Control/SE

Concepts and Facilities Guide

July, 2015

6416 Via De Albur Court
Suite 100
El Paso, TX 79912

P. (800) 252-1400
F. (915) 845-7918

Support@marblecomputer.com
www.marblecomputer.com

Table of Contents

Control/SE Overview

The Need for Control/SE

Managing Source Code for Mainframe Applications has become a difficult task. The shortage of experienced mainframe programmers has left organizations with a major dilemma with respect to maintaining mainframe Legacy Application code by either an offshore solution or with in-house COBOL programmers. Some organizations have opted to develop in-house COBOL programming talent via training courses at their internal "COBOL University".

This creates new problems in the maintenance of the code, the predominance of which is COBOL although Assembler and PL/1 as well as Java are routinely found on today's Mainframes.

Marble Computer believes that until there is a "Silver Bullet" solution to convert COBOL, Assembler and PL/1 to Java, the shortage of experienced mainframe programmers will cause organizations to:

- Induce retirees back to work albeit at higher costs.
- Set up In-House COBOL Universities or COBOL Training Courses
- Continue to look to Offshore Code Remediation companies that provide quality personnel, use efficient Life Cycle Tools and control costs.

IBM and other Independent Software Vendors (ISVs) have pushed efforts to move the coding off the mainframe through new Integrated Development Environments (IDEs) like Rational, Topaz and other products. The adoption rate has been slow and individual developer costs are high.

The preferred platform for developing and maintaining mainframe code is still the Interactive System Productivity Facility (ISPF). ISPF has been in use on mainframes since the release of IBM's Time Sharing option (TSO). Based on IBM studies the current ratio of mainframe developers using ISPF versus an IDE is 4:1. ISPF has been used by generations of COBOL programmers from the earliest days of mainframe computing. It's easy to learn, easy to use and integrated into TSO, thus ISPF is not going away.

Based on our customer surveys, we find that some organizations are investing in operating an in-house University to teach COBOL to their non-programming workforce. This is a more difficult task as these new COBOL programmers need to learn both an Editor and the COBOL Language. They especially need to understand the data usage and program logic. Without the proper tools this could be a difficult challenge. On the plus side, these new programmers are probably familiar with the application. They reside on site and thus can be better managed.

How Control SE Works

Control/SE creates a new powerful way to use the ISPF editor that makes it simple for the COBOL programmer to quickly understand the flow of the program as to data and process. Business rules are easily interpreted, data analysis is made simple, dead code is found and can be eliminated.

By utilizing Control/SE's simple command structure and operational command words programmers can issue an ISPF Find for Macro functions like Code-Not-Used or Forward-Tracing.

Documenting the complete flow of a single data field is available with one simple FIND command to show the complete flow (*and more*) on that data field.

Forward tracing shows the flow of the program logic through the Procedure Division.

The programmer can follow the use of nested performs through their looping functions.

Control/SE flags errors such as differences in Reserved Words, all the way back to COBOL 68 or COBOL 74. Programmers can bring older code up to COBOL Version 5.2 without the fear of introducing errors.

Marble Computer has created a simple command structure, with the addition of one character (An (&) ampersand character) added to the Find Command that works with a program's data names, paragraph names, and more. So the adoption rate is fast and efficient. Control/SE requires very little training.

By using the **&**, as in 'Find **&**Operand, you, the programmer are now in control of the code from the Data Division to the Procedure Division which includes all Record formats as well as all Copybooks.

No compile is necessary prior to using Control/SE. Control/SE produces an alternative compile listing (ACL) that makes the COBOL compiler listing obsolete and the ISPF Control/SE output is viewed under the ISPF editor.

The types of Program Analysis using the FIND &[Operand] are listed below.

