────────────────────────────►◄
```

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.

```
►►──DISPLAY──┬─&FILE────┬──┬─┬─TITLE───┬──────────────┬──────────────►
             └─SYSPRINT─┘  │ └─NOTITLE─┘              │
                           ├─SKIP──&SKIP──────────────┤
                           └─CONTROL──&POSITION───────┘

►──┬─┤ Position information ├─┬─────────────────────────────────────►◄
   └─HEX──┬─&SRCFILE─┬────────┘
          ├─&FIELD───┤
          └─&RECORD──┘
```

Position information:

```
├──┬─&FIELD───┬──┬─+──OFFSET─┬──┬──────────────┬────────────────────►
   └─&LITERAL─┘  └─-──OFFSET─┘  └─COL──&COLUMN─┘

►──┬────────────────┬───────────────────────────────────────────────┤
   └─POS──&POSITION─┘
```

Parameters

&FILE A sequential file name. Use a file name if you want 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 want 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 want 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 quotation marks.

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 allow it, 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.

```
►►──CALL──&PROGRAM ──────────────────────────────────────────────────►

►─┬──────────────────────────────────────────────────────────────┬──►◄
  │              ┌──────────────────┐                             │
  │              ▼                  │                             │
  └─USING──(──┬─&FIELD ──────────┬─┴─┬───────────────────────┬─)─┘
              └─'──&LITERAL ──'─┘   └─RETURNS──&RCODE ───────┘
```

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 quotation marks.
- Generates a dynamic call when the program name is not enclosed in quotation marks.

RETURN-CODE considerations

Migration Utility considers RETURN-CODE after each CALL according to the standard calling conventions. This normally results in a good run but potentially a bad return code can be returned at End of Job if a called sub-program does not set the return code properly.

Easytrieve Plus does not save RETURN-CODE unless specifically requested on the CALL statement. Use the RETURNC=EASYT EASYTRAN/EASYPARAMS option to make RETURN-CODE handling compatible with Easytrieve Plus.

Examples

```
CALL  FSABE01


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

GOTO statement

The GOTO statement alters the flow of processing.

```
►►─┬─GOTO──┬──┬─&LABEL─┬─────────────────────────────────────────────────►◄
   └─GO TO─┘  └─JOB────┘
```

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.

```
►►──STOP──┬─────────┬────────────────────────────────────────────────────►◄
          └─EXECUTE─┘
```

Parameter

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

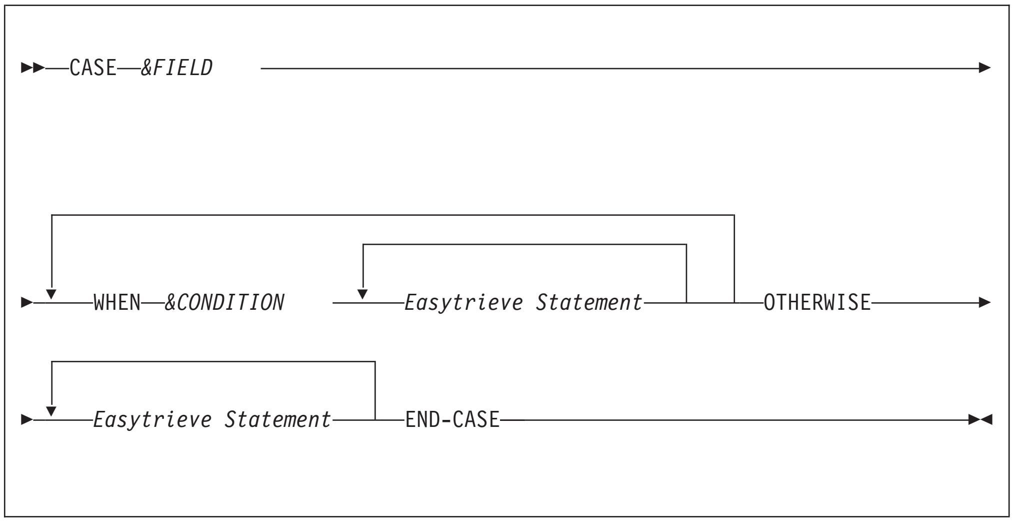
STOP without EXECUTE terminates only current job activity. Any other jobs subsequent to the current one continue processing.

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.

```
►►──DO───┬─WHILE─┬──&CONDITION    ──Easytrieve statements──END-DO──────────────►◄
         └─UNTIL─┘
```

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 91 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 executed again. When the conditional expression is true, the statements are executed again. When the conditional expression is false, the processing continues with the next statement following the END-DO.

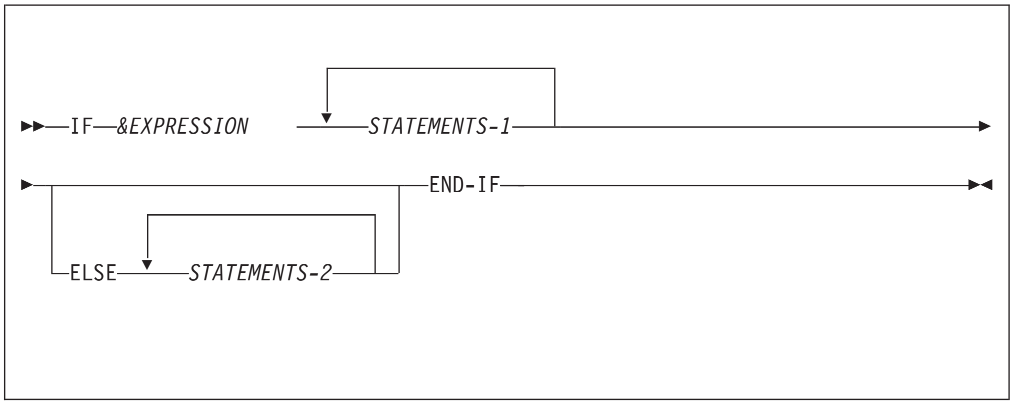
Example

```
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")

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

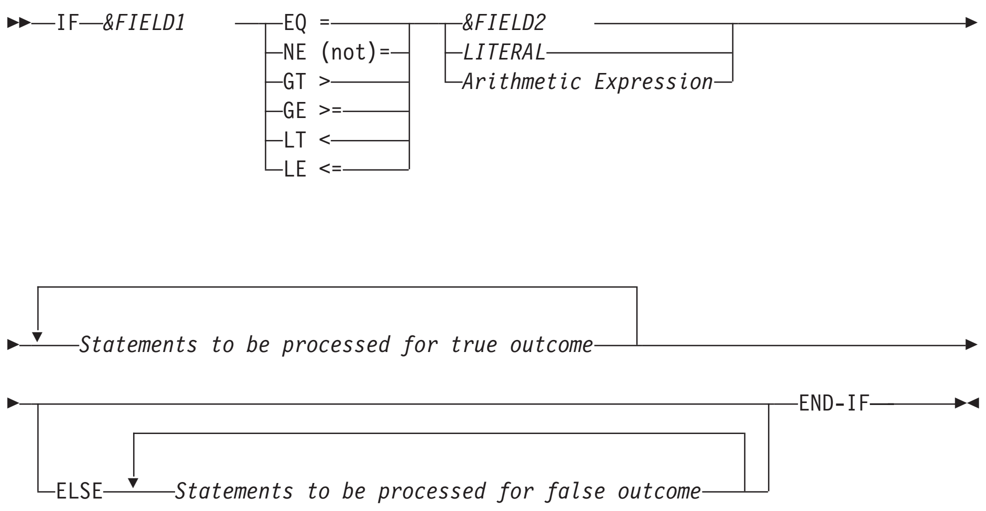
Conditional expressions are used in combination with the IF and DO statements to manipulate and select data in the Job Activity section.

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.

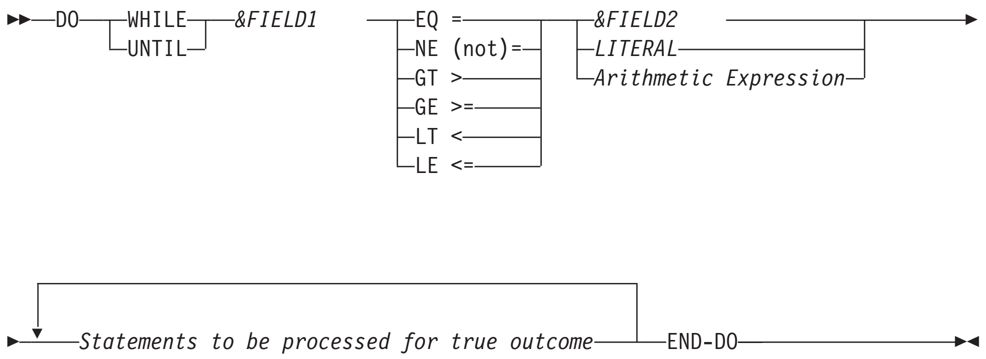
When a DO statement is present, all statements following the DO statement are processed, based on the truth of the conditional expression, until and END-DO statement is encountered.

Conditional expressions

The IF statement syntax


IF
&FIELD1
EQ =
NE (not)=
GT >
GE >=
LT <
LE <=
&FIELD2
LITERAL
Arithmetic Expression
Statements to be processed for true outcome
ELSE
Statements to be processed for false outcome
END-IF

The DO statement syntax



Parameters

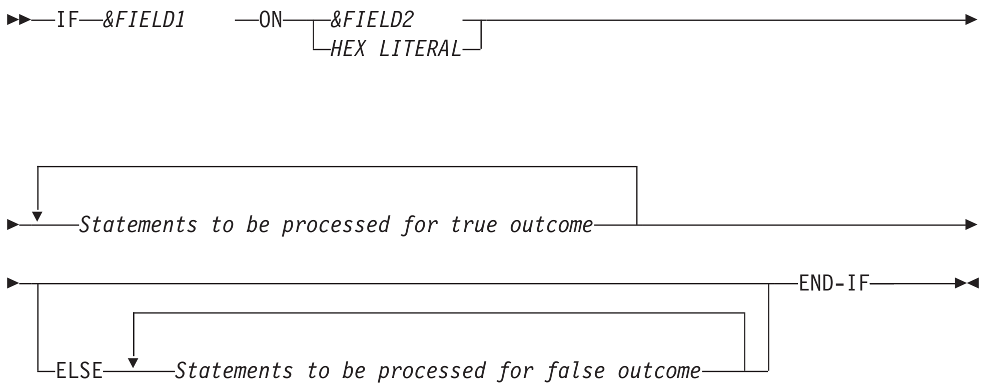
&FIELD1
: A field name used as argument 1 in comparison

&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


IF
&FIELD1
ON
&FIELD2
HEX LITERAL
Statements to be processed for true outcome
END-IF
ELSE
Statements to be processed for false outcome

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)
```

Conditional expressions

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.

Example:

```
FIELDA   W     4  A  VALUE '1234'
FIELDB   W     6  A  VALUE '123456'

IF FIELDA = FIELDB
    DISPLAY 'FIELDS MATCH'
END-IF
```

In Easytrieve Plus, the IF statement results in a true outcome while in COBOL it is false as “1234” is not equal to “123456”.

IMU issues an MNOTE message to alert the user of such potential problem.

If the output produced by IMU does not match the output produced by Easytrieve Plus, check IMU translator listing (SYSLIST1) for potential MNOTEs and change program to comply with the COBOL rules.

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.

```
►►──PRINT──┬──────────┬──────────────────────────────────────────►◄
           └─&REPORT──┘
```

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 typically 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.

```
                         ┌─────────────────┐
                         ▼                 │
►►──&PROCNAME.    ──PROC────STATEMENTS─────┴──END-PROC──────────────────────►◄
```

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 IMS/DLI database records that are input to the JOB activity.

The RETRIEVE statement is described in detail in Chapter 8, "DLI/IMS support," on page 143.

SELECT statement (SORT and REPORT selection)

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

```
►►──SELECT──────────────────────────────────────────────────────────────►◄
```

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.

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)

CHKP-STATUS
A 2-byte alpha field returned by DLI after a CHKP and XRST call. This

field contains spaces for a successful call. Any other value indicates an error. Refer to IMS/DLI Messages and Codes in the IBM IMS/DLI reference for specific values.

DISPLAYSIZE
A 4-byte binary field available in REPORT exits only. It contains the DISPLAYSIZE value, either the default value or the "y" value from the PAGESIZE (x y) coded on the REPORT statement.

Note: This is an Migration Utility option only.

PAGESIZE
A 4-byte binary field available in REPORT exits only. It contains the PAGESIZE value, either the default value or from the PAGESIZE coded on the REPORT statement.

Note: This is an Migration Utility option only.

PATH-ID
2-character path ID for DLI/IMS, available after the RETRIEVE statement. This field contains the last segment ID that was retrieved by the RETRIEVE statement. Note that an optional 2-character ID can be supplied for each segment in the RETRIEVE statement. PATH-ID contains spaces if there is no ID specified for the last retrieved segment in the path.

When using a tickler file to retrieve unique segments, PATH-ID is set to `NF` for the 'record not found' condition, otherwise it contains the ID of the last accessed segment.

***&file*-RECFM**
A 1-byte alpha field that indicates the file record format:

F	Fixed record length
V	Variable record length
U	Undefined record length

&file is the file name defined in your program.

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

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.

File fields (available globally in each activity section)

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 229. 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 2-byte alphanumeric field while in Easytrieve it is a fullword 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 2-digit constant (literal) enclosed in quotation marks. Here are some of the 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)

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

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.

Report fields (available only in report exits)

BREAK-LEVEL
Indicates the break level number

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

&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.

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

LINE-COUNT
A 2-byte binary field that contains the number of lines printed on the current page.

LINE-NUMBER
A 2-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 2-byte binary field that contains the number of the page being printed.

TALLY
A 10-byte packed decimal field that contains the number of detail records in a control break

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.

Easytrieve reserved keywords

Keyword
Description

DUPLICATE
Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 71 for more information.

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.

FIRST-DETAIL
A conditional name available in the BEFORE-DETAIL report exit only. It is set to a true value for the first detail line after control breaks, and when

the current detail line about to be built would cause page overflow. Use the FIRST-DETAIL test to include special information on the first detail line of each page or after control breaks.

The following example prints the contents of WS-DATE on the first detail line of each page and after control breaks:

```
BEFORE-DETAIL. PROC.
   WS-DATE = SPACES
   IF FIRST-DETAIL
       WS-DATE = '02/27/2006'
   END-IF
END-PROC.
```

FIRST-DUP
Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 71 for more information.

HIGH-VALUES
Used to test a field for all hex "FF" or move all hex "FF" to a field.

LAST-DUP
Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 71 for more information.

LOW-VALUES
Used to test a field for all binary zeros or move all binary zeros to a field

MATCHED
Used to test the outcome of synchronized file process. Refer to "Synchronized file processing" on page 71 for more information.

NUMERIC
Used to test data fields for numeric contents

POST-BREAK
A conditional name available in the BEFORE-DETAIL report exit only. It is set to a true value for the first detail line after control breaks. Use the POST-BREAK test to include special information on the first detail line after control breaks.

The following example prints the contents of WS-TAG on the first detail line after a control break:

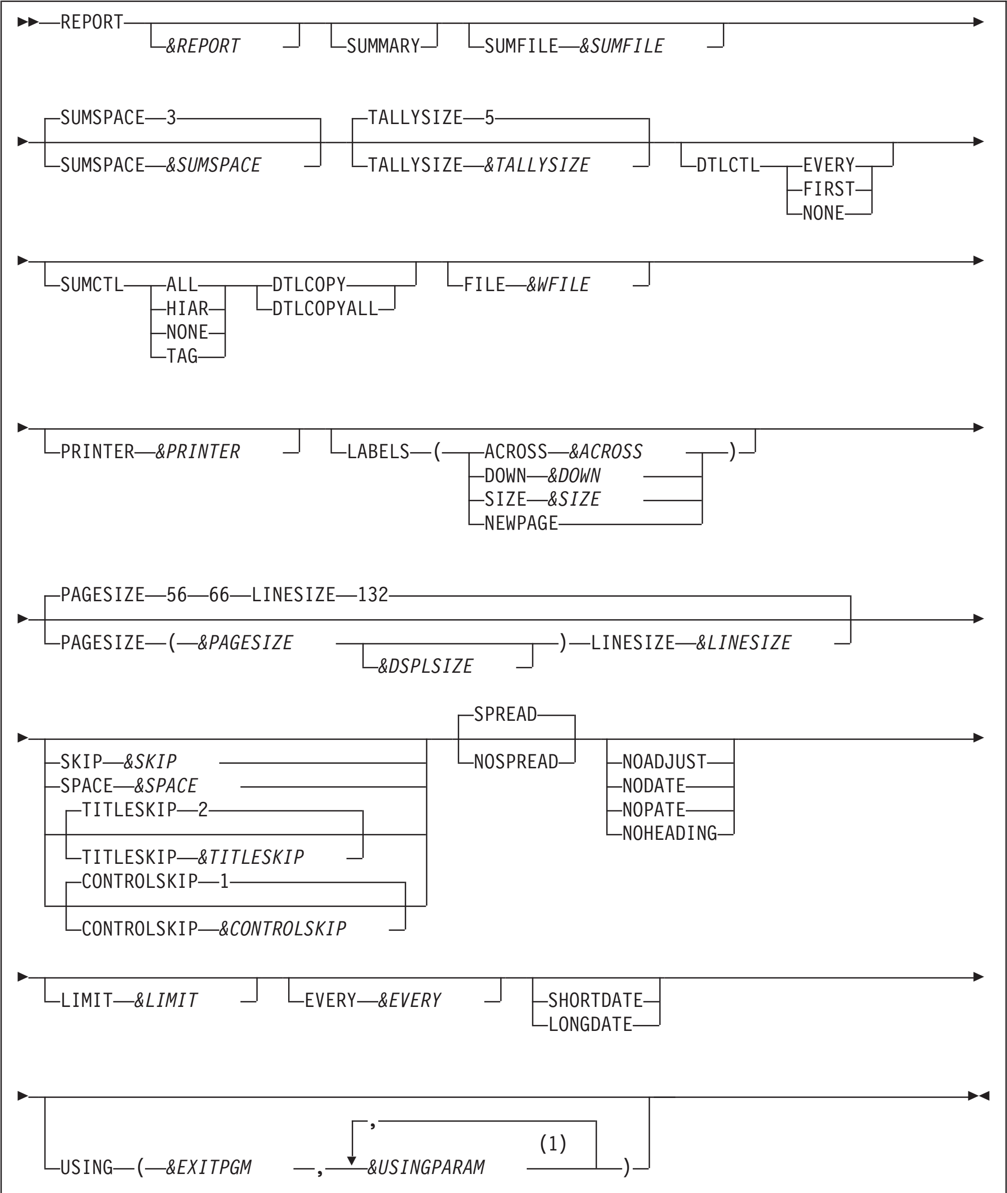
```
BEFORE-DETAIL. PROC.
   WS-TAG = SPACES
   IF POST-BREAK
       WS-TAG = '*after break*'
   END-IF
END-PROC.
```

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.

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, SUM, HEADING, TITLE, and LINE statements as described on the pages that follow.



Notes:

1 1–4 parameters can be passed to the exit program.

Parameters

&REPORT
: Report name. Easytrieve allows a name of up to 128 characters. 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 needed 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

SUMCTL
: 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 detailed information about minor level total lines.

DTLCOPYALL
: Prints detailed 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.

  When a printer file is not specified, the report is written to a file specified by the PRINTER= EASYTRAN/EZPARAMS option. For example, when PRINTER=SYSPRINT is in effect, the report is written to SYSPRINT, when PRINTER=AUTOGEN is in effect, an internal printer file is generated, when PRINTER=REPORT0 is in effect, the report is written to the REPORT0 file.

Note that Migration Utility Version 1 always defaulted to PRINTER=AUTOGEN. There was no override.

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

&DSPLSIZE
Number of lines for each logical page for DISPLAY statements in report exits. If not coded, the default is the &PAGESIZE (if supplied), or as coded in the EASYDTAB default table.

&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.

&EVERY
: Every *&EVERY*th line is to be printed. This is a development option.

SHORTDATE
: The SYSDATE date format is to be used.

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

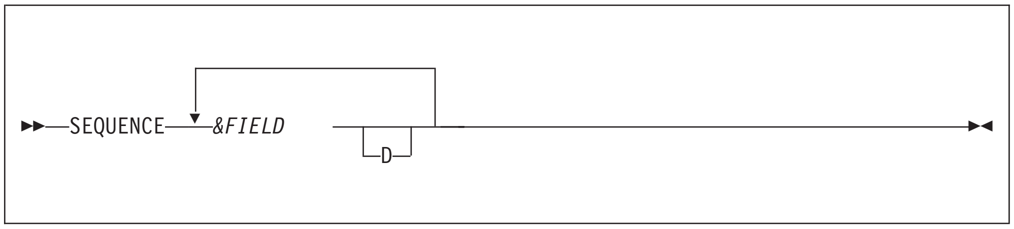
&EXITPGM
: Print I/O exit program name. This program is responsible for performing printer file I/O.

&USINGPARAM
: Parameter to be passed to the exit program. Can be a literal or a field name.

  See "File I/O Exits" on page 189 for non-MODIFY exit program coding conventions.

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.

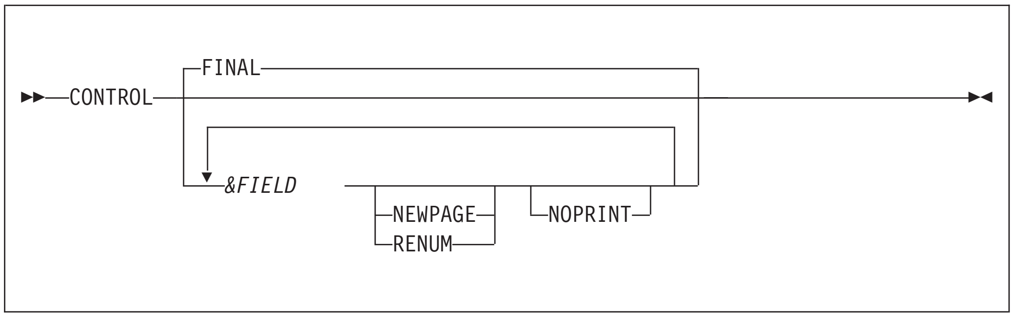
The SEQUENCE statement causes sorting of the report items before printing. The sort adds a substantial amount of processing overhead, therefore the SEQUENCE should be used with care. SEQUENCE is not needed if your input file is in the same order as your reports.

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 a maximum of 64 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.

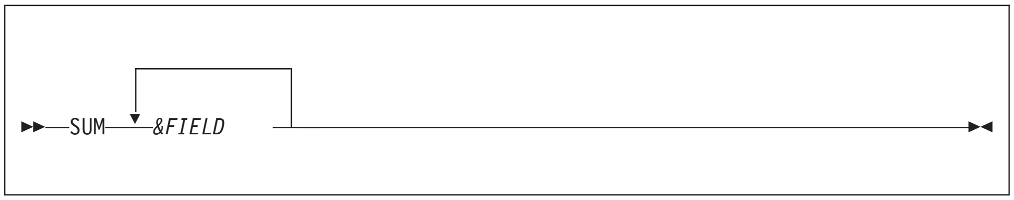
A break level number is assigned to each control break field, with most minor break assigned to level 1, the one before it to 2, and so on. Final control break has the highest level number of N+1, where N is the number of fields specified.

A Level can be tested via the LEVEL keyword in the BEFORE-BREAK and AFTER-BREAK report exits. See “Report exits” on page 111 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

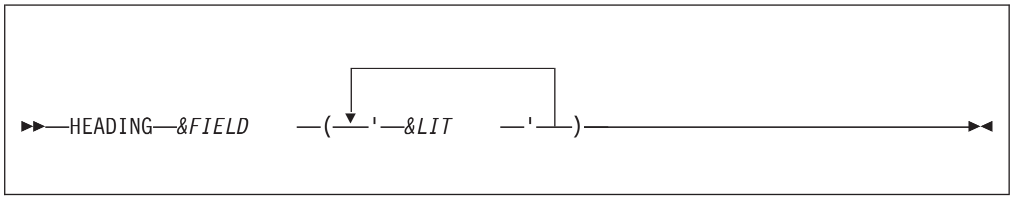
&FIELD
A list of one or more fields to be summed. The fields must be quantitative contained in an active file or working storage.

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 in Easytrieve Plus. Migration Utility allows literal up to 60 characters long (including quotes). Literals are stacked vertically over the print field or column.

Example

Here are various report headings:

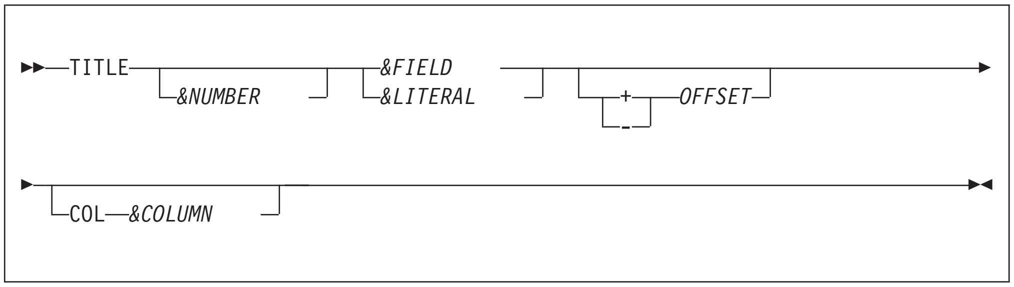
```
FILE FILEIN1 (80)                                 |
   CUST-NAME      01  15 A HEADING ('NAME')       |
   CUST-ADDRESS1  16  15 A HEADING ('ADDRESS1')   | Library Section
   CUST-ADDRESS2  32  15 A HEADING ('ADDRESS2')   |
   CUST-ADDRESS3  48  15 A HEADING ('ADDRESS3')   |

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 quotation marks.

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.

```
►►──LINE──┬──────────────┬──┬─&FIELD───┬──┬──────────────────┬──────────────►
          └─&NUMBER──────┘  └─&LITERAL─┘  └─┬─+─┬──OFFSET────┘
                                            └─-─┘

►──┬──────────────────┬──┬──────────────────┬──────────────────────────────►◄
   └─COL──&COLUMN─────┘  └─POS──&POSITION───┘
```

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 quotation marks.

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.

Page overflow for DISPLAY statements may be different from that of Easytrieve Plus. You can use the REXOVCK macro to make page overflow compliant with Easytrieve Plus.

The default is no overflow check.

Coding `%REXOVCK ON` in this exit activates the overflow check logic for DISPLAYs coded after the %REXOVCK ON statement. The page limit is the number of lines specified by the DISPLAYSIZE.

BEFORE-DETAIL. PROC

Before detail lines are built exit (Migration Utility exit only). This exit is entered before detail lines are built. It lets you change the contents of fields before detail lines are built. When multiple lines are printed, this exit is entered before building the first detail line only.

Page overflow for DISPLAY statements may be different from that of Easytrieve Plus. The REXOVCK macro can be used to make page overflow compliant with Easytrieve Plus.

The default is no overflow check.

Coding `%REXOVCK ON` in this exit activates the overflow check logic for DISPLAYs coded after the %REXOVCK ON statement. The page limit is the number of lines specified by the DISPLAYSIZE.

Usage hints:

Use tests on FIRST-DETAIL, POST-BREAK, PAGESIZE, and DISPLAYSIZE reserved keywords to control print fields on the first detail line of each page or the first detail line after control breaks.

Example 1:

Print a date field on the first detail line of each page and the first detail line after each control break for a report defined with one detail line. (Assume that all fields are defined).

```
REPORT RPT2 NOADJUST SPACE 0 LINESIZE 132 +
            NODATE NOPAGE SUMCTL(TAG) +
            TITLESKIP 0 PAGESIZE 58 PRINTER REPORT2
CONTROL COMPANY BRANCH
TITLE 1 COL 40 'TEST BEFORE-DETAIL REPORT EXIT'
LINE 01 COMPANY  +2 BRANCH +2 OFFICER +2  W-DATE +2 WAGE +2 RATE
⋮
BEFORE-DETAIL. PROC
  MOVE SPACES TO W-DATE
  IF FIRST-DETAIL
     MOVE I-DATE TO W-DATE MASK '99-99-99'
  END-IF
END-PROC
```

Example 2:

Print a date field on the first detail line of each page and the first detail line after each control break for a report defined with two detail lines. (Assume that all fields are defined).

```
REPORT RPT1 NOADJUST SPACE 0 LINESIZE 132 +
            NODATE NOPAGE SUMCTL(TAG) +
            TITLESKIP 0 PAGESIZE 58 PRINTER REPORT1
CONTROL COMPANY BRANCH
TITLE 1 COL 40 'TEST BEFORE-DETAIL REPORT EXIT'
LINE 01 COMPANY  +2 BRANCH +2 W-DATE +2 WAGE +2 RATE
LINE 02 '**L2**' +2 BRANCH +2 W-DATE +2 WAGE +2 RATE
⋮
BEFORE-DETAIL. PROC
  MOVE SPACES TO W-DATE
  IF FIRST-DETAIL OR LINE-COUNT > (PAGESIZE - 2)
     IF LINE-COUNT > (PAGESIZE - 2)
        LINE-COUNT = PAGESIZE
     END-IF
     MOVE I-DATE TO W-DATE MASK '99-99-99'
  END-IF
END-PROC
```

In the above example, a test is made for (PAGESIZE - 2) because our report contains two detail lines. Since FIRST-DETAIL is set to true only if the first detail line would cause the overflow, testing for (PAGESIZE - 2) and setting the LINE-COUNT to PAGESIZE, ensures that page overflow is detected for both lines, giving the desired output.

The test for (PAGESIZE - *n*) must be used whenever there is more than one detail line, including DISPLAYs located in the BEFORE-LINE and AFTER-LINE exits. In such a case, *n* is the total number of detail lines plus the number of DISPLAY statements.

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.

Page overflow for DISPLAY statements may be different from that of Easytrieve Plus. By default, Migration Utility checks overflow for DISPLAY statements as follows:

- For the first DISPLAY, the page limit is specified by PAGESIZE.
- For subsequent DISPLAYs, the limit is specified by DISPLAYSIZE.

You cannot control the overflow with the REXOVCK macro. If coded, the REXOVCK macro is ignored.

AFTER-LINE. PROC

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

Page overflow for DISPLAY statements may be different from that of Easytrieve Plus. You can use the REXOVCK macro to make page overflow compliant with Easytrieve Plus.

The default is no overflow check.

Coding `%REXOVCK ON` in this exit activates the overflow check logic for DISPLAYs coded after the %REXOVCK ON statement. The page limit is the number of lines specified by DISPLAYSIZE.

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 the 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, and so on. 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.

Page overflow for DISPLAY statements may be different from that of Easytrieve Plus. You can use the REXOVCK macro to make page overflow compliant with Easytrieve Plus.

The default is no overflow check.

Coding `%REXOVCK ON` in this exit activates the overflow check logic for DISPLAYs coded after the %REXOVCK ON statement. The page limit is the number of lines specified by DISPLAYSIZE.

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 the 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, and so on. 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.

Page overflow for DISPLAY statements may be different from that of Easytrieve Plus. You can use the REXOVCK macro to make page overflow compliant with Easytrieve Plus.

The default is no overflow check.

Coding `%REXOVCK ON` in this exit activates the overflow check logic for DISPLAYs coded after the %REXOVCK ON statement. The page limit is the number of lines specified by DISPLAYSIZE.

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.

The content of the print fields cannot and should not be changed. The BEFORE-DETAIL exit should be used if field contents must be changed before printing the next detail line.

Changing field contents in this exit results in a different outcome from that of Easytrieve Plus.

Page overflow for DISPLAY statements cannot be controlled by the REXOVCK macro. If coded, the REXOVCK macro is ignored.

Migration Utility defaults to no overflow check.

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.

You can control page overflow for DISPLAY statements by means of the REXOVCK macro.

The default is no overflow check.

Coding `%REXOVCK ON` in this exit activates the overflow check logic for DISPLAYs coded after the %REXOVCK ON statement. The page limit is the number of lines specified by DISPLAYSIZE.

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 118

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 using the Easytrieve Plus syntax and have a need for COBOL statements not supported by Easytrieve Plus.

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

```
                                          ┌──────────────────────┐
                                          ▼                      │
►►──INSPECT──&DATABUFFER  ──VREPLACING────┤ old/new strings ├────┴──┬─────────────────────────┬──►◄
                                                                    └──LENGTH──&STRINGLENGTH──┘
```

old/new strings:

```
├──&OLDSTRING──┬────────┬──┬─────────────────────┬──BY──&NEWSTRING──┬────────┬──────►
               └─IGNORE─┘  └─PADCHAR──&PADCHAR───┘                  └─IGNORE─┘

►──┬─────────────────────┬──────────────────────────────────────────────────────┤
   └─PADCHAR──&PADCHAR───┘
```

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.

Easytrieve macros

The Easytrieve macros allow the user to have record definitions and more frequently used Easytrieve routines defined externally of the program.

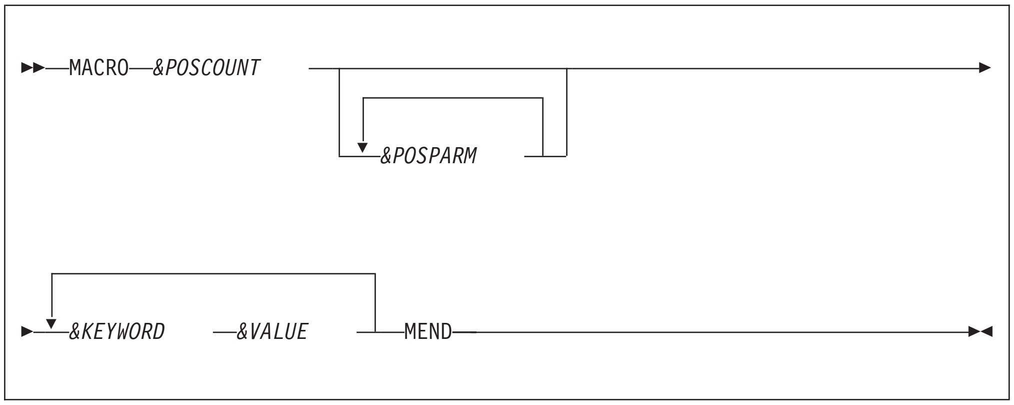
Macros are traditionally kept in a separate PDS or Librarian library.

Migration Utility fully supports all Easytrieve Macro Language conventions.

Macros allow symbolic replacement of the symbols embedded 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. The maximum number of positional parameters allowed by Migration Utility is 1024.

&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 embedded blanks must be enclosed in quotation marks.

User supplied statements are coded between the MACRO prototype and the MEND statements. The embedded 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 embedded in the FILE and the record definitions. Each positional and keyword identifier is preceded by a "&".

Invoking macros

Macros are embedded 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        |
                                              |     Valid
                                              |
%MSTFILE FILEIN2 130 ORG KSDS PREFIX I2       |
                                              |     MSTFILE
                                              |
%MSTFILE FILEIN3 130 PREFIX I3                |     Macro Invocations
```

Chapter 6. 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

The FSYTPA00 program runs the SQL translator step when SSID or BIND parameters are coded on the PARM statement of an Easytrieve Plus program. There are two translators that it chooses from:

- When BIND (STATIC) or BIND(STATIC-ONLY) is coded, the standard DB2/SQL translator supplied with DB2 system is run. The supplied JCMUSQLJ and JCMUSQLP PROCs, located in the SYS1.SFSYJCLS, can be tailored and used to translate and bind DB2 programs that will be running with imbedded static SQL statements.
- When BIND(DYNAMIC) is coded, the Dynamic SQL translator supplied with the Migration Utility system is run. For details, see “Running SQL programs in Dynamic mode” on page 128.

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 207.

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: DECLGEN of a DB2 table

This example shows a real DECLGEN of a DB2 table. COBOL users typically 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_PRIM_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 typically coded with an underscore, the underscores should be changed to dashes to preserve COBOL field naming conventions.
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-, and so on. 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.
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 names 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:
BIND (STATIC/STATIC-ONLY/DYNAMIC)
SQLID (&OWNER)
SSID (&SSID)
PLAN (&PLAN)
LINK (&PROGRAM R)
QUAL (&QUAL)

For SQL/DS:
USERID (&USERID)
PREPNAME (&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')
```

Running SQL programs in STATIC mode

Migration Utility generates COBOL programs with imbedded SQL statements. To run in STATIC mode, the generated programs must be translated with the DB2 translator and bound to DB2 system before use.

To translate and bind DB2 programs, use the supplied JCMUSQLJ and JCMUSQLP PROCS in SYS1.SFSYJCLS library as templates. The BIND parameters are generated as described below.

The PARM statement 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, or you can tailor it for a custom use in a separate BIND job. In addition, the SQLBIND macro, located in the SYS1.SFSYCCLM PDS, can be changed to generate hard-coded parameters as needed. 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
           &PGMNAME = Program to use for connecting to DB2 via CAF
```

Running SQL programs in Dynamic mode

Normally, COBOL programs with imbedded SQL statements must be translated with an SQL translator and bound to DB2 before they can be run. This results in additional system overhead and unnecessary complications associated with binding.

Migration Utility provides a Dynamic SQL interface that allows users to run the DB2 application programs without doing the BIND step. This eliminates problems associated with the BIND access authority and system contention.

General concepts

The FSYSQLIO module, supplied with Migration Utility, performs DB2 calls on behalf of application programs. FSYSQLIO runs in CAF (Call Attachment Interface) mode.

- FSYSQLIO is prepared and bound at installation time with a collection name.
- COBOL programs that contain imbedded SQL statements are run through Migration Utility's Dynamic SQL translator instead of the standard DB2 SQL translator.

  Migration Utility's translator converts imbedded static SQL statements to interface with the FSYSQLIO module. The interface is by means of a standard COBOL CALL statement.
- The resulting COBOL program is compiled and linked, or executed as Link and Go, as a standard COBOL program.

  Migration Utility's standard runtime load library (SYS1.SFSYLOAD) is required at run time.

How does it work in a Migration Utility environment?

Migration Utility uses four parameters from the PARM statement coded in the Easytrieve Plus program to determine DB2 resources to be used at application run time:

- SSID (&ssid)
- PLAN (&plan)
- SQLID(&sqlid)
- BIND(&option)

Example:

```
PARM SSID(DBVA) PLAN(IBMMIGUT) SQLID(GLAPPL02) BIND(DYNAMIC)
```

Parameters not coded on the PARM statement default as follows:

- EZPARAMS file defines the default values for the SQLBIND and SQLSSID parameters.
- The defaults can be overridden by the PARM parameters placed in the Easytrieve Plus program. The BIND overrides the SQLBIND and SSID overrides the SQLSSID.
- There is no default for SQLID. If SQLID is not provided on the PARM statement in the Easytrieve Plus program, the TSO user that submits the application job is assumed by DB2. Note that SQLID can be controlled by imbedding an SQL SET CURRENT SQLID = '&SQLID' statement in the Easytrieve Plus source.
- SQLPARMS file defines the default values for the Dynamic SQL translator step. SQLPARMS is located in SYS1.SFSYEZTS.
- The one-step driver program (FSYTPA00) determines the mode of operation based on information supplied in the first two items above, and executes the DYNTRAN step located in the #EZTPROC. After successful translation, COBOL and LKED, or LKGO steps are executed as found in the #EZTPROC. The BIND is not required.

Available options supplied by EASYTRAN/EZPARAMS

SQLBIND=*mode*

Establishes the default application runtime mode, where *mode* can be:

STATIC
Runs in static mode. Option cannot be overridden by PARM BIND (&option) in Easytrieve Plus source.

DYNAMIC
To run in dynamic mode. Option cannot be overridden by PARM BIND (&option) in Easytrieve Plus source.

ANY BIND is supplied in Easytrieve Plus source. Run mode must be as supplied by PARM BIND (&option) in Easytrieve Plus source.

SQLSSID=(*&SYS*)
establishes the default DB2 system ID if one is not supplied by means of the PARM SSID in the Easytrieve Plus source where *&SYS* is the DB2 system to run with.

SQLMODE=FSYDB250
The FSYDB250 program has been changed to receive the SSID, PLAN and SQLID name from the application program if SQLDD is not supplied in the JCL. It employs the following logic:

- If SQLDD is supplied in the JCL, it extracts the SSID, PLAN and SQLID from the SQLDD file. Note that all parameters are optional.
- If PLAN is not supplied in the SQLDD file, FSYDB250 uses the plan name coded on the PARM PLAN (&plan) in the Easytrieve Plus source.
- If SSID is not supplied in the SQLDD file and SSID is coded in the Easytrieve Plus program, the supplied SSID from the program is used, otherwise FSYDB250 retrieves the SSID from the system module DSNHDECP (DSNEXIT or DSNLOAD load libraries).
- If SQLID is not supplied in the SQLDD file and SQLID is coded in the Easytrieve Plus program, the supplied SQLID from the program is used.

Translating DB2/SQL programs to run in Dynamic SQL mode

You can choose from any supplied JCMUCLG* or JCMUCL* jobs located in the SYS1.SFSYJCLS library. You must add DSNEXIT and DSNLOAD DB2 libraries to the JOBLIB or STEPLIB. Everything else remains as if you were running a non-DB2 program.

You can run programs as Link and Go, or you can choose to create a load module and execute it later. The load module is executed by name, like any non-DB2 program. The SYS1.SFSYLOAD Migration Utility runtime library is needed at run time.

Dynamic SQL translator and runtime errors

The translator errors are reported to SYSTERM DD*name*.

The translator generated program listing is printed to SYSTLIST.

Application runtime errors are printed to SYSOUT. Errors are reported by means of the standard IBM DB2 error-handler DSNTIAR module.

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 typically 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 uses the SYSIBM.SYSCOLUMNS catalog when 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 207.

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 2-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 2-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, the SQL preprocessor 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 the SQL preprocessor as coded. Host variables are potentially renamed 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 SYS1.SFSYEZTC.

Easytrieve Plus SQL files

To process data from an SQL table via Easytrieve SQL file method, you must code the following statements and definitions:

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 or more fields for the columns within the tables that you want to retrieve. The fields can be defined using the DEFINE statement or 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 typically 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 embedded 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 7. SQL File I/O statement reference

CLOSE statement

The CLOSE statement closes an SQL File.

```
►►──CLOSE──&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.

```
►►──DELETE──┬──────┬──&FILE ───────────────────────────────────────►◄
            └─FROM─┘
```

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.

```
►►──FETCH──┬──────┬──&FILE   ──────────────────────────────────────────►◄
           └─FROM─┘
```

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.

```
►►──SQL INCLUDE──┬─────────────────┬──┬──────────────────────────────────────┬──►
                 │    ┌────────┐   │  └─LOCATION──┬─starting-position──┬─┘
                 └─(──▼─column─┴─)─┘              ├─*──┬───────────┬─┤
                                                  │    └─+──offset─┘ │
                                                  ├─W────────────────┤
                                                  └─S────────────────┘

►──┬─────────┬──┬────────┬──┬──────────┬──FROM──┬──────────┬──&TABLE   ──────►◄
   └─HEADING─┘  └─UPDATE─┘  └─NULLABLE─┘        └─owner──.─┘
```

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.

```
►►──INSERT──┬──────┬──&FILE  ──────────────────────────────────────►◄
            └─INTO─┘
```

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.

```
►►──UPDATE──&FILE  ─────────────────────────────────────────────────►◄
```

Parameter

&FILE The name of the SQL file.

UPDATE issues an UPDATE WHERE CURRENT OF cursor.

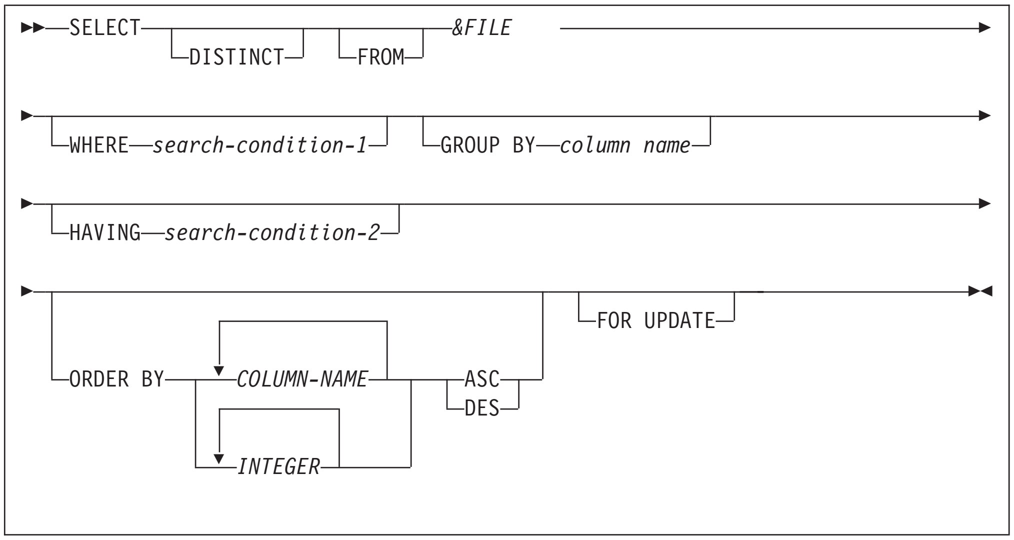
When the file is defined with the UPDATE, all defined columns are updated, otherwise only columns defined with the UPDATE are updated. Refer to the description of the SQL INCLUDE statement.

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.

Chapter 8. DLI/IMS support

Migration Utility provides a DLI/IMS interface. The functions described in this chapter are compatible with Easytrieve Plus, except as otherwise noted.

Three basic statements facilitate the DLI interface:

1. FILE is used to define the DLI files.

   A FILE statement is required for each database to be processed. It defines the database records (segments) and the parent/child relationships.
2. RETRIEVE is used for the Automated Job inputs.

   The RETRIEVE statement is required when you want to read a DLI file automatically via a JOB statement. It must be coded immediately after the JOB statement.

   Note: The file name used in the RETRIEVE statement must be the same file named on the JOB statement.
3. DLI is used to do the DLI I/Os in a controlled environment.

   The DLI statement can be used as needed in your program logic. Its syntax closely resembles the DLI call used by third-generation languages such as COBOL. The only parameter that you do not code is the PCB name. All other parameters are the same or very similar.

IMS/DLI concepts

In-depth description of DLI/IMS database concepts is beyond the scope of this document. However, here are a few points and some terminology to remember.

A DLI/IMS database is defined via the Data Base Definition macros (DBD macros). Among other things, the DBD contains the description of all records/segments, the field definitions in each segment, and the field key of each segment supported by that data base. The DBD is typically built by the DBD Administrator, and a load module/table is created to be used at run time.

To access a data base in an Easytrieve Plus program, the record (segment) definitions must be defined to your program. Creating an Easytrieve Plus macro of these definitions is recommended. Refer to "RECORD definition" on page 145 for details.

The I/O path for accessing your database is defined using the Program Specification Block (PSB) macro. The PSB identifies the data base, the segments, and the segment relationships for a complete I/O path. The PSB must be assembled and linked into a load library (typically by the database administrator) before it can be used. The PSB load module you create is passed to the DLI driver program via this statement in your JCL:

```
EXEC PGM=DFSRRC00,PARM=(PARM='DLI,&PGMNAME,&PSBNAME')
```

The **Program Communication Block (PCB)** is an area in memory created by the DLI DFSRRC00 driver program before passing control to your application routines. One PCB is built for each database to be accessed, plus an IO PCB for doing system calls (if CMPAT=YES is specified on the PSB macro). The pointers of these in-memory PCBs are received by your Easytrieve Plus or Migration Utility program at run time. These PCBs are used for DLI calls by your program.

It is critical that the PSB definitions correspond to the path as coded by the RECORD definitions in your program. Out-of-sequence entries will cause I/O errors.

Translating DLI/IMS programs

The JCMUIMSJ sample JCL can be found in the SYS1.SFSYJCLS Migration Utility library. It is also listed in “Migration Utility JCL,” on page 445.

To run Link and Go jobs, you must provide in the JCL the necessary application files that your program needs. To translate and link (create a load module) only, you can use any other JCL/PROC as described in Chapter 2, “Using Migration Utility,” on page 5.

To run Link and Go or compiled programs, you must execute the DFSRRC00 DLI program driver. The drive program then loads the application program specified on the PARM statement. If you are doing Link and Go, then your application program would be FSYTPA00, otherwise it would be the name of your linked load module. The `"//IMS DD ..."` or `"//IMSACB DD . . . "` statement must be included in the JCL to point to any load libraries where your PSB and DBD modules are located.

Summary of supported features

These features are supported:
- FILE definition
- RECORD definition
- FOR ACCESS statement
- RETRIEVE statement
- DLI I/O call statements
- DLI CHKP basic checkpoint
- DLI CHKP symbolic checkpoint
- DLI XRST extended restart

Summary of unsupported features

These features are not supported:
- Easytrieve Plus extended checkpoint/restart on:
  - PARM statement
  - JOB statement
  - RETRIEVE statement
  - REPORT statement

FILE definition

The FILE statement is coded in your program to identify the database to be used.

```
►►──FILE──&FILE-NAME────DLI──(─&DBDNAME────&PCBNO────)──┬───────┬──────────►◄
                                                       └─RESET─┘
```

Parameters

&FILE-NAME
Is the 1 to 8 character name used in your program to reference the database.

DLI Declares the file as a DLI database file.

&DBDNAME
The name of your IMS database. This is the 1 to 8 character name of your database defined via the IBM DBD macro.

&PCBNO
PCB sequence number within the DBD.

RESET
This parameter is ignored by Migration Utility.

Example

```
FILE JOURNAL DLI (DBXXX100)
```

RECORD definition

The database segments to be used by your program are identified by the RECORD statement.

Segments must be defined in the same sequence as defined in your PSB. You need to define only those segments that are needed in your program, however, the parent segment must be provided for each. Partial paths are not supported.

```
►►──RECORD──&SEGMENT────&SEGSIZE────┬──────────────┬─────────────────────────►
                                    └─&PARENT──────┘

►─┬──────────────────────────────────────────────┬───────────────────────────►◄
  └─KEY──(──&KEY────&POS────&KEYSIZE────)────────┘
```

Parameters

&SEGMENT
A one to eight character segment name as defined in the DBD.

&SEGSIZE
Segment size (in bytes).

&PARENT
The name of the parent segment. This parameter is required for all segments except the root segment.

KEY Identifies the segment key as defined in the DBD. This parameter is required for the RETRIEVE statement when using SAA or the tickler file.

&KEY The key name (sequence field) as defined in the DBD.

Note that this key name must be identical to the key defined in DBD. Using a different name results in a runtime IMS error.

&POS An integer that identifies the location of the key in the segment record.

&KEYSIZE
An integer that specifies the key length.

Note: The file PCB layout can be defined before the first RECORD definition. In the example below, JOURNAL-PCB-STATUS maps the status code returned by DLI. The value is equivalent to JOURNAL:FILE-STATUS, thus the definition is really not needed. A complete description of the PCB layout can be found in the IBM DLI/IMS reference manuals.

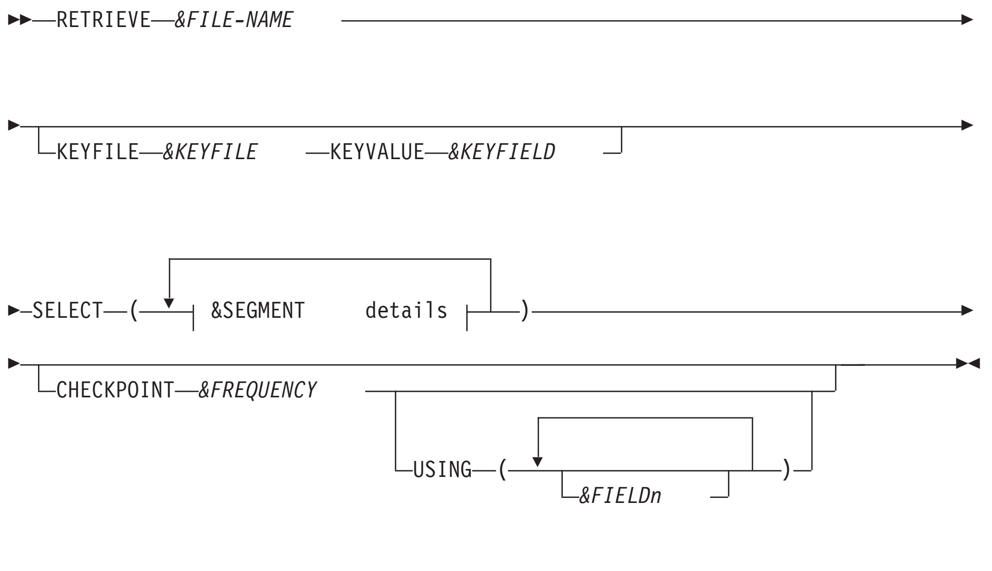
Example

```
FILE JOURNAL DLI (DBXXX100)
JOURNAL-PCB-STATUS 11 2 A

RECORD DBSEGM00 120 KEY (SG00KEY 1 20)
 SG00KEY                         1   20  A
 SG00-DATA                      21  100  A

RECORD  DBSEGM05  154  DBSEGM00
  SG05KEY                        1    12  A
  SG05-DATA                      13  142  A

RECORD  DBSEGM10  140  DBSEGM05
  SG10KEY                        1    10  A
  SG10-DATA                      11  130  A
```

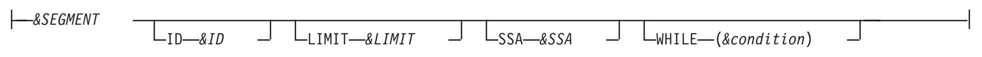
RETRIEVE statement

The RETRIEVE statement encapsulates the logic that performs automatic database input for a complete database path. It must be coded immediately after the JOB statement.

The following parameters are available:



&SEGMENT details:



&FILE-NAME
DLI file name as defined by the FILE statement

KEYFILE
Designates that a tickler file is to be used.

&KEYFILE
Tickler file ddname as defined by the FILE statement. This must be a sequential file that contains the keys to be retrieved from the root segments. The file is read sequentially. The key values are used as SSA (segment search arguments) for accessing the root segments. A Get Unique (GU) call is issued to DLI to retrieve the unique key found on the tickler file. The End of tickler file marks the end of automatic input and the RETRIEVE logic.

Note that the SSA option cannot be used for the root segment when the tickler file is in use.

&KEYFIELD
Identifies the field name to be used as the segment search argument (SSA).

SELECT (...)
Segment select statement. One or more segments can be coded, each followed by the select options:

&SEGMENT
Segment name to retrieve.

ID Segment identification code.

&ID is a 2-digit literal for identifying the retrieved paths (segments). The &ID is moved to the system-defined field PATH-ID for the lowest accessed segment in the path. The literal can be any two characters allowed by the system. When accessing root segments using a tickler file, PATH-ID is automatically set to 'NF' (not found) by the RETRIEVE logic.

LIMIT
Segment limit identifier.

&LIMIT is the number of segments to retrieve. This limit applies to each segment in the path. If LIMIT is not coded, all segments are retrieved in each path.

SSA Segment search identifier

&SSA is the root segment SSA. This option cannot be used when a tickler file is in use. The supplied value is enclosed in parentheses and concatenated with the segment name, by the RETRIEVE logic, to form a valid SSA.

WHILE (&condition)
A conditional expression for selecting on specific segment field values.

&condition is a conditional expression. The syntax for this expression is the same as that of the DO WHILE statement described in “DO and END-DO statements” on page 90. Segments are accepted for input only when the WHILE expression is true, otherwise they are bypassed.

CHECKPOINT
Checkpoint information (not supported by Migration Utility)

&FREQUENCY
An integer indicating the checkpoint interval (frequency)

USING
List of fields to be restored at restart time. &FIELD1 ... &FIELDn are the field names to be restored.

Programming notes:

1. With RETRIEVE, you can sweep the entire database without coding the detailed calls to DLI. Segments are retrieved starting with the root segment and all subordinate child segments in the path, up to the lowest segment in the path. When a path is fulfilled, all retrieved segments are returned in the respective segment records as defined for the DLI file in use. Processing continues with the first instruction coded after the RETRIEVE statement.
2. You can code `ID &ID` for each segment to identify the lowest segment in use. The system defined field PATH-ID can then be tested to find out the lowest segment returned.
3. RETRIEVE is coded immediately after the JOB statement. The file on the JOB statement must reference a valid DLI file that contains segments the RETRIEVE is operating on.

DLI statement

The DLI statement provides controlled access to an IMS/DLI database. DLI statements are used in the Activity Section independently of the RETRIEVE statement. This section describes the four DLI formats.

Format-1: DLI application I/O calls

```
►►──DLI──&FILE-NAME ──&SEGMENT ──&FUNCTION ──┬───────────┬──────────►
                                             └─SSA──&SSA─┘

►──┬─────────────┬─────────────────────────────────────────────────►◄
   └─SSANO──&SSANO─┘
```

Parameters

&SEGMENT
: I/O area segment name coded on a RECORD statement, or a working storage field name. In either case, the size of this storage must be large enough to hold the retrieved segments.

SSA &SSA
: Identifier for SSA.

: &SSA is an optional segment search argument. Multiple search arguments can be coded. &SSA can be a field name or a string (alphanumeric literal). DLI SSA coding rules must be observed.

SSANO &SSANO
: &SSANO is a 4-byte binary field that identifies the number of SSA parameters. This parameter is bypassed by &prod;. The number of SSA parameters is always assumed to be the number of coded SSA statements.

The I/O return code is placed in the system reserved field &FILE:FILE-STATUS:

SPACES
: A good call

'GB' EOF (end of data base)

'GE' End of segment

Any other value indicates an error.

Format-2: Basic checkpoint

This basic CHKP commits the changes your program has made to the database. It establishes a point of restart from which your program can be restarted after an abnormal termination. The CHKP DLI call can be used only when your PSB is generated with the CMPAT=YES option.

```
►►──DLI──CHKP──&AREA ──────────────────────────────────────────────►◄
```

Parameters

&AREA
: An area that contains 8-byte checkpoint ID. This is an input parameter.

After the call, spaces in the CHKP-STATUS system reserved name indicates a good call.

Format-3: Symbolic checkpoint

A symbolic checkpoint is used for the recovery purposes. When symbolic CHKP is used in your program, the XRST must be used as well.

The symbolic CHKP commits the changes your program has made to the database and it saves areas defined in your program for restarting (XRST), should your program terminate abnormally.

An XRST is required before CHKP to activate the IMS symbolic checkpoint interface. The XRST must be issued with a check point ID of blanks.

```
                                         ┌──────────────────────┐
                                         ▼                      │
►►──DLI──CHKP──&IOLENGTH────&IOAREA──────┴──────────────────────┴──────►◄
                                          └─&LENGTH────&AREA───┘
```

Parameters

&IOLENGTH
Length of &IOAREA. This must be a valid integer.

&IOAREA
An area that contains the 8-byte ID for this checkpoint. This is an input parameter.

&LENGTH
Length of &AREA field. This must be a valid integer.

&AREA
An area in your program that you want to checkpoint. This is an input parameter. This area is saved by IMS for restart.

Up to seven pairs of &LENGTH &AREA can be coded. When you restart your program, IMS restores only the areas specified in the CHKP call.

After the call, spaces in the CHKP-STATUS system reserved name indicates a good call.

Format-4: Extended restart

The Extended Restart (XRST) is used to restart your program after it terminates abnormally. If the CHKP is being used, an XRST is required before the CHKP to activate the IMS symbolic checkpoint interface. The XRST must be issued with a check point ID of blanks.

```
                                         ┌──────────────────────┐
                                         ▼                      │
►►──DLI──XRST──&IOLENGTH──&IOAREA──────┬──────────────────────┬─┴──►◄
                                         └─&LENGTH──&AREA─┘
```

Parameters

&IOLENGTH
Length of &IOAREA. This must be a valid integer.

&IOAREA
An area that contains the 14-byte checkpoint ID for this restart, or blanks when starting your program normally. This is an input parameter.

&LENGTH
Length of &AREA field. This must be a valid integer.

&AREA
An area in your program that you want IMS to restore. Up to seven pairs of `&LENGTH &AREA` can be coded. When you restart your program, IMS restores only the areas specified in the CHKP call.

After the call, spaces in the CHKP-STATUS system reserved name indicates a good call.

Restarting your program

IMS determines whether to perform a normal start or restart based on the &IOAREA provided on XRST, or CKPTID= value provided in the PARM field on the EXEC statement in your JCL. If you supply both parameters, IMS will use the one from the CKPTID= parameter.

The ID specified can be any of the following:
- A 1 to 8 character extended checkpoint ID
- A 14 character time stamp ID from DFS0540I message where:
  - IIII is the region ID
  - DDD is the day of the year
  - HHMMSSR is the time in hours, minutes, seconds, and tenths of a second
- The constant `LAST`. (BMPs only: indicates that the last checkpoint issued by BMP will be used.)

The system message DFS0540I supplies the checkpoint ID and the time stamp.

The system message DFS682I supplies the last completed checkpoint which can be used to restart your program.

IMS writes checkpoint-restart information to the Online Log data set or System Log data sets, The //IMSLOGR DD statement must be supplied in the JCL if the Online Log is no longer available. IMS searches these data sets for the checkpoint records with the ID that was specified.

Restriction: The original job name of the job that failed must be used for the restart run. Otherwise, IMS will not be able to locate the checkpoint records it needs, it which case, IMS fails with code U0102.

At successful completion of XRST, the &IOAREA contains an 8-character checkpoint ID, or a 14-character time stamp. The time stamp is returned when the XRST is issued with blanks in the checkpoint ID.

Spaces in the CHKP-STATUS system reserved name indicates a successful restart.

DLI FOR ACCESS statement

The DLI FOR ACCESS statement allows the user to access the DLI file RECORD fields without referencing the DLI file in the job logic. Its main purpose is to allow users to populate DLI file records by means of a CALL to an external program.

Format-1

```
►►──DLI──&FILE-NAME    ──FOR ACCESS──────────────────────────────────────►◄
```

Parameters

&file-name
DLI file name as defined by the FILE statement.

DLI program examples

Example 1—Sweep of database using RETRIEVE statements

This example demonstrates a sweep of an entire journal database called DBXXX100 using RETRIEVE statements.

Our database consists of three segments: DBSEG00, DBSEGM05, and DBSEGM10. DBSEG05 is a child of the DBSEG00 root segment, and DBSEG10 is a child of the DBSEG05 segment.

The logic below reads the entire data base starting with the first root segment and its subordinate child segments. Multiple child segments are read until all segments are processed in each hierarchy. Note that this automatic input follows the same logic as that in Example 2.

```
PARM LINK (TESTIMS2 R)
* EASYTRAN: DEBUG (LIST COBOL)
* END-EASYTRAN

FILE JOURNAL DLI (DBXXX100)
RECORD DBSEGM00 120  KEY (SG00KEY 1 20)
 SG00KEY                         1   20  A
 SG00-DATA                      21  100  A

RECORD  DBSEGM05  154  DBSEGM00
  SG05KEY                        1    12  A
  SG05-DATA                      13  142  A

RECORD  DBSEGM10  140  DBSEGM05
  SG10KEY                        1    10  A
  SG10-DATA                      11  130  A

JOB INPUT JOURNAL START INITIALIZE
RETRIEVE JOURNAL +
         SELECT (DBSEGM00 ID 'RT'                        +
                    SSA 'SG00KEY >                    '  +
                 DBSEGM05 ID '05'                        +
                 DBSEGM10 ID '10')

IF PATH-ID EQ '10'
   * ALL SEGEMNTS ARE AVAILABLE HERE
   CONTINUE
ELSE-IF PATH-ID EQ '05'
   * DBSEG00 AND DBSEG05 ARE AVAILABLE ONLY
   CONTINUE
ELSE-IF PATH-ID EQ 'RT'
   * DBSEG00 IS AVAILABLE ONLY
   CONTINUE
END-IF

GO TO JOB

INITIALIZE. PROC
* INSERT INITIALIZE STATEMENTS HERE
END-PROC
```

Example 2—Sweep of database using controlled DLI statements

This example demonstrates a sweep of the DBXXX100 journal database using controlled DLI statements.

Our database DBXXX100 consists of three segments; DBSEG00, DBSEGM05, and DBSEGM10. DBSEG05 is a child of the DBSEG00 root segment, and DBSEG10 is a child of the DBSEG05 segment.

The logic below reads the entire data base starting with the first root segment and its subordinate child segments. Multiple child segments are read until all segments are processed in each hierarchy.

```
PARM LINK (TESTIMS1 R) SORT (DEVICE TEMP)
* EASYTRAN: DEBUG (LIST COBOL)
* END-EASYTRAN

FILE JOURNAL DLI (DBXXX100)
RECORD DBSEGM00 120
 SG00KEY                           1   20  A
 SG00-DATA                        21  100  A

RECORD  DBSEGM05  154  DBSEGM00
  SG05KEY                          1    12  A
  SG05-DATA                        13  142  A

RECORD  DBSEGM10  140  DBSEGM05
  SG10KEY                          1    10  A
  SG10-DATA                        11  130  A

K00-SSA W 42 A
   KSF K00-SSA 20 A VALUE 'DBSEGM00(SEG00KEY > '
   KSD K00-SSA +20 20 A VALUE LOW-VALUES
   KSE K00-SSA +40 2 A VALUE ') '

S00-SSA W 42 A
   SSF S00-SSA 20 A VALUE 'DBSEGM00(SEG00KEY = '
   SSD S00-SSA +20 20 A VALUE LOW-VALUES
   SSE S00-SSA +40 2 A VALUE ') '

S05-SSA W 23 A
   SLF S05-SSA 20 A VALUE 'DBSEGM05(SEG05KEY = '
   SLD S05-SSA +20 1 A VALUE ' '
   SLE S05-SSA +21 2 A VALUE ') '

WSEGID  W    8   A

JOB INPUT NULL START INITIALIZE
* READ NEXT ROOT SEGMENT
NEXT-ROOT
   DLI JOURNAL DBSEGM00 'GN  ' SSA (K00-SSA)
   IF JOURNAL:FILE-STATUS EQ 'GB' 'GE'
      RETURN-CODE = 0
      STOP
   ELSE-IF JOURNAL:FILE-STATUS NE '  '
      WSEGID  = 'DBSEGM00'
      PERFORM PRINT-ERROR
      RETURN-CODE = 16
      STOP EXECUTE
   END-IF

* READ NEXT DBSEGM05 IN THE CURRENT ROOT SEGMENT
NEXT-SEG0
   SSD = SG00KEY
   DLI JOURNAL DBSEGM05 'GNP ' SSA (S00-SSA, 'DBSEGM05 ')
   IF JOURNAL:FILE-STATUS EQ 'GE'
      GO TO JOB
   ELSE-IF JOURNAL:FILE-STATUS NE '  '
      WSEGID  = 'DBSEGM05'
      PERFORM PRINT-ERROR
      RETURN-CODE = 16
      STOP EXECUTE
   END-IF

* READ NEXT DBSEGM10 IN THE CURRENT DBSEGM05 SEGMENT
NEXT-SEG1
   SLD = SG05KEY
   DLI JOURNAL DBSEGM10 'GNP ' SSA (S00-SSA, S05-SSA 'DBSEGM10 ')
   IF JOURNAL:FILE-STATUS EQ 'GB'
      GO TO JOB
   ELSE-IF JOURNAL:FILE-STATUS EQ 'GE'
      GO TO NEXT-SEG0
   ELSE-IF JOURNAL:FILE-STATUS NE '  '
      WSEGID  = 'DBSEGM10'
      PERFORM PRINT-ERROR
      RETURN-CODE = 16
      STOP EXECUTE
   END-IF

* INSERT PROGRAM LOGIC HERE
   GO TO NEXT-SEG1

PRINT-ERROR. PROC
   DISPLAY WSEGID +
           ' ERROR JOURNAL:FILE-STATUS: ' JOURNAL:FILE-STATUS
END-PROC

INITIALIZE. PROC
* DO THE NECESSARY INITIALIZATION HERE
END-PROC
```

Chapter 9. Creating HTML and spreadsheet files

Migration Utility provides statements that allow you to create reports and files that can be easily accessed by the standard HTML browsers and spreadsheet software. You can create:

- Character Separated Values (CSV) files and reports
- HTML Drill Down reports
- A combination of Drill Down and CSV reports

The HTML and CSV reports can be placed in a PDS file, then downloaded to a server and parsed so that they can be accessed by a browser. To do the parsing, use the supplied JAVA utility, fsyjpars.class.

The CSV files and HTML reports can be written to the UNIX File System on z/OS and accessed by a browser directly from the z/OS server. No file transfer or parsing is needed.

Note:

1. Job JCMUDRL1, located in SYS1.SFSYJCLS, contains a complete example of a program that creates HTML Drill Down and CSV reports (also listed at end of this chapter).
2. Job JCMUDRL2, located in SYS1.SFSYJCLS, contains a complete example of a program that creates a CSV file and a CSV report (also listed at end of this chapter).
3. Job JCMUDRLU, located in SYS1.SFSYJCLS, contains a complete example of a program that creates Drill Down reports and writes HTML documents directly to a mainframe UNIX File System (also listed at end of this chapter).

Character Separated Value (CSV) files and reports

You can download the CSV files and reports from the mainframe, or place them on the mainframe web server, and import them into many commonly used spreadsheets.

To create a CSV file, you use one or more DISPLAY statements with a SEP= parameter.

To create a CSV report , you code a standard REPORT statement with a SEP= parameter.

Syntax

```
►►──SEP=(─'&char'    ───────────)──────────────────────────────────────►◄
                       └─MASK─┘
```

Where:

&char One or more characters to insert

MASK

Mask option:

If MASK is coded, Migration Utility uses the field mask as coded on the field definition or the default mask. For example, 123,456.88.

If MASK is not coded, Migration Utility formats numeric fields without edit characters and inserts the decimal point and a leading sign. For example, -123456.88.

The SEP= parameter triggers Migration Utility to inserts the specified characters after each field.

Examples:

Create comma-separated fields to a file named CSVFILE1 (assume that all fields on the DISPLAY statement are defined and that “FILE CSVFILE1 F (100)” has been defined).

1. To create field values for the CSVFILE1 file without edit characters, write:
   ```
   DISPLAY CSVFILE1 SEP=(',') FIELD1 'CSV TEST RECORD ' FIELD2 FIELD3
   ```
2. To create fields for the CSVFILE1 file with the default MASK, write:
   ```
   DISPLAY CSVFILE1 SEP=(',' MASK) FIELD1 'CSV TEST RECORD '  FIELD2 FIELD3
   ```
3. To create report RPT1 with CSV fields without edit characters, write:
   ```
   REPORT RPT1 SEP=(',') ...
   ```
4. To create report RPT1 with CSV fields with the default MASK, write:
   ```
   REPORT RPT1 SEP=(',' MASK) ...
   ```

Job JCMUDRL2, located in SYS1.SFSYJCLS, contains a complete example of a program that creates a CSV file and a CSV report.

HTML Drill Down reports

The Drill Down reports are HTML-ready reports that can be browsed using standard Internet browsers such as Internet Explorer or Netscape.

You can create reports as a single PDS file, or write them out directly to a UNIX File System on the z/OS as follows:

- When you create a PDS file, you must download the file, parse it, and place it on a server where it can be accessed with a browser. To do the parsing, use the supplied Java utility, fsyjpars.class.
- When you write the reports to a UNIX File System on z/OS, you can access them with a browser directly from the z/OS server. No file transfer or parsing is needed.

Concepts

A Drill Down hierarchy consists of a Drill Down document and Drill Down reports.

A Drill Down document defines:

- One or more Drill Down menu items (lines to be included in the index page).
- The Drill Down report names to be anchored as a group for each menu item.

Figure 1 on page 157 generalizes the concept.

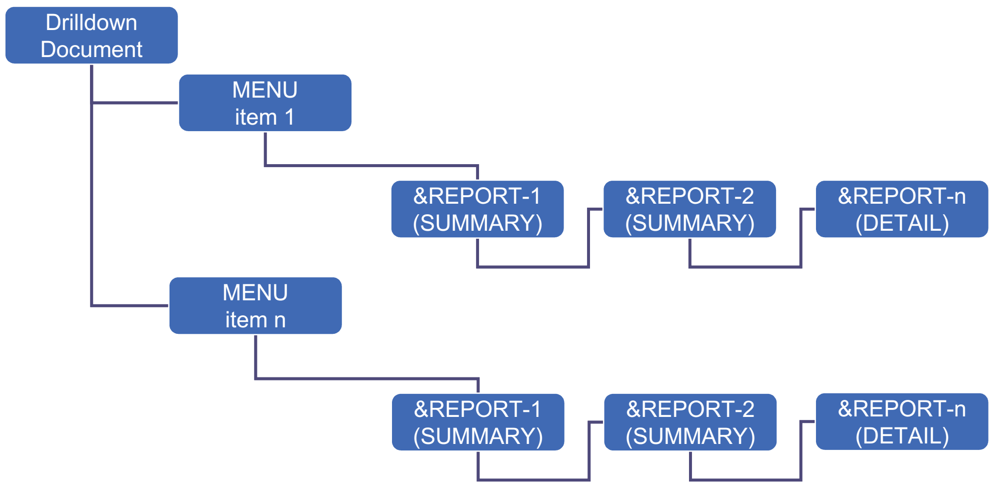


Figure 1. Drill Down reports

For example, to define a Drill Down document called HTMLFL1 that is accessed from the menu as “INCOME FROM SERVICES (TEXT)” with Drill Down reports by CO-NAME and BR-NAME, you need to name three separate reports, one for each control break, and the last one as the DETAIL report:

```
REPORT <DOC> HTMLFL1 VALIDATE
DRILL MENU 1 #(RED BOLD)  +
      'INCOME FROM SERVICES (TEXT)' #GREEN SYSDATE
DRILL DOWN RPT1 CO-NAME   RPT2 BR-NAME   RPT3 DETAIL
```

Each named report RPT1, RPT2 and RPT3 is defined using the standard Easytrieve Plus REPORT statement, in combination with some additional parameters that add font and attribute capabilities for appearance.

For each report, you must code a separate REPORT statement with exactly one control break specifying the field name listed in the DRILL DOWN statement, with the last report declared as the DETAIL report, and all other reports in the hierarchy as SUMMARY reports.

```
REPORT RPT1 SUMMARY ...
CONTROL  CO-NAME
⋮
REPORT RPT2 SUMMARY ...
CONTROL BR-NAME
⋮
REPORT RPT3 ...
CONTROL DETAIL
⋮
```

Migration Utility checks that each control break named in the DRILL DOWN report list is the same as the one defined on CONTROL statement of each report.

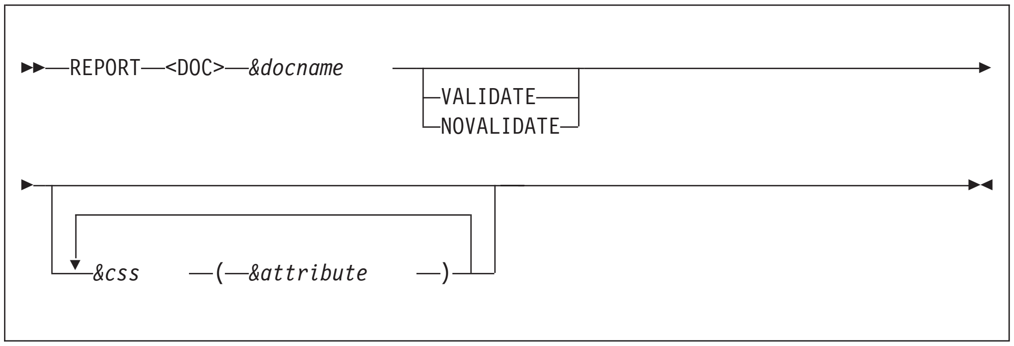
When accessed using a browser, the Drill Down reports follow the hierarchy of control breaks. That is, each control break represents a layer of HTML pages that have links to the lower level reports (parent and child relationship). In the examples above, MENU links to RPT1, the RPT1 entries link to RPT2, and the

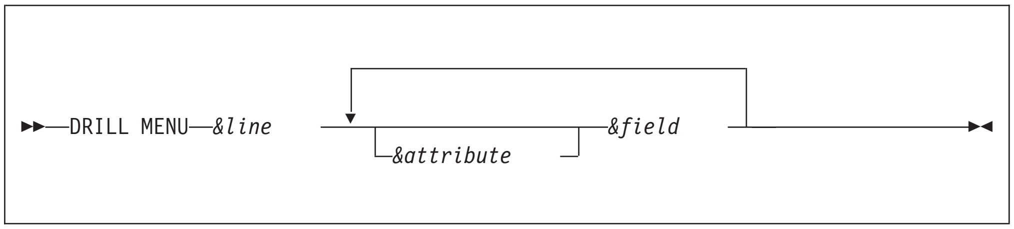
RPT2 entries link to RPT3. Think of it as three distinct reports, with each higher level linking to detail data that belongs to it.

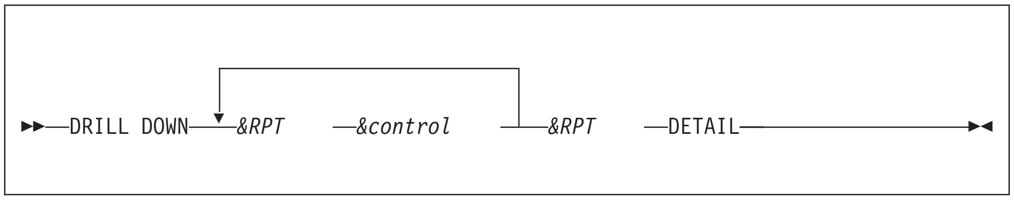
Note that each report is printed using the PRINT statement, just like any other report. Because of dependencies, each report in the Drill Down group must be produced using the same input data. Feeding reports with different data may result in unresolved links.

Job JCMUDRL1, located in SYS1.SFSYJCLS, contains a complete example of a program that creates Drill Down reports (also listed at end of this chapter).

Defining Drill Down documents







Parameters

&docname

Document file ddname. The file must be a PDS file with DCB and DISP coded in the JCL as:

```
DCB=(LRECL=4096,RECFM=VB,BLKSIZE=,DSORG=PO)
DISP=(MOD,CATLG,DELETE)
```

Note: At application run time, a PDS member, *&docname*, is created for all reports identified in the DRILL DOWN lists. To make *&docname* ready for a browser, you must download it into a server and parse it into constituent directories and files using the supplied JAVA program, fsyjpars.class.

VALIDATE
This is the default. Prompts Migration Utility to generate logic that validates the Drill Down URLs (links). When used, you must supply FJWDOC0, SORTIN, and FJEDOC0 ddnames in the application runtime JCL. See "Drill Down JCL requirements" on page 167.

NOVALIDATE
Prompts Migration Utility to ignore the URL validation logic.

&css Default Cascaded Style Sheets (CSS) templates. Migration Utility reads these templates from the SYS1.SFSYDOCS PDS, depending on the DOCTYPE specified on the REPORT statement.

$B1	CSS for HTML BODY
$TB	CSS for table body
$TH	CSS for table headers
$TR	CSS table rows
$TD	CSS table data
$LN	CSS for text line
$CP	CSS table caption

&attribute
Default attribute to be used for building HTML documents. You can specify a color and font. See "Available Attributes" on page 164.

&line Menu item number (the line number on the menu frame). This must be an integer from 1 to 99.

&field Field or literal value to be included on the menu line. Code these elements as you would for a DISPLAY statement.

&RPT Drill Down Report name to be included in the drill down hierarchy.

&control
Control break as specified for each report *&RPT*. *&control* for the last report must be coded as "DETAIL".

DRILL MENU and DRILL DOWN statements are coded in pairs, each pair defining a Drill Down hierarchy. You can define multiple Drill Down hierarchies within a single Drill Down document.

Example

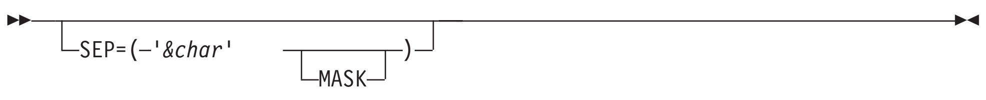
```
REPORT <DOC> HTMLFL1 VALIDATE      +
                    $B1 (BLUE)     +
                    $TB (BLUE)     +
                    $TR (BLUE)     +
                    $TH (BLUE)

DRILL MENU 1 #(RED BOLD) +
     'INCOME FROM SERVICES (TEXT)' #GREEN SYSDATE
DRILL DOWN RPT1 CO-NAME RPT2 BR-NAME RPT3 DETAIL

DRILL MENU 2 #(RED BOLD) +
     'INCOME FROM SERVICES (TABLE)' #GREEN SYSDATE
DRILL DOWN RPT21 CO-NAME RPT22 BR-NAME RPT23 DETAIL
```

Defining Drill Down Reports

Each Drill Down report named in the REPORT document is defined using the standard Easytrieve Plus REPORT statement parameters. The following sections describe additional parameters that you can use.

Insert character for CSV reports

This parameter specifies the insert character for CSV reports.



&char One ore more characters to insert.

MASK
Mask option.

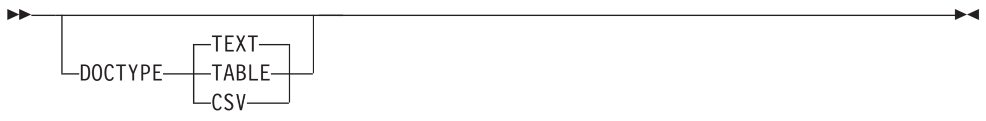
If MASK is coded, Migration Utility uses the field mask as coded on the field definition, or the default mask. For example, 123,456.88.

If MASK is not coded, Migration Utility formats numeric fields without edit characters, and inserts the decimal point and a leading sign. For example, -123456.88.

The SEP= parameter triggers Migration Utility to inserts the specified characters after each field for the DOCTYPE CSV reports.

HTML document type

This parameter specifies the type of HTML document to create.



TEXT Create HTML pages in text format.

TABLE
Create HTML pages in TABLE format.

CSV Create Character Separated Value (CSV) report. These files are created with a .csv file type.

Note: To simplify the accessability of CSV files, adjust the MIME type on your PC browser to use a specific spreadsheet program for .csv files.

Link identifier for PREV and NEXT buttons

This parameter specifies the link identifier for the PREV and NEXT buttons.

```
►►──┬──────────────────────┬──────────────────────────────────────────►◄
    │           ┌─Drill──┐ │
    └─LINKID────┼─NO─────┼─┘
                └─'&id'  ┘
```

NO Do not create PREV/NEXT link buttons.

&id Literal to be included on the NEXT and PREV buttons. Can be 1–16 characters long.

Templates and attributes

This parameter specifies the templates and attributes for a report

```
►►──┬──────────────────────────────┬───────────────────────────────────►◄
    │ ┌──────────────────────────┐ │
    │ ▼                          │ │
    └─&css    ──'&attribute'    ─┴─┘
```

&css Default Cascaded Style Sheets (CSS) templates specific to this report. Migration Utility reads these templates from the SYS1.SFSYDOCS PDS depending on the DOCTYPE specified on the REPORT statement.

$B1	CSS for HTML BODY
$TB	CSS for table body
$TH	CSS for table headers
$TR	CSS table rows
$TD	CSS table data
$LN	CSS for text line
$CP	CSS table caption

&attribute
Default attribute to be used for building HTML documents. You can specify the color and font to be used. See “Defining field attributes” on page 163.

REPORT statement considerations

PAGESIZE *&n*
In most situations, this parameter defines the number of lines for each report page. However, for Drill Down reports, it defines the number of lines for each HTML page.

Each HTML page is generated as a separate file. Therefore, when selecting the PAGESIZE, you should take care to ensure that each HTML page gives you the optimum number of lines. Too few lines results in more frequent paging and additional overhead while creating directories on your server. Too many lines causes longer scrolling pages.

To inhibit paging, specify PAGESIZE 0.

TITLESKIP 1
Use this parameter to avoid unnecessary blank lines in the HTML document.

SPACE 1
Use this parameter to avoid unnecessary spaces between fields in CSV reports.

SUMMARY
Use this option for all reports in the specified DRILL DOWN hierarchy except the last report. The last report can be a SUMMARY or a DETAIL report.

SUMCTL (ALL DTLCOPY)
Use this parameter for SUMMARY reports to make sure that all relevant information is shown on each report line.

CONTROL statement considerations

CONTROL statements must name the same field name as specified for each report named in the DRILL DOWN hierarchy. You can only specify one control break field. The last report in the DRILL DOWN hierarchy must specify DETAIL for control break (even if you specify SUMMARY on the REPORT statement). The last (DETAIL) report can list additional control break fields following the DETAIL parameter.

FINAL totals:
- Can be printed only when one report is named in the DRILL DOWN list.
- Can not be printed when multiple reports are named in the DRILL DOWN list.

EZPARAMS/EASYTRAN options

PRINTER=AUTOGEN
This is a required option. Alternatively, you can specify a PRINTER file for each report. Note that SYSPRINT cannot be used for the Drill Down reports as each report must be written to a separate file.

Field attributes

Migration Utility provides syntax for specifying field attributes such as the font type, font size, and color. You can conditionally set these attributes with the WHEN wizard. For example, you can print negative numeric values using a distinct color. See “Defining field attributes” on page 163 for the attribute coding rules.

When you specify an attribute, it must:
- Be the first item preceding the field name
- Precede the COL or POS values

HEADING attributes

You can not place attributes in the HEADING of field definitions. Instead, code the attributes in the HEADING definitions of the REPORT statement. You place the attributes before each literal.

Example:
```
HEADING AMOUNT (#(120% BOLD) 'CURRENT' #(120%) 'AMOUNT')
```

For rules about coding attributes, see “Defining field attributes” on page 163.

REPORT SEQUENCE and performance issues

The Drill Down reports are expected to be in the same sequence as the CONTROL fields listed in the DRILL DOWN list.

```
DRILL DOWN  RPT1 COMPANY  RPT2 BRANCH   RPT3 DETAIL
```

In the above example:
- RPT1 data must be in sequence by COMPANY
- RPT2 must be in sequence by COMPANY and BRANCH
- RPT3 must be in sequence by COMPANY, BRANCH, and some other fields in the lower hierarchy.

In the example shown, it is important to realize that you are dealing with three separate reports that are connected by means of a Drill Down document. Coding a SEQUENCE statement for each report would create three separate SORT requests, one for each report. This can add a substantial overhead to your program.

You should avoid using the SEQUENCE statement whenever possible. Instead, use the SORT statement in the program (or use an external SORT) to put your records in the desired order before printing.

Defining field attributes

The default attributes are obtained from the Cascaded Style Sheets (CSS) located in the SYS1.SFSYDOCS Migration Utility library.

You can override the global defaults for each document by coding *&css* names with overriding values for each Drill Down document (see “Defining Drill Down documents” on page 158).

You can override the defaults for each report by coding *&css* names with overriding values for each report on the REPORT statement (see “Defining Drill Down documents” on page 158).

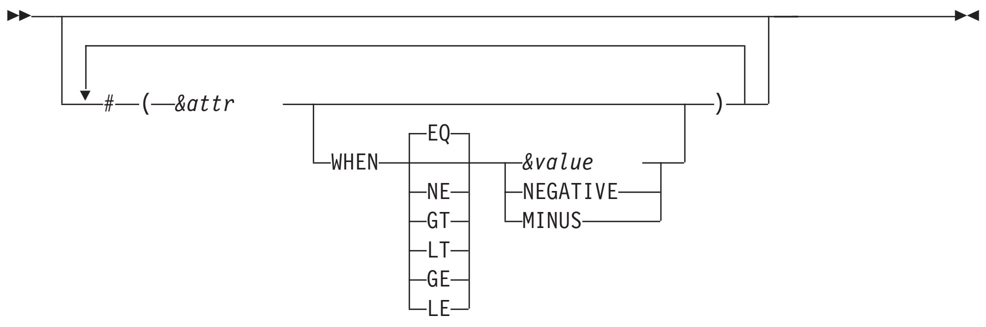
You can specify attributes for each field listed on the MENU, TITLE, LINE, or DISPLAY statements using the syntax below.

Attributes syntax

When you specify an attribute, it must be the first item preceding the field name and preceding any COL or POS values.

Attributes syntax

Format 1



Parameters:

&attr Any attribute from the list of attributes (see "Available Attributes").

(Relational operator)
Can have one of the following values:

EQ	Equal
NE	Not equal
GT	Greater than
LT	Less than
GE	Greater than or equal
LE	Less than or equal

&value Value or literal to compare on. If the literal consists of more than one word, you must enclose the whole literal in quotes.

The length, used in comparison, is the length of field or the length of *&value*, whichever is the shorter. Therefore, when *&value* is shorter than the field, the field is compared partially (left side). When the field is shorter, *&value* is compared partially (left side).

NEGATIVE
Tests for "-" (negative value) at end of the field.

MINUS
Same as NEGATIVE.

Example:

```
TITLE 1   #(BOLD 110% BLUE) COL 10 'Report: RPT1'
LINE  1   #(RED WHEN MINUS) COL 1  AMOUNT1   +
          #(BOLD WHEN EQ .00) COL 20 AMOUNT2 +
          #(RED WHEN EQ 'JOHN SMITH') NAME
```

Available Attributes

Migration Utility looks at the FSYFONTS table (the source is located in the SYS1.SFSYEZTS) to assign a meaning to each attribute. If a specified attribute does not exist in the table, a color attribute is assumed.

Be careful with the color attributes as they are not validated by Migration Utility. The validation was not enforced intentionally, to provide an open system.

The FSYFONTS table can be updated at installation time to reflect your environment.

Attributes

Code attributes as follows:

Background color

Code as BG-*&color*, where *&color* is any valid HTML color (including a hexadecimal number preceded by a "#").

Examples:

```
BG-WHITE
BG-#ffcafff
```

Text color:

Code any valid HTML color (including a hexadecimal number preceded by a "#").

Examples:

```
RED
#adfffff
```

Font style:

ITALIC
OBLIQUE

Font weight:

NORMAL
BOLD

Font decoration:

BLINK
OVERLINE
UNDERLINE
LINE-THROUGH

Font family:

ARIAL
HELVETICA
CG-TIMES
COURIER
'COURIER NEW'
SERIF
TIMES

Text align:

CENTER
LEFT
RIGHT

Default fonts and CSS templates located in SYS1.SFSYDOCS

$B1 Default CSS

```
BODY {
font-family: verdana,arial,helvetica,serif;
font-weight: normal;
font-style: normal;
font-size: 12pt;
background-color: lightblue;
color: blue;
}
```

$B2 Default CSS

```
BODY {
font-family: courier;
font-weight: normal;
font-style: normal;
font-size: 10pt;
background-color: lightgray;
color: black;
}
```

$CP Default CSS

```
caption {
font-family: arial,helvetica,serif;
font-weight: normal;
font-style: normal;
font-size: 100%;
background-color: lightgray;
color: red;
}
```

$EM Default CSS

```
em {
font-family: courier;
font-weight: normal;
font-style: normal;
font-size: 100%;
text-align: center;
background-color: lightgray;
color: black;
}
```

$TB Default CSS

```
tb {
font-family: courier;
font-weight: normal;
font-style: normal;
font-size: 100%;
background-color: lightgray;
color: black;
text-align: left;
border-style: dotted;
padding: 1px;
width: 100%
}
```

$TD Default CSS

```
td {
font-family: courier;
font-weight: normal;
font-style: normal;
font-size: 100%;
text-align: left;
background-color: lightgray;
color: black;
}
```

$TH Default CSS

```
th {
font-family: courier;
font-weight: normal;
font-style: normal;
font-size: 100%;
background-color: lightgray;
color: blue;
}
```

$TR Default CSS

```
tr {
font-family: courier;
font-weight: normal;
font-style: normal;
font-size: 100%;
text-align: left;
background-color: lightgray;
color: blue;
}
```

Drill Down JCL requirements

The following sections describe the Drill Down JCL requirements at application run time.

FJIDOC0—Cascading Style Sheet (CSS) library

This ddname must point to Migration Utility's SYS1.SFSYDOCS PDS. This is an input file. Migration Utility loads the default CSS templates from this PDS.

Example:

```
//FJIDOC0  DD DSN=SYS1.SFSYDOCS,DISP=SHR
```

ddname for the Drill Down documents

The *&docname* name that you specify on the REPORT <DOC> *&docname*... statement is the ddname for the document file. It must be a PDS file with the following attributes:

LRECL=4096
RECFM=VB
DSORG=PO

At end of job, all reports belonging to the *&docname* documents are combined into a single PDS member and written to this PDS as *&dsname*(*&docname*). Make sure that you allocate enough space to hold all reports.

This single file contains all reports in HTML format. Each HTML page is preceded by a header record that specifies a Path ID (directory and file name).

Example:

```
MEMBER=rpt1/rpt2/i0000001.htm
```

The *&docfile* is recorded in EBCDIC. Download this file into a web server in text format. Then use the JAVA program, fsyjpars.class, to parse the file into browser-ready HTML documents. See "Running the Drill Down document parser—fsyjpars" on page 171.

Note: If the //*&docname* DD is not coded in the JCL, Migration Utility ignores this file.

ddname for the report files

When the "DYNALLOC=YES" EZPARAMS/EASYTRAN option is used, the REPORT files are allocated dynamically at run time. To avoid allocation problems, make sure that you specify a sufficient work space using the "WRKSPACE=" EZPARAMS/EASYTRAN option.

When "DYNALLOC=NO" is used, a file for each report must be added to the JCL. You must add 10 bytes to the record length of each Drill Down report. For example, if a report LRECL is 133 (132+1 for CC), then the total length is (133+10) = 143. Reports must be allocated to disk files as FBA files. Allocating temporary files is preferred.

As previously stated, at end of job, all reports are combined into a single document file and written out to the HFS (UNIX File System) on z/OS.

Using DYNALLOC=YES can save you unnecessary JCL overhead.

ddnames for the index/links validator program

FJWDOC0—Index cross-reference file

This file is created internally by the Drill Down logic when you use the VALIDATE option of the "REPORT <DOC>..." statement. The file is used to record, sort, and validate all links in the Drill Down HTML documents. It ensures that all pages are properly chained. The file must be allocated as follows:

```
//FJWDOC0  DD DSN=&FJWDOC0,
//         DISP=(OLD,CATLG,DELETE),
//         SPACE=(CYL,(?,?),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=)
```

Note that `DISP=(OLD...)` is required.

SORTIN—Index file sort ddname

This ddname must be coded whenever FJWDOC0 is required. It must point to the same DSN as the FJWDOC0 file. The file must be allocated as follows:

```
//SORTIN   DD DSN=&FJWDOC0,
//         DISP=SHR
```

FJEDOC0—Error messages

Index and links validator messages are written to this file. This can be a standard SYSOUT file. The LRECL=4096, RECFM=VBA, DSORG=PS.

Special note: When the FJWDOC0 file is required, before your job step is run, define it using a IEFBR14 step as follows:

```
//IEFBR14  EXEC PGM=IEFBR14
//FJWDOC0  DD DSN=&FJWDOC0,
//         DISP=(MOD,CATLG,DELETE),
//         SPACE=(CYL,(5,5),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=)
```

This prevents JCL errors in the Drill Down step.

HFS (UNIX files) requirements

Note: To use this feature, the z/OS UNIX environment and the z/OS Internet server must be activated on the z/OS system. A root directory on the UNIX system must be established for each user. For more information, consult your z/OS System administrator.

UNIX files are handled by the FSYUNIX1 Migration Utility program. This program is dynamically loaded at end of job when combining the Drill Down reports for distribution.

Code the ddnames shown below when you want to write HTML documents directly into a HFS (UNIX Directory) on the z/OS UNIX system.

Note: UNIX is case-sensitive; that is, commands, directory and file names must by typed exactly as shown.

Migration Utility checks the JCL for the FJUNIX0 ddname. If FJUNIX0 exists, Migration Utility assumes that the HTML documents are being written directly into the z/OS UNIX System.

The following ddnames are required when writing HTML documents directly into the z/OS UNIX system.

Note: In the examples shown, assume that the root directory is `/u/migutil/fsoft01`, and that *&docfile*=HTMLFIL1.

FJCONFG
The UNIX system configuration file used to determine the code set of each file type (ASCII or EBCDIC). See your UNIX system administrator about the location of the httpd.conf file.

Migration Utility uses the code set for the file types found in the httpd.conf file. In this way, the HTML documents are always in sync with the UNIX standards on your z/OS system.

Example:

```
//FJCONFG  DD PATHOPTS=(ORDONLY),
//         PATH='/u/vagen1/httpd.conf'
```

FJDMAP0
Log of directories and files created on the UNIX system. This is a standard SYSOUT file.

FJUNIX0
The output directory on the z/OS UNIX System where files are to be written. All HTML documents are written to this ddname. Note that PATH= must point to your root directory.

Example:

```
//FJUNIX0  DD PATHOPTS=(ORDONLY),
//         PATH='/u/migutil/fsoft01'
```

A subdirectory, *&docfile*, is created as coded on the REPORT <DOC> *&docfile*... statement. All subsequent directories and HTML documents are written in the created *&docfile* directory.

The *&docfile* directory is created as a new directory. If the &docfile directory already exists, the job is abnormally terminated.

To reuse a directory, execute the BPXBATCH program before the application step as follows:

```
//BPXBATCH EXEC PGM=BPXBATCH,
//  PARM='SH rm -r /u/migutil/fsoft01/htmlfil1'
```

Be extra careful. The above statements delete the specified directory and all subdirectories within it. It will not give you a second chance.

STDOUT
The BPXBATCH program stdout file. This is an optional file. Point PATH= to your own directory.

Example:

```
//STDOUT DD PATH='/u/migutil/fsoft01/fsyunix1.out',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIXGRP)
```

STDERR
The BPXBATCH program stderr file. This is an optional file. Point PATH= to your own directory.

Example:

```
//STDERR DD PATH='/u/migutil/fsoft01/fsyunix1.err',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIXGRP)
```

Tailoring the FSYFONTS table

To update the FSYFONTS table to support new attributes not included in the default table:

1. Edit the FSYFONTS source located in SYS1.SFSYEZTS. Entries in the table are coded by attribute category (FONT-STYLE, COLOR, BACKGROUND-COLOR, FONT-STYLE, etc). Add the new entries to each category using the same syntax as for the existing entries.
2. Assemble the FSYFONTS table using the JCASMFON job supplied in SYS1.SFSYJCLS. Make sure that you link into a commonly used library (preferably SYS1.SFSYLOAD).

Running the HTML1 document parser - fsybpars

fsybpars JAVA class creates directories from the downloaded special format HTML file created by Migration Utility due to HTML1 option on FILE definition of printer file.

To install fsybpars parser program:

1. On your PC or a server, create a directory for the HTML reports.

   Windows example:

   ```
   MD drilldown
   ```

2. Download from the mainframe (in BINARY format), SYS1.SFSYDOCS(fsybpars) to drilldown\fsybpars.class

To parse an HTML file on your PC or a server do the following:

1. Make sure that Java VM is installed and running.
2. Download the HTML1 document file from the mainframe (in BINARY format) into a directory (assume drilldown directory).
3. Switch to drilldown directory:

   Window example:(from command line)

   ```
   CD\drilldown
   ```
4. Enter on the command line:

   ```
   Java fsybpars &src &dest
   ```

   Where:

&src	Your downloaded Drill Down reports file.
&dest	The target directory for HTML documents.

   Example:

   ```
   Java fsybpars  htmlfil1.txt   htmlfil1
   ```

   On completion, all HTML documents are created in the /drilldown/htmlfil1 directory. To view from your browser, enter the URL that points to the created directory and click on the index.htm file.

   Note: Depending on the platform, you can get sophisticated using scripts.

Running the Drill Down document parser—fsyjpars

To install the parser program, fsyjpars:

1. On your PC or a server, create a directory for the Drill Down reports.

   Dos Windows example:

   ```
   MD drilldown
   ```
2. Download from the mainframe (in BINARY format), SYS1.SFSYDOCS(fsyjpars) to drilldown\fsyjpars.class

To parse a Drill Down file on your PC or a server do the following:

1. Make sure that Java VM is installed and running.
2. Download the HTML document file from the mainframe (in TEXT format) into a directory (assume drilldown directory).
3. Switch to drilldown directory:

   Dos Window example:

   ```
   CD\drilldown
   ```
4. Enter on the command line:

   ```
   Java fsyjpars &src &dest
   ```

   Where:

&src	Your downloaded Drill Down reports file.
&dest	The target directory for HTML documents.

   Example:

   ```
   Java fsyjpars  htmlfil1.txt   htmlfil1
   ```

On completion, all HTML documents are created in the /drilldown/htmlfil1 directory. To view from your browser, enter the URL that points to the created directory and click on the index.htm file.

Note: Depending on the platform, you can get sophisticated using scripts.

JCYUNIX0 (FSYUNIX0)—Drill Down utility for UNIX files

The FSYUNIX0 utility creates HTML documents on z/OS UNIX from a single PDS HTML document file created by your application on z/OS.

The utility provides an alternative to creating HFS z/OS UNIX files directly from your application program.

The utility runs as a standalone batch job. You can add it as a separate job step after your application program, or run it as a separate job.

Use the JCYUNIX0 member, located in SYS1.SFSYJCLS, as a template (see exact copy below).

Note: You must replace the PARM= with your own Directory Path.

```
//FSOFT01B JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(05,LT),REGION=0M
//*
//JOBLIB   DD  DSN=SYS1.SFSYLOAD,DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2005, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.     *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004, 2005.   *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.       *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/10/2005              WRITTEN BY: SRECKO LAZANJA         *
//*******************************************************************
//* FSYUNIX0 - Create/load UNIX files from a single file that       *
//*            contains one or more directory path header records.  *
//*                                                                 *
//*            Example of header record:                            *
//*                                                                 *
//*            MEMBER=homedir/report1.txt                           *
//*                                                                 *
//* hfs/unix files are created to designated directory.             *
//* The directory is provided via PARM=  on the exec statement.     *
//*                                                                 *
//*  example:                                                       *
//*       PARAM=('replace /u/migutil/fsoft01/htmlfl1')              *
//*       PARAM=('create /u/migutil/fsoft01/htmlfl1')               *
//*                                                                 *
//*  notes: use "replace" to replace the old directory if it        *
//*         exists, or create new directory if it does not.         *
//*                                                                 *
//*         use "create" to create a new directory. Job will run    *
//*         only if the directory does not exist.                   *
//*-----------------------------------------------------------------*
//FSYUNIX0 PROC   SYSOUT='*',PARAM=
//*-----------------------------------------------------------------*
//FSYUNIX0 EXEC PGM=FSYUNIX0,
//         PARM=&PARAM
//SYSOUT   DD SYSOUT=&SYSOUT
//*
//* Output log of created files
//FJDMAP0  DD SYSOUT=&SYSOUT
```

```
//*
//* Input flat VB file or a PDS member
//FJIDOC0  DD DSN=???????.INPUT.FILE,DISP=SHR
//*
//* UNIX config file (point to your own config)
//FJCONFG  DD PATHOPTS=(ORDONLY),
//         PATH='/u/vagen1/httpd.conf'
//*
//* UNIX stdout file (point to your own stdout)
//STDOUT   DD PATH='/u/migutil/fsoft01/fsyunix0.out',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIXGRP)
//*
//* UNIX stderr file (point to your own stderr)
//STDERR   DD PATH='/u/migutil/fsoft01/fsyunix0.err',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIXGRP)
// PEND
//*
//STEP000 EXEC PROC=FSYUNIX0,
//        PARAM=('replace /u/migutil/fsoft01/htmlfl1')
//
```

JCMUDRL1—Drill Down reports program

Program JCMUDRL1 creates two Drill Down documents, HTMLFL1 and HTMLFL2. It demonstrates how to create multiple Drill Down documents, Drill Down reports, and CSV reports in the same program. The program is set to run in Link and Go mode.

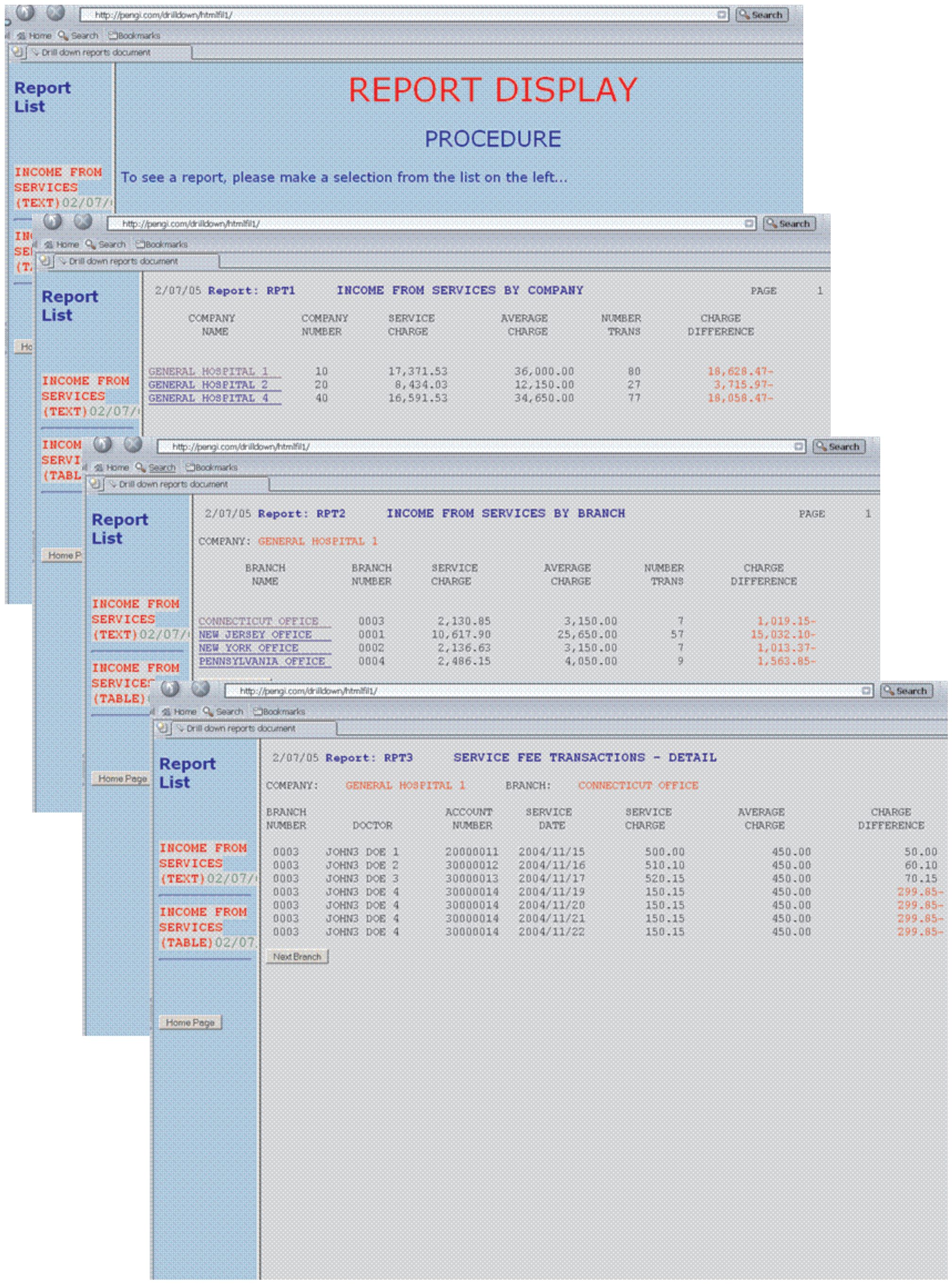
You can find a copy of JCMUDRL1 in the SYS1.SFSYJCL Migration Utility PDS library.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.      *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.         *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.        *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/20/2004              WRITTEN BY: SRECKO LAZANJA         *
//*******************************************************************
//* JCMUDRL1 - COMPILE, LINK  AND EXECUTE IN ONE STEP - INSTREAM PROC *
//*                                                                 *
//* THIS PROC DEMONSTRATES HOW TO RUN DRILL DOWN REPORTS.           *
//*******************************************************************
//FSPENGI  PROC  SYSOUT='*',
//         FILEIN=SYS1.SFSYEZTS(TESTDRL1),
//         SYSLIB1=SYS1.SFSYLOAD,                PENGIEZT/Migration Utility LOADLIB
//         PANDD=SYS1.SFSYEZTC,                  YOUR EZT MACROS
//         SYSIN=SYS1.SFSYEZTS,                  INPUT EZT PROGRAM
//         FJWDOC0=???????.TEMP.FJWDOC0.INDEX,   TEMPORARY INDEX FILE
//         HTMLFL1=???????.TEMP.HTMLFL1,         OUTPUT HTMLFILE
//         FJIDOC0=SYS1.SFSYDOCS,                HTML TEMPLATES
//         MEMBER=GO,                            PROGRAM NAME
//         PARAM=                                POSSIBLE USER PARMS
//*-----------------------------------------------------------------*
//IEFBR14  EXEC PGM=IEFBR14
//FJWDOC0  DD DSN=&FJWDOC0,
//         DISP=(MOD,CATLG,DELETE),
//         SPACE=(CYL,(5,5),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=)
```

```
//HTMLFL1  DD DSN=&HTMLFL1,
//         DISP=(MOD,CATLG,DELETE),
//         SPACE=(CYL,(20,30,64),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=,DSORG=PO)
//*-------------------------------------------------------------------*
//FSYTPA00 EXEC PGM=FSYTPA00,PARM=&PARAM
//SYSLIB   DD DSN=&SYSLIB1,
//         DISP=SHR
//* PANDD MUST POINT TO YOUR OWN Easytrieve Plus MACROS.
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=&PANDD,
//         DISP=SHR
//SYSOUT   DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//FJEDOC0  DD SYSOUT=&SYSOUT
//FJWDOC0  DD DSN=&FJWDOC0,
//         DISP=(OLD,CATLG,DELETE),
//         SPACE=(CYL,(5,5),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=)
//SORTIN   DD DSN=&FJWDOC0,
//         DISP=SHR
//HTMLFL1  DD DSN=&HTMLFL1,DISP=OLD
//FJIDOC0  DD DSN=&FJIDOC0,DISP=SHR
//FILEIN1  DD DSN=&FILEIN,
//         DISP=SHR
//SORTWK01 DD UNIT=SYSDA,
//         SPACE=(CYL,(20,50))
//SORTWK02 DD UNIT=SYSDA,
//         SPACE=(CYL,(20,50))
//* INPUT PROGRAM IF FROM EXTERNAL PDS
//SYSIN    DD DSN=&SYSIN(&MEMBER),
//         DISP=SHR
// PEND
//*
//STEP010  EXEC PROC=FSPENGI
//FSYTPA00.SYSIN  DD *
* EASYTRAN: PROCESS NOLIST,NOXREF,NOMAP
* EASYTRAN: DEBUG (LIST)
* EASYTRAN: PRINTER AUTOGEN
* END-EASYTRAN

FILE FILEIN1 DISK   F (80)
COMP-DEP    1   7    A
COMPANY     1   2    A
BRANCH      4   4    A
ACCOUNT     9   8    A
NAME       18  15    A
FEE        34  07 N  2
SRVDATE    42  10    A

FILE TEMPFIL VIRTUAL
CO-NAME     * 20 A    HEADING ('COMPANY' 'NAME')
BR-NAME     * 20 A    HEADING ('BRANCH'  'NAME')
COMPANY     *  2 A    HEADING ('COMPANY' 'NUMBER')
BRANCH      *  4 A    HEADING ('BRANCH' 'NUMBER')
ACCOUNT     *  8 A    HEADING ('ACCOUNT' 'NUMBER')
NAME        * 15 A    HEADING ('DOCTOR')
FEE         * 07 N  2 HEADING ('SERVICE' 'CHARGE')
SRVDATE     * 10 A    HEADING ('SERVICE' 'DATE')

FILE COMPTAB TABLE INSTREAM
ARG  1  1  A
DESC 4  20 A
10 GENERAL HOSPITAL 1
20 GENERAL HOSPITAL 2
30 GENERAL HOSPITAL 3
40 GENERAL HOSPITAL 4
ENDTABLE

FILE DEPTTAB TABLE INSTREAM
```

```
ARG  1  7  A
DESC 9  20 A
10 0001 NEW JERSEY OFFICE
10 0002 NEW YORK OFFICE
10 0003 CONNECTICUT OFFICE
10 0004 PENNSYLVANIA OFFICE
20 0001 ARIZONA OFFICES
20 0002 CALIFORNIA OFFICES
20 0003 OHIO OFFICES
20 0004 FLORIDA HEALTH
40 0001 MEXICO OFFICES
40 0002 ARKANSAS OFFICES
40 0003 NEW MEXICO COMPLEX
40 0004 ALABAMA SPECIALISTS
ENDTABLE

WFEEDIFF   W   5 P 2  HEADING ('CHARGE' 'DIFFERENCE')
WNOTRANS   W   2 P 0  HEADING ('NUMBER' 'TRANS')  VALUE 1
WFEEBASE   W   5 P 2  HEADING ('AVERAGE' 'CHARGE') VALUE 450.00
WPRCNDIF   S   3 P 2  HEADING ('PERCENT' 'DIFFERENCE').

JOB INPUT FILEIN1
  MOVE LIKE FILEIN1 TO TEMPFIL
  CO-NAME = 'UNKNOWN COMPANY'
  SEARCH COMPTAB WITH FILEIN1:COMPANY GIVING CO-NAME
  BR-NAME = 'UNKNOWN BRANCH'
  SEARCH DEPTTAB WITH FILEIN1:COMP-DEP GIVING BR-NAME
  PUT TEMPFIL
GOTO JOB

SORT TEMPFIL TO TEMPFIL USING (CO-NAME BR-NAME NAME SRVDATE)

JOB INPUT TEMPFIL
WFEEDIFF = (FEE - WFEEBASE)
WPRCNDIF = ((WFEEDIFF * 100) / WFEEBASE)
PRINT RPT1
PRINT RPT2
PRINT RPT3
PRINT RPT21
PRINT RPT22
PRINT RPT23
PRINT RPT31
PRINT RPT32
PRINT RPT33
*---------------------------------------------------------------------*

REPORT <DOC> HTMLFL1 VALIDATE               +
                     $B1 (BLUE)             +
                     $TB (BLUE)             +
                     $TR (BLUE)             +
                     $TH (BLUE)
DRILL MENU 1 #(RED BOLD)  +
      'INCOME FROM SERVICES (TEXT)' #GREEN SYSDATE
DRILL DOWN RPT1 CO-NAME RPT2 BR-NAME RPT3 DETAIL

DRILL MENU 2 #(RED BOLD)  +
           'INCOME FROM SERVICES (TABLE)' #GREEN SYSDATE
DRILL DOWN RPT21 CO-NAME RPT22 BR-NAME RPT23 DETAIL

REPORT <DOC> HTMLFL2 VALIDATE               +
                     $B1 (BLUE)             +
                     $TB (BLUE)             +
                     $TR (BLUE)             +
                     $TH (BLUE)
DRILL MENU 3 #(RED BOLD)  +
           'INCOME FROM SERVICES (CSV)' #GREEN SYSDATE
DRILL DOWN RPT31 CO-NAME RPT32 BR-NAME RPT33 DETAIL

*------- DRILL DOWN REPORTS (TEXT FORMAT) -----------------------------*
REPORT RPT1 SUMMARY NOADJUST DOCTYPE TEXT SKIP 0 LINESIZE 95  +
       SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1 LINKID 'Company'
```

```
CONTROL  CO-NAME
TITLE 1  #(BOLD 110% BLUE) COL 10 'Report: RPT1' +
         #(BOLD 120% BLUE) COL 28 'INCOME FROM SERVICES BY COMPANY'
LINE  1  CO-NAME    +
         COMPANY    +
         FEE        +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT2 SUMMARY NOADJUST DOCTYPE TEXT SKIP 0 LINESIZE 95  +
       SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1 LINKID 'Branch'
CONTROL BR-NAME
TITLE 1 #(BOLD 110% BLUE) COL 10 'Report: RPT2' +
        #(BOLD 120% BLUE) COL 28 'INCOME FROM SERVICES BY BRANCH'
TITLE 3 'COMPANY: ' #(RED) COL 10 CO-NAME
LINE  1  BR-NAME   +
         BRANCH    +
         FEE       +
         WFEEBASE  +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT3 NOADJUST DTLCTL (EVERY) PAGESIZE 40 LINESIZE  115 +
       DOCTYPE TEXT TITLESKIP 1 LINKID 'Detail'
CONTROL  DETAIL
TITLE 1 #(BOLD 110% BLUE) COL 10 'Report: RPT3' +
        #(BOLD 120% BLUE) COL 28 'SERVICE FEE TRANSACTIONS - DETAIL'
TITLE 3 'COMPANY: ' #(RED) CO-NAME ' BRANCH: ' #(RED) BR-NAME
LINE  1  BRANCH  +
         #(RED WHEN 'JOHN DOE 2') NAME    +
         ACCOUNT +
         SRVDATE +
         FEE     +
         WFEEBASE   +
         #(RED WHEN MINUS) WFEEDIFF   +
         #(RED WHEN MINUS) WPRCNDIF

*------- DRILL DOWN REPORTS (TABLE FORMAT) ----------------------------*
REPORT RPT21 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
            SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL  CO-NAME
TITLE 1  #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY COMPANY'
TITLE 2 'Report: RPT21'
LINE  1  CO-NAME    +
         COMPANY    +
         FEE        +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT22 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
       SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL BR-NAME
TITLE 1 #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY BRANCH'
TITLE 2 'Report: RPT22'
TITLE 3 'COMPANY: ' #(RED) COL 10 CO-NAME
LINE  1  BR-NAME   +
         BRANCH    +
         FEE       +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT23 NOADJUST DTLCTL (EVERY) PAGESIZE 40 LINESIZE  115 +
       DOCTYPE TABLE TITLESKIP 1  LINKID NO
CONTROL  DETAIL
TITLE 1 #(BOLD 120% BLUE) COL 24 'SERVICE FEE TRANSACTIONS - DETAIL'
TITLE 2 'Report: RPT23'
TITLE 3 'COMPANY: ' #(RED) CO-NAME ' BRANCH: ' #(RED) BR-NAME
LINE  1  BRANCH  +
         NAME    +
         ACCOUNT +
         SRVDATE +
         FEE     +
         WFEEBASE   +
         #(RED WHEN MINUS) WFEEDIFF   +
         #(RED WHEN MINUS) WPRCNDIF

*------- DRILL DOWN REPORTS (TABLE/CSV FORMAT) ------------------------*
REPORT RPT31 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
            SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL  CO-NAME
TITLE 1  #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY COMPANY'
TITLE 2 'Report: RPT31'
LINE  1  CO-NAME    +
         COMPANY    +
         FEE        +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT32 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
            SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL BR-NAME
TITLE 1 #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY BRANCH'
TITLE 2 'Report: RPT32'
TITLE 3 'COMPANY: ' #(RED) COL 10 CO-NAME
LINE  1  BR-NAME   +
         BRANCH    +
         FEE       +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS ) WFEEDIFF

REPORT RPT33 NOADJUST DTLCTL (EVERY) PAGESIZE 40 LINESIZE  115 +
            DOCTYPE CSV SEP=(',') TITLESKIP 1 SPACE 1
CONTROL  DETAIL
TITLE 1 COL 24 'SERVICE FEE TRANSACTIONS - DETAIL'
TITLE 2 ' '
TITLE 3 'COMPANY: '  CO-NAME ' BRANCH: ' BR-NAME
LINE  1  BRANCH  +
         NAME    +
         ACCOUNT +
         SRVDATE +
         FEE     +
         WFEEBASE   +
         WFEEDIFF   +
         WPRCNDIF
*----------------------- END OF PROGRAM -------------------------------*
```


http://pengi.com/drilldown/htmlfil1/
Search
Home
Search
Bookmarks
Drill down reports document
Report List
INCOME FROM SERVICES (TEXT) 02/07/
INCOME FROM SERVICES (TABLE)
Home Page
Next Branch