1. Data Division Name
2. Performed Routine
3. CALL
4. COPY
5. CODE-NOT-USED
6. ERRORS
7. FORWARD-TRACING
8. HELP
9. INDEX
10. OPEN
11. PERFORM-ANALYSIS
12. PERFORMED-ROUTINES
13. Including COPY Members
14. Direct access to Other Field Narrative

Note: There is no space between the & and the Operand

Benefits of Control/SE

In 2011, a major tools software company commissioned a study of 520 CIOs from enterprise organizations to determine the business impact of the retiring mainframe workforce. Seventy-one percent of those surveyed stated they're concerned that the looming mainframe skills shortage will hurt their business. Specifically, they're concerned the skills shortage will result in increased application risk (58 percent), reduced productivity (58 percent) and project overruns (53 percent).

IBM surveys have shown that the Eclipse based IDE platforms available for editing and maintaining COBOL source code, have an adoption rate of only 20% among the programmer workforce when programming on a Mainframe. Therefore, the ratio of mainframe COBOL programmers using ISPF over existing IDEs is 4:1.

Therefore it is important to understand that the primary Editor on the Mainframe is ISPF. It has been used by generations of COBOL programmers. One can say the ISPF is part of a programmer's DNA.

With Control/SE, productivity is dramatically increased from the time the code is opened for remediation until it's ready for Alpha and Beta Testing. In our tests, the productivity gains of completing a standard set of remediation tasks can increase by 25-35%

Control/SE's command structure is built into the most common Editor on the mainframe, ISPF. There is little or no learning curve. The programmer is utilizing the same user interface, for COBOL, JCL and Data. ISPF has been in use for all of the programmer's working life.

While Eclipse and other IDEs offer some security protections, the security on the Mainframe is handled by the same software products that have been on the mainframe for generations of programmers. This is important, as a lot of an organization's business rules and logic are hard coded in these Legacy Applications. With offshore factories now maintaining a large portion of Mission Critical applications, security needs are more important than ever.

Control/SE provides management statistics so that the final code changes can be analyzed as to the quality of the program as well as the programmer. Programmer productivity and reduced Mean-Time-to-Repair are major advantages for Control/SE users. Project management is now able to deliver projects on time and under budget.

In the end, better code means less Job Failures which creates better utilization of the production environment, makes life easier for Job Schedulers and those that support Job Scheduling systems.

The Return on Investment (ROI) for Control/SE is quick and highly calculable. It's easy to install, just a set of program code modules that are compiled and placed in a load library. Much of the analytics are done in a batch process so to save CPU cycles rather than adding immense real-time overhead. Cost justifications are easy to develop through several recommended methodologies. Junior programmers quickly learn to understand the programs logic and flow thus keeping salary and overhead costs down and in budget.

Using Control/SE

The Control/SE Components

The Analysis Component

The Analysis Component is a batch job that is executed from ISPF via a standard set of JCL. It is creates an EDIT FILE for user editing. An Alternate Compile Listing (ACL) which is the COBOL Source plus the analysis of the Source is also created. The COBOL program does not go through a compiler, but the ACL does find most all of the compiler errors, if any, so that at Compile Time the program may pass that step.

The Analysis component is an upgraded version of Marble Computer's DCD product for COBOL documentation and analysis. DCD has a 38 year history of helping mainframe organizations maintain and enhance their mission critical applications. DCD has been through all the different version of ANSI COBOL and today supports COBOL Version 5.2. DCD, the Analysis Component does the grunt work, in batch, and builds the resulting ACL and the analysis information.

Example 1 as created by the Analysis Component:

F &FORWARD-TRACING

```
FORWARD TRACING
1    229     PROGRAM-ENTRY
2    265       C-BUILD-NARR-FILE-TO-MERGE
3    487       MERGE-THREE-FILES-TO-ONE
4    554         FILEIO-ERROR-ROUTINE  --> (4 Performs)
5    588           FILEIO-RTN-2 --> (Perform/VARYING)
6    612         N-SORT-INPUT-PROCEDURE
7    653           N220-LOOK-FOR-MATCH  --> (Perform/UNTIL)
                 FILEIO-ERROR-ROUTINE  --> (2 Performs)
8    892         P-SORT-OUTPUT-ROUTINE
9   1021       G-BUILD-OPEN-NARR-TO-MERGE
```

The FORWARD TRACING shows the flow of the Perform Logic in the program for a range of Performs or for all of the Perform routines.

Example 2 as created by the Analysis Component:

```
F &PERFORM-ANALYSIS

 PERFORM-ANALYSIS
  PERFORM Warnings & Major Errors
   Count   Type & Sequence Number(s)

   01  GO TOs leaving the range of a PERFORM      MAJOR PERFORM ERROR
    403

  PERFORM & GO TO activity
   Count   Type & Sequence Number(s)
   05   PERFORM SECTIONS
     345       381       452       512       842

   12   PERFORM PARAGRAPHS
     353       370       422       441       492       C4/12     531
     566       598       633       700       757

   04   GO TO Paragraphs
     384       403       502       822
```

The PERFORM-ANALYSIS shows PERFORM Warnings and Major Perform Errors, such as the one **MAJOR PERFORM ERROR** shown that leaves the range of a PERFORM.

The ISPF Interface

The ISPF Interface is accomplished when the user uses the ISPF EDIT FILE created by Control/SE. ISPF is the most widely use Interface for IT Specialists on a mainframe. Mostly used by Programmers the Interface has been in existence since the introduction of IBM's Time Sharing Option (TSO). It is an imbedded program product and supported by IBM. Millions of IT Specialist have used ISPF and compared to Integrated Development Environments (IDEs) has a 4:1 user advantage as the preferred editor for Mainframe Organizations.

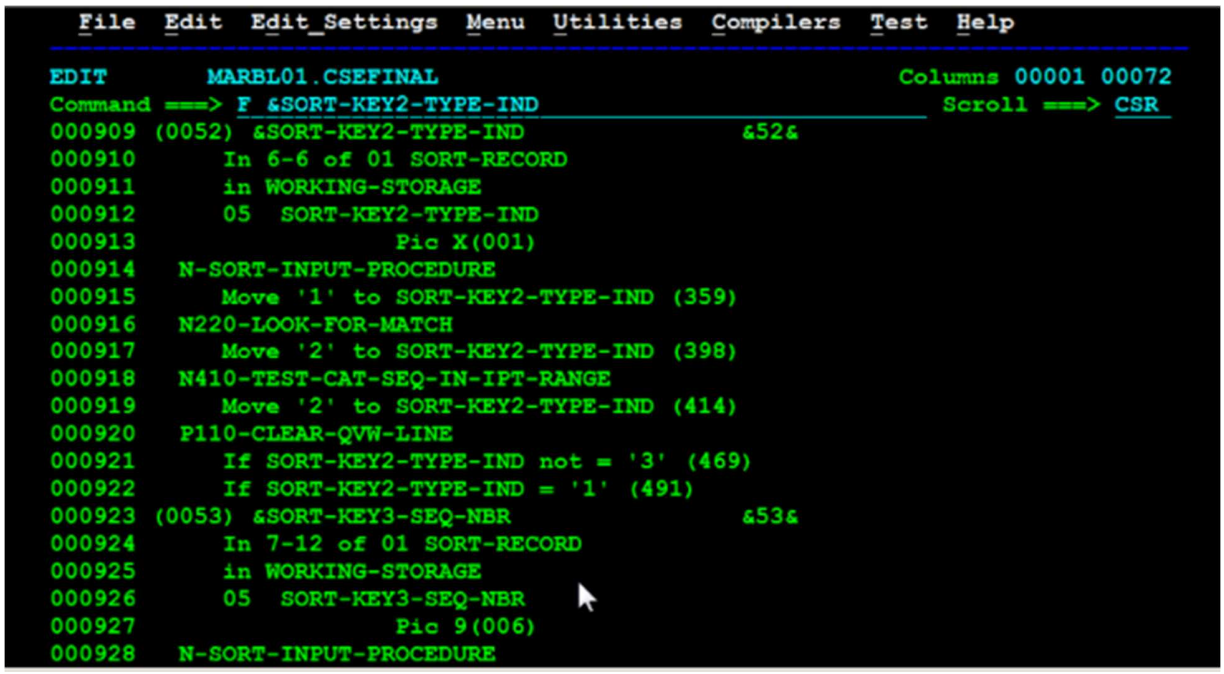
The ISPF Interface takes advantage of one of the ISPF Editors most used commands – FIND. The format for utilizing the power of Control/SE resides with this easy to use and easy to learn command structure:

[FIND &operand] Example: F &PERFORM-ROUTINES

The currently provided Operands are:

1. Data Division Name
2. Performed Routine
3. CALL
4. COPY
5. CODE-NOT-USED
6. ERRORS
7. FORWARD-TRACING
8. HELP
9. INDEX
10. OPEN
11. PERFORM-ANALYSIS
12. PERFORMED-ROUTINES
13. Including COPY Members
14. Direct access to other field Narrative

Example 1 of the User ISPF Interface and the output for Data Name Usage:



Example 2 of the User ISPF Interface and the output for CODE-NOT-FOUND:

The Programmer is able to view and edit the COBOL source within this Control/SE COBOL Edit File. As stated earlier there is no compile required to build the Edit File. Once all of the changes are completed the COBOL Source is returned to its source library or to Change Control products.

Once again, the **creation** of the EDIT FILE and **restoration** of modified COBOL Source is done through two simple PROCs or sets of JCL. The Program is now ready for a compile and save. While Control/SE, the Analysis Component, will catch most of the Compiler errors there are some that could get through and cause a compile error so we recommend a compile and save at least for the first remediation of an individual program.

Control/SE in 3 Easy Steps

1. **Build a COBOL Edit File** which adds analysis information to the COBOL program,
2. **Edit the COBOL program** using the Edit File,
3. **Run The ISOLATE program** to extract just the COBOL program for compiling and saving.

Build the COBOL Edit File

Building the COBOL Edit File:

- See **Sample JCL** shown below, or look for in-house directives, or consult Control SE User's Manual
- See **Control/SE EDIT FILE DEFAULTS** for established defaults
- See **Modifying User DEFAULTS** to change these defaults

Sample JCL

```
//NAME  JOB  ..
//STEP1  EXEC  CSEPROC
//CSE.COBOLIN  DD
//CSE.COPYLIB  DD  DSN=USER.COPYLIB,DISP=SHR
//CSE.VERSION  DD  *
ENTERPRISE              (or older IBM-COBOL, VS-COBOL-II, or VS-COBOL)
//CSE.USERSEL  DD  *
FORMAT 3                (Enter ADD, OMIT  or  FORMAT commands here )
ADD COPY=USERMBR1       (Example of how to bring in 1 COPY member for viewing )
```

Note: **The Quick Start Guide** contains all the Defaults for the COBOL Edit File.

Edit the COBOL Program

Using the ISPF Interface's FIND Command allows the programmer to successfully analyze and remediate the COBOL Source. The Operands are listed below with further explanation.

There are many different type of analysis **operands** available with the FIND command

1. **Data Division Field Name**
2. **Performed Paragraph or Section**
3. **CALL**
4. **COPY**
5. **CODE-NOT-USED**
6. **ERRORS**
7. **FORWARD-TRACING**
8. **HELP**
9. **OPEN**
10. **PERFORM-ANALYSIS**
11. **PERFORMED-ROUTINES**
12. **Cn/**
13. **n FIRST**

- When using on any of the above in a FIND, remember to include the preceding **&** ahead of the operand

More detail on the operands can be found in the following two sections:

- The Find &Operand Examples
- The Control/SE ISPF Examples

The ISOLATE Program

Run The ISOLATE program

```
//NAME  JOB  ..
//STEP1  EXEC  PGM=ISOLATE
//INPUT  DD  DSN=USER.CSEPDS(progname),DISP=SHR
//OUTPUT DD  DSN=USER.COBOLPDS(progname),DISP=SHR
//SYSOUT DD  SYSOUT=*
```

The ISOLATE program job stream is used to strip the analysis comments from the COBOL program and send it off to a source library or change management system from which it will than go to a compile step.

The Management Reports

The Management Reports on a given set of COBOL programs is available for management to review the quality of the COBOL code. Additionally these reports provide an insight as to the difficulty of the code and the flow sequences.