REPORT DISPLAY

PROCEDURE

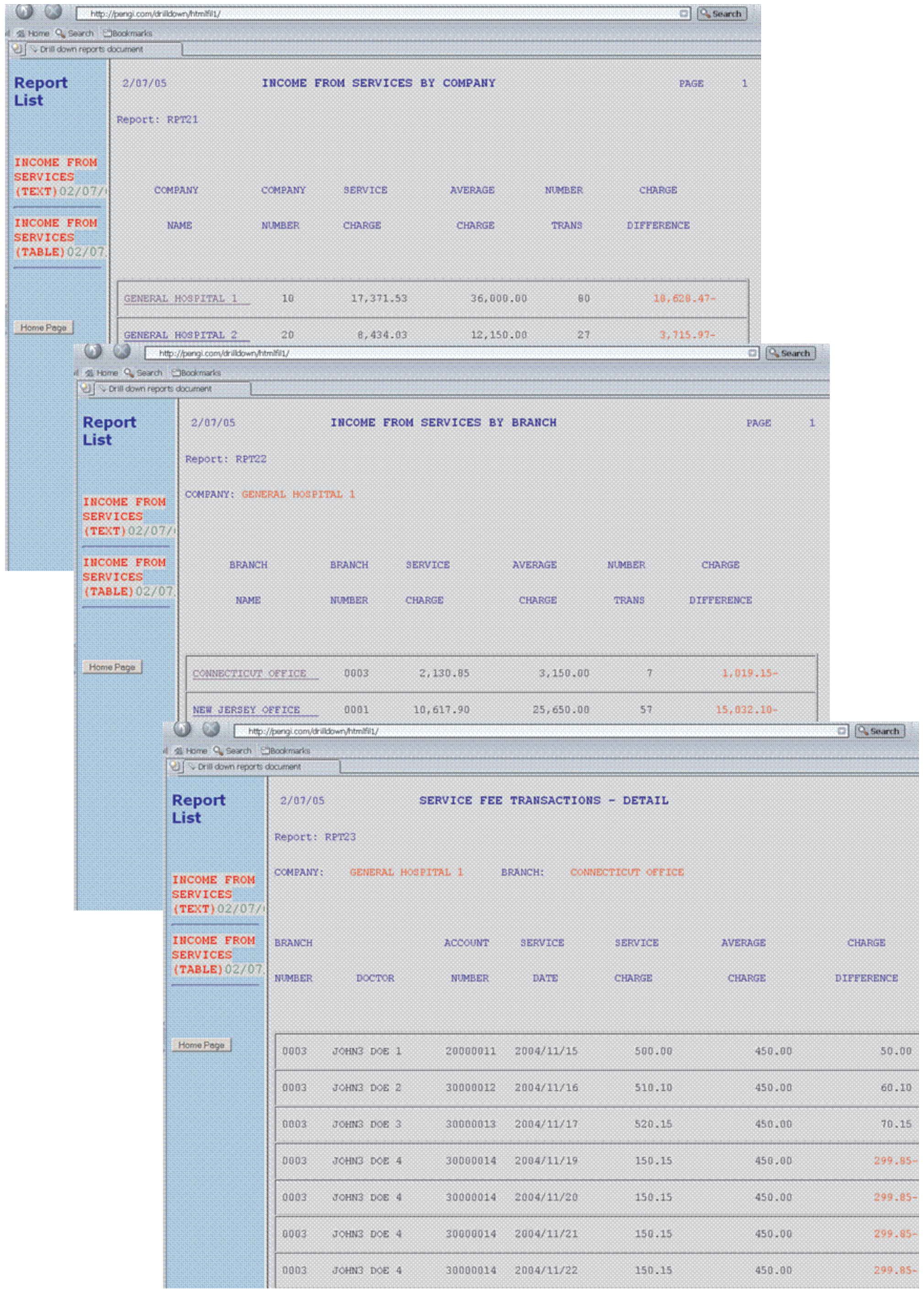
To see a report, please make a selection from the list on the left...

```
2/07/05 Report: RPT1      INCOME FROM SERVICES BY COMPANY                        PAGE      1

     COMPANY            COMPANY    SERVICE         AVERAGE      NUMBER       CHARGE
      NAME              NUMBER     CHARGE          CHARGE        TRANS      DIFFERENCE

GENERAL HOSPITAL 1         10     17,371.53       36,000.00        80        18,628.47-
GENERAL HOSPITAL 2         20      8,434.03       12,150.00        27         3,715.97-
GENERAL HOSPITAL 4         40     16,591.53       34,650.00        77        18,058.47-
```

```
2/07/05 Report: RPT2      INCOME FROM SERVICES BY BRANCH                         PAGE      1

COMPANY: GENERAL HOSPITAL 1

      BRANCH            BRANCH    SERVICE         AVERAGE      NUMBER       CHARGE
       NAME             NUMBER    CHARGE          CHARGE        TRANS      DIFFERENCE

CONNECTICUT OFFICE       0003     2,130.85        3,150.00         7         1,019.15-
NEW JERSEY OFFICE        0001    10,617.90       25,650.00        57        15,032.10-
NEW YORK OFFICE          0002     2,136.63        3,150.00         7         1,013.37-
PENNSYLVANIA OFFICE      0004     2,486.15        4,050.00         9         1,563.85-
```

```
2/07/05 Report: RPT3      SERVICE FEE TRANSACTIONS - DETAIL

COMPANY:    GENERAL HOSPITAL 1       BRANCH:     CONNECTICUT OFFICE

BRANCH                     ACCOUNT     SERVICE        SERVICE        AVERAGE         CHARGE
NUMBER       DOCTOR         NUMBER      DATE          CHARGE         CHARGE        DIFFERENCE

 0003    JOHN3 DOE 1      20000011   2004/11/15       500.00         450.00          50.00
 0003    JOHN3 DOE 2      30000012   2004/11/16       510.10         450.00          60.10
 0003    JOHN3 DOE 3      30000013   2004/11/17       520.15         450.00          70.15
 0003    JOHN3 DOE 4      30000014   2004/11/19       150.15         450.00         299.85-
 0003    JOHN3 DOE 4      30000014   2004/11/20       150.15         450.00         299.85-
 0003    JOHN3 DOE 4      30000014   2004/11/21       150.15         450.00         299.85-
 0003    JOHN3 DOE 4      30000014   2004/11/22       150.15         450.00         299.85-
```

Figure 2. Generated Drill Down reports (HTMLFL1 document only) (Part 1 of 2)


http://pengi.com/drilldown/htmlfil1/
Search
Home
Search
Bookmarks
Drill down reports document
Report List
INCOME FROM SERVICES (TEXT) 02/07/
INCOME FROM SERVICES (TABLE) 02/07
Home Page

```
2/07/05             INCOME FROM SERVICES BY COMPANY                PAGE    1

Report: RPT21

   COMPANY        COMPANY     SERVICE        AVERAGE      NUMBER      CHARGE

    NAME          NUMBER      CHARGE          CHARGE       TRANS    DIFFERENCE

 GENERAL HOSPITAL 1    10    17,371.53      36,000.00       80      18,628.47-

 GENERAL HOSPITAL 2    20     8,434.03      12,150.00       27       3,715.97-
```

```
2/07/05             INCOME FROM SERVICES BY BRANCH                 PAGE    1

Report: RPT22

COMPANY: GENERAL HOSPITAL 1

     BRANCH         BRANCH     SERVICE        AVERAGE      NUMBER      CHARGE

      NAME          NUMBER     CHARGE          CHARGE       TRANS    DIFFERENCE

 CONNECTICUT OFFICE  0003      2,130.85       3,150.00        7       1,019.15-

 NEW JERSEY OFFICE   0001     10,617.90      25,650.00       57      15,032.10-
```

```
2/07/05             SERVICE FEE TRANSACTIONS - DETAIL

Report: RPT23

COMPANY:    GENERAL HOSPITAL 1      BRANCH:    CONNECTICUT OFFICE

BRANCH                   ACCOUNT    SERVICE      SERVICE      AVERAGE       CHARGE

NUMBER     DOCTOR        NUMBER      DATE        CHARGE       CHARGE      DIFFERENCE

 0003    JOHN3 DOE 1    20000011  2004/11/15     500.00       450.00         50.00

 0003    JOHN3 DOE 2    30000012  2004/11/16     510.10       450.00         60.10

 0003    JOHN3 DOE 3    30000013  2004/11/17     520.15       450.00         70.15

 0003    JOHN3 DOE 4    30000014  2004/11/19     150.15       450.00        299.85-

 0003    JOHN3 DOE 4    30000014  2004/11/20     150.15       450.00        299.85-

 0003    JOHN3 DOE 4    30000014  2004/11/21     150.15       450.00        299.85-

 0003    JOHN3 DOE 4    30000014  2004/11/22     150.15       450.00        299.85-
```

Figure 3. Generated Drill Down reports (HTMLFL1 document only) (Part 2 of 2)

JCMUDRL2—Creates a CSV file and a CSV report

Program JCMUDRL2 creates a CSV file and a CSV report. It demonstrates how to create plain CSV files and reports (without using the Drill Down syntax).

The program is set to run in Link and Go mode.

JCMUDRL2—Creates a CSV file and a CSV report

You can find a copy of JCMUDRL1 in the SYS1.SFSYJCL Migration Utility PDS library.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.     *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.         *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.       *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/20/2004              WRITTEN BY: SRECKO LAZANJA         *
//*******************************************************************
//* JCMUDRL2 - COMPILE, LINK  AND EXECUTE IN ONE STEP - INSTREAM PROC *
//*                                                                 *
//* THIS PROC DEMONS HOW TO CREATE CHARACTER SEPARATED VALUE FILES.   *
//*******************************************************************
//FSPENGI  PROC  SYSOUT='*',
//         FILEIN=SYS1.SFSYEZTS(TESTDRL1),
//         SYSLIB1=SYS1.SFSYLOAD,                 PENGIEZT/Migration Utility LOADLIB
//         PANDD=SYS1.SFSYEZTC,                   YOUR EZT MACROS
//         SYSIN=SYS1.SFSYEZTS,                   INPUT EZT PROGRAM
//         CSVFIL1=???????.TEST.CSVFIL1,          CSV FILE 1
//         CSVRPT1=???????.TEST.CSVRPT1,          CSV FILE 2
//         MEMBER=GO,                             PROGRAM NAME
//         PARAM=                                 POSSIBLE USER PARMS
//*-----------------------------------------------------------------*
//IEFBR14  EXEC PGM=IEFBR14
//CSVRPT1  DD DSN=&CSVRPT1,
//         DISP=(MOD,DELETE,DELETE),
//         SPACE=(CYL,(1,5),RLSE),
//         DCB=(LRECL=133,RECFM=FB,BLKSIZE=)
//CSVFIL1  DD DSN=&CSVFIL1,
//         DISP=(MOD,DELETE,DELETE),
//         SPACE=(CYL,(2,5),RLSE),
//         DCB=(LRECL=120,RECFM=VB,BLKSIZE=,DSORG=PO)
//*-----------------------------------------------------------------*
//FSYTPA00 EXEC PGM=FSYTPA00,PARM=&PARAM
//SYSLIB   DD DSN=&SYSLIB1,
//         DISP=SHR
//* PANDD MUST POINT TO YOUR OWN Easytrieve Plus MACROS.
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=&PANDD,
//         DISP=SHR
//SYSOUT   DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//CSVFIL1  DD DSN=&CSVFIL1,
//         DISP=(NEW,CATLG,DELETE),
//         SPACE=(CYL,(5,5),RLSE),
//         DCB=(LRECL=100,RECFM=FB,BLKSIZE=)
//CSVRPT1  DD DSN=&CSVRPT1,
//         DISP=(NEW,CATLG,DELETE),
//         SPACE=(CYL,(5,5),RLSE),
//         DCB=(LRECL=133,RECFM=FBA,BLKSIZE=)
//FILEIN1  DD DSN=&FILEIN,
//         DISP=SHR
//* INPUT PROGRAM IF FROM EXTERNAL PDS
//SYSIN    DD DSN=&SYSIN(&MEMBER),
//         DISP=SHR
// PEND
//*
//GENCSV   EXEC PROC=FSPENGI
//FSYTPA00.SYSIN  DD *
* EASYTRAN: PROCESS LIST,XREF,MAP
* EASYTRAN: DEBUG (NOLIST)
* EASYTRAN: DEBUG (COBOL BLIST LKED)
* EASYTRAN: PRINTER AUTOGEN
* END-EASYTRAN

FILE CSVRPT1 PRINTER

FILE CSVFIL1 F (100)

FILE FILEIN1 DISK   F (80)
COMP-DEP    1   7    A
COMPANY     1   2    A HEADING ('COMPANY' 'NUMBER')
BRANCH      4   4    A HEADING ('BRANCH' 'NUMBER')
ACCOUNT     9   8    A HEADING ('ACCOUNT' 'NUMBER')
NAME       18  15    A HEADING ('DOCTOR')
FEE        34  07 N  2 HEADING ('SERVICE' 'CHARGE')
SRVDATE    42  10    A HEADING ('SERVICE' 'DATE')

FILE COMPTAB TABLE INSTREAM
ARG  1  1  A
DESC 4  20 A
10 GENERAL HOSPITAL 1
20 GENERAL HOSPITAL 2
30 GENERAL HOSPITAL 3
40 GENERAL HOSPITAL 4
ENDTABLE

FILE DEPTTAB TABLE INSTREAM
ARG  1  7  A
DESC 9  20 A
10 0001 NEW JERSEY OFFICE
10 0002 NEW YORK OFFICE
10 0003 CONNECTICUT OFFICE
10 0004 PENNSYLVANIA OFFICE
20 0001 ARIZONA OFFICES
20 0002 CALIFORNIA OFFICES
20 0003 OHIO OFFICES
20 0004 FLORIDA HEALTH
40 0001 MEXICO OFFICES
40 0002 ARKANSAS OFFICES
40 0003 NEW MEXICO COMPLEX
40 0004 ALABAMA SPECIALISTS
ENDTABLE

WBONUS     W   4 P 2  HEADING ('BONUS')
WBONBAS    S   4 P 2  HEADING ('BONUS' 'BASE') VALUE 500.00
WPBONUS    S   4 P 2  HEADING ('%BONUS') VALUE .10
WCO-NAME   W  20 A    HEADING ('COMPANY' 'NAME')
WBR-NAME   W  20 A    HEADING ('BRANCH'  'NAME')
WHFLAG     W   1 A    VALUE 'Y'

JOB INPUT FILEIN1
WCO-NAME = 'UNKNOWN COMPANY'
SEARCH COMPTAB WITH COMPANY GIVING WCO-NAME
WBR-NAME = 'UNKNOWN BRANCH'
SEARCH DEPTTAB WITH COMP-DEP GIVING WBR-NAME
WBONUS = ((FEE - WBONBAS) * WPBONUS)

IF WHFLAG = 'Y'
   DISPLAY CSVFIL1    +
           SEP=(', ') +
          'COMPANY'   +
          'BRANCH'    +
          'NAME'      +
          'ACCOUNT'   +
          'FEE'       +
          'BON-BASE'  +
          '%BONUS'    +
          'BONUS'
   WHFLAG = 'N'
END-IF
DISPLAY CSVFIL1    +
        SEP=(', ') +
        COMPANY    +
        BRANCH     +
        NAME       +
        ACCOUNT    +
        FEE        +
        WBONBAS    +
        WPBONUS    +
        WBONUS

PRINT RPT1
*-------------------------------------------------------------------*

REPORT RPT1 PRINTER CSVRPT1 NOADJUST DTLCTL(EVERY) +
            PAGESIZE 0 SEP=(',') TITLESKIP 1 NOSPREAD
SEQUENCE WCO-NAME WBR-NAME NAME
CONTROL  FINAL
TITLE 1 COL 28 'SERVICE FEES BY DOCTOR'
TITLE 2 ' '
TITLE 3 'COMPANY: ' WCO-NAME ' BRANCH: ' WBR-NAME
LINE  1  COMPANY    +
         BRANCH     +
         NAME       +
         ACCOUNT    +
         FEE        +
         WBONBAS    +
         WPBONUS    +
         WBONUS
*----------------------- END OF PROGRAM -------------------------------*
```

The output CSVFIL1 file would be formatted as follows:

```
COMPANY, BRANCH, NAME, ACCOUNT, FEE, BON-BASE, %BONUS, BONUS,
10, 0001, JOHN DOE 1     , 10000011,    500.00,    500.00,       .10,       .00,
10, 0001, JOHN DOE 1     , 10000011,    510.00,    500.00,       .10,      1.00,
10, 0001, JOHN DOE 1     , 10000011,    520.00,    500.00,       .10,      2.00,
10, 0001, JOHN DOE 1     , 10000011,    400.00,    500.00,       .10,    -10.00,
10, 0001, JOHN DOE 2     , 10000012,    510.10,    500.00,       .10,      1.01,
10, 0001, JOHN DOE 3     , 10000013,    520.15,    500.00,       .10,      2.01,
10, 0001, JOHN DOE 4     , 10000014,    150.15,    500.00,       .10,    -34.98,
```

The output CSVRPT1 file would be formatted as follows (only first 80 columns shown):

```
 1/01/05                    SERVICE FEES BY DOCTOR

COMPANY:    GENERAL HOSPITAL 1      BRANCH:    CONNECTICUT OFFICE

COMPANY   ,   BRANCH   ,                      ,   ACCOUNT    ,     SERVICE      ,
NUMBER    ,   NUMBER   ,       DOCTOR         ,    NUMBER    ,      CHARGE      ,

  10      ,    0003    ,   JOHN3 DOE 1        ,   20000011   ,         500.00   ,
  10      ,    0003    ,   JOHN3 DOE 2        ,   30000012   ,         510.10   ,
  10      ,    0003    ,   JOHN3 DOE 3        ,   30000013   ,         520.15   ,
  10      ,    0003    ,   JOHN3 DOE 4        ,   30000014   ,         150.15   ,
  10      ,    0003    ,   JOHN3 DOE 4        ,   30000014   ,         150.15   ,
  10      ,    0003    ,   JOHN3 DOE 4        ,   30000014   ,         150.15   ,
  10      ,    0003    ,   JOHN3 DOE 4        ,   30000014   ,         150.15   ,
```

JCMUDRLU—Drill Down reports and UNIX environment

The JCMUDRLU program creates two Drill Down documents, HTMLFL1 and HTMLFL2. It demonstrates how to create multiple Drill Down documents, Drill Down reports and CSV reports in the same program.

The output Drill Down reports are written to HFS (UNIX files) on z/OS and standard PDS for download. The program is set to run in Link and Go mode.

JCMUDRLU—Drill Down reports and UNIX environment

You can find a copy of JCMUDRL1 in the SYS1.SFSYJCL Migration Utility PDS library.

The JCMUDRLU program is the same as the JCMUDRL1 example except for the HFS (UNIX files) part of the JCL.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2005, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.      *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004, 2005.    *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.        *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/10/2005              WRITTEN BY: SRECKO LAZANJA         *
//*******************************************************************
//* JCMUDRLU - COMPILE, LINK  AND EXECUTE IN ONE STEP - INSTREAM PROC *
//*                                                                 *
//* NOTES: EZT PROGRAM IS AT THE BOTTOM OF THIS PROC                *
//*                                                                 *
//* This job demonstrates how to write Drill Down Reports directly to *
//* MVS UNIX (HFS) directory. Drill Down Reports can then be accessed *
//* by the WEB Browsers directly via z/OS server.                   *
//*                                                                 *
//*******************************************************************
//FSPENGI  PROC  SYSOUT='*',
//         FILEIN=SYS1.SFSYEZTS(TESTDRL1),
//         SYSLIB1=SYS1.SFSYLOAD,                PENGIEZT/Migration Utility LOADLIB
//         PANDD=SYS1.SFSYEZTC,                  YOUR EZT MACROS
//         SYSIN=SYS1.SFSYEZTS,                  INPUT EZT PROGRAM
//         FJWDOC0=???????.TEMP.FJWDOC0.INDEX,     TEMPORARY INDEX FILE
//         HTMLFL1=???????.TEMP.HTMLFL1,          OUTPUT HTMLFILE
//         HTMLFL2=???????.TEMP.HTMLFL2,          OUTPUT HTMLFILE
//         FJIDOC0=SYS1.SFSYDOCS,                HTML TEMPLATES
//         MEMBER=GO,                            PROGRAM NAME
//         PARAM=                                POSSIBLE USER PARMS
//*-----------------------------------------------------------------*
//* prepare files before run.                                       *
//*-----------------------------------------------------------------*
//IEFBR14  EXEC PGM=IEFBR14
//* temporary work file to hold index/anchor information
//FJWDOC0  DD DSN=&FJWDOC0,
//         DISP=(MOD,CATLG,DELETE),
//         SPACE=(CYL,(5,5),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=)
//* output PDS for drilldown report named HTMLFL1.
//HTMLFL1  DD DSN=&HTMLFL1,
//         DISP=(MOD,CATLG,DELETE),
//         SPACE=(CYL,(20,30,64),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=,DSORG=PO)
//* output PDS for drilldown report named HTMLFL2.
//HTMLFL2  DD DSN=&HTMLFL2,
//         DISP=(MOD,CATLG,DELETE),
//         SPACE=(CYL,(20,30,64),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=,DSORG=PO)
//*-----------------------------------------------------------------*
//FSYTPA00 EXEC PGM=FSYTPA00,PARM=&PARAM
//SYSLIB   DD DSN=&SYSLIB1,
//         DISP=SHR
//* PANDD MUST POINT TO YOUR OWN Easytrieve Plus MACROS.
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=&PANDD,
//         DISP=SHR
//SYSOUT   DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
```

```
//SYSPRINT DD SYSOUT=&SYSOUT
//*
//*-------------------------------------------------------------------*
//* files needed by drilldown application program.                    *
//*-------------------------------------------------------------------*
//*
//* index validation error listing (used only for VALIDATE option)
//FJEDOC0  DD SYSOUT=&SYSOUT
//*
//* log of directories and files created on unix system
//FJDMAP0  DD SYSOUT=&SYSOUT
//*
//* temporary work file to hold index/anchor information
//FJWDOC0  DD DSN=&FJWDOC0,
//         DISP=(OLD,CATLG,DELETE),
//         SPACE=(CYL,(5,5),RLSE),
//         DCB=(LRECL=4096,RECFM=VB,BLKSIZE=)
//*
//* sort work file. must point to the same file as FJWDOC0 ABOVE.
//SORTIN   DD DSN=&FJWDOC0,
//         DISP=SHR
//*
//* output PDS for drilldown report named HTMLFL1.
//HTMLFL1  DD DSN=&HTMLFL1,DISP=OLD
//*
//* output PDS for drilldown report named HTMLFL2.
//HTMLFL2  DD DSN=&HTMLFL2,DISP=OLD
//*
//* input pds where Migration Utility html templates are located (sfsydocs)
//FJIDOC0  DD DSN=&FJIDOC0,DISP=SHR
//*
//FILEIN1  DD DSN=&FILEIN,
//         DISP=SHR
//*
//* INPUT PROGRAM (Migration Utility drilldown program) IF FROM EXTERNAL PDS
//SYSIN    DD DSN=&SYSIN(&MEMBER),
//         DISP=SHR
//*
//* UNIX system configuration file used to determine code ascii/ebcdic
//* Consult with UNIX system administrator for httpd.conf location.
//FJCONFG  DD PATHOPTS=(ORDONLY),
//         PATH='/u/vagen1/httpd.conf'
//*
//* Output directory on UNIX. All reports are written to this ddname.
//* Point PATH= to your own directory.
//FJUNIX0  DD PATHOPTS=(ORDONLY),
//         PATH='/u/migutil/fsoft01'
//*
//* BPXBATCH Program stdout file. This is an optional file.
//* Point PATH= to your own directory.
//STDOUT   DD PATH='/u/migutil/fsoft01/fsyunix1.out',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIXGRP)
//*
//* BPXBATCH Program stderr file. This is an optional file.
//* Point PATH= to your own directory.
//STDERR   DD PATH='/u/migutil/fsoft01/fsyunix1.err',
//         PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//         PATHMODE=(SIRWXU,SIXGRP)
//*
//* sort work files are needed for sorting index records in addition
//* to potential sorts(s) in Migration Utility program.
//SORTWK01 DD UNIT=SYSDA,
//         SPACE=(CYL,(20,50))
//*
//SORTWK02 DD UNIT=SYSDA,
//         SPACE=(CYL,(20,50))
//*
// PEND
//*
//STEP010  EXEC PROC=FSPENGI
```

```
//FSYTPA00.SYSIN  DD *
* EASYTRAN: PROCESS NOLIST,NOXREF,NOMAP
* EASYTRAN: DEBUG (LIST)
* EASYTRAN: PRINTER AUTOGEN
* END-EASYTRAN

FILE FILEIN1 DISK   F (80)
COMP-DEP    1   7    A
COMPANY     1   2    A
BRANCH      4   4    A
ACCOUNT     9   8    A
NAME       18  15    A
FEE        34  07 N  2
SRVDATE    42  10    A

FILE TEMPFIL VIRTUAL
CO-NAME     * 20 A    HEADING ('COMPANY' 'NAME')
BR-NAME     * 20 A    HEADING ('BRANCH'  'NAME')
COMPANY     *  2 A    HEADING ('COMPANY' 'NUMBER')
BRANCH      *  4 A    HEADING ('BRANCH' 'NUMBER')
ACCOUNT     *  8 A    HEADING ('ACCOUNT' 'NUMBER')
NAME        * 15 A    HEADING ('DOCTOR')
FEE         * 07 N  2 HEADING ('SERVICE' 'CHARGE')
SRVDATE     * 10 A    HEADING ('SERVICE' 'DATE')

FILE COMPTAB TABLE INSTREAM
ARG  1  1  A
DESC 4  20 A
10 GENERAL HOSPITAL 1
20 GENERAL HOSPITAL 2
30 GENERAL HOSPITAL 3
40 GENERAL HOSPITAL 4
ENDTABLE

FILE DEPTTAB TABLE INSTREAM
ARG  1  7  A
DESC 9  20 A
10 0001 NEW JERSEY OFFICE
10 0002 NEW YORK OFFICE
10 0003 CONNECTICUT OFFICE
10 0004 PENNSYLVANIA OFFICE
20 0001 ARIZONA OFFICES
20 0002 CALIFORNIA OFFICES
20 0003 OHIO OFFICES
20 0004 FLORIDA HEALTH
40 0001 MEXICO OFFICES
40 0002 ARKANSAS OFFICES
40 0003 NEW MEXICO COMPLEX
40 0004 ALABAMA SPECIALISTS
ENDTABLE

WFEEDIFF   W   5 P 2  HEADING ('CHARGE' 'DIFFERENCE')
WNOTRANS   W   2 P 0  HEADING ('NUMBER' 'TRANS')  VALUE 1
WFEEBASE   W   5 P 2  HEADING ('AVERAGE' 'CHARGE') VALUE 450.00
WPRCNDIF   S   3 P 2  HEADING ('PERCENT' 'DIFFERENCE').

JOB INPUT FILEIN1
  MOVE LIKE FILEIN1 TO TEMPFIL
  CO-NAME = 'UNKNOWN COMPANY'
  SEARCH COMPTAB WITH FILEIN1:COMPANY GIVING CO-NAME
  BR-NAME = 'UNKNOWN BRANCH'
  SEARCH DEPTTAB WITH FILEIN1:COMP-DEP GIVING BR-NAME
  PUT TEMPFIL
GOTO JOB

SORT TEMPFIL TO TEMPFIL USING (CO-NAME BR-NAME NAME SRVDATE)

JOB INPUT TEMPFIL
WFEEDIFF = (FEE - WFEEBASE)
WPRCNDIF = ((WFEEDIFF * 100) / WFEEBASE)
PRINT RPT1
PRINT RPT2
PRINT RPT3
PRINT RPT21
PRINT RPT22
PRINT RPT23
PRINT RPT31
PRINT RPT32
PRINT RPT33
*----------------------------------------------------------------------*

REPORT <DOC> HTMLFL1 VALIDATE               +
                     $B1 (BLUE)             +
                     $TB (BLUE)             +
                     $TR (BLUE)             +
                     $TH (BLUE)
DRILL MENU 1 #(RED BOLD)  +
      'INCOME FROM SERVICES (TEXT)' #GREEN SYSDATE
DRILL DOWN RPT1 CO-NAME RPT2 BR-NAME RPT3 DETAIL

DRILL MENU 2 #(RED BOLD)  +
           'INCOME FROM SERVICES (TABLE)' #GREEN SYSDATE
DRILL DOWN RPT21 CO-NAME RPT22 BR-NAME RPT23 DETAIL

REPORT <DOC> HTMLFL2 VALIDATE               +
                     $B1 (BLUE)             +
                     $TB (BLUE)             +
                     $TR (BLUE)             +
                     $TH (BLUE)
DRILL MENU 3 #(RED BOLD)  +
           'INCOME FROM SERVICES (CSV)' #GREEN SYSDATE
DRILL DOWN RPT31 CO-NAME RPT32 BR-NAME RPT33 DETAIL

*------- DRILL DOWN REPORTS (TEXT FORMAT) ------------------------------*
REPORT RPT1 SUMMARY NOADJUST DOCTYPE TEXT SKIP 0 LINESIZE 95  +
       SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1 LINKID 'Company'
CONTROL  CO-NAME
TITLE 1  #(BOLD 110% BLUE) COL 10 'Report: RPT1' +
         #(BOLD 120% BLUE) COL 28 'INCOME FROM SERVICES BY COMPANY'
LINE  1  CO-NAME    +
         COMPANY    +
         FEE        +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT2 SUMMARY NOADJUST DOCTYPE TEXT SKIP 0 LINESIZE 95  +
       SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1 LINKID 'Branch'
CONTROL BR-NAME
TITLE 1 #(BOLD 110% BLUE) COL 10 'Report: RPT2' +
        #(BOLD 120% BLUE) COL 28 'INCOME FROM SERVICES BY BRANCH'
TITLE 3 'COMPANY: ' #(RED) COL 10 CO-NAME
LINE  1  BR-NAME   +
         BRANCH    +
         FEE       +
         WFEEBASE  +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT3 NOADJUST DTLCTL (EVERY) PAGESIZE 40 LINESIZE  115 +
       DOCTYPE TEXT TITLESKIP 1 LINKID 'Detail'
CONTROL  DETAIL
TITLE 1 #(BOLD 110% BLUE) COL 10 'Report: RPT3' +
        #(BOLD 120% BLUE) COL 28 'SERVICE FEE TRANSACTIONS - DETAIL'
TITLE 3 'COMPANY: ' #(RED) CO-NAME ' BRANCH: ' #(RED) BR-NAME
LINE  1  BRANCH  +
         #(RED WHEN 'JOHN DOE 2') NAME    +
         ACCOUNT +
         SRVDATE +
         FEE     +
         WFEEBASE   +
         #(RED WHEN MINUS) WFEEDIFF   +
         #(RED WHEN MINUS) WPRCNDIF

*------- DRILL DOWN REPORTS (TABLE FORMAT) ----------------------------*
REPORT RPT21 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
            SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL  CO-NAME
TITLE 1  #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY COMPANY'
TITLE 2 'Report: RPT21'
LINE  1  CO-NAME    +
         COMPANY    +
         FEE        +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT22 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
       SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL BR-NAME
TITLE 1 #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY BRANCH'
TITLE 2 'Report: RPT22'
TITLE 3 'COMPANY: ' #(RED) COL 10 CO-NAME
LINE  1  BR-NAME   +
         BRANCH    +
         FEE       +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT23 NOADJUST DTLCTL (EVERY) PAGESIZE 40 LINESIZE  115 +
       DOCTYPE TABLE TITLESKIP 1  LINKID NO
CONTROL  DETAIL
TITLE 1 #(BOLD 120% BLUE) COL 24 'SERVICE FEE TRANSACTIONS - DETAIL'
TITLE 2 'Report: RPT23'
TITLE 3 'COMPANY: ' #(RED) CO-NAME ' BRANCH: ' #(RED) BR-NAME
LINE  1  BRANCH  +
         NAME    +
         ACCOUNT +
         SRVDATE +
         FEE     +
         WFEEBASE   +
         #(RED WHEN MINUS) WFEEDIFF   +
         #(RED WHEN MINUS) WPRCNDIF

*------- DRILL DOWN REPORTS (TABLE/CSV FORMAT) ------------------------*
REPORT RPT31 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
            SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL  CO-NAME
TITLE 1  #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY COMPANY'
TITLE 2 'Report: RPT31'
LINE  1  CO-NAME    +
         COMPANY    +
         FEE        +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS) WFEEDIFF

REPORT RPT32 SUMMARY NOADJUST DOCTYPE TABLE SKIP 0 LINESIZE 95  +
            SUMCTL (ALL DTLCOPY) PAGESIZE 0 TITLESKIP 1
CONTROL BR-NAME
TITLE 1 #(BOLD 120% BLUE) COL 24 'INCOME FROM SERVICES BY BRANCH'
TITLE 2 'Report: RPT32'
TITLE 3 'COMPANY: ' #(RED) COL 10 CO-NAME
LINE  1  BR-NAME   +
         BRANCH    +
         FEE       +
         WFEEBASE   +
         WNOTRANS   +
         #(RED WHEN MINUS ) WFEEDIFF

REPORT RPT33 NOADJUST DTLCTL (EVERY) PAGESIZE 0 LINESIZE  115 +
            DOCTYPE CSV SEP=(',') TITLESKIP 1 SPACE 1
```

```
CONTROL  DETAIL
TITLE 1 COL 24 'SERVICE FEE TRANSACTIONS - DETAIL'
TITLE 2 ' '
TITLE 3 'COMPANY: '  CO-NAME ' BRANCH: ' BR-NAME
LINE  1  BRANCH  +
         NAME    +
         ACCOUNT +
         SRVDATE +
         FEE     +
         WFEEBASE   +
         WFEEDIFF   +
         WPRCNDIF
*----------------------- END OF PROGRAM -------------------------------*
```

Chapter 10. User exits

End of translating macro exit

Optionally, users 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 necessary 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.

File I/O Exits

The FILE EXIT statement allows you to:

- Generate a MODIFY EXIT when the MODIFY option is specified on the file EXIT statement. With this exit, I/O is performed by the generated COBOL. Your exit is invoked to prescreen or modify the input and output records.
- Generate a non-MODIFY EXIT when the MODIFY option is not specified on the file EXIT statement. Your exit program is responsible for the file I/O.

The EXIT can be specified for VSAM and sequential files only. Easytrieve Plus does not support EXIT for DLI/IMS, IDMS, and DB2.

All exits must be compiled as REUSABLE and linked with 31-bit or ANY addressing mode. The run mode can be 24-bit or 31-bit run mode.

Exit calling conventions

The exits are invoked via a COBOL call using the standard IBM linkage conventions. That is, register 1 (R1) points to a list of parameter pointers upon entry into your program.

In this example:

```
CALL  PROGRAM1 USING FIELD1 FIELD2 FIELD3
```

R1 would point to:

```
                              DC   A(FIELD1)
                              DC   A(FIELD2)
                              DC   A(FIELD3)
```

The high order bit of the last pointer would contain X'80'.

To receive these parameters in your COBOL exit, you must code a 01 level in the LINKAGE SECTION for each parameter, and establish addressability to each on the PROCEDURE statement. For example, to access the above 3 parameters you would code:

```
LINKAGE SECTION.
01 FIELD1 PIC    ...
01 FIELD2 PIC    ...
01 FIELD3 PIC    ...

PROCEDURE DIVISION USING FIELD1 FIELD2 FIELD3.
```

From a BAL program, each pointer can be accessed off R1. For example:

```
L    R4,0(R1)  Location of FIELD1
L    R5,4(R1)  Location of FIELD2
L    R6,8(R1)  Location of FIELD3
```

MODIFY Exits

When MODIFY is specified, your exit program receives two standard parameters, followed by the optional USING parameters.

- The first parameter points to the file record. This record contains file input record for input operations, and the output record for output operations.
- The second parameter points to a WORKAREA.
- Additional parameters as specified by the USING option (if any).

The WORKAREA is required for MODIFY option. The file record is the input file record. You must move this file record to the WORKAREA in your exit program. Then you can modify the WORKAREA as needed. Upon return from your exit, the WORKAREA is moved to the file record and made available to your generated COBOL program.

Use the supplied sample FSYTMXIT program in the SYS1.SFSYEZTS Migration Utility library as a skeleton for creating more complex programs. To use the supplied FSYTMXIT, you would code the following file statement in your Easytrieve Plus program:

```
FILE FILE01    DISK F (80)     +
     EXIT (FSYTMXIT MODIFY) WORKAREA 80
```

Non-MODIFY Exits

When MODIFY is *not* specified, your exit program receives two standard parameters, followed by the optional USING parameters.

- The first parameter points to the file record. You must populate this file record with valid information that maps the layout defined in your Easytrieve Plus program.
- The second parameter points to the request code. This code is a 4-byte binary integer.

  The input values are:

`0`	first entry and request for read/write
`8`	file close request

These are received by the exit program.

The output values are:

`8`	end of file reached

These must be set by the exit program.

- Additional parameters as specified by the USING option (if any).

Use the sample FSYTIXIT and FSYTOXIT programs supplied in the SYS1.SFSYEZTS Migration Utility library as skeletons for creating your own exits. For example, to use the supplied FSYTIXIT, you would code the following file statement in your Easytrieve Plus program:

```
    FILE FILE01    DISK F (80)    EXIT (FSYTIXIT)
```

If you are reading one or more files in the exit, you may need a flag to control your file I/O activities, such as a first time switch or file closed switch. For this reason, your program must be linked as a reusable program.

All open files must be closed either when you reach EOF or when a request code of 8 is received.

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.

```
►►──%CBLCNVRT──&NAME    ──LOC(&LOC)    ──PFX(&PFX)    ──SFX(&SFX)    ──────────►

►─HEADING(&H)    ─────────────────────────────────────────────────────────────►◄
```

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 194 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.

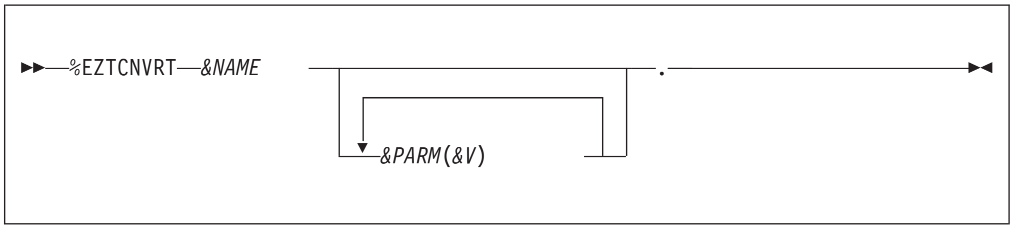
A standalone job must be run to do the conversion:

1. Tailor and use JCEZCNV1 JCL located in SYS1.SFSYJCLS.
2. Make sure that COPYBOOK=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 embedded (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.
3. Generate COBOL copybooks out of Easytrieve Plus macros using the EZTCNVRT macro (refer to “EZTCNVRT macro” on page 192).
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 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.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 preprocessed (byte code) macros
SYS1.SFSYJCLS	Sample JCL distributed with the product
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
FSDATSR2	Date services program for base 360
FSDATSRV	Date services program for base 365
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
FSSQLERR	SQL error display module
FSYCNV01	Automated conversion restart module
FSYCNV02	Automated conversion error logger

FSYCNV03
JCL Merge program (not used)

FSYCNV20
Parallel test compare program

FSYCNV50
Easytrieve Plus discovery utility

FSYCNV55
Easytrieve Plus source analyzer

FSYDB200
Call Attachment Interface (CAF) sample program

FSYDB250
Call Attachment Interface (CAF) module

FSYDLIE0
DLI error display module

FSYTIXIT
I/O EXIT sample exit module (input files)

FSYTMXIT
I/O EXIT sample module (MODIFY option)

FSYTOXIT
I/O exit sample module (output files)

FSYXIT00
Parallel test sample compare exit program/

The following BAL modules are located in SYS1.SFSYLOAD Library:

Name	Description
FSDYN*	Dynamic I/O modules
FSDYNCNV	Convert ASCII to EBCDIC and EBCDIC to ASCII
FSFILL00	Pad a field with a pad characters
FSLOPER0	Perform Logical Operation, AND, OR, XOR
FSLOPER1	Perform Bit Testing for ON condition
FSMVSCN0	Console I/O module
FSSHIFTL	Shift a field 4 bits left
FSSHIFTR	Shift a field 4 bits right
FSSQLER2	Dynamic SQL error handler
FSVLNT03	Get File Record Length after and I/O
FSVLNT90	Get File Information Block (FIB) at Open time
FSYATTCH	Task initiator
FSYDB202	Load DSNHFECP CAF Module
FSYDDCPY	Dynamic allocator interface
FSYDRIL0	Drill Down sub module

FSYDRIL1
Drill Down sub module

FSYDRIL2
Drill Down sub module

FSYEZEND
DEBUG options decoder

FSYFONTS
Drill Down fonts

FSYGDBD0
IMS/DLI SSA string generator

FSYGJOB0
Get Job Information from JOB Scheduler

FSYGPCB0
Resolves DLI/IMS PCB

FSYGTDCB
Obtains file DCB information

FSYGTSO0
DCLQUE0 macro TSO user resolver

FSYLOAD0
Program loader

FSYMIG00
Parallel test JCL builder main program

FSYMIG05
Parallel test run time initiator

FSYMIG10
Parallel test run time driver

FSYMIGLD
Program loader

FSYMIGS0
Parallel test JCL builder submodule

FSYMIGS1
Parallel test JCL builder submodule

FSYMIGS2
Parallel test JCL builder submodule

FSYSQLIO
Dynamic SQL interface

FSYUNIX0
Utility for Unix files

FSYUNIX1
Utility sub module for Unix files

FSYVLN90
Get file information block (FIB) at open time. This module replaces FSVLNT90.

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**
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	Precompiled 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).
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.
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.
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.
FJSYSIN	Input file which contains the EZPARAMS File, normally SYS1.SFSYEZTS
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.
FJSYSP0	The output statistics record normally contains a list of files and macros found in Easytrieve Plus program.

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.

FJSYSPW
: PASSWORD file SYS1.SFSYCCLM(PASSWORD)

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 about this file, see the "LIST=" parameter in the Translator Options described in this section.

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 64,530.

  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 preprocessed 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 preprocessed 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.

The following positional parameters can be coded on the COPTION statement:

LISTM
Print processed Easytrieve Plus macros.

NOLISTM
Do not print processed Easytrieve Plus macros.

EZLREF
Generate cross-reference sequence number of the Easytrieve Plus program source in the generated COBOL columns 1–6. Thus number is used for debugging and relates COBOL statements back to the Easytrieve Plus program.

Chapter 11. Installing Migration Utility

For step by step Easytrieve source conversion refer to Chapter 3, "Conversion guidelines," on page 13.

The Report Modernization Utility (RMU) system is distributed with IMU V4.1 at no additional cost. For the installation of RMU, refer to the *Report Modernization Utility User's Guide and Reference* (publication number SC19-2726-00).

Migration Utility is installed using SMP/E. Refer to the *Program Directory* (GI13-2909) for installation instructions.

Migration Utility works on the OS/390® and z/OS® operating systems.

Upon installation, you must perform tailoring as described in these sections:
- "REPORT statement default options" on page 212
- Chapter 12, "Migration Utility translation options," on page 217
- "Tailoring default PROC for the One-Step driver program" on page 205
- "Activating Call Attachment Facility (CAF) for DB2 users" on page 207
- "Generating Dynamic SQL I/O module (FSYSQLIO)" on page 210

Users outside of the United States should adjust the following EZPARAMS parameters. (For the meaning of each parameter, refer to Chapter 12, "Migration Utility translation options," on page 217.)

NAMETAB=
: Users of non-English alphabets should add the special characters in their alphabet to the NAMETAB= string to avoid the invalid COBOL name messages.

CURRENCY=
: Check currency for proper sign.

DECIMAL=
: Check DECIMAL characters usage.

In addition, check the EASYDTAB macro for date format and other REPORT statement defaults.

Migrating from previous versions

If you are migrating from a previous version of Migration Utility, there are steps you should take to have a smooth migration. The following subsections set out what you should do.

Migrating from IMU Version 2.1, Version 3.1 or Version 3.2

To avoid incompatibility, copy your existing #EZTPROC, EASYDTAB, and EZPARAMS files into the new IMU libraries.

#EZTPROC is distributed in the &sys1.SFSYJCLS library.

EASYDTAB is distributed in the &sys1.SFSYCCLM library.

EZPARAMS is distributed in the &sys1.SFSYEZTS library.

Migrating from Version 1

The EZPARAMS default options changed in Migration Utility Version 2 Release 1 and later versions. If you are migrating from an older version of Migration Utility, you must make sure that your existing programs are not impacted by these changes. These parameters will not affect your program unless it is recompiled.

To take advantage of these new options, you must create two EZPARAMS files, one for the old release and one for the new release. The recommended way is to make a new EZPARAMS member for running the new way, in which case you can change the member name for FJSYSIN in the FSCCL1 step in #EZTPROC and #CNVPROC. The following parameters have changed or have been added. (For the meaning of each parameter, refer to Chapter 12, “Migration Utility translation options,” on page 217.)

ABEND='CON_ABEND00(FSABEC16)'
The old default was FSABECOB. This option impacts how your program abnormally terminates on I/O errors. FSABECOB causes an S0C7 program check, FSABEC16 abends with RETURN-CODE 16, which is compatible with Easytrieve Plus.

COPYNTAB=
The old default was EZTABLE0. It was removed as most companies do not use COBOL copy books. If you are using COBOL copy books, change this parameter to include the same table from your previous release.

DOWHILE=PERFORM
The default was INLINE. PERFORM has fewer compatibility issues, but functionally it is the same as the INLINE option.

IOCODE=EASYT
The default was NATIVE. This may affect how file I/O return codes are interpreted in your program.

IOMODE=DYNAM
The default was NODYNAM. DYNAM impacts your I/O routines and is more compatible with how Easytrieve Plus works.

MOVENUM=EASYT
This is a new option. The previous versions of Migration Utility always defaulted to NATIVE option. Change this option to NATIVE for programs that had been translated with version 1.

OCCURS1=0
The meaning of option 0 has changed. OCCURS=0 in earlier versions flagged OCCURS 1 as an error. OCCURS=0 in this version is ignored, and fields are generated with OCCURS 1.

PRINTER=SYSPRINT
This is a new option. The old default was AUTOGEN, unconditionally. Coding SYSPRINT makes your program compatible with Easytrieve Plus. That is, reports without explicit PRINTER and the DISPLAY lines are printed to SYSPRINT. To preserve report ddnames for programs that were translated with version 1, change this option to AUTOGEN.

SPOOLOPT=YES
This is a new option. The old default was NO, unconditionally. Coding YES will optimize the record size of internally generated files. You must

change this option to NO for programs that were translated with version 1. Failure to change this parameter may result in wrong length records due to DCB LRECL coded in the JCL.

DYNALLOC=YES
This is a new option. The old default was NO, unconditionally. This parameter does not affect your program logic, therefore it can be left as is.

TBMEMORY=DYNAMIC
The old default was STATIC. This option affects how externally loaded tables memory are allocated. Program logic is not affected. Coding DYNAMIC will reduce the size of your load module.

Tailoring default PROC for the One-Step driver program

Migration Utility requires a few JOB steps when translating Easytrieve Plus programs to COBOL. Previous Migration Utility releases supplied jobs and procedures (PROCs) with multiple steps that accomplished the task. This multi-step compilation process is quite different from and is more complex than in Easytrieve Plus.

The One-Step driver program (FSYTPA00) concept resolves these differences by running the required translating steps from a single program. It essentially makes translating look like a single step and compatible with Easytrieve Plus. The differences amount to changing the EZTPA00 program to FSYTPA00 on the EXEC statement, and providing access to Migration Utility instead of Easytrieve Plus load libraries.

Activating the FSYTPA00 program

Note: The addressing mode of the FSYTPA00 program has been changed from AMODE(ANY) to AMODE(31). Effective with IMU V4.1, users who call FSYTPA00 from a third party software can now call FSYTPA00 instead of FSYTPA31. The FSYTPA31 bootstrap is still available for compatibility.