SUMMARY MANAGEMENT REPORT FOR (PERFORM ERRORs)

```
PERFORM Warnings & Major Errors
 Count   Type & Sequence Number(s)

  03   GO TOs leaving the range of a PERFORM         MAJOR PERFORM ERROR
   444       C4/9      C4/16

  03   Backward GO TO's                              MODERATE PERFORM ERROR
   444       C4/9      C4/16

PERFORM & GO TO activity
 Count   Type & Sequence Number(s)

  12   PERFORM SECTIONs
   244       250       259       293       295       299       321
   323       327       446       447       C4/8
  12   PERFORM Paragraphs
   334       376       390       402       420       426       430
   C4/15     458       474       492       532
  07   GO TO Paragraphs
   276       338       422       444       C4/9      C4/16     465
```

The Online Help Command

An Online Help command is built into the ISPF Editor for Control/SE. The format for help uses the same F &Operand format used in Control/SE and is:

```
F &HELP
```

The Help command covers all aspects of the product and is accompanied by a Quick Start Guide.

The following is an example of Control/SE's Help command:

FINDs are used to get to the ANALYSIS information made available.

Analysis FINDs in Control/SE require an &. See next line:

```
FIND &operand
```

What is different from other FINDs is the addition of the &

The operand following an & can be any of the following:

1. **Data-Field**
2. **Performed paragraph or section**
3. **CALL**
4. **COPY**
5. **CODE-NOT-USED**
6. **ERRORS**
7. **FORWARD-TRACING**
8. **HELP**
9. **INDEX**
10. **OPEN**
11. **PERFORM-ANALYSIS**
12. **PERFORMED-ROUTINES**

The next three entries are explained later in the HELP within ISPF or the Quick Start Guide.

13. **Cn/**
14. **n FIRST**
15. **Copy-Member-Name**

Find &Operand Examples

F &SORT-DATA-NAME (Find on Data-Name using **FORMAT 3** NARRATIVE)

```
(0106) &SORT-DATA-NAME                          <-- Narrative ID line
    In 39-68 of 01 SORT-RECORD                  <-- ..
    In WORKING-STORAGE                          <-- ... DD-ATTRIBUTES
        Data field is a GROUP item              <-- ..
  C100-BUILD-RECORDS                            <-- Performed Routine
      MOVE W1-DATA-NAME TO SORT-DATA-NAME (673) <--  P-D activity
  C200-ALT-BUILD                                <-- Performed Routine
      MOVE W1-DATA-NAME TO SORT-DATA-NAME (766) <--  P-D activity
  M860-GET-NEXT-SORT-RECORD                     <-- Performed Routine
      MOVE SORT-DATA-NAME TO CPY1-NAME (C2/32)  <--  P-D activity
```

- Use FORMAT 1 or use OMIT DD-ATTRIBUTES to remove **DD-ATTRIBUTES**
- Use FORMAT 2 or 3, or use ADD DD-ATTRIBUTES to add **DD-ATTRIBUTES**
- Use FORMAT 1 or 2, to remove **Performed Routine**
- **C2/32** in last **P-D activity** line refers to the 32nd line in the 2nd COPY member. If that COPY member is included with (**ADD COPY=member**), use (**F &C2/**) to go to code for 2nd COPY member.

F &FORWARD-TRACING

```
    #FORWARD TRACING    &FORWARD-TRACING
1   229    PROGRAM-ENTRY
2   265      C-BUILD-NARR-FILE-TO-MERGE
3   487      MERGE-THREE-FILES-TO-ONE
4   554        FILEIO-ERROR-ROUTINE  --> (4 Performs)
5   588          FILEIO-RTN-2 --> (Perform/VARYING)
6   612        N-SORT-INPUT-PROCEDURE
7   653          N220-LOOK-FOR-MATCH  --> (Perform/UNTIL)
(See #4)                    FILEIO-ERROR-ROUTINE  --> (2 Performs)
8   892        P-SORT-OUTPUT-ROUTINE
9  1021      G-BUILD-OPEN-NARR-TO-MERGE
```