To activate the FSYTPA00 program, do the following steps:

1. Tailor #EZTPROC, located in SYS1.SFSYJCLS

   This proc is read by FSYTPA00 program. It contains all the information and steps needed to simulate the one-step process.

   To tailor, follow the comments embedded in the proc. You will probably change the following parameters: TWORK=, CWORK=, MODEL=, EZMACDD=, OWNLOAD=, DSNEXIT=, DSNLOAD=, COBLIB=, and IMSRLIB=. You may have to change other parameters, depending on your requirements. Verify the entire PROC for potential changes.

   If you comment out a proc keyword, you must also comment out all references to it.

   Non-SMS users: You must make dynamically-allocated files available to subsequent steps. To do this, globally change the DISP= values:

   - The ",PASS," option to ",CATLG,"
   - The ",KEEP," option to ",CATLG,"

   Important #EZTPROC options

   a. SYSPRDD= and SYSUTDD= options

      The SYSPRDD=&DDname option in #EZTPROC directs listings produced by COBOL, Link step and SQL translator steps to a specific DD name. The default is SYSPRDD=FJSYSPR. This DD name is internally allocated and used by Migration Utility. Do not code it in the JCL.

The SYSUTDD=&DDname option in #EZTPROC is used to replace the SYSUT1-SYSUT7 COBOL compiler work DDnames. The default is SYSUTDD=SYSUT.

SYSPRDD=SYSPRINT
will force Migration Utility to use SYSPRINT as the system printer. Be aware, using SYSPRINT for compiler listings may interfere with your reports archiving system if your Easytrieve Plus reports default printer is SYSPRINT.

b. While testing, you can change SVC99=MSGOFF in #EZTPROC to one of the following options:

MSGOFF
Disable dynamic allocator tracing.

MSGON
Display JES messages.

MSGALL
Display input text and JES messages.

MSGSER
Display messages of serious nature only.

MSGTXT
Display input text only.

Use the above options to trace #EZTPROC and the dynamic allocation problems that might arise during testing. For production use, SVC99=MSGOFF is recommended to avoid excessive console messages.

2. Change the FSYPROCS table, located in SYS1.SFSYEZTS

   You must change this program to point to the Migration Utility libraries installed on your system. Change the PROCLIB0 constant to point to the PDS where #EZTPROC from step #1 is located. Change the PRODUCT0 constant to the high-level qualifier of the Migration Utility libraries.

   Assemble and link the FSYPROCS program using the JCASMBAS job located in SYS1.SFSYJCLS. This program must be linked into Migration Utility's SYS1.SFSYLOAD library, where FSYTPA00 is located.

   Note: FSYTPA00 loads FSYPROCS to locate the #EZTPROC member and to acquire replacement for &SYS1 symbol located in #EZTPROC. The information in FSYPROCS must always point to proper libraries. This means that if you rename or move your Migration Utility libraries, you must adjust the information in FSYPROCS accordingly.

3. Test FSYTPA00

   Tailor and run JCMUCL1J, which is located in SYS1.SFSYJCLS. This job translates MUTEST00 program located in SYS1.SFSYEZTS. If all changes you made are correct, this program should translate and link MUTEST00 program free of errors.

   If you experience allocation problems, change SVC99=MSGOFF to SVC99=MSGALL in #EZTPROC. This will turn dynamic allocation messages on. It may help explain what could be wrong.

   Note: Do not leave your proc with messages turned on for general users. This option prints too many messages for normal use and it degrades the process.

Activating Call Attachment Facility (CAF) for DB2 users

Note: The tasks outlined below should be performed by someone who has an in-depth knowledge of DB2, such as the DB2 administrator.

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. JOBS are located in SYS1.SFSYJCLS.

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 use DCLINCL, refer to the description of DCLINCL in “Embedding options in the program source” on page 239.

   The SYSIBM.COLUMNS table can be accessed automatically in two ways:
   - Using CAF Interface via FSYDB2D1 module
   - Via Dynamic SQL using FSYSQLIO module

   When Dynamic SQL is in use, the SSID and PLAN name provided on the Easytrieve Plus PARM statement are used. The FSYSQLIO module must be generated and bound as described in “Generating Dynamic SQL I/O module (FSYSQLIO)” on page 210. Also, CAFSSID=&PARM and CAFPLAN=&PARM EZPARAMS/EASYTRAN options must be used to tell Migration Utility to use the values specified on the PARM statement. This feature allows you to use a single CAF interface for the Migration Utility translator and the application programs, as well as the capability to choose SSID and Plan at run time.

   To use CAF interface via FSYDB2D1, you must perform installation steps as described below.

   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).

     For example, to create a plan IBMMIGUT and a package (collection) called BATCH for the DBVA system, would use the following BIND parameters: (Tailor and run JCPLNBND job)

     ```
     DSN SYSTEM(DBVA) -
     BIND PLAN(IBMMIGUT) PKLIST(BATCH.*) -
     ISOLATION(CS) ENCODING(EBCDIC) ACTION(REPLACE)
     ```

     After you bind your plan, grant PUBLIC access to it so that all users have the proper authority to use it.

   - 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.

     For example, to bind FSYDB2D1 with the package (collection) called BATCH for the DBVA system, you would use the following BIND parameters: (You can run this as a separate BIND job)

```
DSN SYSTEM(DBVA) -
BIND PACKAGE(BATCH) MEMBER(FSYDB2D1) -
ACTION(REPLACE) ISOLATION(CS) ENCODING(EBCDIC) -
LIBRARY('????????.DBRMLIB')
```

Make sure that you point DBRMLIB to the DBRM library where FSYDB2D1 DBRM is located (from your compile JCCOBSQL job).

Make the following changes to EZPARAMS:

CAFPLAN=&*plan*,
Where &*plan* is the plan name you created.

CAFSSID=&*ssid*,
Where &*ssid* is the DB2 system for which the plan was created.

Note that CAFSSID= is an optional parameter. If you do not provide a default &*ssid*, make sure that the correct DSNLOAD and DSNEXIT load libraries are concatenated on your JOBLIB or STEPLIB when translating DB2 programs. Migration Utility uses the DSNHDECP DB2 module located in these libraries to obtain the DB2 system name (SSID) for Call Attachment Facility.

The translator uses the SYSIBM.SYSCOLUMNS table when it encounters an "SQL INCLUDE FROM &owner.&table" in the Easytrieve Plus program, and a DCLINCL was not previously supplied.

To test the CAF interface, translate TESTCOL0 (located in SYS1.SFSYEZTS), using the JCMUSQLP Proc. Change PARM SSID and PLAN in the program to your own SSID and PLAN.

TESTCOL0 is set for STATIC SQL mode and to run in Call Attachment Facility (CAF) mode. The program contains the `SQL INCLUDE FROM SYSIBM.SYSCOLUMNS` statement to test CAF functionality. After you run JCMUSQLP, look at the SYSOUT file for CAF messages. The program should translate and BIND cleanly.

2. Converted DB2/SQL programs can be run in three ways:
   - Using the Dynamic SQL

     Before using Dynamic SQL, the FSYSQLIO module must be prepared as per "Generating Dynamic SQL I/O module (FSYSQLIO)" on page 210.

     Programs are generated to run in Dynamic SQL mode when SQLBIND=DYNAMIC is specified in the EZPARAMS/EASYTRAN or SQLBIND=ANY is specified in the EZPARAMS/EASYTRAN in combination with the PARM BIND(DYNAMIC) in the program source.

     Binding of the generated programs is not needed.

     When FSYTPA00 runs, it automatically detects the Dynamic SQL mode and invokes the IMU's Dynamic SQL translator in the place of the standard DB2 SQL translator. The step is located in #EZTPROC/#CNVPROC.

     The translated program is compiled, linked and run as a regular COBOL program.

     Use JCMUCL* jobs to translate DB2 programs to run in dynamic mode.

     Note: Make sure that you add the DB2 DSNEXIT and DSNLOAD libraries to the JOBLIB of these PROCs.
   - Attaching to DB2 via Call Attachment Facility (CAF) from the generated COBOL programs (at run time) with imbedded static SQL statements (static mode).

     For this method, you must translate your generated COBOL via the standard DB2 SQL translator and BIND the program as a collection/package. (Refer to the JCMUSQLJ and JCMUSQLP jobs. Note that the bind parameters in these jobs must be changed to bind a PACKAGE. The SQLMODE=FSYDB250 or SQLMODE=FSYDB200 option is required in the EZPARAMS/EASYTRAN).

Use the JCMUSQLJ or JCMUSQLP job to translate. These jobs contain a BIND step. The BIND parameters are generated by the FSYTPA00 from PARM SSID, PLAN and BIND values in the program (or the defaults). The BIND values are punched by the SQLBIND macro located in SYS1.SFSYCCLM library.

You can tailor the SQLBIND macro to conform to your own system or you can provide custom BIND parameters by changing the SYSTSIN to your own BIND file. Note that SQLBIND macro runs in interpretive mode. No macro compiling is needed.

Use JCMUSQLP to translate TESTCOL0 to see how things work. Use JCRSQL0P to test the translated program.

- Using the standard IKJEFT1B TSO loader program.

  For this method, you must translate your generated COBOL via the standard DB2 SQL translator and BIND the program to a PLAN. (Refer to JCMUSQLJ and JCMUSQLP jobs. SQLMODE=BIND option is required in the EZPARAMS/EASYTRAN options table).

  Use JCMUSQLP to translate TESTCOL2 to see how things work. Use JCRSQL2P to test the translated program.

Call attachment programs supplied by IMU

To run your generated DB2 programs using CAF **static mode**, choose from FSYDB200 or FSYDB250 as described below.

To run your generated DB2 programs using CAF **dynamic mode** you must use FSYDB250 as your CAF module (SQLMODE=FSYDB250) as described below.

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, required for SQLBIND=DYNAMIC)

Attaches to DB2 by means of 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:

- Change EZPARAMS option to SQLMODE=FSYDB250 before translating Easytrieve Programs. A call will be generated to FSYDB250 module at the beginning of the generated COBOL program.
- If you are using multi step JCL, make sure that DB2SQL step in the translator JCL contains ATTACH(CAF) on the PARM statement.
- If you will be using CAF in static mode, BIND the generated COBOL program as a PACKAGE. Note that BIND is not needed for dynamic mode.

Optionally, you can provide SSID=&*ssid*, SQLID=&*sqlid* and PLAN=&*plan* by means of //SQLDD JCL at run time.

Note: SQLDD file LRECL=80,RECFM=F.

Comments can be placed in the input SQLDD file by placing an asterisk (*) 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 typically 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.

   The following rules are observed:

   - The PLAN name, SQLID, and SSID are used from the SQLDD file if supplied.
   - Any parameters not supplied in the SQLDD file are used as coded on the PARM statement in the Easytrieve Plus program, PLAN(&plan), SQLID=&*sqlid* and SSID(&ssid).
   - If SSID is not specified on the PARM statement in the Easytrieve Plus program, FSYDB250 obtains the SSID from the DFNHDECP module located in DSNEXIT or DSNLOAD load libraries. In this case, the DSNEXIT and DSNLOAD libraries must be concatenated to the JOBLIB or STEPLIB in your JCL.

Generating Dynamic SQL I/O module (FSYSQLIO)

Before doing this task, make sure that you successfully activate the CAF interface as described in the previous section “Activating Call Attachment Facility (CAF) for DB2 users” on page 207.

Tailor and submit JCASMSQL supplied JOB to assemble, translate and bind FSYSQLIO dynamic I/O interface module. Adjust BIND step and BIND parameters to your needs.

The source for FSYSQLIO is located in SYS1.SFSYEZTS. There are four parameters that you can change to accommodate your requirements:

NONCURS=64,
Maximum number of non-cursor requests

CURSORS=32,
Maximum number of non-scrollable cursor requests

ICURSOR=16,
Maximum number of insensitive scrollable cursors

PREPARE=NOREFRESH,
Where:
REFRESH
Refresh PREPARE statements on every cursor OPEN call.
NOREFRESH
Do not refresh PREPARE statements on every cursor OPEN.

SCURSOR=16;
Maximum number of sensitive scrollable cursors

The number of specified requests are for supporting a single DB2 program. Do not over-allocate. The module size should be kept to a reasonable size.

Bind a PLAN (or as many as you need) that you will be running your application programs with. You can tailor and run the JCPLNBND job to do so. Make sure that you specify (for the PKLIST) the same package name that was used to bind the FSYSQLIO module.

Grant proper authorization to bound plans (you can use SPUFI from an ISPF panel). For example, you can build and run the following request for PLAN IBMMIGUT and Collection BATCH:

```
GRANT EXECUTE ON PLAN IBMMIGUT TO PUBLIC;
GRANT PACKADM ON COLLECTION BATCH TO PUBLIC;
GRANT BINDADD TO PUBLIC;
```

Translate TESTCOL1 using the JCMUCL1P job. The TESTCOL1 program is set to run in Dynamic SQL mode.

The JCMUCL1P JOB is located in SYS1.SFSYJCLS. Uncomment TESTCOL1 at the bottom to translate TESTCOL1. Make sure that the correct DSNLOAD and DSNEXIT libraries are on the JOBLIB.

- Make sure that you adjust SSID and PLAN on the PARM statement: `PARM LINK(TESTCOL1) BIND(DYNAMIC) SSID(&SSID) PLAN(&PLAN)`
- Assuming that you are using SSID=DL11 and PLAN=IBMMIGUT, then the top of your TESTCOL1 program should look like this: `PARM LINK (TESTCOL1) BIND(DYNAMIC) PLAN(IBMMIGUT) SSID(DL11)`
- After a clean compile, run JCRSQL1P located in SYS1.SFSYJCLS. Make sure that TESTCOL1 is specified on the EXEC statement.

DB2 libraries are not needed when your application program is running in Dynamic SQL mode. The program is treated as any other non-DB2 program. However, the translator does need DB2 DSNLOAD and/or DSNEXIT libraries to resolve the DB2 tables column definitions.

REPORT statement default options

The EASYDTAB macro, located in SYS1.SFSYCCLM, 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
```

&GSYSTIMEC,'.';
Insert character for SYSTIME.

Example: `HH.MM.SS` (change to your own character).

&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
```

Performance hints for the generated COBOL

IMU-generated COBOL is more efficient when translated with certain options. The default options supplied in the EZPARAMS make COBOL program more compatible with Easytrieve Plus but add some processing overhead.

If performance is your main concern, you can change the defaults in the EZPARAMS default table or add specific options to the program source via the EASYTRAN.

Keep in mind that changing options below may make your translated program less compatible with Easytrieve Plus. More work may be needed when converting.

PROCESS ADV,FASTSRT,OPT(FULL),NUMPROC(PFD),TRUNC(OPT),NOLIST
: Note: PROCESS is a COBOL compiler option. LIST tells COBOL to produces a BAL listing of COBOL code. This can be useful for debugging, but it may take longer to compile.

IOMODE=NODYNAM
: This option is more efficient than IOMODE=DYNAM. However, the file define statements in the program must specify the correct file type, the record length, and the VSAM file key location. As the adjustments are made only once, you may well consider this option worthwhile.

  The alternative is to use IOMODE=DYNAM or IOMODE=DYNAM24 in combination with the DYNBOOT=FSDYNIO1 option.

IOCODE=NATIVE
: This option eliminates the double management of the file status code in the generated COBOL. The downside is that Easytrieve Plus codes are different from those of COBOL. Therefore, if your program is testing for a specific status code, the code must be changed in the Easytrieve Plus program to the COBOL equivalent. In general, testing for EOF or ZEROS does not require any change.

SYNCREC=NOREFRESH
: This option eliminates statements that save and restore file records when user logic is entered from the synchronized file logic. It therefore saves on CPU cycles. This is compatible with Easytrieve Plus.

SYNCSEQ=NO
: This is a new option. It suppresses the sequence check of files in the synchronized file (match) job. Eliminating the sequence check saves CPU cycles but out-of-sequence files are not detected. Therefore, the outcome of synchronized file process can be wrong without ever detecting the problem. Note that SYNCSEQ=NO is consistent with Easytrieve Plus.

Using EZTPA00 program loader

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: Do not confuse the Migration Utility EZTPA00 program with the Easytrieve Plus compiler program, which is also named EZTPA00 and normally resides in the Easytrieve Plus load library. Migration Utility's EZTPA00 is a program loader, *not* a compiler.

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 EZTPA00 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.

Chapter 12. Migration Utility translation 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 embedded 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 239.

The first line in the EZPARAMS member is the COPTION statement. The COPTION statement describes PEngiCCL options such as the output listing and paragraph re-sequencing options. The COPTION parameters are beyond the scope of this document. 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
```

Installing Migration Utility for the first time

IMU default options in the EZPARAMS table, distributed with the system, provide options that are most compatible with Easytrieve Plus. Options that provide for EASYT/NATIVE choice have been all set to the EASYT option.

Use your text editor to make changes to EZPARAMS if any. Note that a comma must be placed after each parameter.

EASYTRAN/EZPARAMS options

This section provides details of the available options.

Note: The value indicated is the default option as coded in the EASYTRAN macro. The actual active option can be found in the EZPARAMS options table.

ABEND='CON_ABEND00(FSABEC16)'
File I/O error-handler function and abend program.

FSABECOB
Prints I/O errors and causes an S07 abend (data exception).

FSABEC16
Prints I/O errors and returns RETURN-CODE 16.

If you want to use your own abend interface, you can use a special syntax that allows you to pass a 2-byte binary abend code in a field by means of a USING statement. The field name must be unique as Migration Utility generates a 2-byte binary field in working storage using the supplied field name. Note that Migration Utility moves the abend code into the designated field before passing control to the user abend interface program.

Example:

```
CON_ABEND00 (USERAB00 USING MY-ABEND-CODE)
```

Here, USERAB00 is assumed to be a user abend program name and MY-ABEND-CODE the name of 2-byte binary field that is passed to USERAB00.

ALIGNRPT=NATIVE/EASYT
Specifies how is the last “+NN” coded on REPORT LINE or TITLE to be handled.

NATIVE
Ignores the last +NN if not followed by a field/literal.

EASYT
Assumes a space (“ ”) following the +nn to match EZT Plus.

ALLOCATE=(&DDname)
Allocate SYSOUT files selectively. Files are allocated at application program run time.

&DDname
One or more 1–8 character DD names separated by comma.

The default is ALLOCATE=().

Note: Migration Utility generates a call to FSYSVC99, in the generated COBOL code, first to de-allocate files listed in the DEALLOCATE= option, then to allocate files listed in the ALLOCATE= option. The sequence of the ALLOCATE/DEALLOCATE statements, as coded in the program, is not honored.

The DD names are not validated for valid names. Coding improper DD names results in a failed ALLOCATE attempt.

Example:

```
ALLOCATE=(CEEDUMP,FJSYABE)
```

ALPHATEST=NATIVE/EASYT
Alphabetic test option. Note that COBOL test is for upper and lower case, while Easytrieve Plus tests for upper case only.

EASYT
Accepts ALPHABETIC as the only valid test. Statements are converted to ALPHABETIC-UPPER for Easytrieve Plus compatibility.

NATIVE
Accepts the following:

ALPHABETIC
ALPHABETIC-LOWER
ALPHABETIC-UPPER

The default is ALPHATEST=NATIVE.

ALTSEQ=()
Name of the alternating sequence table to be used by internal sorts in the generated COBOL programs. This parameter is for users who use an Alternate Sequence table for Easytrieve Plus sorts. Check the ALTSEQ= parameter in the EZTPOPT (Easytrieve Plus) options table to see if you need to specify this entry. The default is the installation default collating sequence table.

Refer to the section on the SPECIAL-NAMES paragraph in the COBOL Language Reference for information about creating ALPHABET tables. Use the Migration Utility FSYSRTAQ copybook supplied in SYS1.SFSYCCLC as a template. Migration Utility inserts a COPY statement, for the name you specify, in the generated COBOL SPECIAL NAMES.

BLANKFIX=NO/YES
This option suppreses extra blank lines to match Easytrieve Plus logic.

NO Keeps IMU logic as is, that is:

- IMU produces extra blank lines at the bottom of the reports that contain multiple LINE statements and the *n*th line does not contain data.
- IMU is not printing a blank line before the first detail line on each page. This logic applies when REPORT TITLE lines do not exist.

YES Makes blank line handling compatible with Easytrieve Plus.

BLKSIZE=NATIVE/EASYT
Option for BLOCKSIZE handling on FILE definition.

NATIVE
Block size is ignored.

EASYT
Block Size is used as coded on EZT Plus file definition in the program.

The default is BLKSIZE=NATIVE.

BREAKS=32/nn
Maximum number of report control breaks.

nn A number from 00 to 64.

The default is BREAKS=32.

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 207. 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.

&PARM
Use PLAN for CAF interface from the PARM PLAN (&PLAN) found in the Easytrieve Plus program to resolve SYSIBM.SYSCOLUMNS definitions.

CAFSSID=(&SYS)
Up to 4 characters DB2 system (SSID) for Call Attachment Facility (CAF) for resolving DB2 column definitions. This parameter tells FSYDB2D1 module the DB2 system to connect to as follows:

&SYS Use SSID from DSNHDECP.

&PARM
Use SSID for CAF interface from the PARM SSID (&SSID) found in the Easytrieve Plus program to resolve SYSIBM.SYSCOLUMNS definitions.

&*ssid* A four (4) character DB2 SSID.

Example:

```
CAFSSID=(DBVA)
```

CFACTOR=NO
Intermediate results precision option for high-order math operations:

NO Use standard COBOL COMPUTE statement rounding rules.

YES Use 8 decimal places for intermediate results.

Note: Easytrieve Plus automatically uses 8 decimal places in intermediate results when doing high-order math operations (i.e., multiplication and division). COBOL COMPUTE statement uses the maximum number of decimal places as determined from the field definitions used in each term. As a result, the outcome of COMPUTE may not be as precise.

Specifying CFACTOR=YES compensates for the loss by multiplying each math expression that contains high-order operations by 1.00000000.

As most programs do not contain math operations that require precision and to preserve efficiency, you should keep CFACTOR=NO as the default and code CFACTOR=YES in those programs where precision is essential.

CLIPSIGN=NO/YES

Migration Utility users of &FIELD INTEGER = &FIELD syntax can prevent data exception when display numeric field contains a bad sign.

Migration Utility validates the field sign at execution time and causes data exception when the sign is not valid. Easytrieve Plus seems to be ignoring the bad sign, OR-ring it with x"F0".

NO Sign is left as is. A space (x'40' causes 0C7).

YES X'40' in the last byte is replaced by x"F0" for numeric display fields only. The logic is applied when there is only one source field.

Example: (the sign would be changed)

```
DEFINE FIELDA W 6 N
DEFINE FIELDB W 6 N 3

FIELDA INTEGER = FIELDB
```

Example: (the sign would **not** be changed)

```
DEFINE FIELDA W 6 N
DEFINE FIELDB W 6 N 3

FIELDA INTEGER = FIELDB + 3
```

COBOL=COBOL390

Type of COBOL. COBOLII for COBOL II, COBOL390 for COBOL/390 and later versions of COBOL.

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 because 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 quotation marks.

COPYFHDR=NATIVE/EASYT
Controls if the file name is to be included on the report field headings for fields generated due to COPY &file statement.

NATIVE
Includes &file in the field heading.

EASYT
Excludes &file from the field heading.

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. If located and `COPYBOOK=NO`, the translator uses this fact and generates a meaningful fields prefix when the same macro is used in the program more than once.

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 194.

COPYVERB=(COPY)
COBOL copy to be used as COPY statement when COPYBOOK=YES is specified.

Change this default if you have a special copybook preprocessor that recognizes a different verb as a COPY statement. For example, SQL host variables are flagged by the DB2 preprocessor when located in a copybook. To allow copybooks, you can create your own preprocessor 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=== .
```

CURRENCY=($)
Currency sign

DATEABE=(&ABEND,&HIBYTES)
Date handling options for the DATECONV macro.

&ABEND

NO	Do not abend on invalid date.
YES	Abend on invalid date.
RC	Use RETURN-CODE for invalid dates.

The default is NO.

&HIBYTES

CLIP The DATECONV macro zeros out the high-order bytes in the date work field before converting. However, be aware that this option returns a valid date even if the date is not correct due to high-order digits. The high-order bytes are those bytes that exceed the number of digits represented by the date mask. For example, the date 20081213, when used with YYDDMM mask, is adjusted to 081213 before validation.

NOCLIP
The DATECONV macro includes the entire content of the supplied field.

The default is NOCLIP.

DATEFORM=(&type,&format,&mask)
Provides the date usage override by way of EASYTRAN or EZTRVPRM. Adding DATEFORM= by way of EASYTRAN or EZTRVPRM provides a global way of altering Migration Utility options without changing the code or maintaining EASYDTAB in a separate library for Easytrieve Classic. :

&type SHORT for short (6 digit) date format.

LONG for long (8 digit) date format.

&format
Valid SHORT formats: MMDDYY, YYMMDD,DDMMYY.

Valid LONG formats: MMDDCCYY, CCYYMMDD, DDMMCCYY.

&mask
DATE Mask that corresponds to &format.

Examples:

```
DATEFORM=(SHORT,MMDDYY,Z9/99/99)
DATEFORM=(LONG,CCYYMMDD,9999/99/99)
```

DDFILTER=NO/YES
Option for flagging DD names found in the Easytrieve Plus programs that conflict with the COBOL compiler and Link steps.

NO Do not flag conflicting DD names as an error.

YES Flag conflicting DD names found in FSYDDTAB as an error.

- The FSYDDTAB table located in &SYS1.SFSYCCLC contains the default DD names to be flagged. The table can be changed to user requirements, that is, custom DD names can be added or deleted from FSYDDTAB. Follow directions in the FSYDDTAB table for the updating rules.
- An alternative to DDFILTER=YES is the SYSOUTDD= parameter coded in the #EZTPROC. However, this parameter addresses the SYSUT* DD names conflict only. The FSYDDTAB contains additional DD names used by the translator that could potentially conflict with the file names defined in the Easytrieve Plus program.

The default is DDFILTER=NO.

DDFNAME=
ddname of the 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.

DEALLOCATE=(&DDname)
De-allocate files selectively. Files are de-allocated at application program run time.

&DDname
One or more 1–8 character DD names separated by comma.

The default is DEALLOCATE=().

Note: Migration Utility generates a call to FSYSVC99, in the generated COBOL code, first to de-allocate files listed in the DEALLOCATE= option, then to allocate files listed in the ALLOCATE=option. The sequence of the ALLOCATE/DEALLOCATE statements, as coded in the program, is not honored.

The DD names are not validated for valid names. Coding improper DD names results in a failed DEALLOCATE attempt.

Example:

```
DEALLOCATE=(SYSMDUMP,SYSUDUMP)
```

DEBUG=()
This parameter controls the Migration Utility one-step translator listing options when the FSYTPA00 driver program is used. This option should be used to control listings at the individual program level using `'* EASYTRAN: DEBUG (........)'`. The statement is normally placed at the beginning of the Easytrieve Plus program.

EASYTRAN DEBUG statements from the Easytrieve Plus program are merged to the default values coded in the EZPARAMS file. Note that the previous versions of Migration Utility were replacing the defaults values that were coded in the EZPARAMS file.

LIST/NOLIST/LISTALL
Controls FSCCL1 step (first step listing). The default is LIST.

NOBLIST/BLIST/BLISTALL
Controls FSCCL2 step (second step). The default is NOBLIST.

NOCOBOL/COBOL
Controls COBOL listing. The default is NOCOBOL.

LKGO/NOLKGO
Inhibits Link and Go if there is no PARM LINK (&name r) in the Easytrieve Plus program. The default is LKGO.

NOJCL/JCL
Generate JCL to a sequential disk. The default is NOJCL.

SQLTRAN/NOSQLTRAN
Controls SQL/DB2 translator listing. The default is SQLTRAN.

NOPDEBUG/PDEBUG
Generate DISPLAY statements at the beginning of each paragraph name. The default is NOPDEBUG.

MSGOFF/MSGSTART/MSGALL/MSGEND
Controls FSYTPA00 internal steps messages. The default is MSGMOFF. Other choices are:

MSGSTART
Message is issued when steps initiate

MSGALL
Message is issued when steps initiate and terminate

MSGEND
Message is issued when steps terminate

NOESPI
Do not set program check interrupt trap.

ESPI Sets program interrupt trap that prints basic information regarding the interrupt. When this option is in effect, only fields and field groups used in the problem instruction are printed.

ESPI-PART
Sets program interrupt trap that prints basic information regarding the interrupt, file flags, records and file counters as follows:

```
&FILE-IO-COUNT
&FILE-STATUS
&FILE-DYNAM-FIB
&FILE-RECORD
```

ESPI-FULL
Sets program interrupt trap that prints basic information regarding the interrupt and the entire DATA DIVISION (working-storage section, linkage section, and file records).

| **ABEND**
| Abend on field overflow. This parameter is activated when
| ESPI/ESI-PART/ESPI-FULL is active.

| **Note:** By Z/OS architecture, field overflow exception (SC0A) is not
| detected unless bit 21 of PSW is turned on. The user is responsible
| for making sure that data loss does not occur. Refer to “How to
| activate the Interrupt Handler” on page 301 for a detailed
| explanation of the ABEND option.

Here is an example of a DEBUG statement in EZPARAMS:

```
DEBUG=(LIST,BLIST,COBOL)
```

Here is an example of a DEBUG statement in an Easytrieve Plus program:

```
* EASYTRAN: DEBUG (LIST BLIST COBOL JCL)
```

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.

DEFER=NATIVE/EASYT
Defer statement handling on file definition.

NATIVE
Ignore DEFER.

EASYT
Honor DEFER on file statements.

DOWHILE=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, INLINE generates logic that is less fragmented and is easier 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.

DTABNAME=EASYDTAB
Overrides the default EASYDTAB options table with a new name. Coding DTABNAME=&tbname via EASYTRAN, EZPARAMS and EZTRVPRM provides a global way of altering Migration Utility options without changing the code or maintaining EASYDTAB options table in a separate library.

Migration Utility is distributed with 2 options tables in &SYS.SFSYCCLM:

EASYDTAB
This is the default table for both Easytrieve Classic and Easytrieve Plus.

EZTRDTAB
This table is identical with EASYDTAB. It can be modified and used for Easytrieve Classic.

Additional copies of either table can be made and used.

DYNALLOC=YES
COBOL Runtime temporary files allocation option.

YES Generates COBOL statements that allocate temporary files dynamically at run time. This is a required statement when using the parallel testing utility for parallel testing and conversion of existing programs. Also, this option requires fewer changes to your existing Easytrieve Plus JCL.

NO Does not generate COBOL statements for allocating temporary files dynamically. When this option is used, all temporary files generated in your COBOL program must be defined in the JCL.

DYNBOOT=FSDYNIO1
I/O interface when IOMODE=DYNAM/DYNAM24 is in use:

FSDYNIO1
Use FSDYNIO1 MU's bootstrap in static mode when calling dynamic I/O modules. The static CALL eliminates the overhead associated with the COBOL dynamic CALL.

FSDYNIO0
Use FSDYNIO0 by means of a standard COBOL dynamic call to interface with the dynamic I/O modules. This method is less efficient than the FSDYNIO1 option.

DYNINIT=VALUES
Initialization option for file records of files accessed in dynamic mode.

SPACES
Clears file records to spaces

VALUES
Clears file records to spaces and then with COBOL INITIALIZE.

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.

| **EZVERS=STD/11.00**
| Easytrieve Plus version in use.

| **STD** All Easytrieve Plus versions prior to V11.00.

| **11.00** Easytrieve plus Version 11.00 and later versions

| **FDISPLAY=NATIVE/EASYT**
| Option for handling display statements when DISPLAY is to a non-printer
| file.

| **NATIVE**
| Do not use print control character (writes as a regular file).

| **EASYT**
| Use print control character when writing to a non-printer file. One
| byte for print control character is reserved at the beginning of the
| message.

FIELDS=5000
Maximum number of field definitions

FILES=128
Maximum number of supported files

FLDCOMNT=NO/YES
Controls generating of long field names as comments.

NO Migration Utility does not generate any long field name comments.

YES A comment is generated of long field names and placed before the derived field name in the COBOL code.

ESPI=LOWER/UPPER
Designates the way ABEND messages are printed on FJSYABE by the Migration Utility abend analyzer.

LOWER
Print ABEND messages in lower/mixed case.

UPPER
Print ABEND messages in upper case.

FORCEBWZ=NO/YES
By default, Migration Utility changes all "Z"s beyond the decimal point to

"9"s. This option allows you to choose the translating method for MASKS of numeric fields that contain "Z"s beyond the decimal point.

NO Migration Utility handles masks compatible with the previous versions of Migration Utility. That is, the "Z"s beyond the decimal point are changed to "9"s.

YES Migration Utility adds the BWZ option to the adjusted masks.

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

| **IDDCOPY=(COBOL/IDMS)**
| Sets the format of IDMS record layout.

| **COBOL**
| Generate IDMS record layout in COBOL format

| **IDMS** Generate IDMS record layout via “COPY IDMS RECORD &record”

| The default is IDDCOPY=COBOL

IDDLIBS=(&lib . .)
List of PDS members and libraries that contain IDMS IDD COBOL definitions. Migration Utility decodes the layout found in each supplied library and subsequently uses decoded record layouts when "IDD RECORD &name" statement is encountered. in Easytrieve Plus programs.

The &lib are the names of PDS members that contain IDMS layouts, in COBOL format, punched by IDMS utility. One or more libraries can be listed.

Example:

```
IDDLIBS=(IDDLIB1 IDDLIB2 IDDLIB3)
```

INARGS=124
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 124 if absolutely necessary.

INDENT=3
Standard indentation (3 means three spaces)

INDEXS=256
Maximum number of index entries due to OCCURS

IOCODE=EASYT
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 39.

EASYT
Migration Utility translates COBOL Status Code to Easytrieve Plus equivalent after each I/O. This option generates extra code, but the checking of status codes 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=DYNAM
I/O mode:

DYNAM
Use dynamic I/O with buffers acquired above 16MB (AMODE(31))

DYNAM24
Use dynamic I/O with buffers acquired below the line (AMODE(24)). This option changes the COBOL compiler option to DATA(24). When you use this option, make sure that the LE environment options allow programs to run with AMODE(24).

NODYNAM
Use static I/O

LINES=256
Maximum number of report lines for a single report

LINKNAME=YES
Punch option for COBOL “PROCESS NAME” statement.

YES Generate PROCESS NAME statement. Note that “PROCESS NAME” is generated only if PARM LINK (&PGMNAME) is coded in the program.

NO Do not punch PROCESS NAME statement. When NO is specified, the “PROCESS” option supplied by means of the EZPARAMS or EASYTRAN or the default COBOL option is honored.

LXITDTL=NATIVE/EASYT
Print control character handling in BEFORE-LINE exit.

NATIVE
Do not honor heading offset in the BEFORE-LINE exit. This suppresses leading blank lines before the first detail line.

EASYT
Honor heading offset for the first line in the BEFORE-LINE exit as per EZT Plus rule.

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=2048
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

MOVENUM=EASYT
MOVE statement option for numeric fields.

NATIVE
MOVE statements operate according to COBOL rules. With this option, the data in the source field is converted to the type of the target field for all elementary items. For example, data is packed when moving a display numeric field to a packed decimal field. The group items are moved without conversion.

EASYT
MOVE statements operate according to Easytrieve Plus rules. With this option, all moves to or from numeric fields are performed without data type conversion. For example, data would be moved from a display numeric field to a packed decimal field without conversion. If the target field is longer, the remaining bytes are padded with spaces. This option may yield more compatible conversions of existing programs (fewer changes are needed). However, choosing this option inhibits the ability to move fields of non-compatible types with proper conversion.

MOVERPT=NATIVE
Move method for building REPORT LINE and TITLE lines.

NATIVE
Generates field moves in forward order, i.e., the first defined field is moved first, and so on.

EASYT
Generates moves in backward order (the Easytrieve Plus way). Use this option if there are overlapping fields on LINE or TITLE definitions that cause loss of data.

Note: When fields are moved in forward order, and there are overlaps, the field that overlaps the previous field is moved over the tail end (overlapped) portion of the previous field (Migration Utility default).

When fields are moved in backward order, and there are overlaps, the beginning of the field that overlaps the previous field is clipped by the previous field (the Easytrieve Plus way).

As a result of this difference, if overlap exists, the Migration Utility generated reports may not match those generated by Easytrieve Plus. The use of MOVERPT=EASYT corrects this difference.

MPARMS=064
Maximum number of supported macro parameters

NAMETAB='$S/-+A#N@V*-_-%P?Q'
Translate table for special characters found in field names. These are coded in pairs, so “$” 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=64
Maximum number of nested IFs

NEWPAGE=NATIVE
Blank line option when no TITLE lines exist.

NATIVE
Print a blank line at the top of each page if not TITLE lines are present.

EASYT
Do not print a blank line at the top of each page if no TITLE lines are present.

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

PARMPROG=EZTPX01
Provides an option to substitute EZTPX01 with your own program.

Specify a program name of your choice, or one of the following:

EZTPX01
Considers the user-provided buffer length when returning the PARM value.

FSYTPX01
Ignores the user-provided buffer length (compatible with Easytrieve Plus).

| **PRINTIO=(&exitpgm,&option1,&option2,&option3)**
| Print file segmenting option when exit is used for printing.

| **&exitpgm**
| Globally activates &PGMNAME as the default I/O exit.

| **&option1**
|

SEGRPT
Call close function at end of each report (segment at end of each report).

SEGJOB
Call close function at end of each JOB in Easytrieve Plus program (segment at end of each JOB).

SEGEOJ
Call close function at real EOJ (segment at EOJ only).

The default is SEGRPT.

&option2

ALL Use standard DISPLAY for DISPLAY statements in the Activity section, that is, DISPLAY statements not located in REPORT exits and DISPLAY is to a printer file.

RPT Use FSYPXIT1 standard print I/O exit for DISPLAY statements in the Activity section, that is, DISPLAY statements not located in REPORT exits and DISPLAY is to a printer file.

&option3

NOEXPCC
Do not expand print control characters (groups blank lines and prints with print cc).

EXPCC
Expand print control characters (creates blank lines).

The default is NOEXPCC

PRINTER=SYSPRINT
Default ddname for the REPORT statement printer files. This option controls how PRINTER ddname is generated for those reports that do not have a PRINTER associated with them.

SYSPRINT
Print to SYSPRINT ddname

AUTOGEN
Generate a ddname automatically

&ddname
Any valid printer file ddname defined via the FILE statement.

Note: Most installations use SYSPRINT for Easytrieve Plus default printer. Using SYSPRINT for Migration Utility will make parallel testing of existing programs easier. Any other value will make parallel testing impossible.

You can code PRINTER=(&DDname,&lrecl) by means of the EASYTRAN option, locally in your program, when the &lrecl of your default printer (&GLINESIZE specified in EASYDTAB) needs to be changed. The &lrecl is the actual record length. Migration Utility adds 1 byte to it to compensate for the control character.

Example:

```
* EASYTRAN: PRINTER=(SYSPRINT,140)
```

Note that the length of the PRINTER= statement coded in EZPARAMS is ignored by Migration Utility. The length can be coded locally by means of the EASYTRAN option in the Easytrieve Plus program only.

RANGERR=NO/YES
Option to control out of range field definitions.

NO Generates fields out of range.

YES Flags out of range fields as error.

Example:

```
FIELDA      10   1   A OCCURS 20
FIELDB FIELDA +0 2 A
```

For RANGERR=NO, Migration Utility generates FIELDB independent of FIELDA because one byte FIELDA cannot serve as a group for two byte FIELDB.

For RANGERR=NO, Migration Utility flags FIELDB as an out of range error.

RECSIZE=NATIVE
Variable and undefined length record handling:

NATIVE
Handle record length by means of native COBOL statements.

CALL Handle record length by means of CALL to BAL modules.

RESET=NATIVE
Field reset option for fields defined with RESET option.

NATIVE
Fields are reset in the JOB initialization routine.

EASYT
Field are reset for every JOB cycle, i.e., every time the JOB is entered. This is compatible with Easytrieve Plus.

RETURNC=NATIVE/EASYT
Option for handling RETURN-CODE when calling sub-programs.

NATIVE
Migration Utility captures the return code after each CALL (compatible with standard calling conventions).

EASYT
Migration Utility ignores RETURN-CODE (compatible with Easytrieve Plus).

RFIELDS=1024
Maximum number of fields on a single report

SIZERROR=(ABEND,ALL)
Allows the user to disable the Decimal Divide Exception when division by 0 occurs. While dividing by 0 is mathematically incorrect, this option was added for compatibility reasons with Easytrieve Plus.

Migration Utility generates a COBOL COMPUTE statement for each assign statement that involves arithmetic. The COMPUTE statement syntax allows for the "ON SIZE ERROR" option that traps arithmetic exceptions, such as Decimal Divide exceptions, and so on. The “ON SIZE ERROR” traps the following exceptions:

- When product value overflows receiving field capacity
- Division by zero
- Exponent underflow/overflow
- ZERO raised to zero
- ZERO raised to a negative number
- A negative number raised to a fraction power

Use this option with utmost care and only when you are absolutely certain that suppressing the above exceptions will not impact the outcome.

You can code the following combinations:

(ABEND,ALL)
Do not generate “ON SIZE ERROR”. This is the default.

(ZERO,ALL)
Generate “ON SIZE ERROR MOVE ZERO TO &TARGETFIELD” for statements located in the Activity section and Report Exits. The “ON SIZE ERROR” is generated for COMPUTE statements that contain at least one division term, otherwise it is not generated.

(ZERO,EXIT)
Generate “ON SIZE ERROR MOVE ZERO TO &TARGETFIELD” for statements located in the Report Exits only. The “ON SIZE ERROR” is generated for COMPUTE statements that contain at least one division term, otherwise it is not generated.

RXITOVF=NATIVE/EASYT
Page overflow handling in report exits.

NATIVE
Use PAGE= value for page overflow in report exits. This limit is enforced when REXOVCK macro is in use.

EASYT
Use DISPLAY= value for page overflow in report exits.

When this option is in use, REXOVCK is automatically turned on, therefore the %REXOVCK ON is not required.

SORTWORK=0/n
Option for allocating Sort Work files.

n = The number of SORTWK files to allocate.

The default is SORTWORK=0.

Valid values are 0 thru 9. Maximum of 9 sort work files are allowed. Files are named as SORTWK01...SORTWK09.

The amount of space to allocate is obtained from the WRKSPACE= parameter.

Example:

```
SORTWORK=2,      would result in SORTWK01 and SORTWK02
                 to be allocated by the generated program.
```

SPOOLOPT=YES
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 uses storage more efficiently.

SQLBIND=ANY
Establishes the default SQL/DB2 application program run mode.

STATIC
Programs are to run in static mode. This option cannot be overridden via the PARM BIND (&option) in Easytrieve Plus source.

DYNAMIC
Programs are to run in dynamic mode. This option cannot be overridden via the PARM BIND (&option) in Easytrieve Plus source.

ANY BIND is supplied in Easytrieve Plus source. Run mode must be as supplied via the PARM BIND (&option) in Easytrieve Plus source.

SQLFLDS=1024
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.

SQLPLAN=&plan
Allows you to designate a default plan name for runtime of DB2 programs.

&plan must be a valid DB2 PLAN that your programs can use.

The supplied PLAN is used when no PLAN name is specified on the PARM statement.

SQLSSID=(&SYS)
Establishes the default DB2 system (SSID) if one is not supplied by the PARM SSID in Easytrieve Plus source.

&SYS Use SSID from DSNHDECP

&*ssid* A four-character DB2 SSID

Example:

```
SQLSSID=(DBVA)
```

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: &prod; 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.

STATSDD=(&OPTION,&CC)
Allows the user to direct JOB Statistics to the default printer (usually SYSPRINT).

&OPTION

OUTDD
Print statistics using COBOL DISPLAY statement. The COBOL DISPLAY statements are printed to the DDname as per COBOL "PROCESS OUTDD(&DDNAME)" option.

PRINTER
Prints statistics to the default printer specified by the PRINTER=(&DDname) option.

&CC Valid in combination with the PRINTER option only.

CH1 Skip to channel 1 when printing the first line of JOB statistics.

N Number of lines to advance before the first line of JOB statistics.

The default is STATSDD=(OUTDD).

Example:

Print statistics to SYSPRINT and skip to channel 1. Assume that PRINTER=(SYSPRINT) is specified.

```
STATSDD=(PRINTER,CH1)
```

Example:

Print statistics to SYSPRINT and advance 2 lines before printing. Assume that PRINTER=(SYSPRINT) is specified.

```
STATSDD=(PRINTER,2)
```

SYNCREC=REFRESH
Whether to refresh synchronized file records before invoking user-written logic.

REFRESH
Synchronized file records are to be refreshed every time before invoking the user-written logic. This option is compatible with Migration Utility version 1 and earlier.

NOREFRESH
Synchronized file records are not to be refreshed before invoking the user-written logic. This option allows you to modify synchronized file records. When duplicates are present on the master file, the changes previously made to the master file records are preserved and made available to the user written logic again.

SYNCSEQ=YES
Synchronized file sequence check option:

NO Do not generate code for sequence check. Use this option with caution. Out of sequence files do not produce the correct output. Therefore, use the NO option if you are sure that the files are in the correct sequence.

YES Generate code to check file sequence.

SYSOUT='*,REFDD=SYSPRINT'
Dynamic allocation statement for SYSOUT (report) files. To be used when DYNALLOC=YES is in effect. This must be a valid SYSOUT JCL statement enclosed in quotation marks.

TABLES=*nn*
This new parameter can be used to increase the number of tables supported by MU. The maximum number of tables cannot exceed the maximum number of files specified by the FILES= parameter. The default value for *nn* is 32.

TBMEMORY=DYNAMIC
Option for external tables memory.

STATIC
Tables are allocated in working storage.

DYNAMIC
Memory is dynamically allocated at run time and tables are loaded into the acquired memory. Use this option for large tables, to reduce the size of your load module, or for tables that are too large for working storage.

TESTNUM=NATIVE/EASYT
Numeric test sign handling option.

NATIVE
Do test as per previous IMU rules.

```
PROCESS NUMPROC(NOPFD/MIG)valid signs
    type    signed   unsigned
    ------- -------- --------
    packed    F,C      F,C
    display   F        F

PROCESS NUMPROC(PFD)valid signs
    type    signed   unsigned
    ------- -------- --------
    packed    C       F
    display   C       F
```

EASYT
Do test as per EZT Plus rules.

```
(the NUMPROC option has no impact on the outcome)
    type    signed   unsigned
    ------- -------- --------
    packed    F,C      F,C
    display   F        F
```

The default is TESTNUM=NATIVE

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 Chapter 10, “User exits,” on page 189.

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 redefinition of the field is not of the same attributes (for example, length, type)

NONE
No warning message for duplicate fields is issued

VSAMOER=(&option,DUPKEY=&code)
Empty VSAM file open error and duplicate key handling for IOMODE=DYNAM/DYNAM24:

&option

IGNORE
Accept empty VSAM files

ERROR
Issue error for empty VSAM files

The default is ERROR

&code

22 Return 22 in FILE-STATUS on duplicate key condition

21 Return 21 in FILE-STATUS on duplicate key condition

The default is 21.

Example:
```
VSAMOER=(IGNORE,DUPKEY=22)
```

WRKSPACE='(CYL,(10,50),RLSE),UNIT=SYSDA'
Space to be used for dynamically allocated files. Code valid SPACE parameters enclosed in quotation marks. Space must be allocated in track or cylinder units. The UNIT= parameter must also be provided, unless your installation SMS allows otherwise.

Be careful when allocating space, as it is used for all temporary files, whether small or large. A good practice is to allocate some reasonable primary space and provide for more secondary space.

You can override WRKSPACE in your program via the `'* EASYTRAN: WRKSPACE ... '` statement. In addition, you can provide space for unusually large temporary files in the JCL. For example, if there is a file that needs space for millions of records, you can code that specific file ddname with appropriate SPACE in the JCL.

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 in comment form via the "EASYTRAN: DCLINCL" statement to preserve compatibility with Easytrieve syntax. This is an alternative to the "SQL DCLINCL &NAME" form of DECLGEN inclusion. One or more "EASYTRAN: DCLINCL &NAME' statements can be included. 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" statement must be the last statement as shown.

```
**********************************************************************
* 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
```

Because the 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.

The interpreted PARM and EASYTRAN statements are marked with a "+P" on SYSLIST1 produced by Migration Utility. The "+P" to the left of the statement means that the statement was interpreted by Migration Utility.

Example:

```
00001+P PARM LINK (TESTPGM)
00002+P * EASYTRAN: PROCESS LIST,NOXREF,OPTIMIZE
00003+P * EASYTRAN: TABLES=64
```

```
00004   *EASYTRAN: IOMODE=NODYNAM
00005+P * EASYTRAN: DEBUG=(COBOL BLIST)
00006+P * EASYTRAN: PROCESS NONAME
00007+P * END-EASYTRAN
```

The PARM and EASYTRAN options can be provided in the form of one or more Easytrieve Plus macros. The following examples show where you can code macro statements that contain valid PARM/EASYTRAN statements (MACROA is the macro name):

Example 1:

MACROA is the first statement. Migration Utility interprets the MACROA macro when parsing for PARM and EASYTRAN statements.

```
%MACROA
PARM LINK (PROGRAM1)
* EASYTRAN: IOMODE=DYNAM
* EASYTRAN: DEBUG=(ESPI NOLKGO)
* END-EASYTRAN
FILE REPORT1 PRINTER
FILE REPORT2 PRINTER
⋮
```

Example 2:

MACROA is after the PARM statement,but before the first EASYTRAN statement. Migration Utility interprets the MACROA macro when parsing for PARM and EASYTRAN statements.

```
PARM LINK (PROGRAM1)
%MACROA
* EASYTRAN: IOMODE=DYNAM
* EASYTRAN: DEBUG=(ESPI NOLKGO)
* END-EASYTRAN
FILE REPORT1 PRINTER
FILE REPORT2 PRINTER
⋮
```

Example 3:

MACROA is after the PARM and in the middle of the EASYTRAN statements. Migration Utility treats MACROA as a non-EASYTRAN statement and terminates the search for additional EASYTRAN statements, that is, it treats the macro as END-EASYTRAN.

```
PARM LINK (PROGRAM1)
* EASYTRAN: IOMODE=DYNAM
%MACROA
* EASYTRAN: DEBUG=(ESPI NOLKGO)
* END-EASYTRAN
FILE REPORT1 PRINTER
FILE REPORT2 PRINTER
⋮
```

Example 4:

MACROA is after the END-EASYTRAN statement. The PARM statement is in front. Migration Utility treats MACROA as a non-EASYTRAN statement. The macro is not decoded or looked at.

```
PARM LINK (PROGRAM1)
* EASYTRAN: IOMODE=DYNAM
* EASYTRAN: DEBUG=(ESPI NOLKGO)
* END-EASYTRAN
%MACROA
FILE REPORT1 PRINTER
FILE REPORT2 PRINTER
⋮
```

Example 5:

MACROA is after the END-EASYTRAN statement. The PARM statement is not provided before it. Migration Utility decodes MACROA and looks for PARM statements.

```
* EASYTRAN: IOMODE=DYNAM
* EASYTRAN: DEBUG=(ESPI NOLKGO)
* END-EASYTRAN
%MACROA
FILE REPORT1 PRINTER
FILE REPORT2 PRINTER
⋮
```

COBOL Compiler PROCESS options

COBOL generated by Migration Utility is front-ended by the PROCESS statements located at the beginning of the EZPARAMS table.

The following COBOL PROCESS statements are required:

PROCESS ADV
: This option is required for print control character.

PROCESS TRUNC(BIN) or TRUNC(OPT)
: This option is required to make sure that two byte binary numbers are properly handled.

The following statements can be added to EZPARAMS:

PROCESS SQL
: Use this option if you want SQL statements to be translated by the COBOL Compiler. When specified, SQL statements are translated by the COBOL Compiler. The standard SQL translator step is bypassed by FSYTPA00 program. The SQL option is recognized when placed in the EZPARAMS table. The COBOL default is not recognized.

Be aware of consequences of any other options you might use. Describing every COBOL PROCESS option is beyond the scope of this document. Consult your COBOL Reference for a complete list of COBOL options.

The COBOL compiler PROCESS options are considered in the following sequence:

1. Your installation standard default PROCESS options.
2. Options included in the PARM= of the COBOL step.
3. Migration Utility default PROCESS options located at the beginning of the EZPARAMS file. These options are added to the generated COBOL program and override options in items 1 and 2 above. Note that you can code one or more PROCESS statement after the COPTION parameter. PROCESS must start in position 8.

   Example:

```
COPTION LIST=CND,ERRLIMT=32000,BUFSIZE=32000,EZLREF
PROCESS ADV,TRUNC(OPT),NOLIST,OPTIMIZE
⋮
EXCCL EASYTRAN
⋮
```

4. Process statements placed in your Easytrieve Plus program by means of the EASYTRAN override as described in "Embedding options in the program source" on page 239. These options are added to the generated COBOL program and override items 1, 2, and 3 above.

   Example:

   ```
   * EASYTRAN: PROCESS LIST,OPTIMIZE
   * END-EASYTRAN
   ```

The PROCESS options available may vary with different COBOL compiler versions. Consult your COBOL reference manual for detailed information on each option.

When using FSYTPA00 to translate your programs, the DEBUG=(COBOL) EZTRAN/EZPARAMS option is required to produce a COBOL compiler listing.

Chapter 13. 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 want 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 respectively.
- 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
  ```
- You can point to a member name or to a data set with a member name. To point to a data set with a wild card or A member name, use a literal or a working storage field to provide the data set name and member.

  Example:
  ```
  MOVE 'DSN=SYS1.MACLIB(*)' TO WS-DSN
  POINT FILEIN KEY EQ WS-DSN
  ```
  This allows you to retrieve information from multiple PDS files controlled from your program. Migration Utility keeps track of the opened data set name, and if it changes, it closes the previously accessed data set and opens the newly supplied data set.
- 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
– 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 “PDS/PDSE program example” (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.

PDS/PDSE program example

```
***********************************************************************
* 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

*---------------------------------------------------------------------*
* 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 ------------------------------*
```

Chapter 14. 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 in the SYS1.SFSYFJCC library and as macro source in the SYS1.SFSYCCLM library.

All date routines use the Gregorian Leap Year formula as follows:

The year is a leap year if one of these conditions holds:
- It is divisible by 4 and not divisible by 100,
- 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)

REXOVCK
: Page overflow control macro for REPORT exits

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.

Pan Audit replacement macros

The following macros are available:

ATTPCT
: Calculate sample size from given Population Size, Confidence, Precision and Error. (internal use by ATTSAMP2 macros only)

ATTSAMP1
: Prepares working storage for ATTSAMP2.

ATTSAMP2
: Stratified random sampling file and statistics.

DOLUNIT1
: Prepares working storage for DOLUNIT2.

DOLUNIT2
: Performs dollar unit sampling from a file based on dollar interval and top dollar value.

INTERTAB
: Define interval ranges for INTERVL macros.

INTERVL1
: Prepares working storage for INTERVL2.

INTERVL2
Stratified REPORT of a value by intereval range.

INTSAMP1
Prepares working storage for INTSAMP2.

INTSAMP2
Performs random sampling from a file based on an interval value threshold. It allows selection on a random number.

MULTDUP1
Prepares working storage for MULTDUP2.

MULTDUP2
Create a file of duplicate records from an input file.

OCCURS1
Prepares working storage for OCCURS2.

OCCURS2
Count occurences of field values and produces a graph.

POPCOUNT
Adds 1 to POP-COUNT.

POPSIZE1
Prepares fields for POPSIZE2.

POPSIZE2
Counts records in a file.

RANDPCT1
Job initiator for RANDPCT2 macro.

RANDPCT2
JOB logic for random sampling using percentage.

RANDSMP1
Defines fields for RANDPCT1/RANDXCT1 (internal use only).

RANDSMP2
Defines fields for RANDPCT2/RANDXCT2 (internal use only).

RANDSPAN
Generates random number between two values.

RANDTELL
Display random sampling information (internal).

RANDXCT1
Random sampling based on specific size.

RANDXCT2
Random sampling based on percentage.

RGHTJUST
Right-justifies value in a field (internal use only).

SRCECOMP
Compare program source (OLD vs NEW).

SRCHTBL
Search Confidence table (internal use by ATTPCT and STRATIF1 macros only).

STDDEV
Calculate Standard Deviation.

STRATTAB
Define interval ranges for STRATIF macros.

STRATMLT
Stratified random sampling inner macro.

STRATTAR
Stratified random sampling inner macro (used by STRATIF2).

STRATIF1
Prepares working storage for STRATIF2 (used by STRATIF2).

STRATIF2
Stratified random sampling

VERNUMP
: Verify packed decimal field for numeric value.

VERNUM2
: Verifies values for numeric (internal use only).

VERSUS1
: Prepares working storage for VERSUS2 .

VERSUS2
: Frequency distribution of a value by category

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

CYYDDD   CYYDDD,    .EXAMPLE: 0107182         002

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
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

MMYYD    MMYYDD,    .SPECIAL FORMATS FSDATEZ0  001
MMCCYYD  MMCCYYDD
YYMMD    YYMMDD,    .STANDARD FORMAT FOR FSDATEZ0
CCYYMMD  CCYYMMDD;
```

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 ('.')
```

ATTPCT macro: coding rules

Purpose:
Calculate the sample size from the given population size, confidence, precision and error.

Usage: %ATTPCT &size &conf &precision &error

Where:

&size Population size. It can be a field name or an integer.

&conf Confidence level (example: 95 means 95% confidence)

Allowed values are: 50, 68,75,80,85,90,95,96,97,98,99,

&precision
Precision

&error
Error

Notes: The ATTPCT macro is invoked from the ATTSAMP2 macro.

ATTSAMP1 macro: coding rules

Purpose:
Prepares working storage for ATTSAMP2 (Stratified random sampling file and statistics).

Usage: %ATTSAMP1 &infile &size &conf &precision &error &seed LRECL=&lrecl

Where:

&infile
The input file name. Must be a defined file name.

&size Population size. It can be a field name or an integer.

&conf Confidence level (example: 95 means 95% confidence)

Allowed values are: 50, 68,75,80,85,90,95,96,97,98,99.

&precision
Precision

&error
Error

&seed Random seed. This must be an integer or a numeric field.

&lrecl Record length of input file (ignored by IMU).

Notes: ATTSAMP1 prepares all working storage and resources for the ATTSAMP2 macro. When coded, it must be followed by the ATTSAMP2 macro.

ATTSAMP2 macro: coding rules

Purpose:
Stratified random sampling file and statistics.

This macro calculates a sample size from given parameters and produces random sampling file with statistics.

Usage: %ATTSAMP2 &fileout &dbfile PERFORM &procname

Where:

&fileout
DDname of output file. NOFILE suppresses the output file.

&dbfile
DBFILE write from &filein. This option is not supported by IMU.

&procname
User exit proc name. NOPROC means do not use the proc exit.

Notes: The ATTSAMP2 macro must be coded after the ATTSAMP1 macro as ATTSAMP1 prepares working storage and flags for ATTSAMP2.

Inner macros:
CONFTBL, ATTPCT, RANDTELL, SRCHTBL

Example (FILEIN and FILEOUT data is not shown):
```
FILE FILEIN  F (80)
COMPANY     1   2    A HEADING ('COMPANY')
BRANCH      3   3    A HEADING ('BRANCH')
OFFICER     6   4    A HEADING ('OFFICER')
WAGE       10  08 N  2 HEADING ('WAGE')
RATE       18  05 N  3 HEADING ('RATE') MASK 'ZZ.ZZZ'
```

```
FILE FILEOUT F (80)
COPY FILEIN
POP-SIZE S 4 B

JOB INPUT FILEIN
POP-SIZE = POP-SIZE + 1
GOTO JOB

%ATTSAMP1 FILEIN POP-SIZE 95.00 5.0 5.0 100
%ATTSAMP2 FILEOUT
```

Produces this report:

```
             ATTRIBUTE SAMPLING REPORT
                INPUT PARAMETERS
INPUT FILENAME                         FILEIN
TOTAL POPULATION SIZE                       126
REQUIRED PRECISION                            5.00
REQUIRED CONFIDENCE LEVEL                    95
ERROR RATE                                    5.00
               SAMPLE RESULTS
SAMPLE PERCENTAGE REQUIRED                   36.50793650%
SAMPLE SIZE REQUIRED                         46
                 SAMPLE FILE
ATTSAMP SAMPLING REPORT FOR FILE:      FILEIN
NUMBER OF RECORDS PROCESSED                 126
NUMBER OF RECORDS REQUESTED                  46
NUMBER OF RECORDS IN SAMPLE FILE             46
```

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 default threshold value being used 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 default threshold value being used 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 default threshold value being used 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 a valid date, and 'NO' for an 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 want 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 default threshold value being used 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 default threshold value being used 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
```

DOLUNIT1 macro: coding rules

Purpose:
Prepares working storage for DOLUNIT2 (dollar unit sampling).

Usage: %DOLUNIT1 &infile &field &width &cutoff &seed VALUE &value REPORT &report

Where:

&infile
The input file name.

&field The Dollar Unit field name.

&width
Dollar Unit cell width.

&cutoff
Dollar Unit cut off.

&seed Random seed.

&value

ABS	Absolute value.
ACT	Actual value.
POS	Positive value.

&report

YES	Produce a report.
NO	Do not produce a report.

Notes: DOLUNIT1 prepares all working storage and resources for DOLUNIT2 macro. When coded, it must be followed by the DOLUNIT2 macro.

DOLUNIT2 macro: coding rules

Purpose:
Dollar unit sampling from a file based on dollar interval and top dollar value.

Usage: %DOLUNIT2 &fileout &prio1 &prio1o &prio2 &prio2o &dbfile &filein PERFORM &procname

Where:

&fileout
DDname of output file. NOFILE suppress the output file.

&prio1

KEY	Selection by key.
TOP	Selection by top value.

&prio1o
DDname of output file. NOFILE suppress the output file.

&prio2

KEY	Selection by key.
TOP	Selection by top value.

&prio2o
DDname of output file. NOFILE suppress the output file.

&dbfile
DBFILE write from &filein.

&filein
Input file DDname to write from.

&procname
User exit proc name. NOPROC do not use proc exit.

Notes: DOLUNIT2 macro must be coded after the DOLUNIT1 macro as DOLUNIT1 prepares working storage and flags for DOLUNIT2.

In the &procname exit, you can unconditionally test the following for "YES" and do whatever processing is needed:

```
DOLUNIT-SELECTED
DOLU&prio1.-SELECTED
DOLU&prio2.-SELECTED
```

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:** %GETPARM &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
```

| INTERTAB macro: coding rules

| **Purpose:**
| Define interval ranges for INTERVL macros (Stratified REPORT of a value
| by intereval range).

| **Usage:** %INTERTAB &value-1. . . &value-*n*

| Where:

| **&value-1 . . . &value-*n***
| Range values (must be a constant).

Repeat as many values as you need. The last value is the materiality value. Each range is generated between the &value(*n*-1) and &value(*n*).

Notes: The INTERTAB macro is used by the INTERVL1 and INTERVL2 macros. If used, the INTERTAB macro must be prepared in the library section.

Example:

```
%INTERTAB
       1200 +
       1500 +
       1900 +
       2500 +
       4000 +
       5000 +
       9000 +
      19000 +
      27000
```

INTERVL1 macro: coding rules

Purpose:
Prepares working storage for INTERVL2 (Produce stratified REPORT of a value by intereval range)

Usage: %INTERVL1 &infile &amount &intrvlsize &materialioty &graph &graphpct &stars + LOWERLIM (&lowlim) PERCENTAGE (&percentage) LRECL (&lrecl)

Where:

&infile
Input file name. Must be a defined file name.

&amount
Amount

&intrvlsize
Interval size. It can be a field name or an integer.

&materiality
Materiality value (upper value limit)

&graph
Graph option NOGRAPH/GRAPH

&graphpct
Value from 0 to 100. Percentage of value to graph.

&stars Value 1 to 9, Number of stars (*) for each unit

&lowlim
Lowest value to consider: A value. Is a constant or a field name.

INTERTAB
Use the interval table prepared in the program (see INTERTAB macro coding conventions).

LOGARITHM
Produces intervals in increments of 10 to the *n*th power.

&percentage
ABS/NEG/POS (percentage option)

ABS Show the percentage as an absolute value.

NEG Show negative and positive percentages.

POS Show positive percentage only.

&lrecl Record length of input file (ignored by IMU).

Notes: INTERVL1 prepares all working storage and resources for the INTERVL2 macro. The INTERVL2 macro must follow the INTERVL1 macro.

When the GRAPH option is in use, an "(O)" is printed next to any bar line that exceeds the print line capacity.

Inner macros:
ALPHACON

INTERVL2 macro: coding rules

Purpose:
Stratified REPORT of a value by intereval range.

Usage: %INTERVL2 DBFILE(&dbfile) REPORT (&report)

Where:

&dbfile
DBFILE write from &filein. This option is not supported by IMU.

&report
SHORT/LONG. This option is ignored by IMU.

Notes: The INTERVL2 macro must be coded after the INTERVL1 macro as INTERVL1 prepares working storage and flags for INTERVL2.

Inner macros:
INTRVTAR, COBOL

Example:
Produce a stratified report and a graph of LSTSTMBAL tabulated in increments of 100,000 with materiality of 900,000.

```
FILE FILEIN
LSTSTMBAL     1  9  N 0

%INTERVL1 FILEIN LSTSTMBAL 100000.00 900000.00 GRAPH 0 1
%INTERVL2
```

When file FILEIN contains this data:

```
000437668
000586574
000612637
000626964
000767383
000767393
000831763
000835111
000895523
000987076
000934173
000075587
000126396
000215541
000298678
000720322
000783364
000830675
000914162
000981506
000166640
000036716
000386778
000769329
000001109
000053446
000370120
000616266
000658834
```

```
000750964
000907004
000199754
000644309
001133257
```

The example program produces this report:

```
08/12/12                           FREQUENCY ANALYSIS OF LSTSTMBAL                                                     PAGE      1
                                      FROM INPUT FILE FILEIN
                        WITH AN INTERVAL OF          100,000.00  AND A MATERIALITY OF           900,000.00
                                MAXIMUM VALUE:      1,133,257.00  MINIMUM VALUE:         1,109.00

------------  RANGE  -------------          TOTAL                 COUNT        PCT             MEAN                STD DEV

                 <       0.00                        .00                    0       .0                  .00                    .00
                 =       0.00                        .00                    0       .0                  .00                    .00
            0.01 -  100000.00                 166,858.00                    4       .8            41,714.50              27,197.27
       100000.01 -  200000.00                 492,790.00                    3      2.5           164,263.33              29,995.39
       200000.01 -  300000.00                 514,219.00                    2      2.6           257,109.50              41,568.50
       300000.01 -  400000.00                 756,898.00                    2      3.8           378,449.00               8,329.00
       400000.01 -  500000.00                 437,668.00                    1      2.2           437,668.00                    .00
       500000.01 -  600000.00                 586,574.00                    1      2.9           586,574.00                    .00
       600000.01 -  700000.00               3,159,010.00                    5     15.9           631,802.00              17,431.15
       700000.01 -  800000.00               4,558,755.00                    6     22.9           759,792.50              19,990.86
       800000.01 -  900000.00               3,393,072.00                    4     17.0           848,268.00              27,331.62
                  >  900000.00               5,857,178.00                    6     29.4           976,196.33              76,612.02

FINAL TOTALS                               19,923,022.00                   34    100.0           585,971.24             319,687.85
POSITIVE TOTAL                             19,923,022.00
NEGATIVE TOTAL                                       .00
ABSOLUTE VALUE TOTAL                       19,923,022.00

         FREQUENCY ANALYSIS GRAPH

                   <       0.00            .0
                   =       0.00            .0
              0.01 -  100000.00            .8     *
         100000.01 -  200000.00           2.5     ***
         200000.01 -  300000.00           2.6     ***
         300000.01 -  400000.00           3.8     ****
         400000.01 -  500000.00           2.2     **
         500000.01 -  600000.00           2.9     ***
         600000.01 -  700000.00          15.9     ****************
         700000.01 -  800000.00          22.9     ***********************
         800000.01 -  900000.00          17.0     *****************
                   >  900000.00          29.4     *****************************
```

INTSAMP1 macro: coding rules

Purpose:
Prepares working storage for INTSAMP2 (random sampling based on threshold).

Usage: %INTSAMP1 &infile &size &seed

Where:

&infile
The input file name.

&size Interval size.

&seed Random seed.

Notes: INTSAMP1 prepares all working storage and resources for INTSAMP2 macro. When coded, it must be followed by the INTSAMP2 macro.

INTSAMP2 macro: coding rules

Purpose:
Random sampling from a file based on an interval value threshold. It allows selection on a random number.

Usage: %INTSAMP2 &fileout &dbfile &filein PERFORM &procname

Where:

&fileout
DDname of output file. NOFILE suppress the output file.

&dbfile
DBFILE write from &filein.

&filein
Input file DDname to write from.

&procname
User exit proc name. NOPROC do not use proc exit.

Notes: INTSAMP2 macro must be coded after the INTSAMP1 macro as INTSAMP1 prepares working storage and flags for INTSAMP2.

MULTDUP1 macro: coding rules

Purpose:
Prepares working storage for MULTDUP2 (Create a file of duplicate records from an input file)

Usage: %MULTDUP1 &infile LRECL (&lrecl)

Where:

&infile
Input file name. Must be a defined file name.

&lrecl Record length of input file (ignored by IMU).

Notes: MULTDUP1 prepares all working storage and resources for the MULTDUP2 macro. When coded, it must be followed by the MULTDUP2 macro.

Inner macros:
None

MULTDUP2 macro: coding rules

Purpose:
Create a file of duplicate records from an input file

Usage: %MULTDUP2 &infile &sort &outfile &keys

Where:

&infile
Input file name. Must be the same file as used on MULTDUP1.

&sort S = file is sorted, U = file is unsorted. When S is specified, the file is assumend to be in the proper &keys sequence. When U is specified, the file is sorted by the specified &keys (see &keys below).

&outfile
Output file name. NOFILE suppresse the output file.

&keys Compare keys (fields to compare). One or more fields can be specified.

Notes: MULTDUP1 prepares all working storage and resources for the MULTDUP2 macro. MULTDUP2 must be coded after the MULTDUP1 macro.

The MULT-DUP flag contains “Y” when the current record is a duplicate and “N” if the current record is not a duplicate.

The flag can be tested for customized handling after the MULTDUP2 macro has been invoked.

Inner macros:
None

Example:
```
FILE FILEIN1 F (80)
COMPANY     1   2    A HEADING ('COMPANY')
BRANCH      3   3    A HEADING ('BRANCH')
OFFICER     6   4    A HEADING ('OFFICER')
WAGE       10  08 N  2 HEADING ('WAGE')
RATE       18  05 N  3 HEADING ('RATE') MASK 'ZZ.ZZZ'

FILE FILEIN2 F (80)
COPY FILEIN1

%MULTDUP1 FILEIN1
%MULTDUP2 FILEIN1 U FILEIN2 COMPANY BRANCH OFFICER

DISPLAY 'MULTDUP-FLAG: ' MULTDUP-FLAG COMPANY ' ' BRANCH ' ' OFFICER
```

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
```

OCCURS1 macro: coding rules

Purpose:
Prepares working storage for OCCURS2 (Counts occurences of field values and/or produces a graph)

Usage: %OCCURS1 &infile &graph &graphpct &stars

Where:

&infile
Input file name. Must be a defined file name.

&graph
Graph option NOGRAPH/GRAPH

GRAPH
Produce a graph

NOGRAPH
Do not produce a graph

&graphpct
Value from 0 to 100. Percentage of value to graph.

&stars Value 1 to 9, Number of stars (*) for each unit

Notes: OCCURS1 prepares all working storage and resources for the OCCURS2 macro. The OCCURS2 macro must follow the OCCURS1 macro.

When the GRAPH option is in use, an "(O)" is printed next to the bar line that exceeds the print line capacity.

Inner macros:
None

OCCURS2 macro: coding rules

Purpose:
Count occurrences of field values and produces a graph

Usage: %OCCURS2 &valuein

Where:

&valuein
Field name to tabulate values by.

Notes: OCCURS1 prepares all working storage and resources for the OCCURS2 macro. The OCCURS2 macro must follow the OCCURS1 macro..

When the GRAPH option is in use, an "(O)" is printed next to the bar line that exceeds the print line capacity.

Inner macros:
None

Example (MASTER content is not shown):

```
FILE MASTER
BYTE1  1   1  N

%OCCURS1 MASTER GRAPH 0 1
%OCCURS2 BYTE1
```

The program produces this report:

```
8/12/12                                FREQUENCY DISTRIBUTION OF BYTE1                              PAGE      1
                                            INPUT FILENAME MASTER

           BYTE1    COUNT    PCT
             0      12,999   72.2     *************************************************************************
             1       1,000    5.6     ******
             2       1,000    5.6     ******
             3       1,000    5.6     ******
             4       1,000    5.6     ******
             5       1,000    5.6     ******
             6           1     .0
                    18,000  100.0
```

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 respectively.
- &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 quotation marks with the necessary subscript:
```
%PARSE 'I-STRING (SUB1)' TOKEN 80 DELIM (',' '$' '!') PFX('B') SIZE(45)
```

POPCOUNT macro: coding rules

Purpose:
Adds 1 to POP-COUNT.

Usage: %POPCOUNT.

Notes: Use it to increment POP-COUNT field by one.

Example:
```
%POPCOUNT
```

POPSIZE1 macro: coding rules

Purpose:
Prepares fields and initiates a job for POPSIZE2.

Usage: POPSIZE1 &file.

Where:

&file The input file.

Notes: Use it to initiate POP-COUNT job for POPSIZE2.

This macro must be followed by the POPSIZE2 macro.

Example:
```
%POPSIZE1 FILEIN
```

POPSIZE2 macro: coding rules

Purpose:
Counts records in a file and prints total count..

Usage: %POPSIZE2.

Notes: This macro must be coded immediately after the POPSIZE1.

Examples:
```
%POPSIZE1 FILEIN2
%POPSIZE2
```

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.

RANDPCT1 macro: coding rules

Purpose:
Job initiator for RANDPCT2 macro.

Usage: %RANDPCT1 &file POP-SIZE &sample &seed SELECT (&sel)

Where:

&file The input file.

&sample
Percent sampling size.

&seed Random number generator seed.

&sel EXC exclude duplicate records when a random number is generated more than one time. INC generate an alternative random number when a random number is generated more than one time. The default is INC.

Notes: This macro must be followed by RANDPCT2 macro.

Example:
```
%RANDPCT1 FILEIN2 POP-SIZE 10 SYS-TIME SELECT (INC)
```

RANDPCT2 macro: coding rules

Purpose:
JOB logic for random sampling using a percentage of population size.

Usage: %RANDPCT2 NOFILE PERFORM &userproc

Where:

&fileout
Output file (optional)

&filein
Input file (optional). If not coded, &file from RANDPCT1 is used.

&userproc
User procedure entered for each selected record.

Notes: This macro must be coded immediately after the RANDPCT1 macro.

Example 1:
```
%RANDPCT2 NOFILE PERFORM PRT-SAMP2
```

Example 2:
```
%RANDPCT2 FILEOUT DBFILE FILEIN2 PERFORM PRT-SAMP2
```

RANDSPAN macro: coding rules

Purpose:
Generate random numbers within a range of values.

Usage: %RANDSPAN &field &seed &minvalue &maxvalue

Where:

&field User field to receive the random number. This must be a numeric field capable of holding &maxvalue.

&seed Random number seed.

&minvalue
Minimum value (low value).

&maxvalue
Maximum value (high value).

Notes: Each time macro is executed, a random number is returned in the &field field.

Example :
```
%RANDSPAN WS-RANDOM-NO 500 5000000 6000000
```

RANDXACT macro: coding rules

Purpose:
Create an output file of sample size number of records from a file.

Usage: %RANDXACT &poplsize &sampsize &seed PREFIX (&prefix)

Where:

&poplsize
Population size. It can be a field name or an integer.

&sampsize
Sample size (number of items to select).

&seed Random seed. This must be an integer or a numeric field.

&prefix
Prefix for unique field names generated by RANDXACT. The default is RX.

Notes: The RANDXACT macro is invoked from the ATTSAMP2 macro for random selection of sample size records.

The &prefix_SELECTED flag, available after the macro, indicates if the record was selected or not. When &prefix_SELECTED = 'YES', then the record is a random selection, otherwise it is not.

RANDXCT1 macro: coding rules

Purpose:
Job initiator for RANDXCT2 macro.

Usage: %RANDXCT1 &file POP-SIZE &sample &seed SELECT (&sel)

Where:

&file The input file.

&sample
Sample size.

&seed Random number generator seed.

&sel EXC exclude duplicate records when a random number is generated more than one time. INC generate an alternative random number when a random number is generated more than one time. The default is INC.

Notes: This macro must be followed by RANDXCT2 macro.

Example :
```
%RANDXCT1 FILEIN2 POP-SIZE 1000 SYS-TIME SELECT (INC)
```

RANDXCT2 macro: coding rules

Purpose:
JOB logic for random sampling using a specific sample size.

Usage: %RANDXCT2 NOFILE PERFORM &userproc or

%RANDXCT2 &fileout DBFILE &filein PERFORM &userproc

Where:

&fileout
Output file (optional).

&filein
Input file (option al). If not coded, &file from RANDXCT1 is used.

&userproc
User procedure entered for each selected record.

Notes: This macro must be coded immediately after the RANDXPCT1 macro.

Example 1:
```
%RANDXCT2 NOFILE PERFORM PRT-SAMP2
```

Example 2:
```
%RANDXCT2 FILEOUT DBFILE FILEIN2 PERFORM PRT-SAMP2
```

REXOVCK macro: coding rules

Purpose:
Controls page overflow handling for DISPLAY statements in REPORT exits.

Usage: %REXOVCK &option

where &option can have the value:

ON Turn page overflow test on.

OFF Turn page overflow test off.

If &option is not supplied, the default is ON.

Notes: The REXOVCK macro can be used in report exits only. When coded, the selected &option is active until END-PROC is reached or a new REXOVCK statement is encountered within the same exit. This allows you to turn overflow test ON or OFF as needed.

REXOVCK affects page overflow handling as shown in Table 1.

Table 1. Effect of %REXOVCK ON with different report exits

Report exit	Effect of %REXOVCK ON
REPORT-INPUT	Overflow check is activated for all DISPLAY statements. The page limit is DISPLAYSIZE lines.
BEFORE-LINE	Macro is ignored (not applicable).
AFTER-LINE	Overflow check is activated for all DISPLAY statements. The page limit is DISPLAYSIZE lines.
BEFORE-BREAK	Overflow check is activated for all DISPLAY statements. The page limit is DISPLAYSIZE lines.
AFTER-BREAK	Overflow check is activated for all DISPLAY statements. The page limit is DISPLAYSIZE lines.
END-PAGE	Macro is ignored (not applicable).
TERMINATION	Overflow check is activated for all DISPLAY statements. The page limit is DISPLAYSIZE lines.

Example:

```
AFTER-BREAK.PROC
   %REXOVCK ON
   DISPLAY 'AFTER BREAK'
END-PROC
```

RGHTJUST macro: coding rules

Purpose:
Right justify content of a field

Usage: %RGHTJUST &field &size

Where:
&field Field name to right justify.
&size Field size

Notes: Content of the field is right justified. The leading characters are padded with spaces.

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
```

SRCECOMP macro: coding rules

Purpose:
Compare program source (OLD vs NEW)

Usage: %SRCECOMP &filein1 &textin1 &filein2 &textin2 &option SUPCLST (&supclst)

Where:

&filein1
Old file

&textin1
Old file field to be compared

&filein2
New file

&textin2
New file field to compare

&option
CHANGES/ALL (this parameter is vaslidated but ignored by IMU).

&supclst
NO/YES. When YES, a replica of the IMB SUPERC list is produced to SUPCLST DDname. The default is NO.

Notes: The SRCECOMP macro compares source of two programs using the IBM ISRSUPC program. The SUPERC report is spooled to a temporary file then reformatted into a format produced by Easytrieve Plus Pan Audit SRCECOMP macro. The report produced by ISRSUPC is retained if the SUPCLST (YES) option is in effect, otherwise it is released.

The &textin1 and &textin2 fields must be of the same type and length.

Inner macros:
None

Example:

```
FILE OLDDD F (80)
COMPARE-OLD                  1  72   A
FILE NEWDD F (80)
COMPARE-NEW                  1  72   A

%SRCECOMP OLDDD COMPARE-OLD NEWDD COMPARE-NEW CHANGES SUPCLST(NO)
```

STDDEV macro: coding rules

Purpose:
Calculate Standard Deviation

Usage: %STDDEV &intramt &indicator &intstddev &totstddev

Where:

&intramt
Value to be sampled (amount)

&indicator
1 or 2 (entry indicator)

1 Establish new Strata (range)

2 Continue with existing Strata

&intstddev
: Standard deviation for current interval. This must be a valid numeric field defined by the user with two decimal places.

&totstddev
: Total Standard Deviation for the entire population. This must be a valid numeric field defined by the user with two decimal places.

Notes: The &textin1 and &textin2 fields must be of the same type and length.

Inner macros:
None

Example (FILEIN content is not shown):

```
FILE FILEIN  F (80)
 COMPANY     1   2    A
 BRANCH      3   3    A
 OFFICER     6   4    A
 WAGE       10  08 N  2
 RATE       18  05 N  3

FILE SRTFILE F (80)  VIRTUAL
COPY FILEIN

DEFINE INTAMT     W 8 P 2
DEFINE TOTAMT     W 8 P 2
DEFINE INTDEV     W 8 P 2
DEFINE TOTDEV     W 8 P 2
DEFINE SVCOMP     W 2 A VALUE SPACES
DEFINE TOTNUM     W 4 B VALUE ZERO
DEFINE FIRSTSW    W 1 N VALUE ZERO
DEFINE INDICATOR  S 1 N VALUE 1

SORT FILEIN TO SRTFILE USING (COMPANY)
JOB INPUT SRTFILE FINISH END-OF-JOB
IF COMPANY NE SVCOMP
   IF FIRSTSW EQ 1
      PRINT STDDEV-REPORT
   END-IF
   PERFORM NEW-INTERVAL
ELSE
   PERFORM OLD-INTERVAL
END-IF
PERFORM STANDARD-DEVIATION

NEW-INTERVAL. PROC
  FIRSTSW = 1
  INTAMT = WAGE
  SVCOMP = COMPANY
  INDICATOR = 1
  TOTNUM = 0
END-PROC

OLD-INTERVAL. PROC
  INTAMT = INTAMT + WAGE
  INDICATOR = 2
END-PROC

STANDARD-DEVIATION. PROC
  TOTAMT = TOTAMT + WAGE
  TOTNUM = TOTNUM + 1
  %STDDEV WAGE INDICATOR INTDEV TOTDEV
END-PROC

END-OF-JOB. PROC
```

```
      PRINT STDDEV-REPORT
   END-PROC

   REPORT STDDEV-REPORT LINESIZE 132
   TITLE 1 'STANDARD DEVIATION OF BONUS BY COMPANY'
   HEADING SVCOMP  ('COMPANY')
   HEADING INTAMT ('TOTAL', 'FOR COMPANY')
   HEADING INTDEV ('STD. DEVIATION', 'FOR COMPANY')
   HEADING TOTAMT ('TOTAL', 'ALL COMPANIES')
   HEADING TOTDEV ('STD. DEVIATION', 'ALL COMPANIES')
   LINE SVCOMP TOTNUM INTAMT INTDEV TOTAMT TOTDEV
```

The program produces this report:

```
8/12/12                                  STANDARD DEVIATION OF BONUS BY COMPANY                                      PAGE      1

                                      TOTAL            STD. DEVIATION             TOTAL            STD. DEVIATION
      COMPANY     TOTNUM          FOR COMPANY           FOR COMPANY          ALL COMPANIES          ALL COMPANIES

         10           11           705,300.00              2,992.36            705,300.00               2,992.36
         11            5           115,000.30              1,414.24            820,300.30              19,235.84
         12            2            40,300.04                 49.98            860,600.34              20,604.53
         13            3            60,600.18                 81.66            921,200.52              21,383.40
         14            3            61,500.06                 81.66            982,700.58              21,443.26
         15            5           104,600.30                623.90          1,087,300.88              20,924.26
         16            4            98,140.10              1,129.23          1,185,440.98              20,069.76
         17            4            82,700.17                346.58          1,268,141.15              19,536.61
         18            5           101,500.00                141.42          1,369,641.15              18,887.11
         19            5           103,300.00                205.91          1,472,941.15              18,230.61
```

STRATTAB macro: coding rules

Purpose:
Define interval ranges for STRATIF macros (Stratified random sampling).

Usage: %STRATTAB &value-1. . . &value-*n*

Where:

&value-1 . . . &value-*n*
Range values (must be a constant)

Repeat as many values as you need. The last value is the materiality value. Each range is generated between the &value(*n*-1) and &value(*n*).

Notes: The STRATTAB macro is used by the STRATIF1 and STRATIF2 macros. If used, the STRATTAB macro must be prepared in the library section.

Example:

```
%STRATTAB     +
       205.58 +
       524.00 +
      1001.68 +
      2007.84 +
      3273.92 +
      5101.10 +
      9203.88 +
     19106.65 +
     47375.38 +
     250000.00
```

STRATIF1 macro: coding rules

Purpose:
Prepares working storage for STRATIF2 (Stratified random sampling)

Usage: %STRATIF1 &infile &amount &targetin &conf &precision &materialioty &seed MAXSTRATA (&maxstrata) LRECL (&lrecl)

Where:

&infile
: Input file name. Must be a defined file name.

&amount
: Amount

&targetin
: Limit amount. It can be a field name or an integer. STRATTAB use table for strata instead of limit amount.

&conf Confidence level (example: 95 means 95% confidence)

Allowed values are: 50, 68,75,80,85,90,95,96,97,98,99.

&precision
: Precision

&materiality
: Materiality value (upper value limit, can be a constant or a numeric field). STRATTAB use last value in the table for materiality.

&seed Random seed. This must be an integer or a numeric field.

&maxstrata
: Maximum number of strat (not used by IMU).

&lrecl Record length of input file (ignored by IMU).

Notes: STRATIF1 prepares all working storage and resources for the STRATIF2 macro. The STRATIF2 macro must follow the STRATIF1 macro.

When STRATTAB is in use, code STRATTAB for &targetin and &materiality.

Inner macros:
CONFTBL, SRCHTBL

STRATIF2 macro: coding rules

Purpose:
Stratified random sampling

Usage: %STRATIF2 &fileout

Where:

&fileout
: Output file of randomly selected records.

Notes: The STRATIF2 macro must follow the STRATIF1 macro since STRATIF1 prepares all working storage and resources for STRATIF2.

Inner macros:
RANDXACT, STRATMLT, STRATTAR, COBOL

Example 1: (FILEIN content is not shown)

```
FILE FILEOUT

FILE FILEIN
    CHARGE-IN       32  5  P 2


*         FILE   FLD       TARGET CN PREC     MATER SEED
* -----------------------------------------------------------
  %STRATIF1 FILEIN CHARGE-IN 13010408 90 6505204 250000 1920
%STRATIF2 FILEOUT
```

The program produces this report:

```
 8/14/12                                             RESULTS OF STRATIFIED SAMPLING                                   PAGE      1
                                            INPUT FILENAME: FILEIN   INPUT FIELD: CHARGE-IN
                                                         VARIABLE STRATUM SIZE
                                               PRECISION:  6,505,204.00    CONFIDENCE:90%


                                                                                                 STD                SAMP         PCT
         FROM                   TO                FREQ               TOTAL                        DEV                SIZE         INT

                 1.00-                .01-              0                    .00                     .00               0        .0
                  .00                  .00              0                    .00                     .00               0        .0
                  .01               205.58        167,658          13,010,603.42                   54.19              20        .0
               205.59               524.00         40,808          13,011,370.80                   87.73               8        .0
               524.01             1,001.68         18,306          13,016,206.09                  123.80               5        .0
             1,001.69             2,007.84          9,288          13,005,208.60                  275.72               6        .0
             2,007.85             3,273.92          5,100          13,015,410.10                  342.53               4        .0
             3,273.93             5,101.10          3,165          13,007,553.27                  530.16               4        .1
             5,101.11             9,203.88          1,999          13,011,267.37                1,094.27               5        .2
             9,203.89            19,106.65          1,000          13,026,535.31                2,782.97               6        .6
            19,106.66            47,375.38            454          13,045,477.51                7,522.25               8       1.7
            47,375.39           250,000.00            173          12,954,450.55               35,214.68              14       8.0
           250,000.01 9,999,999,999,999.99              3           3,532,294.98              489,218.17               3     100.0

 FINAL TOTAL                                      247,954         133,636,378.00                5,228.06              83        .0

                                                   THE RECOMMENDED SAMPLE SIZE LESS MATERIALITY                       80


                       AN OUTPUT FILE OF SELECTED RECORDS WILL BE PRODUCED
```

Example 2: (FILEIN content is not shown. Program uses STRATTAB table for strata limits)

```
  FILE FILEOUT

  FILE FILEIN
      CHARGE-IN       32  5  P 2

  %STRATTAB   +
          1205.58 +
          2524.00 +
          3273.92 +
          5101.10 +
          9203.88 +
         19106.65 +
         47375.38 +
        260000.00


  *          FILE   FLD      TARGET    CN PREC     MATER   SEED
  * ---------------------------------------------------------------
  %STRATIF1 FILEIN CHARGE-IN STRATTAB  90 6505204 STRATTAB 1920
%STRATIF2 FILEOUT
```

The program produces this report:

```
 8/14/12                                             RESULTS OF STRATIFIED SAMPLING                                   PAGE      1
                                            INPUT FILENAME: FILEIN   INPUT FIELD: CHARGE-IN
                                                         VARIABLE STRATUM SIZE
                                               PRECISION:  6,505,204.00    CONFIDENCE:90%


                                                                                                 STD                SAMP         PCT
         FROM                   TO                FREQ               TOTAL                        DEV                SIZE         INT

                 1.00-                .01-              0                    .00                     .00               0        .0
                  .00                  .00              0                    .00                     .00               0        .0
                  .01             1,205.58        229,593          42,141,494.63                  220.76             222        .0
             1,205.59             2,524.00          9,156          16,028,403.94                  398.18              16        .1
             2,524.01             3,273.92          2,411           6,888,900.44                  216.01               2        .0
             3,273.93             5,101.10          3,165          13,007,553.27                  530.16               7        .2
             5,101.11             9,203.88          1,999          13,011,267.37                1,094.27              10        .5
             9,203.89            19,106.65          1,000          13,026,535.31                2,782.97              12       1.2
            19,106.66            47,375.38            454          13,045,477.51                7,522.25              15       3.3
            47,375.39           260,000.00            173          12,954,450.55               35,214.68              27      15.6
           260,000.01 9,999,999,999,999.99              3           3,532,294.98              489,218.17               3     100.0

 FINAL TOTAL                                      247,954         133,636,378.00                5,228.06             314        .1

                                                   THE RECOMMENDED SAMPLE SIZE LESS MATERIALITY                      311
```

```
AN OUTPUT FILE OF SELECTED RECORDS WILL BE PRODUCED
```

TIMECONV macro: coding rules

Purpose:
Converts input time provided in hundreds of a second to HHMMSSTT.

Where:

&time-in
Time to convert in hundreds of a second units.

&time-out
Converted time as HHMMSSTT.

Usage: %TIMECONV &time-in &time-out.

Notes: &time-in and &time-out fields must be numeric fields.

Example:

```
%TIMECONV WS-TIME-IN WS-TIME-OUT
```

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
```

VERNUMP macro: coding rules

Purpose:
Verify packed decimal field for numeric value

Usage: %VERNUMP &valuein

Where:

&valuein
Field name to evaluate. This must be a packed field.

Notes: The packed field &valuein is tested for proper numeric value, including sign. If the field value is not numeric, the process is stopped with RC=16.

Inner macros:
None

Example (MASTER content is not shown):

```
FILE MASTER
AMOUNT 1   5  P

JOB INPUT MASTER
%VERNUMP AMOUNT
GO TO JOB
```

VERNUM2 macro: coding rules

Purpose:
Verify a field for for numeric value

Usage: %VERNUM2 &valuein &recid

Where:

&valuein
Field name to evaluate.

&&recid
Identifier displayed when a non numeric field is detected.

Notes: The field &valuein is tested for a proper numeric value, including sign. If the field value is not numeric, the process is stopped with RC=16.

Inner macros:
None

Example (MASTER content is not shown):

```
 DEFINE WCOUNT  W  4 B
`    FILE MASTER
 AMOUNT-N 1   5  N

 JOB INPUT MASTER
 WCOUNT = WCOUNT + 1
 %VERNUM2 AMOUNT WCOUNT
 GO TO JOB
```

VERSUS1 macro: coding rules

Purpose:
Prepares working storage for the VERSUS2 (Frequency distribution of a value by category)

Usage: %VERSUS1 &infile &amount &category &graph &graphpct &stars LRECL (&lrecl)

Where:

&infile
Input file name. Must be a defined file name.

&amount
Amount to be tabulated

&category
Category field (control field).

&graph
Graph option NOGRAPH/GRAPH

&graphpct
Value from 0 to 100. Percentage of value to graph.

&stars Value 1 to 9, number of stars (*) for each unit

&lrecl Record length of input file (ignored by IMU).

Notes: VERSUS1 prepares all working storage and resources for the VERSUS2 macro. The VERSUS2 macro must follow the VERSUS1 macro. When the GRAPH option is in use, an "(O)" is printed next to the bar line that exceeds the print line capacity.

Inner macros:
ALPHACON

VERSUS2 macro: coding rules

Purpose:
Frequency distribution of a value by category

Usage: %VERSUS2 DBFILE (&dbfile)

Where:

&dbfile
This parameter is ignored by IMU.

Notes: VERSUS1 prepares all working storage and resources for the VERSUS2 macro. The VERSUS2 macro must follow the VERSUS1 macro. When the GRAPH option is in use, an "(O)" is printed next to the bar line that exceeds the print line capacity.

Inner macros:
None

Example (FILEIN1 content is not shown):

```
FILE REPORT1 PRINTER (141)

FILE FILEIN1 F (80)
COMPANY     1   2    A HEADING ('COMPANY')
BRANCH      3   3    A HEADING ('BRANCH')
OFFICER     6   4    A HEADING ('OFFICER')
WAGE       10  08 N  2 HEADING ('WAGE')
RATE       18  05 N  3 HEADING ('RATE') MASK 'ZZ.ZZZ'

WS-BASE S  4  B VALUE 0
WS-XDUP S  4  B VALUE 2
WS-BIN3 S  3  B


%VERSUS1  FILEIN1 WAGE COMPANY GRAPH 0 2
%VERSUS2
```

The program produces this report:

```
08/12/12                                   FREQUENCY DISTRIBUTION OF                                      PAGE      1
                                              WAGE VERSUS COMPANY
                                             INPUT FILENAME FILEIN1

                  TOTAL
    COMPANY       GROSS           COUNT    PCT     MEAN      STD DEV    MINIMUM    MAXIMUM

      10        1,020,000.00        17  22.4  60,000.00        .00  60,000.00  60,000.00
      11          100,000.00         5   2.2  20,000.00        .00  20,000.00  20,000.00
      12           20,000.00         1    .4  20,000.00        .00  20,000.00  20,000.00
      13           60,000.00         3   1.3  20,000.00        .00  20,000.00  20,000.00
      14           60,000.00         3   1.3  20,000.00        .00  20,000.00  20,000.00
      15          100,000.00         5   2.2  20,000.00        .00  20,000.00  20,000.00
      16           80,000.00         4   1.8  20,000.00        .00  20,000.00  20,000.00
      17           80,000.00         4   1.8  20,000.00        .00  20,000.00  20,000.00
      18          100,000.00         5   2.2  20,000.00        .00  20,000.00  20,000.00
      19          220,000.00        11   4.8  20,000.00        .00  20,000.00  20,000.00
      20          853,600.00        16  18.7  53,350.00   4,854.89  44,500.00  60,000.00
      40          222,000.00         4   4.9  55,500.00   2,500.00  53,000.00  58,000.00
      50          960,000.00        16  21.1  60,000.00        .00  60,000.00  60,000.00
      60           60,000.00         1   1.3  60,000.00        .00  60,000.00  60,000.00
      61          100,000.00         5   2.2  20,000.00        .00  20,000.00  20,000.00
      62           20,000.00         1    .4  20,000.00        .00  20,000.00  20,000.00
      63           60,000.00         3   1.3  20,000.00        .00  20,000.00  20,000.00
      64           60,000.00         3   1.3  20,000.00        .00  20,000.00  20,000.00
      65          100,000.00         5   2.2  20,000.00        .00  20,000.00  20,000.00
      66            80,000.00         4   1.8  20,000.00        .00  20,000.00  20,000.00
      67            80,000.00         4   1.8  20,000.00        .00  20,000.00  20,000.00
      68           100,000.00         5   2.2  20,000.00        .00  20,000.00  20,000.00
      69            20,000.00         1    .4  20,000.00        .00  20,000.00  20,000.00

 TOTAL           4,555,600.00       126 100.0  36,155.55  18,846.17  20,000.00  60,000.00

08/12/12    FREQUENCY DISTRIBUTION GRAPH                                                        PAGE      1

                TOTAL
 COMPANY        GROSS         PCT

   10          1,020,000.00  22.4      ********************************************
   11            100,000.00   2.2      ****
   12             20,000.00    .4
   13             60,000.00   1.3      **
   14             60,000.00   1.3      **
   15            100,000.00   2.2      ****
   16             80,000.00   1.8      ****
   17             80,000.00   1.8      ****
   18            100,000.00   2.2      ****
   19            220,000.00   4.8      **********
   20            853,600.00  18.7      **************************************
   40            222,000.00   4.9      **********
   50            960,000.00  21.1      ******************************************
   60             60,000.00   1.3      **
   61            100,000.00   2.2      ****
   62             20,000.00    .4
   63             60,000.00   1.3      **
   64             60,000.00   1.3      **
   65            100,000.00   2.2      ****
   66             80,000.00   1.8      ****
   67             80,000.00   1.8      ****
   68            100,000.00   2.2      ****
   69             20,000.00    .4
```

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 default threshold value being used 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 15. Easytrieve Classic translator

Migration Utility (MU) V3R1M0 and later versions provide for translating of Easytrieve (Easytrieve Classic) programs to Easytrieve Plus and/or COBOL. The translating process is similar to the process employed for translating Easytrieve Plus programs to COBOL via the FSYTPA00 one step driver program.

Activating Easytrieve Classic conversion PROCs

The following three PROCS, used by the Migration Utility FSYTPA00 one-step driver program for Easytrieve Classic, are located in `SYS1.SFSYJCLS`:

#TRVPROC
: Generates Easytrieve Plus and COBOL, compiles COBOL and optionally links the load module.

#TRVPLUS
: Generates Easytrieve Plus only.

#CNVTREV
: Used by the Automated Conversion Engine (JCTYRVCV1) for conversion of Easytrieve Classic programs.

Before you use one of these PROCs, the following steps must be performed by the Migration Utility administrator and installer:

1. Tailor the PROCs to use the correct IMSRLIB, LE COBOL libraries, and so on. &SYS1 symbols are replaced by the Migration Utility translator (FSYTPA00 program) automatically. Symbols are replaced by the values found in the FSYPROCS table that was compiled and linked during the Migration Utility installation process. The PROCs are internally interpreted by FSYTPA00. They are not suitable for external use.

   Important tailoring options

   a. SYSPRDD= and SYSUTDD= options

      The SYSPRDD=&DDname option in #TRVPROC/#TRVPLUS/#CNVTREV directs listings produced by COBOL, Link step and SQL translator steps to a specific DD name.

      The default is SYSPRDD=FJSYSPR. This DD name is internally allocated and used by Migration Utility. Do not code it in the JCL.

      The SYSUTDD=&DDname option in TRVPROC/#TRVPLUS/#CNVTREV is used to replace the SYSUT1-SYSUT7 COBOL compiler work DDnames. The default is SYSUTDD=SYSUT.

      SYSPRDD=SYSPRINT
      : will force Migration Utility to use SYSPRINT as the system printer. Be aware, using SYSPRINT for compiler listings may interfere with your reports archiving system if your Easytrieve Plus reports the default printer is SYSPRINT.

   b. While testing, you can change SVC99=MSGOFF in the PROCs to one of the following options:

      MSGOFF
      : Display dynamic allocator tracing.

MSGON
Display JES messages.

MSGALL
Display input text and JES messages.

MSGSER
Display messages of serious nature only.

MSGTXT
Display input text only.

Use the above options to trace the dynamic allocation problems that might arise during testing. For production use, SVC99=MSGOFF is recommended.

c. Upon tailoring, copy (from SYS1.SFSYJCLS) tailor and run JCTRVCLG and JCTRVCLK jobs to see if everything is okay.

Easytrieve Classic translator default options table

The translator uses the EZTRVPRM table to establish default options at initiation time.

The table is located in the SYS1.SFSYEZTS library. It can be tailored to installation standards.

The table is pointed to by the //FJSYSIN statement defined in the #TRVPROC, #TRVPLUS and #COVTREV procs.

Custom versions can be pointed to by placing //FJSYSIN in the JCL.

Some options can be imbedded in each Easytrieve Classic program source. This method allows the user to have specific options for each Easytrieve Classic program. The parameters are supplied at the beginning of the program but after the PARM/SUPPRESS, as comment statements. The method is fully described in "Embedding options in the program source" on page 239.

The first line in the EZTRVPRM member is the COPTION statement. The COPTION statement describes PENGICCL options such as the output listing and paragraph re-sequencing options. The COPTION parameters are fully described in Chapter 12, "Migration Utility translation options," on page 217. The defaults as set in the distributed EZTRVPRM member are sufficient, thus there is no need for change.

Options are processed by the EZTRVBAS macro. All options are keyword parameters.

Options must be coded using the standard PENGICCL conventions for coding macro instructions. That is, the word EXCCL starts in position 8 followed by the macro name. Any keyword and positional parameters on subsequent lines must be coded starting in position 12 or after.

The supplied default table follows:

```
COPTION ERRLIMT=32,BUFSIZE=32000
EXCCL EZTRVBAS
      FIELDS=2048,            MAX NUMBER OF FIELDS
      FILES=128,              MAX NUMBER OF FILES
      OBJECTS=1024,           MAX NUMBER OF OBJECTS
      MASKS=64,               MAX NUMBER OF MASK IDS
      INCLNST=16,             MAX NUMBER OF NESTED INCLUDES
      INCPRMS=64,            MAX NUMBER OF INCLUDE PARAMETERS
      COPY=(COPY),           COPY,++INCLIDE,-INC (N/A)
      IOMOD=FSYSQLIO,        DYNAMIC SQL I/O MODULE (N/A)
      DECIMAL=PERIOD,        PERIOD / COMMA
      CURRENCY=($),          CURRENCY SIGN
      TITLECOL=80,           TITLES END COL
      FOOTECOL=80,           FOOTER END COL
      SQLTRAN=EZTRV001,      TRANSLATOR MAIN DRIVER MACRO
      DEBUG=(LIST)
*--------------------------------------------------------------*
* COPYRIGHT 1989-2006, FOUNDATION SOFTWARE, INC.               *
* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.  *
*--------------------------------------------------------------*
      CLASSIC EASYTRIEVE TRANSLATOR;
```

Here are descriptions of the available keywords. The value is the default value:

COPY=(COPY)
: This option is not currently used. Leave it as is.

FIELDS=2048
: The fields queue depth.

FILES=128
: Maximum number of supported files.

OBJECTS=1024
: Maximum number of supported objects, i.e., files, lines, etc.

MASKS=64
: Maximum number of mask IDs.

INCLNST=16
: Maximum number of nested includes (macros).

INCPRMS=64
: Maximum number of macro parameters (INCLUDE parameters).

DECIMAL=PERIOD
: Decimal point character. PERIOD or COMMA.

CURRENCY=($)
: Currency symbol.

TITLECOL=80
: End column for title lines; that is, T1, T2, T3, T4. Code a valid number (72 or 80). Consider changing this parameter when columns 72–80 do not qualify for title text.

FOOTECOL=80
: End column for end of report footers; that is, cards that start with an "F". Code a valid number (72 or 80). Consider changing this parameter when columns 72–80 do not qualify for footer text.

SQLTRAN=EZTRV001
: Main translator macro driver. Do not change.

DEBUG=(LIST)
: Listing and debugging options. For details, refer to "Embedding options in the program source" on page 239. `DEBUG=(ESPI)` activates Migration Utility's program interrupt trap.

After translating Easytrieve source to Easytrieve Plus, Migration Utility uses the EZPARAMS and EASYDTAB options tables for translating the derived Easytrieve

Plus code. The default options set in EASYDTAB for Easytrieve Plus may be incompatible with Easytrieve Classic needs.

To overcome the differences and to be able to separate the Easytrieve Classic options from the Easytrieve Plus options, consider adding the following options to EZTRVPRM:

DATEFORM=(&type,&format,&mask)
Provides the date usage override via EZTRVPRM.

&type SHORT for short (6 digit) date format

Available formats are: MMDDYY, YYMMDD, DDMMYY.

&mask
DATE Mask that corresponds to &format.

Examples:
DATEFORM=(SHORT,MMDDYY,99/99/99),

DTABNAME=EZTRDTAB
Overrides the default EASYDTAB options table. EZTRDTAB is located in &SYS.SFSYCCLM library. EZTRDTAB table is identical with EASYDTAB. Modify it to suit your Easytrieve Classic standards.

Additional copies of the table can be made and used.

Easytrieve Classic conversion jobs and PROCs

The following jobs, PROCs and Control files are located in SYS1.SFSYJCLS library:

#FJECNTL
The Parallel testing control file for Easytrieve Classic (located in SYS1.SFSYEZTS).

JCTRVCLG
In-stream PROC, runs Easytrieve Classic as Link and Go.

JCTRVCLK
In-stream PROC, translates Easytrieve Classic to COBOL, compiles and links load module.

JCTRVCV0
Automated conversion initiator for Easytrieve Classic.

JCTRVCV1
Automated conversion engine for Easytrieve Classic.

JCTRVPLS
In-stream PROC (generates Easytrieve Plus only).

#PGMSDIR
A table of program names used by the JCTRVCV0/JCTRVCV1 engines.

JCGENFIL
This utility creates the conversion environment libraries (the same job is used for Easytrieve Plus programs).

JCYMIG00
The Standard Parallel Testing utility JCL adjuster (the same job is used for Easytrieve Plus programs).

JCYMIG20
: Standalone compare utility (the same job is used for Easytrieve Plus programs).

JCYCNV60
: Easytrieve Classic programs discovery utility. This utility finds Easytrieve Classic programs by scanning PDS libraries. It is functionally equivalent to JCYCNV50 as described in Chapter 3, “Conversion guidelines,” on page 13. Use the supplied JCL JCYCNV60 to run this job. Embedded comments in the JCL describe all needed files.

JCMUCL1P
: Easytrieve Plus translator to COBOL (for ongoing use) (the same job is used for Easytrieve Plus programs).

#FJECNTL and #FJECNT2 are located in SYS1.SFSYEZTS. They are control files, similar to #FJICNTL, used by the JCYMIG00 job. For tailoring, read comments imbedded in the file.

The #PGMSDIR is located in SYS1.SFSYEZTS. This is a table of program names to be automatically converted by JCTRVCV0/JCTRVCV1 engines. You can include your own program names for the automated conversion. Note that the program names are coded in pos 1-8. The $END stops the engine.

How to use the supplied jobs and PROCs

You can choose to do conversions in one of the following ways:

1. Run JCTRVPLS and JCMUCL1P, or run JCTRVCLK only
   a. Use JCTRVPLS job to create Easytrieve Plus program source. Then, use JCMUCL1P Easytrieve Plus PROC to translate Easytrieve Plus to COBOL and create load modules. or Use JCTRVCLK to do everything in one step.
   b. To run the translated program, change the EXEC statement (PARM= statement for IMS), in your old Easytrieve Classic job, to execute your program instead of EASYTREV. You must use Migration Utility's SYS1.SFSYLOAD and the load library where you linked your program, on the JOBLIB or STEPLIB. The "//MACROS" DDname can be removed.

   For both methods above, you can run both Easytrieve and Migration Utility jobs, with the output files re-directed to disk, then use JCYMIG20 job (located in SYS1.SFSYJCLS) or any other suitable compare utility, to compare the outcome.
2. You can run Migration Utility, using Easytrieve Classic programs, in Link and GO mode.

   Example is the JCTRVCLG job.

   To run in Link and Go mode, change your existing Easytrieve job as follows (this method is suitable for continuous use of Easytrieve Classic syntax):
   a. Change EASYTREV on EXEC statement (PARM statement for IMS) to FSYTPA00.
   b. Add Migration Utility's SYS1.SFSYLOAD to the JOBLIB or STEPLIB
   c. Add Migration Utility Translator FJPROC0 DD to JCL (if not there)
   ```
   //FJPROC0 DSN=SYS1.SFSYJCLS(#TRVPROC),DISP=SHR
   ```

   You can run both, Easytrieve and Migration Utility jobs, with output files re-directed to disk, then use JCYMIG20 job (located in SYS1.SFSYJCLS) or any other suitable compare utility, to compare the outcome.

3. If you have more than 20 programs, consider using the Automated Conversion engine.

   The Automated Conversion engine generates the Easytrieve Plus source, the COBOL source and the load modules.

   Also, you must prepare a separate job such as JCMUCL1P for ongoing use of Easytrieve Plus.

   To run the Conversion Engine:

   a. Tailor and run JCGENFIL job to create conversion environment libraries (if you have not done so)
   b. Prepare a PDS file (such as #PGMSDIR located in SYS1.SFSYEZTS) of Easytrieve program names.
   c. Tailor JCTRVCV0 and JCTRVCV1 (if you have not done so). Note that #PGMSDIR is used by JCTRVCV0.
   d. Submit JCTRVCV0 and let it run.

      JCTRVCV0 submits JCTRVCV1 for each program, and JCTRVCV1 restarts JCTRVCV0 until all programs are done. If you have a substantial number of programs, the job can be left to run at night. Make sure that there is enough space allocated for the output files (FJSYSPH, FJEPLUS, FJSYSER, etc.)

      Upon completion, look at &USERID.MUCONV.ERRORS file for potential programs in error. JOBIDs and JOB names are listed in the error file.

      If programs require tweaking, make changes an translate individual programs with JCTRVCLK job.

      If errors are of general nature (i.e., JCL, options. etc.), correct the problem, create a PDS member of programs in error and re-submit JCTRVCV0.
   e. Run tests as suggested in paragraph #1.

Easytrieve Classic conversion hints

1. The Easytrieve Plus source is punched into FJEPLUS DDname. The COBOL Source is punched into FJSYSPH DDname.
2. Migration Utility provides special hooks for running Easytrieve Plus programs derived from the Easytrieve Classic source. Thus, the derived Easytrieve Plus programs will not run with the standard CA's Easytrieve Plus compiler. Here are some major differences:
   - The PARM RUNMODE(CLASSIC) tells Migration Utility that the program is to be compatible with Easytrieve Classic.
   - The summary files are of different layout from those in Easytrieve Plus. Migration Utility preserves the Easytrieve Classic summary files layout for maximum compatibility.
   - The SORT statement, when it is not the first statement in Easytrieve Classic logic, makes two passes thru the JOB activity section that follows the generated SORT statement. "JOB INPUT <SORT>" syntax has been adopted.
   - Easytrieve Classic MERGE and MATCH statements are handled by Migration Utility the way Easytrieve Classic does. There is no equivalent in CA's Easytrieve Plus compiler.
   - Totals on control breaks are printed with a tag compatible with Easytrieve Classic. There is no equivalent in CA's Easytrieve Plus compiler.
   - Other miscellaneous differences exist.
3. The translated programs should be well tested before promoting them into production.