- Forward tracing shows the structure of PERFORMs with in the program in **indented** format.
- Multiple PERFORMs **OR** different types of Performs, such as UNTIL, VARYING, or TIMES are noted.
- Sequence numbers on left show where the PERFORMed routine resides.
- **(See #4)** refers to a performed routine that has been performed before AND tracing further to **FILEIO-RTN-2** is not shown after the first occurrence of **FILEIO-ERROR-ROUTINE.**

F &PERFORM-ANALYSIS

```
#PERFORM-ANALYSIS   &PERFORM-ANALYSIS
PERFORM Warnings & Major Errors
 Count   Type & Sequence Number(s)

   01   GO TOs leaving the range of a PERFORM   MAJOR PERFORM ERROR
    403

PERFORM & GO TO activity
 Count   Type & Sequence Number(s)
  05   PERFORM SECTIONS
   345       381       452       512       842

  12   PERFORM PARAGRAPHS
   353       370       422       441       492       C4/12     531
   566       598       633       700       757

  04   GO TO Paragraphs
   384       403       502       822
```

- The first report in PERFORM-ANALYSIS shows PERFORM Warnings and Major Perform Errors, such as the one **MAJOR PERFORM ERROR** shown that leaves the range of a PERFORM

F &OPEN

```
  #OPEN REPORT       &OPEN REPORT
(0038)  &MAST-INP-FILE
   C-HANDLE-MAST-REC
      Open Input MAST-INP-FILE (338)
(0041)  &MAST-OUTPUT-FILE
   L-PUT-OUT-SELECTED-MAST-RECS
      Open Extend MAST-OUTPUT-FILE (742)
  End of FD OPEN & SORT Info
```

- All files that are used in an **OPEN** or a SORT **USING** or **GIVING** are shown with a reference to where they reside.
- In **Format 3**, the most recent Performed Routine name (e.g. **C-HANDLE-MAST-REC**) is shown ahead of the OPEN verb or SORT USING or GIVING clause.

F &INDEX

```
#INDEX              &INDEX
#COBOL PROGRAM     HAS      854 RECORDS
#HELP              HAS       18 RECORDS
#CALL REPORT       HAS       18 RECORDS
#COPY REPORT       HAS        7 RECORDS
#PERFORM-ANALYSIS  HAS       24 RECORDS
#COPY=FILEINFO     HAS       18 RECORDS    C1/..
#COPY=DDACLNBR     HAS       50 RECORDS    C2/..
#COPY=DCDPREC      HAS       39 RECORDS    C3/..
#OPEN REPORT       HAS        6 RECORDS
#NARRATIVE REPORT  HAS    1,969 RECORDS
#FORWARD TRACING   HAS       36 RECORDS
#CODE-NOT-FOUND    HAS       52 RECORDS
```

- The INDEX report shows a total of all possible records **available** by option in the COBOL PLUS file.

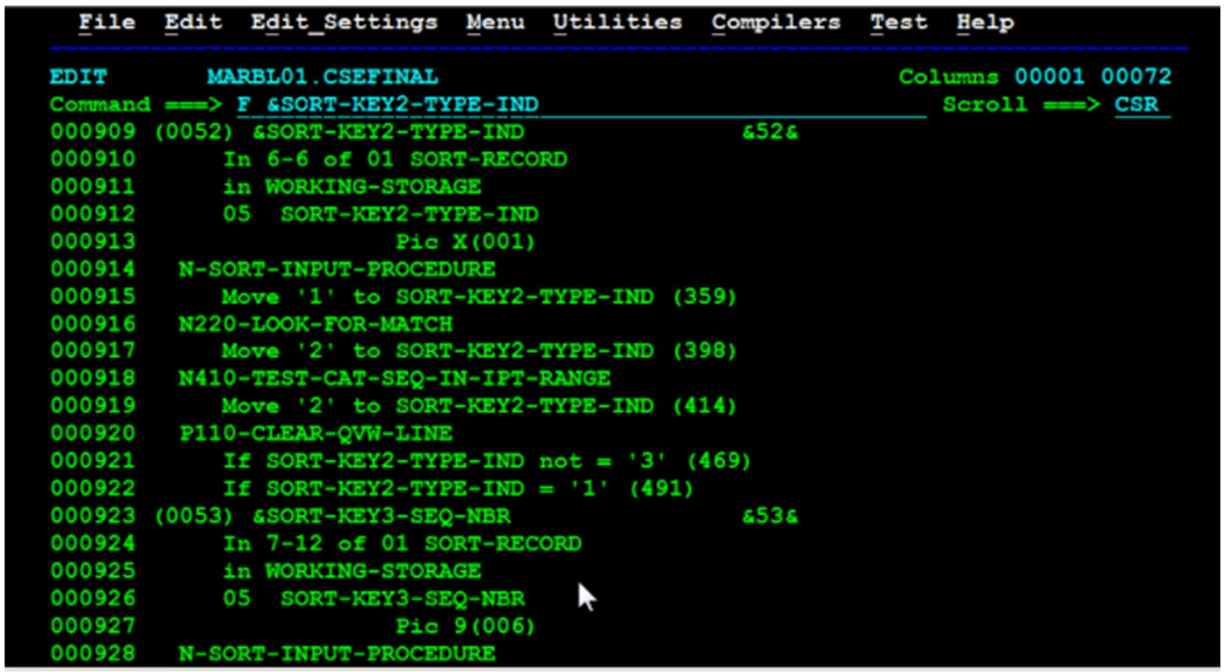
F &CODE-NOT-FOUND

```
 #CODE-NOT-FOUND     &CODE-NOT-FOUND
   &Unreferenced PARAGRAPHs and SECTIONs for TESTPROG
There are no unreferenced PARAGRAPHs or SECTIONs

   &Unused 01 Records and 77 Entries for Prog: TESTPROG
Seq Number         Lvl    Field Name
172                 01  WS-COUNT-SEQS
187                 01  SN1-RECORD
                Caution: The REDEFINES record after above record is USED!
201                 01  S1-RECORD

   &Unused Data Names for Prog: TESTPROG
Seq Number         Lvl    Field Name
(65)************01**WS-AW-RECORD*************** contains unused fields **
67                  10  WS-AW-SORT-PREFIX-ID
                         Indirectly Changed @65 Used @65
68                  10  WS-AW-SORT-POS-3
                         Indirectly Changed @65 Used @65
```

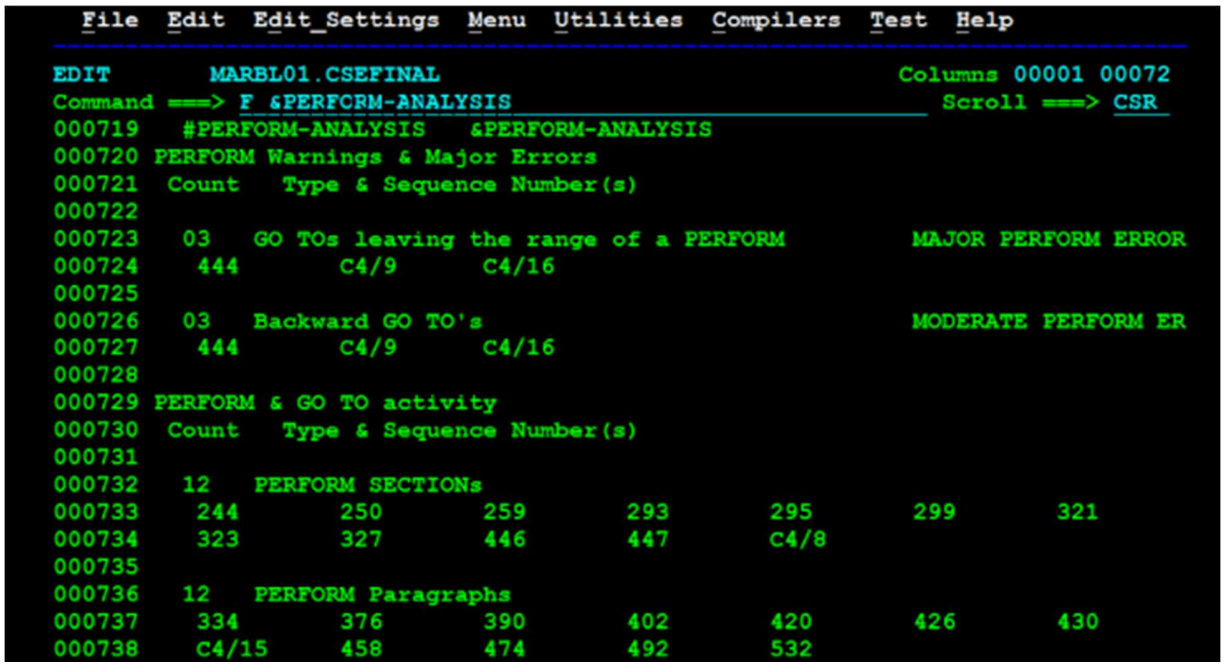
- These three reports show **DEAD** Procedure Division Code, **Unused** 01 records and **Unused** ***(non COPY Member)*** fields.
- In the 01 RECORD unused report, if a redefined record is used, it is reported.
- In the DATA FIELD unused report, INDIRECT References such as a move to whole record or redefines is shown.