4. If you remove PARM RUNMODE(CLASSIC) from the generated Easytrieve Plus source, you may be able to make programs compatible with the standard Easytrieve Plus. The major issue would be the SORT statement as described above.
5. The Migration Utility translator for Easytrieve Classic supports the same '* EASYTRAN: ' syntax used for Easytrieve Plus. Because the Easytrieve Classic programs are converted to Easytrieve Plus, you can take advantage of all supported options of EASYTRAN/EZPARAMS as described in the reference manual. Options can be placed after the PARM or SUPPRESS statement in the Easytrieve Classic source.
6. If you have Easytrieve Classic macros, they must be placed into a PDS. Code //MACROS DDname in the Migration Utility JCL for Easytrieve Classic macros.
7. In general, if you code SYSLMOD in the JCL, the program will be translated and linked. If SYSLMOD is not detected in the JCL, the program is translated and run as Link and GO. The SYSLIB statement in the JCL should point to object libraries that contain includes/sub-modules.
8. The Migration Utility translator for Easytrieve Classic supports the same %COBOL syntax used for Easytrieve Plus. The difference is that Easytrieve Classic scans columns 1–71. Use %COBOL as a last resort to resolve statements that cannot be handled by the translator.

Easytrieve Classic conversion compatibility issues

File organization and exits

Migration Utility translates Easytrieve Classic programs that use VSAM, QSAM, SAM, DLI/IMS, tape files, and unit record devices.

ISAM, IDMS and DA files are not supported.

LIST exit is supported but be aware that if your exit is running below the line (that is, AMODE 24), then you must translate your EZT Classic program to run in AMODE(24). You do so by specifying the EASYTRAN option at the beginning of EZT Classic source as follows:

- When running in IOMODE=DYNAM, code:

```
* EASYTRAN: IOMODE=DYNAM24
* END-EASYTRAN
```

- When Running with IOMODE=NODYNAM, code:

```
* EASYTRAN: PROCESS DATA(24)
* EASYTRAN: IOMODE=NODYNAM
* END-EASYTRAN
```

Exits coded on the FILE statement are supported. However, the same consideration exists as for the LIST exits above.

Record length and format

The correct record length and format must be coded on the FILE statement(s) when running in static mode (IOMODE=NODYNAM) for input and output files.

When running in dynamic mode (IOMODE=DYNAM), record length and format are not needed for input files. For output files, the record length and record format must be always coded.

Variable length record files

The record length coded on the FILE statement must include four extra bytes over the defined record length. For example, if the defined record length is 100, then the LRECL must be coded as 104. This rule applies to both input and output files.

PARM statement
The following PARM statements are recognized by MU:
- `LINESIZ=`*`nnn`*
- `NARROW=`*`nnn`*
- `SCANCOL=`*`nn`*

All other PARM and SUPPRESS statements located in the Easytrieve Classic source are bypassed by Migration Utility.

VSAM File key
When running with IOMODE=NODYNAM, the first defined field for a VSAM file must be the file key. The key must be defined as an alpha type and represent the correct starting position and length.

When running with IOMODE=DYNAM, the key definition is not needed as it is allocated at run time.

Sign of numeric fields
Easytrieve uses "C" for sign of positive numeric fields, even for unsigned fields (fields that are not defined as a quantity). COBOL uses an "F" for unsigned fields and a "C" for signed fields.

To combat this situation, refer to FSIGN=NO/YES/ALL EASYTRAN option in Chapter 12, “Migration Utility translation options,” on page 217. The Migration Utility default for Easytrieve Classic is FSIGN=ALL.

FILEB record length
FILEB record definition is derived from FILEA. If the FILEA record is partially defined, the FILEB record length will include only the defined portion of FILEA record.

In general this is not a problem if FILEB file is not saved and used by other programs. However, if FILEB is passed on to other programs, the FILEA record length should be correct in order to force the correct FILEB record length.

Program labels inside IF statements
Any reference labels places inside IF statement(s) are flagged by Migration Utility.

SORT statement issues
Easytrieve Classic makes two passes through the main logic when the SORT statement is encountered and the SORT statement is not the first statement of the main logic.

To duplicate the Easytrieve Classic logic, Migration Utility also makes two passes through the main logic.

You must be careful to make explicit tests for SORTED EQ NO or SORTED EQ YES when manipulating fields and writing to the output files. The Migration Utility output might differ from Easytrieve Classic output if an explicit decision is not made. For example: (the use of SORT is assumed)

```
IF FILE EQ FILE
    PUT FILEOUT
```

would write twice to the output file, one PUT before SORT and one after the SORT. The Easytrieve Classic opens the output file twice, one time before the sort and one time after the sort, thus unknown to the user, retains the records after the sort only. The overhead of double write is still there, however. Migration Utility MODs on to the file, keeping two records on the output file, one before the sort and one after the sort. The correct way of coding the above IF would be:

```
IF SORTED EQ YES
    PUT FILEOUT
```

Report Issues
: On Migration Utility generated reports, the starting position of printed fields may differ from those of Easytrieve Classic. In most cases, this is an acceptable difference, as long as the field values are correct. If you encounter a difference and you absolutely must have an identical output, tweak spacing between the fields using the &field- nn syntax.

Page overflow
: Unless overridden by the LIST statement, Migration Utility sets the number of lines per page to 60. There is not external method for changing this default.

Easytrieve Classic macros
: Easytrieve Classic macros must be placed into a PDS for Migration Utility. If you do have any macros, make sure that you place them into an 80 byte record length PDS before attempting conversion.

LINECTR differences
: Easytrieve Classic seems to be counting one extra line (the line about to be printed). Migration Utility counts lines that have been printed only. If your program is depending on some hard coded value, adjust the value by minus 1.

Other issues
: When you encounter an issue, use your intuition and the process of elimination to resolve it. Quite frequently, you will recognize ambiguous logic and/or object definitions. Simplifying things often results in a positive outcome.

The Parallel Testing Utility (JCYMIG00)

The most time consuming task is the validating process of the output files created by the Migration Utility generated program vs the output created by the Easytrieve JOB.

JCYMIG00 job generates parallel test jobs, from the original Easytrieve jobs that compares the outputs automatically.

In depth description of JCYMIG00 is described in "The Automated Parallel Testing utility" on page 22.

You must tailor #FJECNTL control before using JCYMIG00.

Conversion overview

The Easytrieve Classic processing revolves around a prearranged set of files, the FLUNK/SUCCESS flags, a primitive IF statement, a GOTO statement and a few assumptions (depending on the statements in use).

Easytrieve Plus is a procedural language. It includes instructions for structures programming, example, IF...END-IF, nested IF, PERFORM statement, separate procedures (like paragraph in COBOL), etc. It is a very powerful language.

Easytrieve Plus supports many statements that do not exist in Easytrieve Classic. (for example, multiple reports per job, multiple sorts, multiple summary files, nested IFs, parentheses in arithmetic expressions, special report exists).

The FILEA/FILEB concept does not exist in Easytrieve Plus. Programmers define files and names them in every I/O request, including the JOB statement. There is no automated FLUNK/SUCCESS mechanism. Every choice must be made by the programmer.

Migration Utility automatically compensates for the differences by generating appropriate logic that simulates non compatible statements. In some instances, there is one to one correspondence. The following table outlines the similarities and the differences in two languages:

Table 2. A comparison of Easytrieve Classic and Easytrieve Plus

Easytrieve Classic	Easytrieve Plus	Comments
*	*	Comment lines (when the 1st character is a star "*")
assign	assign	a = bParentheses can be used in arithmetic. Example: a = (B + (C * D)/100)
CALL	CALL	Call another program
COMPUTE	N/A	Report related computations(Migration Utility generates Report exits)
CONTROL	CONTROL	Report control breaks(EZP uses Reversed order)
N/A	DO WHILE	Do While Statement
N/A	DO UNTIL	Do Until statement
ELSE	ELSE	Used with IF statement
END	N/A	END of program/JOB
EDIT M	N/A	PUTLIST logic is invoked by Migration Utility for each implied SUCCESS
N/A	END-PROC	Marks end of a PROC
FILE	FILE	Define a file (defines files)
FINALS	TERMINATION exit	Display Final information
FLUNK	N/A	The flunk statement (de-selects records)
GET	GET	Get statement (reads a record)
GO TO	GO TO	GO TO statement (jumps to a reference label)
GOTO	GOTO	GO TO statement (jumps to a reference label)
HEXPRINT	DISPLAY HEX	Hex print (prints a field in hex)
IF	IF	The IF statementParentheses can be used for compounded IF. Example: IF A EQ B AND (C EQ D)
INCLUDE	%&MACNAME	Includes EZT macro
JOB	JOB INPUT	New JOB when multiple jobs in a single program
LABELS	REPORT LABELS	Print labels
LIST HITS	REPORT	List every matched or unmatched record
LIST MULTIPLE	REPORT	List every matched and unmatched record, including secondary file duplicates
LIST	REPORT LINE n	Report detail or summary line
SKIP	REPORT SKIP	
DETAIL	N/A	

Table 2. A comparison of Easytrieve Classic and Easytrieve Plus (continued)

Easytrieve Classic	Easytrieve Plus	Comments
SPACE	REPORT SPACE	
NOPAGE	REPORT NOPAGE	
LIMIT	REPORT LIMIT	
EVERY	REPORT EVERY	
NARROW	N/A	
LJ	REPORT NOADJUST	
SUMMARY	REPORT SUMMARY	
MATCH	N/A Special code	Match one or more files(Simulated for RUNMODE=CLASSIC)
MERGE	N/A Special code	Merge two file(Simulated for RUNMODE=CLASSIC)
N/A	Synch file	Synchronized file process MATCH/MERGE
MODE	N/A	Ignored by Migration Utility
MOVE	MOVE	MOVE statement
PARM	PARM	Ignored by Migration Utility
N/A	PERFORM	Perform a PROC
POINT	POINT	Point for VSAM I/O
PRESORT	SORT	Presort statement (sorts input FILEA
PRINT	DISPLAY	Display/Print (displays a line)
N/A	PROC	Marks beginning of a PROC
PUT	PUT	PUT statement(writes to a file)
PUTLIST	PRINT &REPORT	Invokes reports/LIST logic
READ	READ (VSAM only)	Read statement (used for IMS calls in classic)
N/A	DLI	IMS I/O Requests
N/A	SEARCH	Search a table
SELECT	RETRIEVE	IMS (Automatic input)
SORT	REPORT SEQUENCE	Sort statement (Sorts FILEA to FILEB in Eazytrieve Classic)Sorts report input in EZT Plus
STOP	STOP	Stop statement(stops run)
STOP AFTER	N/A	Stop after record is processed(Simulated for RUNMODE=CLASSIC)
N/A	STOP EXECUTE	Abend (Easytrieve Plus)
SUCCESS	N/A	Success statement (selects records)
SUMMARIZE	REPORT SUMMARY	Summarize statement(creates summary records to FILEB) (Record layout is different but simulated for RUNMODE=CLASSIC)
SUPPRESS	N/A	Ignored by Migration Utility
THEN	N/A	Used with IF statement (selection is not affected)
TTLEQT	N/A	Title equate (equates fields to the titles)
WRITE	N/A	Write statement (writes a record to FILEB)
N/A	WRITE	Write to a VSAM file
T1 - T4	REPORT TITLE &n	(report titles)
W fields	W	Working storage fields
S fields	W	Working storage fields
X fields	N/A	Working storage fields
SUB	N/A	SUB option
ADDRESS	N/A	RECORD pointer
TABLEs	TABLE	Support for table files
Ref Label	Ref Label	Reference label for GOTO(":" not needed)
Reserved fields		
EOF	EOF	End of file

Table 2. A comparison of Easytrieve Classic and Easytrieve Plus (continued)

Easytrieve Classic	Easytrieve Plus	Comments
FILE	N/A	FILE Flag
LENGTH	&FILE:RECORD-LENGTH	
LINECTR	LINE-COUNT	Line counter
MATCHED	MATCHED	Match flag
NO	N/A	
PRESORTED	N/A	
READSTAT	FILE-STATUS	IMS Status
RTNCODE	RETURN-CODE	Return code
SORTED	N/A	Sort flag
SUCCESS	N/A	Success flag
YES	N/A	'YES'

Chapter 16. Debugging Migration Utility programs

Migration Utility generates COBOL programs that are compiled with standard IBM COBOL, therefore, the runtime problems that you will experience are the same as for any other non-Migration Utility generated COBOL programs.

COBOL runtime problems fall into one of the following categories:

1. File I/O Errors

   File problems such as wrong LRECL, or no DD in the JCL. Migration Utility intercepts I/O errors automatically and logs relevant information to FJSYABE.
2. Program check interrupts

   Processing problems such as addressing exception (bad pointers), data exception, and specification exception.

   Migration Utility provides an optional Program Check Interrupt handler that intercepts program interrupts and logs relevant information to SYSOUT or FJSYABE (if available). This feature is activated by the DEBUG options ESPI or ESPI-PART or ESPI-FULL as described in Chapter 12, "Migration Utility translation options," on page 217.
3. Environmental errors and ABENDs

   These are matters such as JCL issues, RACF/security problems, and memory problems. Such problems are intercepted and logged by the operating system. In most instances, your job is cancelled by the operating system with appropriate messages.

This chapter explains how to read I/O Error messages logged to FJSYABE, how to activate and read Program check interrupt reports created by the Migration Utility's interrupt handler and what to do when the Migration Utility's Program Interrupt handler is not active.

File I/O errors

The I/O errors are automatically trapped by the generated COBOL program. Messages are printed to the system log, FJSYABE and SYSOUT. The FJSYSABE provides the most comprehensive description of the problem. The following information is included:

USER ABEND code (identifier number in the COBOL program)
FILE-STATUS Code as per COBOL Status Codes
Description of error as per COBOL manual
0C7 intentionally caused or RC=16, depending if FSABECOB or FSABEC16 program is in use. (See ABEND= EASYTRAN/EZPARAMS option).

Common I/O errors:

- Wrong LRECL
- Wrong VSAM Key definition (Static I/O only)
- No DD-name in the JCL
- Wrong file organization (KSDS, ESDS, etc)
- Wrong Record format (F/FB, V/VB, U)
- Pointing to wrong files
- Not enough disk space

Example:

```
06/01/06  TIME: 25:19:72  COPYRIGHT (C), FOUNDATION SOFTWARE, INC.
FSABEC16 - COBOL FILE I/O ABEND SUMMARY REPORT
USER ABEND CODE ===>  3003
PROGRAM NAME ======>  NONAME
PROGRAM TYPE ======>  COBOL-3
OPERATING SYS =====>  MVS
DATE COMPILED =====>  01/06/06
TIME COMPILED =====>  22.25
ABEND MESSAGE =====>  FILE I/O ERROR, RETURN CODE=35 FILEIN
MESSAGE DESCRIPTION:  PERMANENT I/O ERROR:
                      AN OPEN STATEMENT WITH THE INPUT, I-O, OR EXTEND
                      PHRASE WAS ATTEMPTED ON A NON-OPTIONAL FILE THAT
                      NOT PRESENT. HINT: CHECK JCL FOR MISSING DDNAME.
*END OF ABEND SUMMARY*
```

In the above example, the DD-name for FILEIN was not coded in the JCL as suggested by the hint. The User abend code of 3003 can be located in the COBOL source to see the exact location at the time of error (i.e. MOVE 3003 TO WS-PENGI-RETURN).

Program check interrupts

Processing problems such as addressing exception (bad pointers), data exception, and specification exception can be intercepted by the Migration Utility's optional Program Check Interrupt handler that intercepts program interrupts and logs relevant information to SYSOUT or FJSYABE (if available).

The Interrupt Handler supports programs compiled with z/OS Enterprise and LE COBOL.

Program interrupts are intercepted, analyzed and then cascaded down to other program check handlers that might be active on your system, including the LE COBOL abend handler. Processing terminates abnormally.

When Migration Utility's interrupt handler is active, you can potentially get multiple abend reports if other interrupt handlers are active.

How does the Interrupt Handler work

Migration Utility provides two runtime modules that intercept and analyze the interrupts:

1. FSYESPI1

   This modules activates the program check trap. It also receives control when an interrupt occurs. When a DEBUG ESPI option is specified, Migration Utility generates a call to FSYESPI1, at the beginning of the PROCEDURE DIVISION, to set an interrupt trap.
2. FSYESPI2

   This modules is the Interrupt Analyzer program. Upon a program check interrupt, FSYESPI2 receives control from FSYESPI1, opens COBLIST1 file if it exists in the JCL and prints an analysis log as follows:
   a. PSW, condition code along with the appropriate description
   b. COBOL statement that caused the interrupt
   c. Easytrieve Plus statement number found in the COBOL source position 1-6
   d. Value of each field used in the COBOL statement
   e. A trace of COBOL "PERFORM" statements (when NOOPTIMIZE is used)
   f. Working storage and RECORD fields (see ESPI-PART and ESPI-FULL options)

If COBLIST1 is not provided in the JCL, an analysis log of basic information is shown only.

How to use Migration Utility's program check Interrupt Handler

To use the interrupt handler, you must do the following:

1. When running with a load module (compiled programs)

   In the application JCL, include //COBLIST1 DD pointing to the COBLIST produced by the Migration Utility translator.

   To prepare COBLIST1:
   a. Translate Easytrieve Plus program with DEBUG=(ESPI/ESPI-PART/ESPI-FULL).
   b. Save Easytrieve Plus program listing (SYSLIST1 DD) produced by the translator.
   c. Save COBOL compiler listing (//COBLIST DD DSN=&DSN(&MEMBER)) into a PDS, where `&DSN` is a PDS name, `&MEMBER` is the program name.
2. When running in Link and Go mode Translate Easytrieve Plus program with DEBUG=(ESPI/ESPI-PART/ESPI-FULL). SYSLIST1 and COBLIST1 are readily available when running in Link and Go mode.

Note: For ESPI to function, programs must be compiled with the "PROCESS LIST,MAP" COBOL options. and the "COPTION LISTM,EZLREF" PEngiCCL options. Migration Utility automatically generates the required PROCESS and COPTION options when an ESPI option is in effect. You have no control over it.

How to activate the Interrupt Handler

Migration Utility's Interrupt Handler is activated by the DEBUG options ESPI, ESPI-PART or ESPI-FULL. The option can be supplied in the default EZPARAMS table, or it can be locally included in your Easytrieve Plus program, via EASYTRAN, as shown in the example below. The meaning of each option is as follows:

| **ABEND**

| This option traps the field overflow condition and abends run with
| RC=3000.

| The option is activated when ABEND in combination with an
| ESPI/ESPI-PART or ESPI-FULL on DEBUG= parameter is specified.

| By Z/OS architecture, the field overflow exception (SC0A) is not detected
| unless bit 21 of PSW is turned on. The user is responsible for making sure
| that data loss does not occur.

| When IMU's ESPI is not on, there is no way to trap the field overflow
| condition. Thus overflow goes unnoticed.

| When IMU's ESPI is on, overflow is detected and an ABEND report is
| created, however, MU percolates the ESPI exit to other potential handlers
| such as LE. Since LE does not trap overflow condition the program ends
| with RC=0.

| The IMU's ABEND option of DEBUG=() makes sure that overflow
| conditions result in an ABEND.

| When DEBUG=(ABEND, ESPI/ESPI-PART/ESPY-FULL) is specified, IMU's
| ESPI handler issues “ABEND 3000” to terminate the process. When

ESPI/ESPI-PART/ESPI-FULL is specified without the ABEND option, the IMU's ESPI handler percolates condition to LE and other potential interrupt handlers.

The LE runtime options TRAP(ON,SPIE) and ABTERMENC(ABEND) are required. If your installation is using different parameters for these two options, override them by coding a DD statement in the application JCL.

Example:

1. Create the CEEOPTS file in SYS1.V4R1M0.SFSYEZTS PDS and add the following to it:
   ```
   ABTERMENC(ABEND)
   TRAP(ON,SPIE)
   ```
2. Add the following DD statement to your application JCL when you executing linked programs, or to IMU translator JCL when you are running Link and Go.
   ```
   //CEEOPTS  DD  DSN=SYS1.V4R1M0.SFSYEZTS(CEEOPTS),DISP=SHR
   ```

   Note that for Link and Go, a DD statement can be added to the LKGO step in the #EZTPROC PROC.

ESPI This option sets the interrupt trap that prints basic information regarding the interrupt. Only fields and field groups used in the problem instruction are printed.

ESPI-PART
This option sets the interrupt trap that prints basic information regarding the interrupt, file flags, records and file counters as follows:
&FILE-IO-COUNT
&FILE-STATUS
&FILE-DYNAM-FIB
&FILE-RECORD

ESPI-FULL
This option sets the interrupt trap that prints basic information regarding the interrupt and the entire DATA DIVISION (working storage section, linkage section, and file records).

Example when ESPI-PART is placed in the default EZPARAMS file:

```
DEBUG=(LIST,ESPI-PART),
```

Example when ESPI-PART is placed locally in the Easytrieve Plus program:

```
***********************************************************************
* EASYTRAN: DEBUG (LIST ESPI-PART)                                    *
* END-EASYTRAN                                                        *
***********************************************************************
Program statements
     .
     .
```

Note that other DEBUG parameters are accepted on the same DEBUG statement.

Common cause of Program Interrupt exceptions

If data exception, identify the field in error

Check Easytrieve program for improper numeric field usage:
- Numeric field usage as alpha (if redefined as alpha)
- Initial value if defined in working storage

- Incoming data if from a file
- Damaged memory due to bad subscripts

If protection exception 0C4 or addressing exception 0C5

Verify run mode (COBOL PROCESS and Linker options):
- Use DATA(ANY) RMODE(24) for dynamic mode
- STATIC mode can run in 24 or 31 bit addressing mode

If calling sub modules:
- Make sure addressing mode is compatible
- Language compatibility (i.e., VS COBOL vs z/OS/COBOL/390)

If accessing beyond table range:
- Make sure DO loops are properly incrementing subscript/index (a common problem is that the subscript goes one beyond the top of the range).
- If program needs to access data beyond field range, compile with SSRANGE (be careful that other logic does not move beyond the table/array range).
- Check subscript field definition, make sure subscript is positive

Debugging example

The following program is intentionally programmed to cause a Data Exception (S07).

```
//JOBLIB   DD DSN=SYS1.SFSYLOAD,DISP=SHR
//         DD DSN=CEE.SCEERUN,DISP=SHR
//FSYTPA00 EXEC PGM=FSYTPA00
//FILEIN   DD DSN=FSOFT01.PENGI400.DATA(TESTFIL0),DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSLIST  DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//FJSYABE  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//FJSYABE  DD SYSOUT=*
//SYSIN    DD *
***********************************************************************
* EASYTRAN: DEBUG (LIST ESPI)                                         *
* END-EASYTRAN                                                        *
***********************************************************************
FILE FILEIN FB (80 0)
COMPANY   1  2  A
BRANCH    3  3  A
OFFICERR  6  4  A
WAGE      10 08 N 2  MASK ('ZZZ,ZZZ.99')
RATE      18 05 N 3  MASK ('ZZ.999')
*
WBONUS    W 05 P 2  MASK ('ZZZZ,ZZZ.99')

JOB INPUT FILEIN
PERFORM CREATE-REPORT

CREATE-REPORT. PROC
  MOVE X'FFFFFFFF' TO RATE
  WBONUS = (WAGE * (RATE / 100))
  PRINT RPT1
END-PROC

REPORT RPT1 SUMCTL (TAG) LINESIZE 80
CONTROL FINAL COMPANY BRANCH
```

```
TITLE 1 'TEST REPORT 1'
LINE  1 COMPANY BRANCH OFFICERR WAGE RATE WBONUS
/*
//
```

When the above program is submitted, the program check information printed on JESMSGLG is:

```
+FSYESPI1 - INTERRUPT OCCURRED
+FSYESPI2 - S07 Data exception
+FSYESPI2 - Problem Program: NONAME
+FSYESPI2 - (compiled with NOOPTIMIZE)
+FSYESPI2 - Failing PSW: 078D0E00 80031800
+FSYESPI2 - Addr: 00030240 Offset: 000015C0
+FSYESPI2 - <refer to FJSYABE for details>
```

The analysis log on FJSYABE would show the following:

```
FSYESPI2 V1.00 - ABEND Analysis Log
-----------------------------------------------------------------------

S07 Data exception
Problem Program: NONAME
(compiled with NOOPTIMIZE)
Failing PSW: 078D0E00 80031800
Addr: 00030240 Offset: 000015C0

EZTRV statement: 00019  <==

COBOL statement: *****
    ----+----1----+----2----+----3----+----4----+----5----+----6----+--
    000736 A01040-CREATE-REPORT.
                   .
                   .
    000739     COMPUTE WBONUS =
    000740         (WAGE * (RATE / 100))
    ...................................................................

Field values at the time of interrupt
-------------------------------------
02 WBONUS 5P  AT X'17BB1438'
 hex 000000000C
**

02 FILEIN-GRECORD  0CL80
--------------------------

  03 COMPANY 2C  AT X'17B0A2E0'
    char  10
    zone  FF
    numr  10
        0+1.

  03 BRANCH 3C  AT X'17B0A2E2'
    char  001
    zone  FFF
     numr  001
         0+1..

   03 OFFICERR 4C  AT X'17B0A2E5'
     char  BBBB
     zone  CCCC
     numr  2222
         0+1...

   03 WAGE 8C  AT X'17B0A2E9'
     char  05500000
     zone  FFFFFFFF
     numr  05500000
         0+1...!...

   03 RATE 5C  AT X'17B0A2F1'
     char
     zone  FFFF4
     numr  FFFF0
         0+1...!
  03 FILLER 58C  AT X'17B0A2F6'
**

02 FILEIN-RECORD-VEND 4B  AT X'17BB1764'
 hex 00000000
**

***********************************************************************
Trace of latest PERFORM statement(s):
***********************************************************************
==> Addr: 000317D2 Offset: 00001592

   EZTRV statement: 00015  <==

   COBOL statement: *****

   ----+----1----+----2----+----3----+----4----+----5----+----6----+--
   000724 A01020-JOB1-RUN.
                   .
                   .
   000726     PERFORM A01040-CREATE-REPORT
   000727        THRU A01050-CREATE-REPORT-EXIT.

   ...................................................................
***********************************************************************
```

Explanation:

```
EZTRV statement: 00019  <==
```

Is the SYSLIST1 statement number (of Easytrieve Plus program, section shown below) that caused the error.

```
     00017    CREATE-REPORT. PROC
     00018      MOVE X'FFFFFFFF' TO RATE
     00019      WBONUS = (WAGE * (RATE / 100))
     00020      PRINT RPT1
     00021    END-PROC
```

In some instances, the statement number may be off by one, especially if the statement that caused the error is followed by an Easytrieve Plus macro. Use your intuition to recognize the statement.

```
COBOL statement: *****
    ----+----1----+----2----+----3----+----4----+----5----+----6----+--
    000736 A01040-CREATE-REPORT.
                   .
                   .
    000739     COMPUTE WBONUS =
    000740         (WAGE * (RATE / 100))
```

The COBOL statements that caused the error are statements 739 and 740 located in the generated `A01040-CREATE-REPORT` paragraph name. As you can see, Easytrieve statement 19 was converted by the translator, to a COBOL COMPUTE statement as shown on statements 739-740.

```
Field values at the time of interrupt
-------------------------------------
```

This part shows the content of the fields at the time of interrupt. Fields that are a part of a group are printed along with all fields within the group. In this case, WBONUS is an elementary field, and WAGE and RATE fields are located within FILEIN-GRECORD.

```
**********************************************************************
Trace of latest PERFORM statement(s):
**********************************************************************
```

This part shows perform statements as they were executed before the interrupt. PERFORM statements are shown as first in first out, therefore, the last PERFORM trace will always match the paragraph name where the interrupt occurred.

```
**********************************************************************
PARTIAL/FULL DATA DIVISION MAP
**********************************************************************
```

Partial/Full DATA Division Map was not produced by this run because our DEBUG specified ESPI option only. If we coded ESPI-PART or ESPI-FULL, we would have seen an additional section (map of field values).

What to do when Interrupt handler is not active

Follow the steps below to locate the Easytrieve Plus instruction in error when the Interrupt Handler is not active:

- If you have a COBOL debugger, use the debugger to find COBOL statement in error.
- If you do not have a debugger:
  - Compile COBOL with PROCESS LIST,MAP option and "`* EASYTRAN: DEBUG (COBOL)`"
  - In the Log or Dump file, find the hex offset of PSW into your program as displayed by the Operating System.
  - Do a find on offset address in your COBOL compiler listing (expanded assembler listing).
  - COBOL source statement/line number can be found in the neighborhood before the offset.
  - Look at the COBOL listing identified by the line number in error.
  - You should be able to reason out the cause from the field name and/or logic in error. For example, bad data can be passed to reports logic, or a problem can occur in the activity section.

To locate the Easytrieve Plus statement, trace backward in the COBOL source. The first number in columns 1-6 points to the neighborhood of statement in Easytrieve Plus source that corresponds to the generated code.

Chapter 17. IDMS support

Migration Utility provides for an IDMS interface supported by Easytrieve Plus. The functions described in "IDMS statement syntax" on page 309 are compatible with Easytrieve Plus, except as otherwise noted.

There are four basic statements that facilitate the IDMS interface:

IDD statement
: Acts as an interface between Easytrieve Plus and the IDMS Data Dictionary.

FILE statement
: Defines the IDMS files. A FILE statement is required for each database to be processed. It defines the database records (segments) and the parent/child relationships.

RETRIEVE statement
: Used for the Automated Job inputs. The RETRIEVE statement is required when you want to sweep an area automatically by means of a JOB statement. It must be coded immediately after the JOB statement. The RETRIEVE statement is not supported by Migration Utility.

  Note: The file name used in the RETRIEVE statement must be the same file named on the JOB statement.

IDMS statement
: Performs the IDMS I/Os in a controlled environment. The IDMS statement can be used as needed in your program logic. Its syntax closely resembles the imbedded IDMS statements in COBOL.

An in-depth description of IDMS database concepts is beyond the scope of this document. However, you should be aware that:

- The IDMS database is defined by a system administrator using the IDMS supplied utilities.
- The administrator defines the databases required by application programs as well as all records, field definitions, and the dependencies on each other. These definitions are known as the Integrated Data Dictionary (IDD).
- The database access and the IDD for the data base are controlled by the schema and subschema hierarchy. Typically, one or more subschemas exist within a schema name.

Accessing an IDMS database in Easytrieve Plus

To access a database in an Easytrieve Plus program, an IDMS FILE, subschema name and the record definitions must be defined to your program. Creating an Easytrieve Plus macro for each of these definitions is recommended.

The I/O path for accessing your database must be coded using IDMS statements in the Activity Section. This type of I/O is referred to as *controlled processing*.

The RETRIEVE statement allows you to sweep an area. To do so, the IDMS file must be named on the JOB statement as an input file. The RETRIEVE statement must follow the JOB statement.

Translating and running IDMS programs

The installer of the Migration Utility IDMS support feature must create a schema table as described in “Subschema/Schema look-up table (SCHEMATB).” This table provides a means of identifying the schema associated with each subschema found in the Easytrieve Plus program. Note that COBOL requires a schema name, but Easytrieve Plus does not. A schema table must be created and maintained by the administrator before attempting to translate IDMS programs.

There are three jobs provided in the &sys1.SFSYJCLS Migration Utility library that can be tailored to translate Easytrieve Plus IDMS programs:

JCEZIDM0
Includes all steps to translate, compile, and link an Easytrieve Plus program.

JCEZIDM1
Runs the Migration Utility translator steps and saves the generated COBOL in a PDS. The resulting source is COBOL code with imbedded IDMS statements. The output of this job is used as input to JCEZIDM2.

JCEZIDM2
Runs the IDMSDMLC translator, COBOL compiler, and the Link step. The input to this job is the output of JCEZIDM1.

Use JCEZIDM0, JCEZIDM1, and JCEZIDM2 to translate and link Easytrieve Plus programs that contain IDMS statements.

To execute the linked program, you must create a separate job with all the required files as if the program was a standard IDMS COBOL program.

To run Link and Go jobs, clone the JCEZIDM0 job and add a step after the LINK step to execute the linked program.

Note: The one-step driver program, FSYTPA00, cannot translate IDMS programs. This support is to be provided in the future.

Subschema/Schema look-up table (SCHEMATB)

Migration Utility looks at the SCHEMATB table to obtain the schema name for each subschema accessed in the program. A sample table is supplied in the &sys1.SFSYEZTS library. The SCHEMATB table must be updated to reflect all subschemas available on the system.

This table must be maintained to reflect the current schemas and subschemas available to programmers. The layout is described at the beginning of the SCHEMATB table.

The table is pointed to by the SCHEMATB ddname in the JCL of the FSCCL1 step in the JCEZDIM0 and JCEZDIM1 jobs.

Unsupported IDMS features

These IDMS features are not supported:

- Multiple databases cannot be accessed from a single COBOL program, therefore Easytrieve Plus programs that access multiple SCHEMAS must be split into multiple jobs and run separately.

Supported IDMS statements

These IDMS features are supported:

- ACCEPT
- BIND
- COMMIT
- CONNECT
- DISCONNECT
- ERASE
- FINISH
- FIND
- GET
- IDD
- IF
- KEEP
- MODIFY
- OBTAIN
- READY
- RETRIEVE
- ROLLBACK
- STORE

IDMS statement syntax

This chapter describes each of the IDMS statements.

FILE statement

The FILE statement is coded in your program to identify the IDMS database to be used.

Syntax

Syntax

```
►►──FILE──file-name──IDMS──(subschema)──────────────────────────────►◄
```

file-name
: 1 to 8 character name used in your program to reference the database.

IDMS Declares the file as an IDMS database file.

subschema
: The subschema to be used.

Example:

```
FILE JOURNAL IDMS (SUBSCH01)
```

IDD statement

The Integrated Data Dictionary to be used by your program.

There are three IDD statement formats.

Format 1: IDD is coded as a standalone statement

```
►►──IDD──RECORD──record──LOCATION──location──────────────────────────►◄
```

Parameters

record 1 to 16 character RECORD name as defined in the IDD.

location
W generate definitions in working storage as W fields. S generate definitions in working storage as S fields.

Format 2: IDD is coded following the FILE IDMS statement

```
►►──IDD──RECORD──record──LOCATION──location──────────────────────────►◄
```

Parameters

record 1 to 16 character RECORD name as defined in the IDD.

location
W generate definitions in working storage as W fields. S generate definitions in working storage as S fields.

Format 3: IDD is coded as a standalone statement

```
►►──IDD──NAME──record──PROGRAM-NAME──program──VERSION──version──DBNAME──dbname──NODE──node──────►◄
```

Parameters

program
1 to 8 characters program name authorized to access subschema (not used by IMU).

version The program version (not used ny IMU).

dbname
IDD DB name (not used by IMU).

node The central version node name (not used by IMU).

RECORD statement

The database records to be used by your program are identified by the RECORD statement. You need to define only those records that are needed in your program; however, the parent segment must be provided for each. Partial paths are not supported. The record key parameter (KEY) is required for the RETRIEVE statement when using a Tickler File to sweep a database.

Syntax

Syntax

```
►►──RECORD──record──recsize──KEY──(──key──pos──keysize──)──────────────►◄
```

record	1 to 16 character name as defined in the IDD.
recsize	The record size (in bytes).
key	The key name.
pos	The key starting position in the record.
keysize	The key size in bytes.

Example:

```
RECORD DBSEGM00 120 KEY (SG00KEY 1 20)
SG00KEY     1   20 A
SG00-DATA   21 100 A

RECORD DBSEGM05 154 DBSEGM00
SG05KEY     1   12 A
SG05-DATA   13 142 A

RECORD DBSEGM10 140 DBSEGM05
SG10KEY     1   10 A
SG10-DATA   11 130 A
```

ACCEPT PROCEDURE statement

The ACCEPT PROCEDURE statement retrieves information from the Application Program Information Block associated with a database procedure.

Syntax

IDMS syntax

```
►►──ACCEPT──PROCEDURE──field1──TO──field2──────────────────────────────►◄
```

COBOL syntax

```
►►──ACCEPT──field2──FROM──field1───────────────────────────────────────►◄
```

field1	A field name or literal that identifies the name of a database procedure.
field2	A 256-byte field into which the procedure is copied.

ACCEPT STATISTICS statement

The ACCEPT STATISTICS statement retrieves the system run time statistics into an area in the program.

Syntax

IDMS syntax

```
►►──ACCEPT──STATISTICS──field──────────────────────────────────────────►◄
```

COBOL syntax

```
►►──ACCEPT──field──FROM──IDMS-STATISTICS───────────────────────────────►◄
```

field A 100-byte field in working storage.

ACCEPT DBKEY statement

There are two formats of the ACCEPT DBKEY statement.

Format 1

This format of the ACCEPT DBKEY statement retrieves a database key into working storage.

Syntax

IDMS syntax - Format 1

```
►►──ACCEPT──DBKEY──db-key──┬─RECORD─┬──field2──────────────────────────►◄
                           ├─AREA───┤
                           └─SET────┘
```

COBOL syntax - Format 1

```
►►──ACCEPT──db-key──FROM──field2──CURRENCY─────────────────────────────►◄
```

db-key A 4-byte binary field that receives the database key.

RECORD/SET/AREA
Specifies the object type.

field2 A 16-byte alphanumeric field name or a 16-byte literal that identifies the key currency. The default is the current record of the run unit.

Format 2

This format of the ACCEPT DBKEY statement retrieves the NEXT, PRIOR, or OWNER database key of the specified set into working storage.

IDMS syntax - Format 2

```
►►──ACCEPT──DBKEY──db-key──┬─NEXT──┬──setid──────────────────────────────►◄
                           ├─PRIOR─┤
                           └─OWNER─┘
```

COBOL syntax - Format 2

```
►►──ACCEPT──db-key──FROM──setid──┬─NEXT──┬──CURRENCY─────────────────────►◄
                                 ├─PRIOR─┤
                                 └─OWNER─┘
```

db-key A 4-byte binary field that receives the database key.

setid The name of the desired set or 16-byte literal that identifies the name of the desired set.

ACCEPT PAGE-INFO statement

The ACCEPT PAGE-INFO statement retrieves the current page information for the specified record.

Syntax

IDMS syntax

```
►►──ACCEPT──PAGE-INFO──field1──RECORD──field2────────────────────────────►◄
```

COBOL syntax

```
►►──ACCEPT──PAGE-INFO──field1──WITHIN──field2────────────────────────────►◄
```

field1 A 4-byte binary field that receives the page information.

field2 A record name or a 16-byte literal that identifies the record for which the page information is to be retrieved.

BIND FILE statement

The BIND FILE statement binds a file record to IDMS for access.

Syntax

IDMS syntax

```
►►──BIND──FILE──file──RECORD──record─────────────────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──record──────────────────────────────────────────────────────────────►◄
```

file The file the record belongs to.

record The record name to be bound with IDMS.

BIND statement (general format) statement

This format of the BIND statement is converted to COBOL BIND RUN-UNIT.

Syntax

IDMS syntax

```
►►──BIND──field1──────────────────────────────────────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──RUN-UNIT────────────────────────────────────────────────────────────►◄
```

field1 The subschema name to be processed by IDMS. The name can be an 8-byte alphanumeric field or an 8-byte literal.

BIND PROGRAM-NAME statement

The BIND PROGRAM-NAME statement identifies the program name to IDMS.

Syntax

IDMS syntax

```
►►──BIND──field1──PROGRAM-NAME──field2──────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──RUN-UNIT──FOR──field1──PROGRAM-NAME──field2───────────────►◄
```

field1 The subschema name to be processed by IDMS. The name can be an 8-byte alphanumeric field or an 8-byte literal.

field2 An 8-byte alphanumeric field or an 8-byte literal that identifies the application program to IDMS.

BIND DBNAME statement

The BIND DBNAME statement identifies the database name to be used by IDMS.

Syntax

IDMS syntax

```
►►──BIND──field1──DBNAME──field2────────────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──RUN-UNIT──FOR──field1──DBNAME──field2─────────────────────►◄
```

field1 The subschema name to be processed by IDMS. The name can be an 8-byte alphanumeric field or an 8-byte literal.

field2 An 8-byte alphanumeric field or an 8-byte literal that identifies the database name to be used by IDMS.

BIND NODE statement

The BIND NODE statement names the Central Version Node that hosts the IDMS activities.

Syntax

IDMS syntax

```
►►──BIND──field1──NODE──field2──────────────────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──RUN-UNIT──FOR──field1──NODE──field2───────────────────────────►◄
```

field1 The subschema name to be processed by IDMS. The name can be an 8-byte alphanumeric field or an 8-byte literal.

field2 An 8-byte alphanumeric field or an 8-byte literal that identifies the Central Version Node.

BIND DICTNAME statement

The BIND DICTNAME statement identifies the Dictionary Name of a Secondary Load Area.

Syntax

IDMS syntax

```
►►──BIND──field1──DICTNAME──field2──────────────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──RUN-UNIT──FOR──field1──DICTNAME──field2───────────────────────►◄
```

field1 The subschema name to be processed by IDMS. The name can be an 8-byte alphanumeric field or an 8-byte literal.

field2 An 8-byte alphanumeric field or an 8-byte literal that identifies the Dictionary Name.

BIND DICTNODE statement

The BIND DICTNODE statement identifies the Dictionary Node name of a Secondary Load Area.

Syntax

IDMS syntax

```
►►──BIND──field1──DICTNODE──field2────────────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──RUN-UNIT──FOR──field1──DICTNODE──field2─────────────────────►◄
```

field1 The subschema name to be processed by IDMS. The name can be an 8-byte alphanumeric field or an 8-byte literal.

field2 An 8-byte alphanumeric field or an 8-byte literal that identifies the Dictionary Node Name.

BIND PROCEDURE statement

The BIND PROCEDURE statement establishes communication between the application program and an IDMS procedure.

Syntax

IDMS syntax

```
►►──BIND──PROCEDURE──field1──TO──field2───────────────────────────────►◄
```

COBOL syntax

```
►►──BIND──PROCEDURE──field1──TO──field2───────────────────────────────►◄
```

field1 An 8-byte alphanumeric field, or an 8-byte literal that identifies the name of the user-written database procedure to be used.

field2 A 256-byte alphanumeric field.

COMMIT statement

The COMMIT statement creates a check point.

Syntax

IDMS syntax

```
           ┌─ALL─┐
►►──COMMIT─┴─────┴───────────────────────────────────────────────►◄
```

COBOL syntax

```
           ┌─ALL─┐
►►──COMMIT─┴─────┴───────────────────────────────────────────────►◄
```

CONNECT statement

The CONNECT statement connects a record to a set.

Syntax

IDMS syntax

```
►►──CONNECT──RECORD──record──SET──setname──────────────────────────►◄
```

COBOL syntax

```
►►──CONNECT──record──TO──setname───────────────────────────────────►◄
```

record A record name or a 16-byte literal that identifies the record to be connected.

setname
A set name or a 16-byte literal that identifies the set name to connect.

DISCONNECT statement

The DISCONNECT statement disconnects a record from a set.

Syntax

IDMS syntax

```
►►──DISCONNECT──RECORD──record──SET──setname─────────────────────────────►◄
```

COBOL syntax

```
►►──DISCONNECT──record──FROM──setname────────────────────────────────────►◄
```

record A record name or a 16-byte literal that identifies the record to be disconnected.

setname
A set name or a 16-byte literal that identifies the set name to be disconnected.

ERASE RECORD statement

The ERASE statement removes a record from the database. After ERASE, the record is detached from all sets and is made unavailable for further processing.

Syntax

IDMS syntax

```
                              ┌─MEMBERS───┐
►►──ERASE──RECORD──record─────┼───────────┼──────────────────────────────►◄
                              ├─PERMANENT─┤
                              ├─SELECTIVE─┤
                              └─ALL───────┘
```

COBOL syntax

```
                      ┌─MEMBERS───┐
►►──ERASE──record─────┼───────────┼──────────────────────────────────────►◄
                      ├─PERMANENT─┤
                      └─SELECTIVE─┘
```

record A record name or a 16-byte literal that identifies the record erased.

MEMBERS, PERMANENT, SELECTIVE, ALL
Specifies the type of erase.

FINISH statement

The FINISH statement disconnects the disconnecting.

Syntax

IDMS syntax

```
►►──FINISH───────────────────────────────────────────────────────────►◄
```

COBOL syntax

```
►►──FINISH──────────────────────────────────────────────────────────►◄
```

GET statement

The GET statement retrieves current data records from the database.

Syntax

IDMS syntax

```
►►──GET──RECORD──record──────────────────────────────────────────────►◄
```

COBOL syntax

```
►►──GET──record──────────────────────────────────────────────────────►◄
```

record A record name or a 16-byte literal that identifies the record to be retrieved.

MODIFY statement

The MODIFY statement updates a database record.

Syntax

IDMS syntax

```
►►──MODIFY──RECORD──record──────────────────────────────────────────►◄
```

COBOL syntax

```
►►──MODIFY──record──────────────────────────────────────────────────►◄
```

record A record name or a 16-byte literal that identifies the record to be updated.

FIND/OBTAIN statement

The FIND and OBTAIN statements have the same format. They are therefore described together here. The difference between the two statements is that FIND locates records but it does not retrieve them, whereas OBTAIN locates *and* retrieves the records.

There are six different FIND and OBTAIN formats available.

Format 1

Obtain or locate a record using the DBKEY.

Syntax

IDMS syntax - Format 1

```
►►──┬─FIND───┬──DBKEY──field1──┬─RECORD────┬──field2──┬─SHARE─────┬──────────►◄
    └─OBTAIN─┘                 └─PAGE-INFO─┘          ├─SHR───────┤
                                                      ├─EXCLUSIVE─┤
                                                      └─EXC───────┘
```

COBOL syntax - Format 1

```
►►──┬─FIND───┬──field2──┬─DB-KEY────┬──IS──field1─────────────────────────►
    └─OBTAIN─┘          └─PAGE-INFO─┘

►──┬────────────────────────────────────┬─────────────────────────────────►◄
   └─USAGE-MODE──IS──┬─SHARE─────┬──────┘
                     ├─SHR───────┤
                     ├─EXCLUSIVE─┤
                     └─EXC───────┘
```

field1 A 4-byte binary field or a numeric literal that identifies the database record key.

field2 For RECORD, *field2* is a record name or a 16-byte literal that identifies the record to be retrieved.

For PAGE-INFO, *field2* must be a 4-byte binary field or a 4-byte hex value.

SHARE, SHR, EXCLUSIVE, EXC
Specifies the type of lock to be placed on the record.

Format 2

Obtain or locate CURRENT record, set, or area.

IDMS syntax - Format 2

```
►►──┬─FIND───┬──CURRENT──┬─RECORD─┬──field1──┬─SHARE─────┬────────────────►◄
    └─OBTAIN─┘           ├─SET────┤          ├─SHR───────┤
                         └─AREA───┘          ├─EXCLUSIVE─┤
                                             └─EXC───────┘
```

COBOL syntax - Format 2

```
►►──┬─FIND───┬──CURRENT──field1──┬────────────────────────────────────┬────►◄
    └─OBTAIN─┘                   └─USAGE-MODE──IS──┬─SHARE─────┬──────┘
                                                   ├─SHR───────┤
                                                   ├─EXCLUSIVE─┤
                                                   └─EXC───────┘
```

RECORD/SET/AREA
Specifies the object type.

field1 A record, set, area name, or a 16-byte literal that identifies the record, set or an area to be retrieved or located.

SHARE, SHR, EXCLUSIVE, EXC
Specifies the type of lock to be placed on the record.

Format 3

Obtain or locate NEXT, PRIOR, FIRST, LAST, *n*th record occurrence in a set or an area.

IDMS syntax - Format 3

```
►►──┬─FIND───┬──┬─NEXT───┬─RECORD─field2──┬─SET──┬─field3──┬─SHARE─────┬───►◄
    └─OBTAIN─┘  ├─PRIOR──┤                └─AREA─┘         ├─SHR───────┤
                ├─FIRST──┤                                 ├─EXCLUSIVE─┤
                ├─LAST───┤                                 └─EXC───────┘
                ├─n──────┤
                └─field1─┘
```

COBOL syntax - Format 3

```
►►──┬─FIND───┬──┬─NEXT───┬─field2─WITHIN─field3───────────────────────────►
    └─OBTAIN─┘  ├─PRIOR──┤
                ├─FIRST──┤
                ├─LAST───┤
                ├─n──────┤
                └─field1─┘

►─┬───────────────────────────────────┬──────────────────────────────────►◄
  └─USAGE-MODE─IS──┬─SHARE─────┬──────┘
                   ├─SHR───────┤
                   ├─EXCLUSIVE─┤
                   └─EXC───────┘
```

field1 A 4-byte binary field, or a positive or negative numeric literal that identifies the record occurrence of a set or area to be retrieved or located.

field2 A record name or a 16-byte literal that identifies the record to be retrieved or located.

field3 A set or area name or a 16-byte literal that identifies the set or an area to be searched.

n A positive or negative integer that identifies the record occurrence of a set or area to be retrieved or located.

SHARE, SHR, EXCLUSIVE, EXC
Specifies the type of lock to be placed on the record.

Format 4

Obtain or locate sorted record from a set.

IDMS syntax - Format 4

```
                                                               ┌──────────┐
                                                               ▼          │
►►──┬─FIND───┬──CURRENT──RECORD──field1──SET──field2──USING──(───fieldn──┴─)────►
    └─OBTAIN─┘

►──┬─SHARE─────┬────────────────────────────────────────────────────────────────►◄
   ├─SHR───────┤
   ├─EXLUSIVE──┤
   └─EXC───────┘
```

COBOL syntax - Format 4

```
                                                          ┌──────────┐
                                                          ▼          │
►►──┬─FIND───┬──CURRENT──field1──WITHIN──field2──USING──(───fieldn──┴─)─────────►
    └─OBTAIN─┘

►──┬─────────────────────────────────┬──────────────────────────────────────────►◄
   └─USAGE-MODE──IS──┬─SHARE─────┬───┘
                     ├─SHR───────┤
                     ├─EXLUSIVE──┤
                     └─EXC───────┘
```

CURRENT
Controls the start of the search.

field1 A record name or a 16-byte literal that identifies the record to be retrieved or located.

field2 A set name or a 16-byte literal that identifies the set be searched.

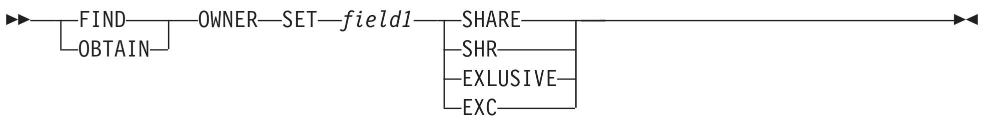
fieldn An alphanumeric field or a literal that contains the control statements.

SHARE, SHR, EXCLUSIVE, EXC
Specifies the type of lock to be placed on the record.

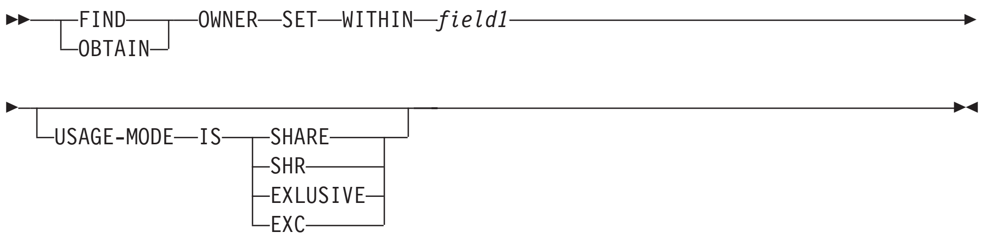
Format 5

Obtain or locate the owner record within a set.

IDMS syntax - Format 5



COBOL syntax - Format 5



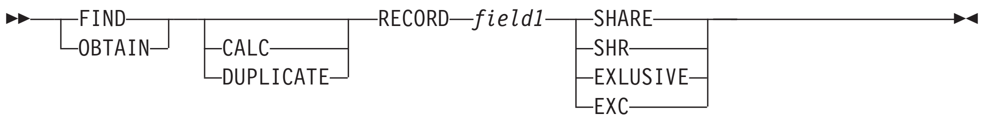
field1 A field name or a 16-byte literal that identifies the set to be retrieved or located.

SHARE, SHR, EXCLUSIVE, EXC
Specifies the type of lock to be placed on the record.

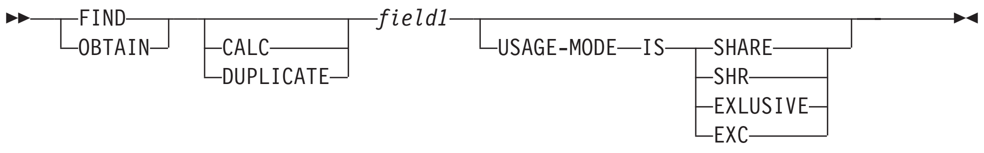
Format 6

Obtain or locate a record using the CALC key.

IDMS syntax - Format 6



COBOL syntax - Format 6



CALC Retrieves the first record.

DUPLICATE
Retrieves duplicate records (records after the first record).

field1 A record name or a 16-byte literal that identifies the record to be retrieved or located.

SHARE, SHR, EXCLUSIVE, EXC
Specifies the type of lock to be placed on the record.

IF statement

The IF statement tests the set status.

Syntax

IDMS syntax

```
►►──IF──SET──setnam──┬─MEMBER───┬──────────────────────────────────────►◄
                     ├─NOMEMBER─┤
                     ├─EMPTY────┤
                     └─NOEMPTY──┘
```

COBOL syntax

```
►►──IF──┬─┬─────┬─setnam──MEMBER──┬────────────────────────────────────►◄
        │ └─NOT─┘                 │
        └─setnam──IS──┬─────┬─EMPTY─┘
                      └─NOT─┘
```

setnam A 16-byte field or a 16-byte literal identifying the set to be tested.

MEMBER
Tests if current record is a member of the set.

NOMEMBER
Tests if current record is not a member of the set.

EMPTY
Tests if the specified set is empty (that is, no records exist).

NOEMPTY
Tests if the specified set contains any records.

KEEP statement

The KEEP statement puts a lock on a record, set, or area.

Syntax

IDMS syntax

```
                            ┌─SHARED────┐
►►──KEEP──┬─RECORD─┬──field1─┤           ├────────────────────────────►◄
          ├─SET────┤         └─EXCLUSIVE─┘
          └─AREA───┘
```

COBOL syntax

```
►►──KEEP──WITHIN──field1──┬─────────────────────────────────┬─────────►◄
                          │                  ┌─SHARED────┐   │
                          └─USAGE-MODE──IS───┴─EXCLUSIVE─┴───┘
```

RECORD/SET/AREA
Specifies the object type.

field1 A 16-byte field or a 16-byte literal that identifies the record, area, or set to be locked.

SHARED
Shared lock.

EXCLUSIVE
Exclusive lock.

READY statement

The READY statement makes an area available for processing.

Syntax

IDMS syntax

```
►►──READY──AREA──area──┬─RETRIEVAL─┬──┬─PROTECTED─┬───────────────────►◄
                       └─UPDATE────┘  └─EXCLUSIVE─┘
```

COBOL syntax

```
►►──READY──area──USAGE-MODE──IS──┬─RETRIEVAL─┬──┬─PROTECTED─┬─────────►◄
                                 └─UPDATE────┘  └─EXCLUSIVE─┘
```

area An area name or a 16-byte literal that identifies the area to be made available for processing.

RETRIEVE statement

The RETRIEVE statement encapsulates the logic that performs automatic database input for a complete database path. It must be coded immediately after the JOB statement.