Control/SE ISPF Examples

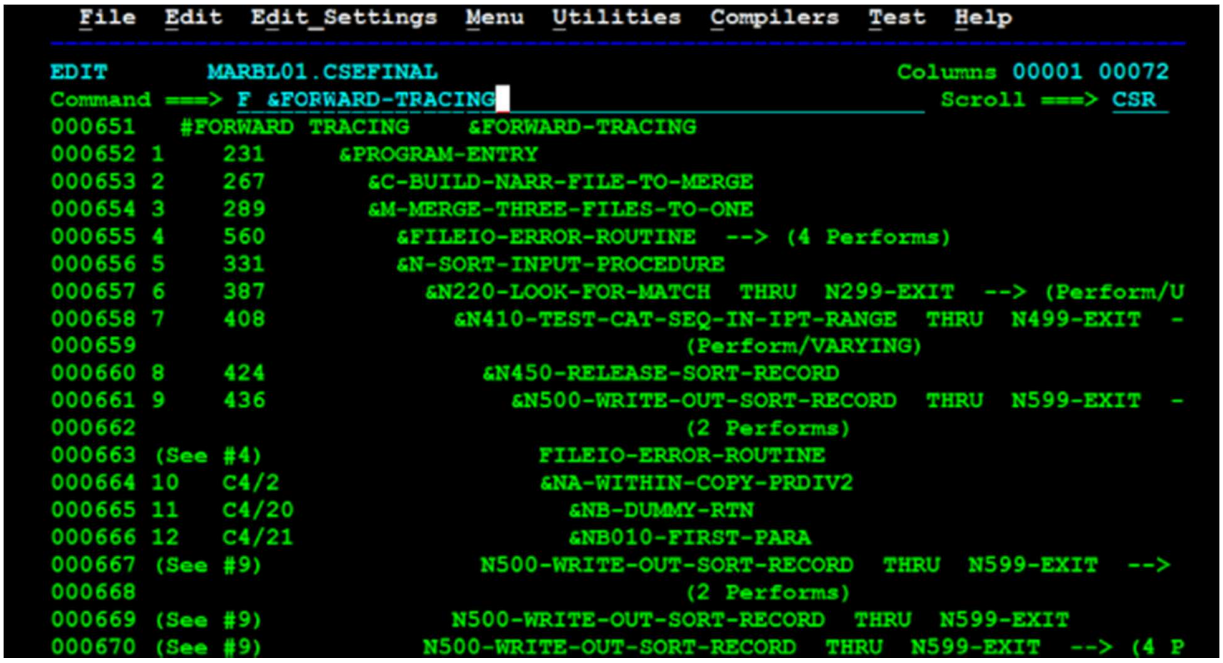
Data Field Analysis



Perform Analysis



Forward Tracing



Code Not Found