The following parameters are available:

```
►►──RETRIEVE──idmsfile──┬────────────────────────┬──┬───────────────────────┬──────────►
                        └─PROGRAM──┬─field1──┬───┘  └─DBNAME──┬─field2──┬───┘
                                   └─'lit1'──┘                └─'lit2'──┘

►──┬──────────────────────┬──┬────────────────────────┬──┬────────────────────────┬────►
   └─NODE──┬─field3──┬────┘  └─DICTNAME──┬─field4──┬──┘  └─DICTNODE──┬─field5──┬──┘
           └─'lit3'──┘                   └─'lit4'──┘                 └─'lit5'──┘

►──┬──────────────────────────────────────────────────────────────────────────┬────────►
   │                                 ┌─────────────────────────┐              │
   │                                 ▼                         │              │
   └─KEYFILE──tickfile──KEYVALUE──(──┴─calcfield──EQ──tickfield─┴──)──┬────────┬┘
                                                                     ├─DUPS───┤
                                                                     └─NODUPS─┘

              ┌──────────────────────┐
              ▼                      │
►─SELECT──(───┤ SELECT options ├─────┴──)────────────────────────────────────────────►◄
```

SELECT options:

```
├──record──AREA──'lit6'──SET──'lit7'──INDEX──'lit8'───────────────────────────────────►

             ┌─────────┐
             ▼         │
►─USING──(───'lit9'────┴──)──┬───────────┬──┬───────────────┬─────────────────────────►
                             └─ID──'id'──┘  └─LIMIT──limit──┘

►──┬──────────────────────────┬───────────────────────────────────────────────────────┤
   └─WHILE──(condition)───────┘
```

Parameters

idmsfile
The IDMS file name as defined by the FILE statement.

field1 | *lit1*
The program name.

field2|*lit2*
: The IDD DB name.

field3|*lit3*
: The central version node name.

field4|*lit4*
: The IDD Dictionary name.

field5|*lit5*
: The IDD Dictionary node name.

tickfile The tickler file ddname as defined by the FILE statement. This must be a sequential file that contains the keys to be retrieved from IDMS data base. The file is read sequentially. The key values are used as search arguments for accessing the data base.

The end of the tickler file marks the end of automatic input and also the end of the RETRIEVE logic.

calcfield
: A calculated key value.

tickfield
: A tickler file field.

DUPS|**NODUPS**
: Duplicate condition handling:

DUPS Duplicate keys are allowed.

NODUPS
: Keys must be unique.

record Segment name to retrieve.

lit6 The name of the area to be used.

lit7 The set name to be used.

lit8 The index name to be used.

lit9 The index usage fields to be used.

id The segment identification code.

id is a two digit literal for identifying the retrieved paths (segments). The id is moved to the system-defined field PATH-ID for the lowest accessed segment in the path. The literal can be any two characters allowed by the system. When accessing root segments using a tickler file, PATH-ID is automatically set to 'NF' (not found) by the RETRIEVE logic.

limit The segment limit identifier.

limit is the number of segments to retrieve. This limit applies to each segment in the path. If LIMIT is not coded, all segments are retrieved in each path.

condition
: *condition* is a conditional expression for selecting on specific segment field values. The syntax for this expression is the same as that of the DO WHILE statement described in "DO and END-DO statements" on page 90. Records are accepted for input only when the WHILE expression is true, otherwise they are bypassed.

ROLLBACK statement

The ROLLBACK statement recovers previous updates.

Syntax

IDMS syntax

```
►►──ROLLBACK──┬──────────┬──────────────────────────────────────────────►◄
              └─CONTINUE─┘
```

COBOL syntax

```
►►──ROLLBACK──┬──────────┬──────────────────────────────────────────────►◄
              └─CONTINUE─┘
```

CONTINUE
Identifies the action to be taken after the recovery. The default terminates the run unit.

STORE statement

The STORE statement adds a new record to the database.

Syntax

IDMS syntax

```
►►──STORE──RECORD──record───────────────────────────────────────────────►◄
```

COBOL syntax

```
►►──STORE──record───────────────────────────────────────────────────────►◄
```

record A record name or a 16-byte literal that identifies the record to be added.

Status codes

Migration Utility includes a copy of the IDMSCOM communication area that contains numerous fields and an indicator that can be tested after each IDMS call.

IDMSSTATUS is a 4-byte alphanumeric field that contains the completion code. IDMSSTATUS must be tested for a valid return after each operation.

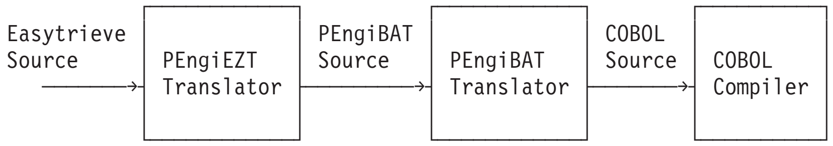
IDMSSTATUS can have these values:

0000	Good return.
0307	End of set.
0326	Record not found.
3101, 3201, 3401, 3901	Dead lock.
3202, 3402	No storage available.
4303	Area ID is unknown.
4404	Queue ID is unknown.
4305, 4405	Record not found.
3908	Resource not available.
3909	Resource is available.
3210	New storage.
3711	Maximum tasks.
4317	Record replaced.
4319 4419 4519 4719	Truncated data.
4525	Attention.
4604	Second Start page.
4664	Detail not found.
4668	More update details Maximum space reached.
4672	No details.
4676	First page sent.
4680	Page Ready.
4684	Start page missing.
4743	Operator cancel.

Chapter 18. 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 335. These messages relate to user and 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 392. From the description of the message, you will have to determine whether you caused the message with 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 358 and "Migration Utility function generated messages" on page 381. 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 because 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 typically 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 392, 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, messages are preceded by the word **MNOTE**. These messages are described in “Migration Utility macro generated messages” on page 358 and “Migration Utility function generated messages” on page 381, 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 embedded 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 392 for convenience. Because the PEngiCCL preprocessor is a macro interpreter, most messages are related to the interpretation of the macro directives embedded in the PEngiCCL macros. Such messages are encountered during the development of the new PEngiCCL macros.
2. It is possible to get PEngiCCL preprocessor messages due to the previously detected errors or MNOTES. You should first resolve all obvious errors by process of elimination. PEngiCCL preprocessor errors are typically 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, contact the IBM service center.
3. The most frequent errors are caused by a 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, check the following:
   - Every macro instruction must be terminated by a ":".
   - 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.
   - Quoted strings must contain paired quotes. If you need a quote as a data item, code double quotes.
   - 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 11, "Installing Migration Utility," on page 203).

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 the 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 11, “Installing Migration Utility,” on page 203.

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 field 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 40.

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=COBOL390` for compatibility with COBOL/390 and later versions, or `COBOL=COBOLII` for COBOL II.

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 11, "Installing Migration Utility," on page 203).

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 11, "Installing Migration Utility," on page 203).

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 11, "Installing Migration Utility," on page 203).

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 11, “Installing Migration Utility,” on page 203).

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 11, “Installing Migration Utility,” on page 203).

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 11, “Installing Migration Utility,” on page 203.

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 11, “Installing Migration Utility,” on page 203).

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 11, “Installing Migration Utility,” on page 203).

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 11, “Installing Migration Utility,” on page 203).

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 11, “Installing Migration Utility,” on page 203).

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 11, “Installing Migration Utility,” on page 203).

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 CONFLICTING FILE REFERENCE

Explanation: Redefining object (file) does not match the redefined field object.

User response: Code the correct file qualifier for the redefined object (field).

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 rearranging 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: Rearrange 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 the following format, 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: Rename 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 89 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 89 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 6, “SQL/DB2 support,” on page 125 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 6, “SQL/DB2 support,” on page 125 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, or use the SSMODE=GEN option to eliminate this message.

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 field 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 255.

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.

EZT000-BJ "&WSEG" :RECURSIVE SEGMENT IN SELECT STATEMENT

Explanation: &WSEG was previously specified in the same RETRIEVE statement.

User response: Remove the duplicate name.

EZT000-BK "&WSEG" :SELECTION OF PARENT "&CURPATH" IS REQUIRED

Explanation: &WSEG segment was specified in the RETRIEVE statement but its parent (root) &CURPATH segment was not.

User response: Child segments cannot be accessed without the parent segment. Add the parent segment to the RETRIEVE to fulfill the requirements.

EZT000-BL REQUIRED "SELECT" IS MISSING FROM RETRIEVE

Explanation: A RETRIEVE was coded without SELECT.

User response: Add SELECT statements as per RETRIEVE statement syntax rules.

EZT000-BM SSA AND TICKLER FILE USAGE CONFLICT

Explanation: You have specified a KEYFILE and SSA for the root segment in RETRIEVE.

User response: The KEYFILE and SSA are mutually exclusive. Only one option can be used at a time.

EZT000-C1 &FILE: EXIT IS NOT ALLOWED

Explanation: "EXIT" option was specified for a file organization that does not support I/O exit.

User response: Remove the EXIT statement or change file organization to comply with the file EXIT rules.

EZT000-C2 &FILE: MODIFY OPTION REQUIRES WORKAREA

Explanation: "MODIFY" option was specified but there was no work area coded.

User response: CODE WORKAREA NNNN following the MODIFY statement. Refer to FILE statement coding rules.

EZT000-C3 "DRILL MENU" STATEMENT IS MISSING

Explanation: REPORT <DOC> was coded but no DRILL MENU was supplied.

User response: See Chapter 9, "Creating HTML and spreadsheet files," on page 155 for coding conventions.

EZT000-C4 "DRILL DOWN" STATEMENT IS MISSING

Explanation: REPORT <DOC> was coded but no DRILL DOWN was supplied.

User response: See Chapter 9, "Creating HTML and spreadsheet files," on page 155 for coding conventions.

EZT000-C5 "&field" DOES NOT MATCH DRILL DOWN FIELD

Explanation: The CONTROL field does not match the field named on the DRILL DOWN statement for this report.

User response: Correct the field name to match that of the DRILL DOWN statement.

EZT000-C6 "PARM BIND(DYNAMIC)" CONFLICTS WITH SQLMODE=BIND. CAF FACILITY IS REQUIRED FOR DYNAMIC MODE. ADJUST SQLMODE= TO USE A CAF PROGRAM NAME.

Explanation: BIND(DYNAMIC) requires SQLMODE=FSYDB250.

User response: Correct SQLMODE= parameter. If SQLMODE=BIND is coded as a default in EZPARAMS, and you wish to use Dynamic SQL mode, add SQLMODE=FSYDB250 to your Easytrieve Plus program using EASYTRAN syntax.

EZT000-C7 "&GSEP" SEPERATOR MUST BE ENCLOSED IN QUOTES

Explanation: The SEP=(&GSEP) syntax error.

User response: Enclose "&GSEP" in quotes. For example: `SEP=(',')`

EZT000-C8 DOCTYPE=&QDOCTYPE REQUIRES "CONTROL DETAIL"

Explanation: The report was declared as a DETAIL report on the DRILL DOWN statement.

User response: Code CONTROL DETAIL for this report.

EZT000-C9 MATCH/MERGE NUMBER OF INPUT FILES EXCEEDS TWO (2)

Explanation: The type of synchronized JOB requires maximum of two files. This is probably an Easytrieve Classic issue.

User response: Code the correct number of files on the JOB statement.

EZT000-CA &FILE: MODIFY OPTION FOR PRINTER NOT SUPPORTED

Explanation: MODIFY was coded on the PRINTER file definition.

User response: Migration Utility does not support the MODIFY option for printer files. Find a different solution.

EZT000-CB &FIELD: BINARY AND PACKED FIELDS CANNOT BE CODED WITH LENGTH MODIFIER. RE-DEFINE FIELD AS ALPHA TYPE.

Explanation: Length modifier was coded for binary or packed decimal field.

User response: Redefine the field as an alpha field and use modifier on the new field.

EZT000-CC &FIELD: BL1/BL3 AND PU FIELDS CANNOT BE USED IN DO WHILE/UNTIL TEST

Explanation: The &FIELD cannot be used in a DO WHILE/UNTIL test because of its special format, as the outcome would be unpredictable.

User response: Change logic to use a more appropriate field type, such as BL2, BL4, or packed decimal.

EZT000-CD &record RECORD IS NOT IN ANY IDDLIBS= LIBRARY

Explanation: The record name &record cannot be located in the supplied IDD library.

User response: Check the spelling of your record name and include in IDDLIBS the correct library that contains the &record definition.

EZT000-CE "LOCATION &IDDLOC" CANNOT BE RESOLVED

Explanation: The specified location is not W or S.

User response: Code the correct location.

EZT000-CF &RPTNAME: CONFLICTING USAGE OF &RPTDDNM

Explanation: There is a conflict in printer DD name usage. A report printer exit was specified using a DDname previously used in the program.

User response: Choose a different DD name that does not conflict with any other names in the program.

EZT000-CG &WORD: ILLEGAL OR MISPLACED PARAMETER

Explanation: An unknown or misplaced parameter was found while decoding the IDD statement.

User response: Correct the parameter.

EZT000-CH &SYSTOKEN(1): SUBSCHEMA NOT IN SCHEMATB

Explanation: IDMS file subschema name is not in SCHEMATB.

User response: Correct the parameter.

EZT000-CI CANNOT RESOLVE "&WIXORG" INDEX

Explanation: This is an internal IMU error.

User response: Call the IBM Support Center.

EZT000-CJ &GCTLBRKS CONTROL BREAKS EXCEEDED. INCREASE BREAKS=nn IN THE EZPARAMS/EASYTRAN

Explanation: The number of control breaks specified by BREAKS=nn is exceeded.

User response: Increase BREAKS=*nn* value to accommodate your requirement. The *nn* can be maximum of 64.

EZTRV-01 "&SYSPARM" - INVALID SYSPARM STATEMENT

Explanation: SYSPARM coded on the FSYTPA00 EXEC statement is not "EASYTRAN:".

User response: Correct the erroneous statement.

EZTRV-02 &NAME - MEMBER NOT LOCATED

Explanation: Program name on the SYSPARM cannot be located.

User response: Check input member name / SYSIN for proper input.

EZTRV-03 INCOMPLETE STATEMENT

Explanation: The statement is incomplete as written. More arguments are expected.

User response: Correct the erroneous statement.

EZTRV-04 &WORD: TOO MANY ARGUMENTS

Explanation: The statement contains too many arguments near word &word.

User response: Correct the erroneous statement.

EZTRV-05 THE USE OF "LENGTH" IS INVALID FOR IMS

Explanation: Length was coded on IMS file statement.

User response: Remove the length.

EZTRV-06 &FIELD :DUPLICATE FIELD NAME

Explanation: N/A

User response: N/A

EZTRV-07 &WP1 :FIELD QUEUE OF N'&QFIELD EXCEEDED. INCREASE FIELDS= ON EZTRVBAS MACRO.

Explanation: The fields queue capacity is exceeded.

User response: Increase FIELDS=*nn* in EZTRVPRM default table.

EZTRV-08 &INCLUDE :INCLUDE QUEUE OF nn EXCEEDED. INCREASE INCLNST= IN EZTRVBAS MACRO.

Explanation: The number of *nn* nested INCLUDES exceeded.

User response: Increase INCLNST=*nn* in EZTRVPRM default table.

EZTRV-09 &FILE :UNDEFINE FILE NAME

Explanation: Referenced file is undefined.

User response: Code a defined file.

EZTRV-10 &WORD :UNKNOWN PARAMETER

Explanation: The &word cannot be resolved.

User response: Correct the erroneous statement.

EZTRV-11 &WORD :PARAMETER IS NOT NUMERIC

Explanation: The &WORD parameter contains non-numeric characters.

User response: Correct the erroneous statement.

EZTRV-12 &FILE :CONFLICTS WITH "NOINPUT" USAGE

Explanation: MERGE or MATCH was coded for a file define as NOINPUT.

User response: Correct the erroneous statement.

EZTRV-13 &SEGNENT :UNKNOWN SEGMENT NAME

Explanation: A SELECT was coded for undefined segment name.

User response: Correct the erroneous statement.

EZTRV-14 CONFLICT WITH CONTROL/ SUMMARIZE/LABELS

Explanation: More than one summary request was detected.

User response: Correct the duplicate statement.

EZTRV-15 DUPLICATE SORT/PRESORT NOT ALLOWED

Explanation: More than one (1) SORT/PRESORT was detected.

User response: Remove extraneous SORT/PRESORT statement.

EZTRV-16 &FIELD :NOT DEFINED IN &QFILE(2)

Explanation: The field is not defined in the first file.

User response: Correct the field name.

EZTRV-17 &FIELD :NOT DEFINED IN &QFILE(3)

Explanation: The field is not defined in the second file.

User response: Correct the field name.

EZTRV-18 MATCH/MERGE REQUIRES TWO INPUT FILES

Explanation: MATCH or MERGE was specified but there is only one (1) defined file.

User response: Define the second file.

EZTRV-19 STATEMENT CANNOT BE RESOLVED

Explanation: The statement cannot be resolved as written.

User response: Correct the erroneous statement.

EZTRV-20 FIELD NAME IS REQUIRED BEFORE "CHANGE"

Explanation: CHANGE in IF but no field name was found.

User response: Correct the erroneous statement.

EZTRV-21 "&FILE" FILE NAME CONFLICTS A FIELD NAME

Explanation: There is a defined field name that conflicts with &file name.

User response: Change the file name to a different name that does not conflict with any field names. Note that Easytrieve Classic allows file names to be the same as field names. Migration Utility requires unique names.

EZTRV-22 "&FILE" NOT DECLARED AS A TABLE

Explanation: The statement implies a table file usage but the file was not declared as a TABLE file.

User response: Change file definition to be a TABLE file.

EZTRV-23 PROGRAM NAME "&NAME" CONFLICTS WITH FILE/FIELD NAME

Explanation: The Easytrieve Classic program name is in conflict with a field name.

User response: Change the program name to a unique name.

EZTRV-24 &WORD :INVALID NUMERICS

Explanation: The &word contains non-numeric characters or its type cannot be used in this context.

User response: Correct the erroneous statement.

EZTRV-25 &WORD :INVALID SEGMENT NAME

Explanation: Segment name exceeded 8 characters.

User response: Reduce the segment name to 8 characters in length.

EZTRV-26 &WORD :INVALID KEY NAME

Explanation: Key name exceeds 8 characters.

User response: Reduce the key name to 8 characters in length.

EZTRV-27 &DLIREL :INVALID RELATIONAL OPERATOR

Explanation: Invalid relational operator on DLI statement.

User response: Code the correct relational operator.

EZTRV-28 &WORD :DUPLICATE PARAMETER

Explanation: Duplicate parameter detected on DLI statement.

User response: Remove the duplicate parameter.

EZTRV-29 &WORD :IMS FILE CANNOT BE USED IN GET/PUT/PRESORT

Explanation: IMS file was improperly used.

User response: Correct the erroneous statement.

EZTRV-30 &WORD :FIELD NAME IS NOT UNIQUE

Explanation: Field referenced for NOINPUT file.

User response: Correct the erroneous statement.

EZTRV-31 "SELECT" IS REQUIRED FOR IMS FILE

Explanation: Improper access of IMS file

User response: Correct the erroneous statement.

EZTRV-32 WRITE STATEMENT IS SUPERFLUOS DUE TO SORT/PRESORT

Explanation: WRITE for FILEB was used in combination with SORT/PRESORT.

User response: Correct the erroneous statement.

EZTRV-33 &WP3 :ILLEGAL FIELD LENGTH

Explanation: Field name is invalid.

User response: Correct the erroneous name.

EZTRV-34 :PARSING ERROR IN YREADWORD ROUTINE

Explanation: Too many unresolved commas in input statement.

User response: Commas are used as separators. Add at least one space after each comma to simplify the string.

EZTRV-35 :CONFLICTING FILE USAGE "FILEB" VS SUMMARIZE.

Explanation: Logic calls for the use of FILEB but FILEB is needed for the SUMMARIZE statement.

User response: Correct the erroneous logic.

EZTRV-36 &WORD :&&ACROSS-&&LINES IS INVALID

Explanation: LABELS statement is invalid as written.

User response: Correct the erroneous statement.

EZTRV-37 &WORD :EXPECTED LINE STATEMENT NOT FOUND

Explanation: Expected LINE statement not found.

User response: Code LINE statement as required.

EZTRV-38 &WORD :LINE NUMBER IS INVALID

Explanation: N/A

User response: N/A

EZTRV-39 LABELS CONFLICTS WITH COMPUTE STATEMENT

Explanation: LABELS and COMPUTE in use

User response: Use one or the other but not both.

EZTRV-40 CURRENCY &CHR IS LLEGAL AS WRITTEN

Explanation: Currency sign exceeds one character.

User response: Correct erroneous statement.

EZTRV-41 DECIMAL &DEc IS LLEGAL AS WRITTEN

Explanation: DECIMAL is not COMMA or PERIOD.

User response: Correct the erroneous statement.

EZTRV-42 STATEMENT IS OUT OF SEQUENCE

Explanation: SELECT is not the first statement of the main logic.

User response: Place SELECT at the beginning of the main logic.

EZTRV-43 STATEMENT CONFLICTS WITH MATCH/MERGE

Explanation: SORT/PRESORT used in MATCH/MERGE logic.

User response: Remove the extraneous statement.

EZTRV-44 &FILE :NOT IN UPDATE MODE

Explanation: UPDATE used for a non VSAM file, or VSAM file was not coded for UPDATE.

User response: Correct the erroneous statement.

EZTRV-45 "&TYPE" :INVALID FIELD TYPE

Explanation: Field type is not A, P, U, N or B, or type cannot be recognized.

User response: Correct the erroneous statement.

EZTRV-46 &WORD :NOT SUPPORTED IN "MODE A"

Explanation: The statement is not supported in MODE A.

User response: Correct the erroneous statement.

EZTRV-47 "SORTED" VARIABLE IN USE BUT NO SORT WAS FOUND

Explanation: SORTED variable used in IF but no SORT is in use.

User response: Correct the erroneous statement.

EZTRV-48 &FIELD :MOVE WOULD EXCEED 80 BYTES

Explanation: The &field on the PUNCH statement would span beyond column 80.

User response: The maximum punch record size is 80 bytes. Reduce the fields being punched to fit 80 bytes.

EZTRV-49 &FILE :FILE NAME EXCEEDS 8 CHARACTERS

Explanation: The file name &file exceeds 8 bytes.

User response: Reduce the file name to 8 bytes or less.

EZTRV-50 &FIELD :FIELD IS NOT A QUANTITY

Explanation: The &field is not defined as a numeric quantitative field.

User response: Correct the erroneous field.

EZTRV-51 SUMMARIZE STATEMENT IS INCOMPLETE

Explanation: The statement does not contain any control fields and/or quantitative fields.

User response: Add the necessary fields.

EZTRV-53 NUMBER OF COMPUTE STATEMENTS EXCEEDS *nnn*

Explanation: The number of COMPUTE statements exceeds the maximum allowed by Migration Utility.

User response: Resort to other means of performing the same calculation.

EZTRV-54 &FIELD: NUMBER OF SUMs EXCEEDS *nnn*

Explanation: The number of SUM fields exceeds the maximum allowed by Migration Utility.

User response: Reduce the number of fields to be summed.

EZTRV-55 &FIELD: CANNOT BE USED IN ARITHMETIC

Explanation: An alphanumeric field of length greater than 18 bytes used in arithmetic.

User response: The maximum length of numeric fields is 18 bytes. Reduce the field size to 18 bytes.

EZTRV-56 &endcol:VALUE IS OUT OF VALID RANGE (32-80)

Explanation: &endcol value is invalid..

User response: Code a valid end column 32 thru 80.

TRVBAS-00 CURRENCY=&CURRENCY IS ILLEGAL AS WRITTEN

Explanation: The &currency is longer than one (1) character.

User response: Correct the erroneous character.

TRVBAS-00 DECIMAL=&DECIMAL IS ILLEGAL AS WRITTEN

Explanation: The &decimal is not COMMA or PERIOD.

User response: Correct the erroneous character.

Dynamic SQL Translator macro generated messages

DYNCPY-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 1-30 characters long and must follow COBOL field-naming conventions.

DYNCPY-02 XXXXX IS ILLEGAL LEVEL NUMBER

Explanation: An invalid COBOL field level number has been detected.

User response: Valid level numbers are 01-99.

DYNCPY-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.

DYNCPY-04 ILLEGAL COBOL PICTURE

Explanation: An invalid COBOL field picture has been detected.

User response: Code a valid COBOL field picture.

DYNCPY-05 COPY IS IMPROPER AS WRITTEN

Explanation: An improperly coded COPY was detected.

User response: Refer to COPY statement format in the COBOL reference manual. A copybook name can be 1-8 characters long.

DYNCPY-06 TOO MANY REPLACING IDENTIFIERS

Explanation: More than the maximum number of 256 ordered REPLACING statements was detected.

User response: Reduce the number of REPLACING pairs to less than 256.

DYNCPY-07 FIELD DEFINITION IS INCOMPLETE

Explanation: The definition of the last field on the copybook is not complete.

User response: Correct the problem.

DYNCPY-08 XXXXX FIELD IS UNDEFINED OR LEVELS ARE INCONSISTENT IN REDEFINES EXPRESSION

Explanation: The redefined field is undefined or the level numbers of the redefined and redefining fields are inconsistent.

User response: Correct the problem.

DYNCPY-09 Text1 Text2 Text3; INVALID REPLACING OPTION

Explanation: An improperly coded COPY REPLACING was detected.

User response: Refer to COPY statement format in the COBOL reference manual.

DYNCPY-10 Text1 Text2 Text3 Text4; RECURSIVE USE OF PSEUDO TEXT

Explanation: Multiple pairs of pseudo text has been detected.

User response: DYNAMSQL translator allows only one ordered pair of pseudo text replacement. Delete extra statements.

DYNCPY-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.

DYNCPY-13 RECURSIVE 01 LEVEL INSIDE COPY

Explanation: Multiple 01 levels were detected in the copybook.

User response: Remove extraneous 01 levels.

DYNCPY-14 LENGTH OF REDEFINED FIELD XXXXX IS INCONSISTENT

Explanation: The length of the redefined field is not equal to the length of the redefining field.

User response: Correct the problem.

DYNCPY-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.

DYNCPY-16 "RENAMES" IS NOT SUPPORTED, USE REDEFINES

Explanation: RENAMES was detected in the copybook.

User response: PEngi does not support RENAMES. The alternative is to use REDEFINES.

DYNCPY-20 &WFIELD: MAX OF N'&GFLDSQUE FIELDS EXCEED. INCREASE FIELDS=NNN ON DYNAMBAS MACRO

Explanation: COBOL fields queue has been exceeded.

User response: Increase FIELDS=NNN on DYNAMBAS macro located in SQLPARMS member.

DYNCPY-21 XXXX: OCCURS VALUE IS NOT NUMERIC

Explanation: The OCCURS value is not numeric.

User response: Code a numeric value for OCCURS.

DYNFQU-02 XXXXX IS ILLEGAL LEVEL NUMBER

Explanation: An invalid COBOL field level number has been detected.

User response: Valid level numbers are 01-99.

DYNFQU-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.

DYNFQU-04 ILLEGAL COBOL PICTURE

Explanation: An invalid COBOL field picture has been detected.

User response: Code a valid COBOL field picture.

DYNFQU-07 FIELD DEFINITION IS INCOMPLETE

Explanation: The definition of the last field in the group is not complete.

User response: Correct the problem.

DYNFQU-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.

DYNFQU-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.

DYNFQU-13 RECURSIVE 01 LEVEL INSIDE COPY

Explanation: Improper 01 level was detected.

User response: Remove extraneous 01 levels.

DYNFQU-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.

DYNFQU-16 "RENAMES" IS NOT SUPPORTED, USE REDEFINES

Explanation: RENAMES was detected in the copybook.

User response: PEngi does not support RENAMES. The alternative is to use REDEFINES.

DYNFQU-20 &WFIELD: MAX OF N'&GFLDSQUE FIELDS EXCEED. INCREASE FIELDS=NNN ON DYNAMBAS MACRO

Explanation: COBOL fields queue has been exceeded.

User response: Increase FIELDS=NNN on DYNAMBAS macro located in SQLPARMS member.

DYNCPY-21 XXXX: OCCURS VALUE IS NOT NUMERIC

Explanation: The OCCURS value is not numeric.

User response: Code a numeric value for OCCURS.

DYNFQU-22 INCOMPLETE COPY STATEMENT

Explanation: COPY was detected without any arguments.

User response: Correct the erroneous COPY.

DYNFQU-22 "END-EXEC" statement is missing

Explanation: “EXEC SQL” was not ended with “END-EXEC”.

User response: Add “END-EXEC” to the statement.

DYNSQL-01 "&SYSPARM" - INVALID SYSPARM STATEMENT

Explanation: SYSPARM on EXEC statement is not (DYNAMSQL:&pgmname,. .)

User response: Verify your program name. Program name can be 1-8 characters long.

DYNSQL-02 &name - MEMBER NOT LOCATED

Explanation: SYSIN cannot be opened.

User response: Make sure that you are pointing to a valid program in a PDS or a QSAM 80-byte record file.

DYNSQL-03 INCOMPLETE "EXEC SQL" STATEMENT

Explanation: END-EXEC statement is missing, or premature EOF.

User response: Add END-EXEC as needed.

DYNSQL-04 "&field" NOT PRECEDED BY A ":"

Explanation: Host variable is expected.

User response: Verify SQL statements. Code a ":" before host variables.

DYNSQL-05 "&word" STATEMENT IS MISSING

Explanation: The "&word" statement is expected but not located.

User response: Code the required statement.

DYNSQL-05 "END-EXEC" STATEMENT IS MISSING

Explanation: "EXEC SQL" was not ended with "END-EXEC".

User response: Add "END-EXEC" to the statement.

DYNSQL-06 "&curs" MAXIMUM OF N'&GCURSORS EXCEEDED. INCREASE OBJECTS=NN IN SQLPARMS TABLE

Explanation: Cursors queue has been exceeded.

User response: Increase OBJECTS=NNN on DYNAMBAS macro located in SQLPARMS member.

DYNSQL-07 "&curs" CURSOR HAS NOT BEEN DECLARED

Explanation: The "&curs" cursor is not declared.

User response: Declare the cursor in question.

DYNSQL-08 "&field" UNDEFINED HOST VARIABLE

Explanation: The host variable was not defined as a field name.

User response: Define the required field.

DYNSQL-09 "&field: &type" UNABLE TO RESOLVE FIELD TYPE

Explanation: The field type cannot be resolved.

User response: Report this problem to the IBM Support Center.

DYNSQL-10 "SELECT" WITHOUT DECLARED CURSOR IS NOT SUPPORTED

Explanation: Unsupported SELECT statement.

User response: Resort to other available methods.

DYNSQL-11 "&word" IS NOT SUPPORTED IN DYNAMIC MODE

Explanation: The "&word" is not supported by Dynamic SQL.

User response: Resort to other available methods.

DYNSQL-12 "&cursor" NAME IS TOO LONG. MAXIMUM IS 18 CHARACTERS.

Explanation: The cursor name exceeds the allowable DB2 cursor size.

User response: Reduce the cursor name to 18 characters or less.

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. 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 the OBJECTS= keyword of COBOLBAS/CICSBASE macro. If necessary, 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 accordingly.

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: Contact IBM Support Center.

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.

GCCL-51 &REPORT DRILL DOWN OBJECTS ARE MISSING

Explanation: DRILL parameter request was passed to GENERATQ macro, but there are no valid DRILL DOWN reports queued by the DEFINE macro.

User response: Report this problem to the IBM Support Center.

| **GCCL-52 CONFLICTING USAGE OF &DDNAM**

| **Explanation:** There is a conflict in printer DD name
| usage. A report printer exit was specified using a
| DDname previously used in the program.

| **User response:** Choos a different DD name that does
| not conflict with any other names in the program.

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: Report this problem to the IBM Support Center.

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 Migration Utility 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 &ALIAS : IMPROPER ALIAS NAME

Explanation: Alias 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 &ALIAS IS NOT DEFINED IN &MEMBER

Explanation: Improper use of Alias 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 Migration Utility 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 Migration Utility 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 Migration Utility 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 Migration Utility 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 Migration Utility 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 Migration Utility 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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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 KICKS=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.

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.

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.

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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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 Easytrieve Macros in Chapter 5 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 reached. 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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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 GSECT 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 embedded 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, this statement is syntactically incorrect:

```
IF SEL_OBJECT (OBJECT OPTION) = ZERO
```

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.

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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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.

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 (typically 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.

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 “Translator CCL1 preprocessor options” on page 200). Also see note 3.

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. 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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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 systems increase the REGION size on the EXEC statement. On VM/CMS systems 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 systems increase the REGION size on the EXEC statement. On VM/CMS systems increase the virtual storage of your CMS machine.

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.

Parallel testing utility messages

FSYATTCH-01,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator (installer) for the correct Migration Utility load libraries.

FSYESPI1 ESPIE TRAP ESTABLISHED

Explanation: Advisory message. Program check interrupt trap is established.

User response: N/A

FSYESPI1 INTERRUPT OCCURRED

Explanation: Advisory message. Program check interrupt has occurred.

User response: N/A

FSYESPI1-01,012 CANNOT LOAD "FSYESPI2" PROGRAM

FSYESPI1-02,012 *ANALYSIS OF INTERRUPT IGNORED

Explanation: Program check interrupt has occurred. but FSYESPI1 cannot load FSYESPI2 program.

User response: Detail program check analysis is ignored. FSYESPI2 is dynamically loaded at run time. Probable cause of load failure is missing Migration Utility SYS1.SFSYLOAD from the JOBLIB or STEPLIB.

FSYESPI2-02,012 COBOL LISTING DATE/TIME STAMP DOES NOT MATCH

FSYESPI2-03,012 *ANALYSIS OF INTERRUPT IGNORED'

Explanation: Program check interrupt has occurred. FSYESPI1 program passed control to FSYESPI2. FSYESPI2 program opened COBLIST1 file but the source listing time or date stamp does not match that of the load module.

User response: Your COBOL listing of the program you are running is out of synch with the load module. ABEND analysis continues without the listing.

FSYGDBD0-01,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator (installer) for the correct Migration Utility load libraries.

FSYGDBD0-02,012 ERROR OPENING IMS DDNAME

Explanation: IMS ddname is not specified in the JCL.

User response: Add "//IMS DD . . ." name to your JCL. To access IMS, you need the PSBLIB(s) and the DBDLIB(s) specified in the JCL via the IMS ddname.

FSYGDBD0-03,012 CANNOT LOCATE DBD: &DATABASE, SEG: &SEGMENT

Explanation: The segment name &SEGMENT cannot be located in the &DATABASE table.

User response: Verify that you have the correct PSB name specified in the PARM= on your EXEC statement and that you have the correct IMS files defined in the JCL. &SEGMENT is the name of the DLI segment in your program. The &DATABASE is the DBD name passed to your program by the DLI interface, in the PCB for the data base you are accessing. Make sure that the segment you are accessing is defined in your data base. If problem persists, seek help from your DLI administrator.

FSY003E–FSYGPCB0 DBD &DATABASE CANNOT BE RESOLVED

Explanation: Your COBOL program cannot resolve &DATABASE from the PCBs passed to it by the DFSRRC00 DLI interface program. FSYGPCB0 is the program that issued this error.

User response: &DATABASE is the DBD name specified in your program. This DBD cannot be located in any PCBs passed to your program by the DLI interface. Make sure that the DBD name you are accessing is correct. Verify that the correct PSB name is specified in the PARM= on your EXEC statement and that you have the correct IMS files defined in the JCL. If the problem persists, seek help from your DLI administrator.

FSYMIGS0-01,012 FJYMIG0 FILE OPEN ERROR

Explanation: FJYMIG0 file cannot be opened. Main program is FSYMIG00, the subprogram that issued this message is FSYMIGS0.

User response: FJYMIG0 file is dynamically allocated by the FSYMIG00 program as follows:

1. For job steps that execute a linked Easytrieve Plus program, the data set name is obtained from the definition of //FJYMIG0 in the #FJICNTL control file. Make sure that //FJYMIG0 is pointing to a valid PDS file that contains the file information member generated by the conversion PROC.
2. For Link and Go Easytrieve Plus job steps, this file is allocated to the FJSYSP0 file produced by the Migration Utility translator step. This file is internally generated for all Link and Go jobs by FSYMIG00. Check the console for any informational messages that might indicate a failed allocation attempt. If this problem occurred with a Link and Go step, verify the MODEL= parameter in the #EZTPROC to make sure that you have read and write access to such a qualifier.

FSYMIGS1-01,012 FJPROC0 FILE OPEN ERROR

Explanation: FJPROC0 file cannot be opened. Main program is FSYMIG00, the subprogram that issued this message is FSYMIGS1. The #EZTPROC pointed to by FSYPROCS default table cannot be located.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator (installer) for the correct Migration Utility load libraries.

FSYMIGS1-02,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator.

FSYMIGS2-02,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator.

FSYMIG00 PROC=&MEMBER ERROR STMT=*NNNNNN* NEAR TEXT "&TEXT"

Explanation: The FSYMIG00 program detected an error in the &MEMBER file. &MEMBER is the name of the PDS member being decoded. *NNNNNN* is the statement in error. &TEXT is the text in error. This error can be issued for two reasons:

1. The JOB/JCL pointed to by the FJIJOB0 does not contain a valid JOB statement on the first line.
2. The "//TEST-COMPILERS" or "//PROD-COMPILERS" statement in the #FJICNTL file is not followed by one or more &EZNAME=&MUNAME valid module names.

User response: Correct the statements in error.

FSYMIG00 JOB ---> &JOBNAME STEP=&STEPNAME &EXEC

Explanation: This is an informational message that shows the progress of the FSYMIG00 run. This message is issued for each job step while interpreting the input job. This information is useful for debugging, if there is a problem with the FSYMIG00 run.

User response: None.

FSYMIG00 PROC ---> &PROCNAME STEP=&STEPNAME &EXEC

Explanation: This is an informational message that shows the progress of the FSYMIG00 run. This message is issued for each PROC step while interpreting PROCS in the input job. This information is useful for debugging, if there is a problem with the FSYMIG00 run.

User response: None.

FSYMIG00-01,012 &FILE FILE OPEN ERROR

Explanation: The file &FILE cannot be opened.

User response: Make sure the &FILE ddname is defined in the JCL. If &FILE is FJPROC0, and you have FJPROC0 coded in the JCL, make sure that it points to a valid PDS with a valid proc name. If you do not have FJPROC0 coded in the JCL, verify that the proc name in the FSYPROCS table specifies the correct proc name located in the SYS1.SFSYJCL Migration Utility library.

FSYMIG00-02,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator.

FSYMIG00-03,012 PROC "&PROCNAME" NOT FOUND

Explanation: The input job being interpreted by FSYMIG00 program contains a PROC that cannot be located in the FJIPROC library specified in your FSYMIG00 JCL.

User response: FSYMIG00 program scans all PROCS found in the JOB it is tailoring. Make sure that the FJIPROC in the FSYMIG00 JCL concatenates all data sets where PROCS are located.

FSYMIG05-01,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator.

FSYMIG10-01,012 &FILE FILE OPEN ERROR

Explanation: The file &FILE cannot be opened.

User response: Make sure the &FILE ddname is defined in the JCL. If &FILE is FJPROC0, and you have FJPROC0 coded in the JCL, make sure that you are pointing to a valid PDS with a valid proc name. If you do not have FJPROC0 coded in the JCL, verify that the proc name in the FSYPROCS table specifies the correct proc name located in the SYS1.SFSYJCL Migration Utility library.

FSYMIG10-02,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator.

FSYSTAE0 ABENDCHK INTERRUPT OCCURRED

Explanation: A program check or an ABEND occurred in the main task while running a job step.

User response: This is an informational message. Refer to system generated messages, abend information, and log files for the actual cause of the interrupt.

FSYSTAE0 ABENDCK2 INTERRUPT OCCURRED

Explanation: A program check or an ABEND occurred in a sub task while running a job step.

User response: This is an informational message. Refer to system generated messages, abend information, and log files for the actual cause of the interrupt.

FSYSVC99 TEXT ERROR NEAR "&TEXT"

Explanation: An inappropriate dynamic allocation parameter/text was passed to the FSYSVC99 dynamic allocator program.

User response: This error can occur for two reasons:

1. Due to incorrect statements in the #EZTPROC/#CNVPROC. These PROCS are normally tailored at system installation time. Report any errors to your Migration Utility system administrator.
2. Due to inappropriate text passed to FSYSVC99 by the Migration Utility programs. Report these types of errors to the IBM Support Center. To assist debugging, turn on FSYSVC99 program messages by coding the following statement in the JCL:

```
//FJSVC99 DD DSN=&FJSVC99(MSGALL)
```

This statement will cause printing of the system-generated messages that may lead to the actual cause.

Note: This switch is not for normal use. Remove //FJSVC99 from your JCL when no longer needed.

FSYTPA00 PROC=&PROCNAME ERROR STMT=*NNNNNN* NEAR TEXT "&TEXT"

Explanation: The translator driver program detected an error in the &PROCNAME proc.

User response: &PROCNAME is the name of the proc. This proc is normally obtained from the system default FSYPROCS table or from the FJPROC0 DD name, if coded in the JCL. *NNNNNN* is the statement in error. &TEXT is the text in error. The proc is expected to contain valid JCL statements. If you are using your own proc, correct the statement in error. Otherwise consult with the Migration Utility installer (administrator) for corrective action.

FSYTPA00-01,012 FJPROC0 FILE OPEN ERROR

Explanation: FJPROC0 file cannot be opened.

User response: This proc is normally obtained from the system default FSYPROCS table or from the FJPROC0 ddname, if coded in the JCL. If you have FJPROC0 coded in the JCL, make sure that it points to a valid PDS with a valid proc name. If you do not have FJPROC0 coded in the JCL, verify that the proc name in the FSYPROCS table specifies the correct proc name located in the SYS1.SFSYJCL Migration Utility library.

FSYTPA00-02,012 PROGRAM "&PROGRAM" LOAD ERROR

Explanation: The program &PROGRAM cannot be located in the specified load libraries.

User response: Verify that you are using the correct Migration Utility libraries. Verify the //STEPLIB and the //JOBLIB respectively. Consult with your Migration Utility administrator (installer) for the correct Migration Utility load libraries.

Runtime I/O error messages

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.

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, typically 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.

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.

Appendix. Migration Utility JCL

This appendix lists the supplied JCL for running Migration Utility and the FSYMIG00 control file.

JCMUCLGJ—Translate, Link and Go (no proc)

JCMUCLGJ can be found in SYS1.SFSYJCLS. It performs a one-step translate, link and go without a proc.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//*********************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.                *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.       *
//*                                                                   *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.           *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.         *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.             *
//*                                                                   *
//* REV. 01/20/2004                                                   *
//*********************************************************************
//* JCMUCLGJ - COMPILE & LINK AND EXECUTE PENGIEZT/MU PROGRAM         *
//*            (LINK AND GO)                                          *
//*********************************************************************
//*
//FSYTPA00 EXEC PGM=FSYTPA00,PARM=
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=???????.MUCONV.EZPLUS.MACROS,
//         DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSLIST  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//REPORT1  DD SYSOUT=*
//FILEIN   DD DSN=SYS1.????(TESTFIL0),
//         DISP=SHR
//SYSIN    DD DSN=SYS1.SFSYEZTS(TESTMU00),
//         DISP=SHR
/*
//
```

JCMUCLGP—Translate, Link and Go (instream proc)

JCMUCLGP can be found in SYS1.SFSYJCLS. It performs a one-step translate, link and go using an instream proc.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//*********************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.                *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.       *
//*                                                                   *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.           *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.         *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.             *
//*                                                                   *
//* REV. 01/20/2004                                                   *
//*********************************************************************
//* JCMUCLGP - COMPILE, LINK  AND EXECUTE IN ONE STEP - INSTREAM PROC *
//*********************************************************************
//FSPENGI  PROC  SYSOUT='*',
//**        FJSYSPH=???????.MUCONV.COBSRC,          OPTIONAL GENED COBOL
//**        FJPROC0=SYS1.SFSYJCLS(#EZTPROC),        OPTIONAL PROC
//          SYSLIB1=SYS1.SFSYLOAD,                  PENGIEZT/MU LOADLIB
//          PANDD=???????.MUCONV.EZPLUS.MACROS,     YOUR EZT MACROS
//          SYSIN=SYS1.SFSYEZTS,                    INPUT EZT PROGRAM
//* INPUT FILE FOR THE TEST PROGRAMS. CONSRUCT YOUR OWN FILES AS NEEDED
//          FILEIN=SYS1.????(TESTFIL0),
//          MEMBER=GO,                              PROGRAM NAME
//          PARAM=                                  POSSIBLE USER PARMS
//*-------------------------------------------------------------------*
//FSYTPA00 EXEC PGM=FSYTPA00,PARM=&PARAM
//**FJPROC0  DD DSN=&FJPROC0,
//**          DISP=SHR
//**FJSYSPH  DD DSN=&FJSYSPH(&MEMBER),
//**          DISP=OLD
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB   DD DSN=&SYSLIB1,
//          DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=&PANDD,
//          DISP=SHR
//SYSOUT   DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//REPORT1  DD SYSOUT=&SYSOUT
//FILEIN   DD DSN=&FILEIN,
//          DISP=SHR
//SYSIN    DD DSN=&SYSIN(&MEMBER),
//          DISP=SHR
// PEND
//*
//STEP010  EXEC PROC=FSPENGI,MEMBER=MUTEST01
/*
//
```

JCMUCL1J—Translate and link (no proc)

JCMUCL1J can be found in SYS1.SFSYJCLS. It performs a one-step translate and link without a proc.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//          CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,DISP=SHR
//*********************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.                *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.       *
//*                                                                   *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.           *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.         *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.             *
//*                                                                   *
//* REV. 01/20/2004                                                   *
//*********************************************************************
```

```
//* JCMUCL1J - COMPILE & LINK PENGIEZT/MU PROGRAM IN ONE STEP         *
//*********************************************************************
//STEP010   EXEC PGM=FSYTPA00
//SYSOUT    DD SYSOUT=*
//SYSLIST   DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//* LINK TO LOADLIB (WHERE YOUR PROGRAM WILL BE LINKED)
//SYSLMOD   DD DSN=???????.MUCONV.LOADLIB,
//          DISP=SHR
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB    DD DSN=SYS1.SFSYLOAD,
//          DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD     DD DSN=???????.MUCONV.EZPLUS.MACROS,
//          DISP=SHR
//SYSIN     DD DSN=SYS1.SFSYEZTS(MUTEST00),DISP=SHR
/*
//
```

JCMUCL1P—Translate and link (instream proc)

JCMUCL1P can be found in SYS1.SFSYJCLS. It performs a one-step translate and link using an instream proc.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//         DD DSN=???????.MUCONV.LOADLIB,
//         DISP=SHR
//*********************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.                *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.       *
//*                                                                   *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.           *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.         *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.             *
//*                                                                   *
//* REV. 01/20/2004                                                   *
//*********************************************************************
//* JCMUCL1P - COMPILE & LINK IN ONE STEP (INSTREAM PROC)             *
//*********************************************************************
//FSYTPA00 PROC  SYSOUT='*',
//**       FJSYSPH=???????.MUCONV.COBSRC,          OPTIONAL GENED COBOL
//         SYSLMOD=???????.MUCONV.LOADLIB,         LINK TO LOADLIB
//         SYSLIB1=SYS1.SFSYLOAD,                  PENGIEZT/MU LOADLIB
//**       FJPROC0=SYS1.SFSYJCLS(#EZTPROC),         OPTIONAL PROC
//         PANDD=???????.MUCONV.EZPLUS.MACROS,     YOUR EZT MACROS
//         SYSIN=SYS1.SFSYEZTS,                    INPUT EZT PROGRAM
//         MEMBER=GO
//*-------------------------------------------------------------------*
//* FSYTPA00 STEP: TRANSLATE, COMPILE AND LINK.                       *
//*-------------------------------------------------------------------*
//FSYTPA00  EXEC PGM=FSYTPA00
//**FJPROC0   DD DSN=&FJPROC0,
//**          DISP=SHR
//**FJSYSPH   DD DSN=&FJSYSPH(&MEMBER),
//**          DISP=OLD
//SYSOUT    DD SYSOUT=&SYSOUT
//SYSLIST   DD SYSOUT=&SYSOUT
//SYSPRINT  DD SYSOUT=&SYSOUT
//SYSLMOD   DD DSN=&SYSLMOD,
//          DISP=SHR
```

```
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB   DD DSN=&SYSLIB1,
//         DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD     DD DSN=&PANDD,
//          DISP=SHR
//SYSIN     DD DSN=&SYSIN(&MEMBER),
//          DISP=SHR
// PEND
//*
//MUTEST00 EXEC PROC=FSYTPA00,MEMBER=MUTEST00
/*
//
```

JCMUCL2J—Two-step translate and link without a proc

JCMUCL2J can be found in SYS1.SFSYJCLS. It performs a two-step translate and link without a proc.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//*********************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.                *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.       *
//*                                                                   *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.           *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.         *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.             *
//*                                                                   *
//* REV. 01/20/2004                                                   *
//*********************************************************************
//* JCMUCL2J - COMPILE & LINK PENGIEZT/MU PROGRAM IN TWO STEPS        *
//*********************************************************************
//STEP010  EXEC PGM=FSYTPA00
//SYSOUT   DD SYSOUT=*
//SYSLIST  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD UNIT=VIO,
//         DSN=&&LOADSET,
//         DISP=(NEW,PASS,DELETE),
//         SPACE=(CYL,(3,5),RLSE),
//         DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=)
//*
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=???????.MUCONV.EZPLUS.MACROS,
//         DISP=SHR
//SYSIN    DD DSN=SYS1.SFSYEZTS(MUTEST00),
//         DISP=SHR
//*
//*-------------------------------------------------------------------*
//* LINK EDIT STEP.  CREATES LOAD MODULE IN SYSLMOD.                  *
//*-------------------------------------------------------------------*
//LKED     EXEC PGM=IEWL,PARM='LIST,LET,XREF,MAP,AMODE(31),RMODE(24)'
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE,DELETE)
//* LINK TO LOADLIB (WHERE YOUR PROGRAM WILL BE LINKED)
//SYSLMOD  DD DSN=???????.MUCONV.LOADLIB,
//         DISP=SHR
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//SYSUT1   DD  UNIT=VIO,SPACE=(CYL,(5,10),RLSE)
/*
//
```

JCMUCL2P—Two-step translate and link (instream proc)

JCMUCL2P can be found in SYS1.SFSYJCLS. It performs a two-step translate and link using an instream proc.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//         DD DSN=???????.MUCONV.LOADLIB,
//         DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.      *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.         *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.       *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/20/2004                                                 *
//*******************************************************************
//* JCMUCL2P - COMPILE & LINK IN TWO STEPS (THE EZT WAY), INSTREAM PROC
//*******************************************************************
//FSYTPA00 PROC  SYSOUT='*',
//**       FJSYSPH=???????.MUCONV.COBSRC,          OPTIONAL GENED COBOL
//**       FJPROC0=SYS1.SFSYJCLS(#EZTPROC),         OPTIONAL PROC
//         SYSLMOD=???????.MUCONV.LOADLIB,         LINK TO LOADLIB
//         SYSLIB1=SYS1.SFSYLOAD,                  PENGIEZT/MU LOADLIB
//         PANDD=???????.MUCONV.EZPLUS.MACROS,     YOUR EZT MACROS
//         SYSIN=SYS1.SFSYEZTS,                    INPUT EZT PROGRAM
//         MEMBER=GO
//*-----------------------------------------------------------------*
//* FSYTPA00 STEP: TRANSLATE, COMPILE AND LINK.                     *
//*-----------------------------------------------------------------*
//FSYTPA00 EXEC PGM=FSYTPA00
//**FJPROC0  DD DSN=&FJPROC0,
//**         DISP=SHR
//**FJSYSPH  DD DSN=&FJSYSPH(&MEMBER),
//**         DISP=OLD
//SYSOUT   DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//SYSLIN   DD UNIT=VIO,
//         DSN=&&LOADSET,
//         DISP=(NEW,PASS,DELETE),
//         SPACE=(CYL,(3,5),RLSE),
//         DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=)
//*
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=&PANDD,
//         DISP=SHR
//SYSIN    DD DSN=&SYSIN(&MEMBER),
//         DISP=SHR
//*
//*-----------------------------------------------------------------*
```

```
//* LINK EDIT STEP.  CREATES LOAD MODULE IN SYSLMOD.                 *
//*------------------------------------------------------------------*
//LKED     EXEC PGM=IEWL,PARM='LIST,LET,XREF,MAP,AMODE(31),RMODE(24)'
//SYSOUT   DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//SYSLIN   DD DSN=&&LOADSET,DISP=(OLD,DELETE,DELETE)
//SYSLMOD  DD DSN=&SYSLMOD,
//         DISP=SHR
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB   DD DSN=&SYSLIB1,
//         DISP=SHR
//SYSUT1   DD  UNIT=VIO,SPACE=(CYL,(5,10),RLSE)
// PEND
//*
//MUTEST00 EXEC PROC=FSYTPA00,MEMBER=MUTEST00
//
```

JCMUIMSJ—Sample job for translating IMS/DLI programs

JCMUIMSJ can be found in SYS1.SFSYJCLS. It is a sample job for translating IMS/DLI programs.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=FSOFT01.FSYMG400.LOADLIB,
//         DISP=SHR
//********************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.               *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.      *
//*                                                                  *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.          *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.        *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.            *
//*                                                                  *
//* REV. 01/26/2004                                                  *
//********************************************************************
//* JCMUIMSJ - COMPILE, LINK AND EXECUTE IMS/DLI PENGIEZT/MU PROGRAM  *
//*          (LINK AND GO)                                           *
//********************************************************************
//*
//FSYTPA00 EXEC  PGM=DFSRRC00,REGION=2048K,
//         PARM='DLI,FSYTPA00,&PSBNAME'
//* NOTE: ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES
//SYSLIB   DD DSN=FSOFT01.FSYMG400.LOADLIB,
//         DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).

//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD    DD DSN=FSOFT01.MUCONV.EZPLUS.MACROS,
//         DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSLIST  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*
//* ADD YOUR OWN JCL FOR ACCESSING DLI, AND WHATEVER FILES YOU MAY NEED
//IMS      DD DSN=???????.???????.PSBLIB,DISP=SHR
//         DD DSN=???????.???????.DBDLIB,DISP=SHR
//SYSIN    DD *
??? ADD YOUR OWN PROGRAM HERE
/*
//
```

JCMUSQLJ—Two-step translate, link and bind for SQL

JCMUSQLJ can be found in SYS1.SFSYJCLS. It performs a two-step translate, link and bind for SQL.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//         DD DSN=DSN710.SDSNEXIT.DBVA,
//         DISP=SHR
//         DD DSN=DB2.V7R1M0.SDSNLOAD,
//         DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.     *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.         *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.       *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/20/2004                                                 *
//*******************************************************************
//* JCMUSQLJ - TRANSLATE, COMPILE, LINK AND BIND PENGIEZT/MU PROGRAMS *
//*******************************************************************
//*
//*-----------------------------------------------------------------*
//* FSYTPA00 STEP: TRANSLATE, SQLTRAN, COMPILE AND LINK.            *
//* "PARM LINK (&PGMNAME)" IS REQUIRED IN EZT PLUS SOURCE.          *
//*-----------------------------------------------------------------*
//FSYTPA00  EXEC PGM=FSYTPA00
//SYSOUT    DD SYSOUT=*
//SYSLIST   DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB    DD DSN=SYS1.SFSYLOAD,
//          DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD     DD DSN=???????.MUCONV.EZPLUS.MACROS,
//          DISP=SHR
//SYSLMOD   DD DSN=???????.MUCONV.LOADLIB,
//          DISP=SHR
//FJBIND0   DD UNIT=SYSDA,
//          DSN=&&FJBIND0,
//          SPACE=(TRK,(15,15,)),
//          DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//          DISP=(NEW,PASS,DELETE)
//DBRMLIB   DD DSN=???????.MUCONV.DBRMLIB(TESTCOL0),
//          DISP=SHR
//SYSIN     DD DSN=SYS1.SFSYEZTS(TESTCOL0),
//          DISP=SHR
//*
//*-----------------------------------------------------------------*
//*  BIND COMPILED PROGRAM.                                         *
//* "PARM BIND (STATIC-ONLY)" IS REQUIRED IN EZT PLUS SOURCE.       *
//*-----------------------------------------------------------------*
//BIND     EXEC PGM=IKJEFT01,COND=(5,LT,FSYTPA00)
//DBRMLIB  DD DSN=???????.MUCONV.DBRMLIB,
//         DISP=SHR
//SYSUT1   DD UNIT=SYSDA,
//         SPACE=(CYL,(4,5))
//* SYSTSIN CAN BE RE-DIRECTED TO YOUR OWN BIND STATEMENTS
//SYSTSIN  DD DSN=&&FJBIND0,
//         DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//$RUNBYRN DD DUMMY
/*
//
```

JCMUSQLP—Two-step translate, link and bind for SQL (using proc)

JCMUSQLP can be found in SYS1.SFSYJCLS. It performs a two-step translate, link and bind for SQL using a proc.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,COND=(5,LT),REGION=0M
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//         DD DSN=???????.MUCONV.LOADLIB,
//         DISP=SHR
//         DD DSN=DSN710.SDSNEXIT.DBVA,
//         DISP=SHR
//         DD DSN=DB2.V7R1M0.SDSNLOAD,
//         DISP=SHR
//********************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.                *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.       *
//*                                                                  *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.          *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.         *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.            *
//*                                                                  *
//* REV. 01/20/2004                                                  *
//********************************************************************
//* JCMUSQLP - INSTREAM PROC (CODE EXEC AT THE BOTTOM OF THIS PROC)   *
//*                                                                  *
//* TRANSLATE, COMPILE, LINK AND BIND PENGIEZT/MU PROGRAMS.          *
//* "PARM LINK (&PGMNAME)" IS REQUIRED IN EZT PLUS SOURCE.           *
//********************************************************************
//FSPENGI  PROC SYSOUT='*',
//         CWORK=VIO,                              WORK DASD/UNIT
//**       FJSYSPH=???????.MUCONV.COBSRC,          OPTIONAL GENED COBOL
//         SYSLMOD=???????.MUCONV.LOADLIB,         LINK TO LOADLIB
//         SYSLIB1=SYS1.SFSYLOAD,                  PENGIEZT/MU LOADLIB
//         SYSLIB2=DSN710.SDSNEXIT.DBVA,           DB2 EXIT LIB
//         SYSLIB3=DB2.V7R1M0.SDSNLOAD,            DB2 LOADLIB
//         DBRMLIB=???????.MUCONV.DBRMLIB,         DBRMLIB
//**       FJPROC0=SYS1.SFSYJCLS(#EZTPROC),         OPTIONAL PROC
//         PANDD=???????.MUCONV.EZPLUS.MACROS,     YOUR EZT MACROS
//         SYSIN=SYS1.SFSYEZTS,                    INPUT EZT PROGRAM
//         MEMBER=GO                               PROGRAM NAME
//*
//*------------------------------------------------------------------*
//* FSYTPA00 STEP: TRANSLATE, SQLTRAN, COMPILE AND LINK.             *
//*------------------------------------------------------------------*
//FSYTPA00  EXEC PGM=FSYTPA00
//**FJPROC0   DD DSN=&FJPROC0,
//**          DISP=SHR
//**FJSYSPH   DD DSN=&FJSYSPH(&MEMBER),
//**          DISP=OLD
//SYSOUT    DD SYSOUT=&SYSOUT
//SYSLIST1  DD SYSOUT=&SYSOUT
//SYSPRINT  DD SYSOUT=&SYSOUT
//SYSLMOD   DD DSN=&SYSLMOD,
//          DISP=SHR
//* ADDITIONAL SYSLIB DSNS CAN BE CODED FOR YOUR OWN INCLUDES.
//SYSLIB    DD DSN=&SYSLIB1,
//          DISP=SHR
//          DD DSN=&SYSLIB2,
//          DISP=SHR
//          DD DSN=&SYSLIB3,
//          DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD     DD DSN=&PANDD,
//          DISP=SHR
//FJBIND0   DD UNIT=&CWORK,
//          DSN=&&FJBIND0,
//          SPACE=(TRK,(15,15,)),
//          DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//          DISP=(NEW,PASS,DELETE)
//DBRMLIB   DD DSN=&DBRMLIB(&MEMBER),
//          DISP=OLD
//SYSIN     DD DSN=&SYSIN(&MEMBER),
//          DISP=SHR
//*
//*------------------------------------------------------------------*
//*  BIND COMPILED PROGRAM.                                          *
//* "PARM BIND (STATIC-ONLY)" IS REQUIRED IN EZT PLUS SOURCE.        *
//*------------------------------------------------------------------*
//BIND     EXEC PGM=IKJEFT01,COND=(5,LT,FSYTPA00)
//DBRMLIB  DD DSN=&DBRMLIB,
//         DISP=SHR
//SYSUT1   DD UNIT=&CWORK,
//         SPACE=(CYL,(5,10))
//* SYSTSIN CAN BE RE-DIRECTED TO YOUR OWN BIND STATEMENTS
//SYSTSIN  DD DSN=&&FJBIND0,
//         DISP=SHR
//SYSPRINT DD SYSOUT=&SYSOUT
//SYSTSPRT DD SYSOUT=&SYSOUT
//$RUNBYRN DD DUMMY
// PEND
//*
//TESTCOL0 EXEC PROC=FSPENGI,MEMBER=TESTCOL0
/*
//
```

JCMUMIG1—Automated conversion engine

JCMUMIG1 can be found in SYS1.SFSYJCLS. It is Migration Utility's automated conversion engine.

```
//FSOFT01G JOB ,'user name',
//         CLASS=C,
//         MSGCLASS=X,
//         REGION=0M,
//&NOTIFY  NOTIFY=&USERID,
//         COND=(256,LT)
//*
//JOBLIB   DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//         DD DSN=???????.MUCONV.LOADLIB,
//         DISP=SHR
//******************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.             *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.    *
//*                                                                *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.        *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.      *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.          *
//*                                                                *
//* REV. 01/20/2004                                                *
//******************************************************************
```

```
//* JCMUMIG1 - CONVERSION (COMPILE & LINK - VIA AN INSTREAM PROC)     *
//*           USAGE: AUTOMATED CONVERSION OF DB2 AND NON-DB2 PROGRAMS*
//*                                                                   *
//* /======> REFER TO JCMUMIG2 JCL FOR THE MANUAL CONVERSION PROCESS. *
//*-------------------------------------------------------------------*
//* /=> THIS JOB IS INITIATED VIA THE JCMUCNV1 JOB FOR MASS CONVERSION*
//*     OF EASYTRIEVE PLUS PROGRAMS TO COBOL.                         *
//*                                                                   *
//* /=> THE "JCMUCNV1" READS A TABLE OF PROGRAMS TO BE TRANSLATED,    *
//*     PLUS THIS JCL, AND INITIATES THE AUTOMATED CONVERSION ENGINE. *
//*                                                                   *
//* /=> THE "JCMUCNV1" WAS SUPPLIED WITH THE PRODUCT. IT CAN BE       *
//*     LOCATED IN THE SAME LIBRARY WHERE THIS JCL RESIDES/RESIDED.   *
//*                                                                   *
//* /=> THIS INSTREAM "FSPENGI" PROC TRANSLATES THE SUPPLIED &MEMBER  *
//*     PROGRAM AS PER #CNVPROC PENGIEZT/MU CONVERSION PROC.          *
//*                                                                   *
//* /=> THE INSTREAM "CONVPD1" LOCATED AT THE BOTTOM OF THIS JOB,     *
//*     RE-INITIATES THIS PROGRAM WITH THE NEXT MEMBER NAME LOCATED   *
//*     IN THE PROGRAM TABLE.                                         *
//*                                                                   *
//* /=> THIS PROCESS STAYS IN A LOOP UNTIL ALL PROGRAMS ARE TRANSLATED.
//*                                                                   *
//* /=> YOU CAN STOP THIS PROCESS BY RE-DIRECTING THE RDRFILE         *
//*     IN //STEP010, TO AN OUTPUT OTHER THAN THE INTERNAL READER.    *
//*                         --- OR ---                                *
//*     YOU CAN INSERT A $END STATEMENT INTO YOUR PROGRAM TABLE,      *
//*     AFTER A PROGRAM NAME THAT HAS NOT BEEN TRANSLATED YET.        *
//*********************************************************************
//*
//FSPENGI  PROC SYSOUT='*',
//         CWORK=VIO,
//*-------------------------------------------------------------------*
//* FJPROC0 POINTS TO THE PENGIEZT/MU CONVERSION PROC                 *
//* #CNVPROC PROC WAS SUPPLIED WITH THE PROGRAM PRODUCT.              *
//*-------------------------------------------------------------------*
//         FJPROC0=SYS1.SFSYJCLS(#CNVPROC),
//*
//*-------------------------------------------------------------------*
//* FJERLOG IS A CUMULATIVE LOG OF PROGRAMS THAT FAILED TO TRANSLATE  *
//* THIS MUST BE A PRE-ALLOCATED SEQUENTIAL FILE (LRECL=80,RECFM=FB)  *
//*-------------------------------------------------------------------*
//         FJERLOG=???????.MUCONV.ERRORS,
//*
//*-------------------------------------------------------------------*
//* EASYTRIEVE FILES INFORMATION IS PUNCHED INTO THIS PDS             *
//* THIS MUST BE A PRE-ALLOCATED PDS FILE (LRECL=80,RECFM=FB)         *
//* THIS FILE IS NEEDED BY THE PARALLEL TEST (FSYMIG10) PROGRAM.      *
//*-------------------------------------------------------------------*
//         FJSYSP0=???????.MUCONV.FJSYSP0,
//*
//*-------------------------------------------------------------------*
//* PDS INTO WHICH THE GENERATED COBOL IS SAVED                       *
//*-------------------------------------------------------------------*
//         FJSYSPH=???????.MUCONV.COBSRC,
//*
//*-------------------------------------------------------------------*
//*            SQL (DB2) DBRMLIB                                      *
//*     MUST BE REPLACED BY YOUR OWN LIBRARIES                        *
//*       (COMMENT OUT IF NOT USING DB2/SQL)                          *
//*-------------------------------------------------------------------*
//         DBRMLIB=???????.MUCONV.DBRMLIB,
//*
//*-------------------------------------------------------------------*
//* THE TARGET LOAD LIBRARY FOR LKED STEP.                            *
//* TRANSLATED PROGRAMS ARE LINKED INTO THIS LOADLIB.                 *
//*-------------------------------------------------------------------*
```

```
//         SYSLMOD=???????.MUCONV.LOADLIB,
//*
//*-----------------------------------------------------------------*
//* LINK TO BOOTSTRAP LIBRARY (BOOT STRAPS ARE LINKED INTO THIS LIB)  *
//*-----------------------------------------------------------------*
//         BOOTMOD=???????.MUCONV.BOOTSTRP.LOADLIB,
//*
//*-----------------------------------------------------------------*
//* SYSLIB THAT CONTAINS SUBMODULES CALLED BY EASYTRIEVE PROGRAMS.    *
//* /==> THE SYSLIB IN THIS JCL IS CONCATENATED TO THE SYSLIB IN      *
//*      THE LKED STEP OF #CNVPROC.                                   *
//*                                                                   *
//* /==> YOU CAN INCLUDE ONE OR MORE LIBRARIES THAT CONTAIN MODULES   *
//*      NEEDED TO RESOLVE EXTERNAL CALLS, INCLUDING DB2 LIBRARIES,   *
//*      IF YOU ARE CONVERTING DB2 PROGRAMS.                          *
//*-----------------------------------------------------------------*
//         SYSLIB=???????.MUCONV.LOADLIB,
//*
//*-----------------------------------------------------------------*
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.              *
//*                                                                   *
//* PANDD CAN BE CHANGED AT INSTALLATION TIME. (SEE #CNVPROC).        *
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)        *
//*-----------------------------------------------------------------*
//         PANDD=???????.MUCONV.EZPLUS.MACROS,
//*
//*-----------------------------------------------------------------*
//* SYSIN MUST POINT TO PDS WHERE YOUR EASYTRIEVE PLUS PROGRAMS ARE   *
//*-----------------------------------------------------------------*
//         SYSIN=SYS1.SFSYEZTS,
//*
//*-----------------------------------------------------------------*
//         MEMBER=GO                                PROGRAM NAME
//*
//*-----------------------------------------------------------------*
//* PENGIEZT/MU RUN PROCEDURE                                         *
//*-----------------------------------------------------------------*
//FSYTPA00 EXEC PGM=FSYTPA00
//STEPLIB  DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//* DB2 LIBRARIES BELOW CAN BE REMOVED IF DB2 IS NOT USED
//         DD DSN=DSN710.SDSNEXIT.DBVA,
//         DISP=SHR
//         DD DSN=DB2.V7R1M0.SDSNLOAD,
//         DISP=SHR
//SYSOUT   DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//FJPROC0  DD DSN=&FJPROC0,
//         DISP=SHR
//FJERLOG  DD DSN=&FJERLOG,
//         DISP=MOD
//FJSYSPH  DD DSN=&FJSYSPH(&MEMBER),
//         DISP=OLD
//FJSYSP0  DD DSN=&FJSYSP0(&MEMBER),
//         DISP=OLD
//DBRMLIB  DD DSN=&DBRMLIB(&MEMBER),
//         DISP=OLD
//FJBOOT0  DD DSN=&BOOTMOD,
//         DISP=SHR
//SYSLMOD  DD DSN=&SYSLMOD,
//         DISP=SHR
//SYSLIB   DD DSN=&SYSLIB,
//         DISP=SHR
//PANDD    DD DSN=&PANDD,
//         DISP=SHR
//SYSIN    DD DSN=&SYSIN(&MEMBER),
//         DISP=SHR
//FJBIND0  DD UNIT=&CWORK,
//         DSN=&&FJBIND0,
//         SPACE=(TRK,(15,15,)),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//         DISP=(NEW,PASS,DELETE)
//*
//*-----------------------------------------------------------------*
//*  BIND COMPILED PROGRAM (SQL/DB2) PROGRAMS ONLY.                 *
//*                                                                 *
//* /=> THIS STEP IS REQUESTED BY FSYTPA00, VIA RC=2, WHEN INPUT    *
//* PROGRAM CONTAINS SQL/DB2 STATEMENTS.                            *
//*                                                                 *
//* "PARM BIND (STATIC-ONLY)" IS REQUIRED IN EZT PLUS SOURCE.       *
//*-----------------------------------------------------------------*
//   IF (FSYTPA00.RUN=TRUE AND FSYTPA00.RC = 2)  THEN
//BIND     EXEC PGM=IKJEFT01
//DBRMLIB  DD DSN=&DBRMLIB,
//         DISP=SHR
//SYSUT1   DD UNIT=&CWORK,
//         SPACE=(CYL,(5,10))
//* SYSTSIN CAN BE RE-DIRECTED TO YOUR OWN BIND STATEMENTS
//SYSTSIN  DD DSN=&&FJBIND0,
//         DISP=SHR
//SYSPRINT DD SYSOUT=&SYSOUT
//SYSTSPRT DD SYSOUT=&SYSOUT
//$RUNBYRN DD DUMMY
//   ENDIF
//*
//   IF  (BIND.RUN=TRUE AND BIND.RC > 0 )  THEN
//BINDER   EXEC PGM=FSYCNV02,
//         REGION=4096K,PARM=(BIND,&MEMBER)
//SYSOUT   DD SYSOUT=&SYSOUT
//FJSYABE  DD SYSOUT=&SYSOUT
//REPORT1  DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//FJERLOG  DD DSN=&FJERLOG,
//         DISP=MOD
//   ENDIF
// PEND
//*
//*******************************************************************
//* FSYCNV01 - PROCEDURE TO INITIATE A NEW JOB                      *
//*******************************************************************
//CONVPD1  PROC   SYSOUT='*',
//*
//*-----------------------------------------------------------------*
//* PDS AND MEMBER NAME OF PROGRAM NAMES TO BE CONVERTED.           *
//* LEAVE THIS ENTRY AS IS. FSYCNV01 PROGRAM REPLACES THIS LINE     *
//* BY THE DIRLIST FILE DURING THE INITIATION (JCMUCNV1) JOB.       *
//*-----------------------------------------------------------------*
//&DIRLST  DIRLIST=,                .FSYCNV01 PROGRAM INSERTS IT
//*
//*-----------------------------------------------------------------*
//* PDS THAT CONTAINS THIS (JCMUMIG1) JOB.                          *
//*-----------------------------------------------------------------*
//         FILEIN=SYS1.SFSYJCLS,
//*-----------------------------------------------------------------*
//         CWORK=VIO,               .WORK FILES UNIT
//         ALLOC=10,                .SPACE FOR WORK FILES
//         PARAM=                   .PARM OPTIONS (LEAVE AS IS)
//*
//*-----------------------------------------------------------------*
//* FSYCNV01 RUN PROCEDURE.                                         *
//*-----------------------------------------------------------------*
//RESUB    EXEC PGM=FSYCNV01,REGION=4096K,COND=EVEN,PARM=(&PARAM)
//SYSOUT   DD SYSOUT=&SYSOUT
```

```
//FJSYABE  DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//REPORT1  DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//SYSUDUPM DD SYSOUT=&SYSOUT
//SYSABOUT DD SYSOUT=&SYSOUT
//*
//*-------- FILES INPUT TO FSYCNV01 ---------------------------------*
//DIRLIST  DD  DSN=&DIRLIST,
//         DISP=SHR
//FILEIN   DD  DSN=&FILEIN,
//         DISP=SHR
//RDRFILE  DD  SYSOUT=(*,INTRDR)
//*
//* THIS RDRFILE CAN BE UNCOMMENTED FOR PUNCHING PROC TO SYSOUT
//*RDRFILE  DD  SYSOUT=&SYSOUT
//*
//*-------- WORK FILES USED IN FSYCNV01 ------------------------------*
//TEMPWK1  DD UNIT=&CWORK,
//         DSN=&&TEMPWK1,
//         SPACE=(CYL,(&ALLOC,20))
//*
//SORTFL1  DD UNIT=&CWORK,
//         DSN=&&SORTFL1,
//         SPACE=(CYL,(&ALLOC,20))
//*
//* SORT WORK FILES NEEDED BY THE SORT PROGRAM.
//SORTWK01 DD UNIT=&CWORK,
//         SPACE=(CYL,(&ALLOC,20))
//*
//SORTWK02 DD UNIT=&CWORK,
//         SPACE=(CYL,(&ALLOC,20))
// PEND
//*
//&MEMBER  EXEC FSPENGI,MEMBER=&MEMBER
//&RESTART EXEC PROC=CONVPD1,PARAM=(JCMUMIG1,&RESTART)
//*
//
```

JCMUCNV1—Automated conversion initiation job

JCMUCNV1 can be found in SYS1.SFSYJCLS. It initiates the automated conversion job.

```
//FSOFT01Y JOB ,,NOTIFY=FSOFT01,
//          CLASS=C,MSGCLASS=X,COND=(5,LT)
//*
//JOBLIB  DD DSN=SYS1.SFSYLOAD,
//        DISP=SHR
//        DD DSN=???????.MUCONV.LOADLIB,
//        DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.     *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.         *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.       *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/20/2004                                                 *
//*******************************************************************
//* FSYCNV01 - TRANSLATOR CONVERSION INITIATION MODULE (RESTART)    *
//*                                                                 *
//* /=> THIS JOB INITIATES THE AUTOMATED TRANSLATING PROCESS BY     *
//*     SUBMITTING THE JOB NAME SPECIFIED IN THE PARM=(&JOB,&PROGRAM),*
//*     SPECIFIED AT THE BOTTOM OF THIS PROC, INTO INTERNAL READER.  *
//*                                                                 *
//* /=> YOU CAN TEST THIS JOB BY RE-DIRECTING THE RDRFILE IN          *
//*     //STEP010, TO AN OUTPUT OTHER THAN THE INTERNAL READER.       *
//*                                                                   *
//*********************************************************************
//*
//CONVPD1  PROC   SYSOUT='*',
//*-------------------------------------------------------------------*
//* PDS WHERE DIRECTORY TABLE OF PROGRAMS TO BE CONVERTED RESIDES.    *
//*-------------------------------------------------------------------*
//         DIRLIST=SYS1.SFSYEZTS,
//*                                                                   *
//*-------------------------------------------------------------------*
//* NAME OF THE DIRECTORY TABLE MEMBER.                               *
//*-------------------------------------------------------------------*
//         DIRMEMB=,
//*                                                                   *
//*-------------------------------------------------------------------*
//* PDS WHER JCMUMIG1 JOB/PROC IS LOCATED.                            *
//*-------------------------------------------------------------------*
//         FILEIN=SYS1.SFSYJCLS,
//*                                                                   *
//*-------------------------------------------------------------------*
//         WORK=SYSDA,               .WORK FILES
//         ALLOC=10,                 .WORK FILES SPACE
//         PARAM=                    .PARM OPTIONS IF ANY
//*
//*-------------------------------------------------------------------*
//* FSYCNV01 RUN PROCEDURE.                                           *
//*-------------------------------------------------------------------*
//STEP010  EXEC PGM=FSYCNV01,REGION=4096K,PARM=(&PARAM)
//SYSOUT   DD SYSOUT=&SYSOUT
//FJSYABE  DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//SYSPRINT DD SYSOUT=&SYSOUT
//SYSUDUPM DD SYSOUT=&SYSOUT
//SYSABOUT DD SYSOUT=&SYSOUT
//*
//*-------- FILES INPUT TO FSYCNV01 ---------------------------------*
//FILEIN   DD DSN=&FILEIN,
//         DISP=SHR
//DIRLIST  DD DSN=&DIRLIST(&DIRMEMB),
//         DISP=SHR
//RDRFILE  DD SYSOUT=(*,INTRDR)
//*
//* THIS RDRFILE CAN BE UNCOMMENTED FOR PUNCHING PROC TO SYSOUT
//*RDRFILE  DD  SYSOUT=&SYSOUT
//*
//*-------- WORK FILES USED IN FSYCNV01 ------------------------------*
//TEMPWK1  DD UNIT=&WORK,
//         DSN=&&TEMPWK1,
//         SPACE=(CYL,(&ALLOC,20))
//*
//SORTFL1  DD UNIT=&WORK,
//         DSN=&&SORTFL1,
//         SPACE=(CYL,(&ALLOC,20))
//*
//* SORT WORK FILES NEEDED BY THE SORT PROGRAM
//SORTWK01 DD UNIT=&WORK,
//         SPACE=(CYL,(&ALLOC,20))
//*
//SORTWK02 DD UNIT=&WORK,
//         SPACE=(CYL,(&ALLOC,20))
//*
// PEND
//* NOTE: $ALL IN THE PARAM BELOW INITIATES TRANSLATION OF ALL
//*       PROGRAMS LOCATED IN THE DIRMEMB= TABLE (UP TO THE $END).
//*       YOU CAN CHANGE $ALL TO A SPECIFIC MEMBER NAME IF DESIRED.
```

```
/*
//PENGI000 EXEC PROC=CONVPD1,PARAM='(JCMUMIG1,$ALL)',DIRMEMB=#PGMSTAB
/*
//
```

JCMUMIG2—Manual conversion engine with no restart

JCMUMIG2 can be found in SYS1.SFSYJCLS. It initiates the manual conversion engine.

```
//FSOFT01V JOB ,,NOTIFY=FSOFT01,
//         CLASS=A,MSGCLASS=X,REGION=0M
//*
//JOBLIB   DD DSN=???????.MUCONV.LOADLIB,
//         DISP=SHR
//         DD DSN=SYS1.SFSYLOAD,
//         DISP=SHR
//*******************************************************************
//* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.              *
//* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.     *
//*                                                                 *
//* 5697-I89 (C) COPYRIGHT IBM CORP 2001, 2002, 2003, 2004.         *
//* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.       *
//* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.           *
//*                                                                 *
//* REV. 01/20/2004              WRITTEN BY: SRECKO LAZANJA         *
//*******************************************************************
//* JCMUMIG2 - CONVERSION (COMPILE & LINK - VIA AN INSTREAM PROC)   *
//*           USAGE: MANUAL CONVERSION OF DB2 AND NON-DB2 PROGRAMS  *
//*                  (THIS JOB MUST BE SUBMITTED MANUALLY)          *
//*                                                                 *
//* /=====> REFER TO JCMUMIG1 JCL FOR AUTOMATED CONVERSION ENGINE.  *
//*******************************************************************
//FSPENGI  PROC SYSOUT='*',
//         CWORK=VIO,                             UNIT/SYSDA
//         FJPROC0=SYS1.SFSYJCLS(#CNVPROC),        CONVERSION PROC
//         FJERLOG=???????.MUCONV.ERRORS,         OUTPUT ERROR FILE
//         FJSYSP0=???????.MUCONV.FJSYSP0,        OUTPUT FILE INFO
//         FJSYSPH=???????.MUCONV.COBSRC,         OUTPUT COBOL SOURCE
//         SYSLMOD=???????.MUCONV.LOADLIB,          LINK TO LIBRARY
//         BOOTMOD=???????.MUCONV.BOOTSTRP.LOADLIB,   LINK TO BOOTSTRP
//         DBRMLIB=???????.MUCONV.DBRMLIB,        DB2 DBRMLIB
//         PANDD=???????.MUCONV.EZPLUS.MACROS,    YOUR EZT MACROS
//         SYSLIB=???????.MUCONV.LOADLIB,         OWN INCLUDES
//         SYSIN=SYS1.SFSYEZTS,                   INPUT EASYTRIEVE
//         MEMBER=GO                              PROGRAM NAME
//*-----------------------------------------------------------------*
//* FSYTPA00 RUN PROCEDURE                                          *
//*-----------------------------------------------------------------*
//FSYTPA00  EXEC PGM=FSYTPA00
//STEPLIB   DD DSN=SYS1.SFSYLOAD,
//          DISP=SHR
//* DB2 LIBRARIES BELOW CAN BE REMOVED IF DB2 IS NOT USED
//          DD DSN=DSN710.SDSNEXIT.DBVA,
//          DISP=SHR
//          DD DSN=DB2.V7R1M0.SDSNLOAD,
//          DISP=SHR
//SYSOUT    DD SYSOUT=&SYSOUT
//SYSLIST   DD SYSOUT=&SYSOUT
//SYSPRINT  DD SYSOUT=&SYSOUT
//FJPROC0   DD DSN=&FJPROC0,
//          DISP=SHR
//FJERLOG   DD DSN=&FJERLOG,
//          DISP=MOD
//FJSYSPH   DD DSN=&FJSYSPH(&MEMBER),
//          DISP=OLD
//FJSYSP0   DD DSN=&FJSYSP0(&MEMBER),
//          DISP=OLD
//FJBOOT0   DD DSN=&BOOTMOD(&MEMBER),
//          DISP=SHR
//SYSLMOD   DD DSN=&SYSLMOD,
//          DISP=SHR
//* PANDD MUST POINT TO YOUR OWN EASYTRIEVE PLUS MACROS.
//* CONSULT WITH SYSTEM PROGRAMMER FOR THE CORRECT DDNAME AS
//* IT CAN BE CHANGED AT INSTALLATION TIME. (SEE #EZTPROC).
//* (THIS IS AN OPTIONAL STATEMENT. COMMENT OUT IF NOT NEEDED)
//PANDD     DD DSN=&PANDD,
//          DISP=SHR
//SYSLIB    DD DSN=&SYSLIB,
//          DISP=SHR
//* DB2 LIBRARIES BELOW CAN BE REMOVED IF DB2 IS NOT USED
//DBRMLIB   DD DSN=&DBRMLIB(&MEMBER),
//          DISP=SHR
//SYSIN     DD DSN=&SYSIN(&MEMBER),
//          DISP=SHR
//FJBIND0   DD UNIT=&CWORK,
//          DSN=&&FJBIND0,
//          SPACE=(TRK,(15,15,)),
//          DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//          DISP=(NEW,PASS,DELETE)
//*
//*-------------------------------------------------------------------*
//*  BIND COMPILED PROGRAM (SQL/DB2) PROGRAMS ONLY.                   *
//*                                                                   *
//* /=> THIS STEP IS REQUESTED BY FSYTPA00, VIA RC=2, WHEN INPUT      *
//* PROGRAM CONTAINS SQL/DB2 STATEMENTS.                              *
//*                                                                   *
//* "PARM BIND (STATIC-ONLY)" IS REQUIRED IN EZT PLUS SOURCE.         *
//*-------------------------------------------------------------------*
//   IF (FSYTPA00.RUN=TRUE AND FSYTPA00.RC = 2)  THEN
//BIND     EXEC PGM=IKJEFT01
//DBRMLIB  DD DSN=&DBRMLIB,
//         DISP=SHR
//SYSUT1   DD UNIT=&CWORK,
//         SPACE=(CYL,(5,10))
//* SYSTSIN CAN BE RE-DIRECTED TO YOUR OWN BIND STATEMENTS
//SYSTSIN  DD DSN=&&FJBIND0,
//         DISP=SHR
//SYSPRINT DD SYSOUT=&SYSOUT
//SYSTSPRT DD SYSOUT=&SYSOUT
//$RUNBYRN DD DUMMY
//   ENDIF
//*
//   IF  (BIND.RUN=TRUE AND BIND.RC > 0 )  THEN
//BINDER   EXEC PGM=FSYCNV02,
//         REGION=4096K,PARM=(BIND,&MEMBER)
//SYSOUT   DD SYSOUT=&SYSOUT
//FJSYABE  DD SYSOUT=&SYSOUT
//REPORT1  DD SYSOUT=&SYSOUT
//SYSLIST  DD SYSOUT=&SYSOUT
//FJERMOG  DD DSN=&FJERLOG,
//         DISP=MOD
//   ENDIF
// PEND
//*
//*IBMDEMO3 EXEC PROC=FSPENGI,MEMBER=IBMDEMO3
//TESTCOL0 EXEC PROC=FSPENGI,MEMBER=TESTCOL0
//
```

#FJICNTL—Control file for JCL adjuster program (FSYMIG00)

#FJICNTL can be found in SYS1.SFSYEZTS. It is the control file that contains the parameters for the JCL adjuster program (FSYMIG00).

```
*-------------------------------------------------------------------*
* COPYRIGHT (C) 1989-2004, FOUNDATION SOFTWARE, INC.                *
* ALL RIGHTS RESERVED - PROPERTY OF FOUNDATION SOFTWARE, INC.       *
*                                                                   *
* 5697-I89 (C) COPYRIGHT IBM CORP 2001 - 2004.                      *
* ALL RIGHTS RESERVED. LICENSED MATERIAL - PROPERTY OF IBM.         *
* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP.             *
*                                                                   *
* REV. 03/14/2004               WRITTEN BY: SRECKO LAZANJA          *
*-------------------------------------------------------------------*
*                                                                   *
* THIS FILE PROVIDES PARAMETERS FOR BUILDING PARALLEL TEST JCL.     *
*                                                                   *
* USAGE: PENGIEZT/MU FSYMIG00 PROGRAM.                              *
*-------------------------------------------------------------------*
* SYNTAX CONVENTIONS FOR THIS PARAMETER FILE:                       *
*                                                                   *
* A "*" ON POSITION 1 DENOTES A COMMENT.                            *
* A "//" IN POSITION 1-2 DENOTES AN OPTION (PARAMETER)              *
* A BLANK LINE DENOTES THE END OF PARAMETER (OPTION) VALUE          *
* ANY OTHER VALUE POST PARAMETER ARE TAKEN AS PARAMETER VALUES      *
*                                                                   *
* HARD CODED SYMBOLS: (REPLACED BY FSYMIG00 PROGRAM)                *
*                                                                   *
* &JOBNAME - JOB NAME FROM FJIJOB0 FILE JOB STATEMENT               *
* &JOBACCT - ACCOUNTING INFORMATION FROM FJIJOB0 FILE JOB STATEMENT *
* &JOBUNAM - USER NAME INFORMATION FROM FJIJOB0 FILE JOB STATEMENT  *
* &STEPNAM - STEP NAME FROM FJIJOB0 FILE EXEC STATEMENTS            *
* &USERID  - TSO USER NAME THE JOB WAS SUBMITTED BY                 *
*                                                                   *
*-------------------------------------------------------------------*
* FSYMIG00 READS THIS PARAMETER FILE ALONG WITH THE USER SUPPLIED   *
* JCL (FROM FJIJOB0) AND POTENTIAL PROCS (FROM FJIPROC) AND CREATES *
* THE FOLLOWING:                                                    *
*                                                                   *
* 1. UPDATED JCL FOR PRODUCTION USE                                 *
* ---------------------------------                                 *
*    PUTS UPDATED JCL IN FJOJOB0 AND PROCS IN FJOPROC PDS. CHANGES  *
*    ARE MADE AS DEFINED IN THIS FILE FOR COMPILED AND LINK&GO      *
*    EASYTRIEVE JOB STEPS:                                          *
*                                                                   *
*    A) PRODUCT LOAD LIBRARIES                                      *
*       - "EZLOAD0" DSN IS REPLACED BY THE "CCL1LIB" LOADLIB        *
*         /=> THIS CHANGE AFFECTS ALL JCL STATEMENTS, INCLUDING PROC *
*             SUBSTITUTION ARGUMENTS.                               *
*                                                                   *
*    B) APPLICATION LOAD LIBRARIES                                  *
*       - "EJLOAD0" DSN IS REPLACED BY THE "FJLOAD0" LOADLIB        *
*         /=> THIS CHANGE AFFECTS ALL JCL STATEMENTS, INCLUDING PROC *
*             SUBSTITUTION ARGUMENTS.                               *
*                                                                   *
*    C) EASYTRIEVE PLUS PRODUCT MACRO LIBRARIES                     *
*       - EASYTRIEVE PLUS PRODUCT MACRO LIBRARIES ARE REPLACED BY   *
*         THE PENGIEZT/MU PRODUCT MACRO LIBRARY.                    *
*         /=> THIS CHANGE AFFECTS ALL JCL STATEMENTS, INCLUDING PROC *
*             SUBSTITUTION ARGUMENTS.                               *
*                                                                   *
*    D) EXEC PGN=EZTPA00 IS CHANGED TO EXEC PGM=FSYTPA00            *
*       EXEC PGN=EZTPA00X IS CHANGED TO EXEC PGM=FSYTPA00           *
*                                                                   *
*    E) DLI/IMS STEPS ARE ASSUMED TO BE THE "EXEC PGM=DFSRRC00"     *
*       /=> PARM=(DLI,EZTPA00..) IS CHANGED TO PARM=(DLI,FSYTPA00...) *
*           (FOR LINK AND GO JOBS)                                  *
*       /=> COMPILED PROGRAM NAMES ARE NOT CHANGED AS THEY ARE      *
*           FETCHED FROM APPLICATION LOADLIB USING THE SAME NAME.    *
*                                                                   *
* 2. UPDATED JCL FOR PARALLEL TEST                                  *
* --------------------------------                                   *
*    PUTS UPDATED JCL IN FJOJOB0. ALL PROCS ARE PUNCHED AS INSTREAM   *
*    PROCS, INCLUDING THE EXTERNAL PROCS. CHANGES ARE MADE AS DEFINED *
*    IN THIS FILE FOR COMPILED AND LINK&GO EASYTRIEVE PLUS JOB STEPS  *
*    AS FOLLOWS:                                                      *
*                                                                     *
*    NOTE: LINK&GO PROGRAMS ARE TRANSLATED IN FLIGHT TO OBTAIN OUTPUT *
*          FILE ATTRIBUTES VIA FJSYSP0 PENGIEZT/MU FSCCL1 STEP.       *
*                                                                     *
*    A) A NEW JOB STEP IS CREATED BEFORE THE ORIGINAL JOB STEP FOR    *
*       PENGIEZT/MU RUN AND MASSAGED AS FOLLOWS:                      *
*                                                                     *
*       PRODUCT LOAD LIBRARIES                                        *
*       - "EZLOAD0" DSN IS REPLACED BY THE "CCL1LIB" LOADLIB          *
*         /=> THIS CHANGE AFFECTS ALL JCL STATEMENTS, INCLUDING PROC  *
*             SUBSTITUTION ARGUMENTS.                                 *
*                                                                     *
*       APPLICATION LOAD LIBRARIES                                    *
*       - "EJLOAD0" DSN IS REPLACED BY THE "FJLOAD0" LOADLIB          *
*         /=> THIS CHANGE AFFECTS ALL JCL STATEMENTS, INCLUDING PROC  *
*             SUBSTITUTION ARGUMENTS.                                 *
*                                                                     *
*       EASYTRIEVE PLUS PRODUCT MACRO LIBRARIES                       *
*       - EASYTRIEVE PLUS PRODUCT MACRO LIBRARIES ARE REPLACED BY     *
*         THE PENGIEZT/MU PRODUCT MACRO LIBRARY.                      *
*         /=> THIS CHANGE AFFECTS ALL JCL STATEMENTS, INCLUDING PROC  *
*             SUBSTITUTION ARGUMENTS.                                 *
*                                                                     *
*       EXEC PGN=EZTPA00 IS CHANGED TO EXEC PGM=FSYMIG05              *
*       EXEC PGN=EZTPA00X IS CHANGED TO EXEC PGM=FSYMIG05             *
*       /=> COMPILED PROGRAM NAMES ARE NOT CHANGED AS THEY ARE        *
*           FETCHED FROM THE BOOTSTRP LOADLIB USING THE SAME NAME.    *
*                                                                     *
*       DLI/IMS STEPS ARE ASSUMED TO BE "EXEC PGM=DFSRRC00"           *
*       /=> PARM=(DLI,EZTPA00..) IS CHANGED TO PARM=(DLI,FSYMIG05...) *
*           (FOR LINK AND GO JOBS)                                    *
*       /=> COMPILED PROGRAM NAMES ARE NOT CHANGED AS THEY ARE        *
*           FETCHED FROM THE BOOTSTRP LOADLIB USING THE SAME NAME.    *
*                                                                     *
*       JOBLIB - NEW JCL STATEMENTS ARE ADDED TO THE JOBLIB:          *
*       - BOOTSTRP LIBRARY IS ADDED TO THE JOBLIB UNCONDITIONALLY     *
*            /=> BOOTSTRAP IS CONCATENATED FIRST IN CHAIN.            *
*       - CCL1LIB LIBRARY IS ADDED TO THE JOBLIB UNCONDITIONALLY      *
*            /=> CCL1LIB IS CONCATENATED SECOND IN CHAIN.             *
*                                                                     *
*       STEPLIB - NEW JCL STATEMENTS ARE ADDED TO THE STEPLIB OF EACH *
*       JOB STEP THAT EXECUTES A COMPILED EASYTRIEVE PLUS PROGRAM.    *
*            /=> BOOTSTRAP IS CONCATENATED FIRST IN CHAIN.            *
*            /=> CCL1LIB IS CONCATENATED SECOND IN CHAIN.             *
*                                                                     *
*       COMPILED AND LINK&GO EASYTRIEVE STEPS:                        *
*            /=> FJLOAD0 DD STATEMENTS ARE CREATED                    *
*                                                                     *
*    B) THE ORIGINAL JOB STEP IS MASSAGED AS FOLLOWS:                 *
*                                                                     *
*       JOBLIB (IF IT EXISTS) IS CONCATENATED TO THE STEPLIB.         *
*       /=> STEPLIB IS CREATED IF ONE DOES NOT EXIST.                 *
*                                                                     *
*       JCL STATEMENTS FOR OUTPUT DATA SETS ARE DISCARDED AND NEW JCL  *
*       WITH DATA SET NAMES ARE GENERATED.                            *
*                                                                     *
*    C) A JOB STEP FOR PARALLEL TEST COMPARE IS ADDED.               *
*                                                                     *
*    D) IEFBR14 STEP TO INITIALIZE PURGE JCL FILE IS ADDED BEFORE THE *
*       FIRST JOB STEP.  SEE "//INITIAL-STEP" OPTION IN THIS FILE.    *
*                                                                     *
*    E) A VSAM REPRO/IDCAMS STEP IS ADDED BEFORE EACH EASYTRIEVE      *
*       PLUS STEP, COMPILED OR LINK &GO, FOR VSAM FILES THAT ARE      *
*       UPDATED IN PLACE (OPENED IN I-O MODE). THIS STEP IS ADDED     *
*       IF THE INPUT VSAM FILE OPENED IN I-O MODE IS NOT CREATED      *
*       BY THE SAME STEP.                                             *
*---------------------------------------------------------------------*

*------------ FSYMIG00 PROGRAM OPTIONS FOLLOW -------------------------*
*---------------------------------------------------------------------*
* //PROD-COMPILERS                                                    *
* //TEST-COMPILERS                                                    *
* EASYTRIEVE PLUS AND PENGIEZT/MU COMPILER PROGRAM NAMES FOR          *
* PRODUCTION AND PARALLEL TEST (PARM=PROD OR PARM=TEST).              *
*                                                                     *
* &EZNAME=&MUNAME                                                     *
*         WHERE: &EZNAME = EASYTRIEVE PLUS COMPILER PROGRAM           *
*                &MUNAME = CONVERT TO PENGIEZT/MU COMPILER PROGRAM    *
*                                                                     *
* /=> SOME DATA CENTERS USE MORE THAN ONE EASYTRIEVE PLUS COMPILER    *
*     PROGRAM. EQUATE ALL NAMES TO FSYTPA00 BELOW FOR PRODUCTION, AND *
*     FSYMIG05 FOR PARALLEL TEST.                                     *
*                                                                     *
* USAGE: LINK & GO (PARALLEL TEST AND PRODUCTION)                     *
*---------------------------------------------------------------------*
* PRODUCTION COMPILERS TO BE USED
//PROD-COMPILERS
EZTPA00=FSYTPA00
EZTPA00X=FSYTPA00

* PARALLEL TEST COMPILERS TO BE USED.
//TEST-COMPILERS
EZTPA00=FSYMIG05
EZTPA00X=FSYMIG05
FSYTPA00=FSYMIG05

*---------------------------------------------------------------------*
* //INITIAL-STEP                                                      *
* FSYMIG00 PROGRAM INSERTS THIS INITIAL IEFBR14 JOB STEP BEFORE THE   *
* FIRST JOB STEP.                                                     *
*                                                                     *
* USAGE: AUTOMATED COMPARE OF PARALLEL TEST FILES.                    *
*        COMPILED PROGRAMS                                            *
*        LINK & GO PROGRAMS                                           *
*                                                                     *
* /=> THE PURPOSE OF THIS STEP IS TO CLEAR THE FILE INTO WHICH OUTPUT *
*     FILES PURGE JOB IS PLACED BY THE COMPARE PROGRAM.               *
*---------------------------------------------------------------------*
//INITIAL-STEP
//*<>------------------ IEFBR14 INITIAL STEP -------------------------*
* KEEP THE IEF#BR14 AS WRITTEN BELOW.
//IEF#BR14 EXEC PGM=IEFBR14
* THIS DSN MUST BE THE SAME AS FOR FJPURGE IN THE COMPARE-STEP BELOW.
//FJPURGE  DD UNIT=SYSDA,
//         SPACE=(TRK,(1,1),RLSE),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//         DISP=(MOD,DELETE,DELETE),
//         DSN=FSOFT01.MUCONV.PURGJCL.&JOBNAME
//*

*---------------------------------------------------------------------*
* //COMPARE-STEP                                                      *
* COMPARE STEP JCL TO BE EXECUTED FOR COMPARING EASYTRIEVE PLUS       *
* VS PENGIEZT/MU OUTPUT APPLICATION FILES.                            *
*                                                                     *
* USAGE: AUTOMATED COMPARE OF PARALLEL TEST FILES.                    *
*        COMPILED PROGRAMS                                            *
*        LINK & GO PROGRAMS                                           *
*                                                                     *
* /=> CODE JCL STATEMENTS FOR YOUR COMPARE PROGRAM, STARTING WITH     *
* "//&STEPNAM EXEC PGM=FSYMIG20,COND=EVEN,"                           *
* "           PARM=(&P1,&P2,&P3,&P4,&P5,&P6)"                         *
*                                                                     *
*     WHERE: &P1 = FILE PURGE OPTION                                  *
*                  PURGE   FILES ARE PURGED AFTER COMPARE. PURGE      *
*                          IS ACTIVATED FOR GOOD COMPARES ONLY.       *
*                  KEEP    FILES ARE KEPT (EVEN IF GOOD COMPARE)      *
*            &P2 = CONDITION CODE OPTION                              *
*                  NOLET   RETURN ERROR CODE                          *
*                  LET     RETURN RC=00 EVEN IF COMPARE ERRORS EXIST  *
*            &P3 = USER EXIT PROGRAM. CODE 1-8 CHARS PROGRAM NAME.    *
*                  THE DEFAULT IS NOEXIT.                             *
*            &P4 = COMPARE MODE FOR PRINTER FILES                     *
*                  BYTE    COMPARES BYTE BY BYTE                      *
*                  WORD    COMPARES WORD BY WORD SEPARATED BY SPACES  *
*            &P5 = PRINTER FILES CC HANDLING OPTION                   *
*                  EXPCC   REPLICATES LINES ACCORDING TO CONTROL CHAR.*
*                  NOEXPCC COMPARES LINE BY LINE AS FOUND ON INPUT FILE
*            &P6 = COMPARE ERROR LIMIT. THE DEFAULT IS 64.            *
*                  COMPARE TERMINATES WHEN THIS LIMIT IS REACHED      *
*            &P7 = JOB STATEMENT ACCOUNTING INFORMATION FOR PURGE JCL *
*                  MUST BE ENCLOSED IN QUOTES, EX: '(1234,N0000000)'  *
*                  THE "&JOBACCT" IS REPLACED BY FSYMIG00 WITH THE    *
*                  ACCOUNTING INFORMATION FOUND IN THE INPUT FJIJOB0  *
*                  FILE. &P7 IS AN OPTIONAL PARAMETER.                *
*            &P8 = JOB STATEMENT USER NAME FOR PURGE JCL.             *
*                  MUST BE ENCLOSED IN QUOTES, EX: 'JOHN-SMITH'       *
*                  THE "&JOBUNAM" IS REPLACED BY FSYMIG00 WITH THE    *
*                  USER NAME INFORMATION FOUND IN THE INPUT FJIJOB0   *
*                  FILE. &P8 IS AN OPTIONAL PARAMETER.                *
*                                                                     *
* SAMPLE EXIT "FSYXIT00" IS DISTRIBUTED WITH THE PRODUCT. THE SOURCE  *
* IS LOCATED IN SYS1.EZTSRC.                                          *
* /=> YOU CAN MAKE A COPY OF FSYXIT00 AND CUSTOMIZE IT TO YOUR NEEDS. *
* IN THAT CASE, CHANGE NOEXIT IN THE PARM BELOW TO YOUR PROGRAM NAME. *
*                                                                     *
*---------------------------------------------------------------------*
//COMPARE-STEP
//*<>------------------ FILE COMPARE STEP ----------------------------*
* KEEP THE &STEPNAM AS WRITTEN BELOW. CHANGE COND= AND PARM= AS NEEDED.
//&STEPNAM EXEC PGM=FSYMIG20,COND=EVEN,
//         PARM=(PURGE,NOLET,NOEXIT,BYTE,EXPCC,64,                     X
//         '&JOBACCT',                                                 X
//         '&JOBUNAM')
//SYSOUT   DD SYSOUT=*
//SYSLIST  DD SYSOUT=*
//HEXLIST  DD SYSOUT=*
//* NOTE: ERROR LOGS ARE DYNAMICALLY ALLOCATED BY FSYMIG00 PROGRAM
//* ----- AT RUN TIME. THE FILE NAMES IN ERROR ARE USED AS DDNAMES.
//*
//* COMPARE PROGRAM PUTS SUMMARY STATISTICS INTO THIS FILE
//FJSTATS  DD UNIT=SYSDA,
//         SPACE=(CYL,(5,20),RLSE),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//         DISP=(MOD,CATLG,CATLG),
//         DSN=FSOFT01.MUCONV.PARALLEL.FJSTATS
//*
//* TEMPORARY WORK FILE FOR PURGE JCL. DO NOT REMOVE.
//FJPURGW  DD UNIT=SYSDA,
//         SPACE=(CYL,(1,2),RLSE),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//         DISP=(NEW,PASS,DELETE),
//         DSN=&&FJPURGW
//*
//* COMPARE PROGRAM PUTS DELETE JCL INTO THIS FILE FOR DELETING
//* TEMPORARY FILES CREATED BY EASYTRIEVE PLUS AND PENGIEZT/MU STEPS.
```

```
* NOTE: &JOBNAME IS REPLACED BY THE JOBNAME OF FSYMIG00 PROGRAM.
//FJPURGE  DD UNIT=SYSDA,
//         SPACE=(CYL,(1,2),RLSE),
//         DCB=(LRECL=80,RECFM=FB,BLKSIZE=),
//         DISP=(MOD,CATLG,CATLG),
//         DSN=FSOFT01.MUCONV.PURGJCL.&JOBNAME
//*
//* WORK FILE OF TEMPORARY DATA SET NAMES CREATED BY PENGIEZT/MU AND
//* EASYTRIEVE PLUS STEPS. DATA SET NAMES TO BE COMPARED ARE OBTAINED
//* FROM THIS FILE. (LEAVE THIS STATEMENT AS IS).
//FJCOMP0  DD DSN=&&FJCOMP0,
//         DISP=(SHR,DELETE,DELETE)
//*
//*-------------------- END OF COMPARE STEP --------------------------*
//*

*-------------------------------------------------------------------*
* //FJIBOOT                                                         *
* DSN WHERE CONVERTED PROGRAMS BOOT STRAPS RESIDE, NORMALLY CREATED *
* WITH JCMUMIG1 OND JCMUMIG2 CONVERSION ENGINES.                    *
*                                                                   *
* USED TO INITIATE PARALLEL RUN FOR COMPILED PROGRAMS.              *
*                                                                   *
* /=> MULTIPLE LIBRARIES CAN BE LISTED                              *
*                                                                   *
* USAGE: COMPILED PROGRAMS (PARALLEL TEST ONLY)                     *
*-------------------------------------------------------------------*
//FJIBOOT
FSOFT01.MUCONV.BOOTSTRP.LOADLIB

*-------------------------------------------------------------------*
* //FJLOAD0                                                         *
* DSN WHERE CONVERTED PROGRAMS COBOL MODULES RESIDE.                *
*                                                                   *
* USAGE: COMPILED PROGRAMS (PARALLEL TEST AND PRODUCTION)           *
*                                                                   *
* /=> MULTIPLE LIBRARIES CAN BE LISTED                              *
*-------------------------------------------------------------------*
//FJLOAD0
FSOFT01.MUCONV.LOADLIB

*-------------------------------------------------------------------*
* //EJLOAD0                                                         *
* DSN WHERE EASYTRIEVE PLUS COMPILED APPLICATION PROGRAMS RESIDE.   *
*                                                                   *
* USAGE: COMPILED PROGRAMS (PARALLEL TEST)                          *
*                                                                   *
* /=> MULTIPLE LIBRARIES CAN BE LISTED                              *
*-------------------------------------------------------------------*
//EJLOAD0
*FSOFT01.FSYMG400.EJLOAD.LOADLIB
FSOFT01.FSYMG400.EZLOAD.LOADLIB

*-------------------------------------------------------------------*
* //CCL1LIB                                                         *
* TRANSLATOR LOAD LIBRARY (PENGIEZT/MU LOAD LIBRARY)                *
*                                                                   *
* USAGE: COMPILED PROGRAMS (PARALLEL TEST AND PRODUCTION)           *
*        LINK & GO (PARALLEL TEST AND PRODUCTION)                   *
*                                                                   *
* /=> MULTIPLE LIBRARIES CAN BE LISTED                              *
*-------------------------------------------------------------------*
//CCL1LIB
FSOFT01.FSYMG400.LOADLIB
CEE.SCEERUN

*-------------------------------------------------------------------*
```

```
* //EZLOAD0                                                           *
* EASYTRIEVE PLUS LOAD LIBRARY (EASYTRIEVE PLUS PRODUCT LIBRARY)      *
*                                                                     *
* USAGE: COMPILED PROGRAMS (PARALLEL TEST ONLY)                       *
*        LINK & GO (PARALLEL TEST ONLY)                               *
*                                                                     *
* /=> MULTIPLE LIBRARIES CAN BE LISTED                                *
*                                                                     *
* /=> THESE DSN'S ARE REPLACED BY THE 1ST CCL1LIB (CODED ABOVE)       *
*---------------------------------------------------------------------*
//EZLOAD0
FSOFT01.FSYMG400.EZLOAD.LOADLIB

*---------------------------------------------------------------------*
* //FJYMIG0                                                           *
* DSN WHERE TRANSLATOR GENERATED FJSYSP0 MEMBERS ARE LOCATED.         *
*                                                                     *
* USAGE: COMPILED PROGRAMS (PARALLEL TEST ONLY)                       *
*                                                                     *
* /=> THIS DSN MUST BE A PDS. FJSYSP0 IS NORMALLY PUNCHED BY THE      *
*     TRANSLATOR DURING THE TRANSLATING PROCESS.                      *
*                                                                     *
* /=> THE MEMBER NAME IS EXTRACTED FROM EACH JOB STEP BY FSYMIG00     *
*     PROGRAM AND ADDED TO THIS PDS TO MAKE IT:                       *
*     "//FJYMIG0  DD  DSN=&DSNAME(&MEMBER),DISP=SHR"                  *
*                                                                     *
* /=> ONLY ONE PDS CAN BE SPECIFIED                                   *
*---------------------------------------------------------------------*
//FJYMIG0
FSOFT01.MUCONV.FJSYSP0

*---------------------------------------------------------------------*
* //REMOVE-MACLIB                                                     *
* EASYTRIEVE PLUS PRODUCT MACRO LIBRARY(S) TO BE REMOVED FROM JCL.    *
*                                                                     *
* THESE ARE TOOLKIT AND SPECIAL MACROS THAT CAME WITH EASYTRIEVE      *
* PLUS PRODUCT (!! TOOLKIT MACLIB).                                   *
*                                                                     *
* /=> MULTIPLE LIBRARIES CAN BE LISTED. LIST ENDS WITH 1ST BLANK LINE.*
* /=> DSNS LISTED HERE ARE REPLACED BY MU'S &SYS1.SFSYCCLM LIBRARY.   *
*---------------------------------------------------------------------*
//REMOVE-MACLIB
*SYS1.EZTV62.MACLIB
FSOFT01.EZTPLUS.PANDD2.MACROS

*---------------------------------------------------------------------*
* //SPACE                                                             *
* SPACE TO BE ALLOCATED TO OUTPUT FILES FOR PARALLEL TESTING.         *
*                                                                     *
* THE SPACE STATEMENT CODED HERE IS USED FOR PRINTER AND THE OUTPUT   *
* FILES THAT DO NOT HAVE THE SPACE PARAMETER CODED IN THE JCL.        *
*---------------------------------------------------------------------*
//SPACE
SPACE=(CYL,(50,200),RLSE)

*---------------------------------------------------------------------*
* //MODEL                                                             *
*                                                                     *
* THE 1ST TWO QUALIFIERS TO BE USED FOR OUTPUT DATA SET NAMES.        *
*                                                                     *
* /=> THE OUTPUT DATA SET NAMES ARE GENERATED AS FOLLOWS:             *
*     DSN=&MODEL.&JOBID.&STEPNAME.&DDNAME.&QUAL                       *
*                                                                     *
* WHERE:  &MODEL    = THE MODEL YOU SUPPLY HERE                       *
*         &JOBID    = THE JOB NAME AS FOUND ON THE JOB STATEMENT      *
*         &STEPNAME = THE JOB STEP NAME AS FOUND IN THE JCL           *
*         &DDNAME   = THE FILE DDNAME                                 *
*         &QUAL     = 'F01' FOR PENGIEZT/MU CREATED FILES             *
*                     'E01' FOR EASYTRIEVE PLUS CREATED FILES         *
*                                                                     *
* /=> THE TOTAL STRING LENGTH OF &MODEL CANNOT EXCEED 13 CHARACTERS.  *
* /=> &USERID IS REPLACED BY THE TSO USER ID.                         *
* /=> YOU CAN HARD CODE A HIGH QUALIFIER INSTEAD.                     *
*---------------------------------------------------------------------*
//MODEL
* PLEASE NOTE: THE MODEL CANNOT EXCEED 13 CHARACTERS IN LENGTH.
* FSOFT01.TEMP
&USERID.TEMP

*----------------------- END OF PARAMETERS ---------------------------*
```

Notices

Trademarks

Index