

CA-IDMS[®]

Dictionary Loader User Guide 15.0



Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

THIS DOCUMENTATION MAY NOT BE COPIED, TRANSFERRED, REPRODUCED, DISCLOSED, OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF CA. THIS DOCUMENTATION IS PROPRIETARY INFORMATION OF CA AND PROTECTED BY THE COPYRIGHT LAWS OF THE UNITED STATES AND INTERNATIONAL TREATIES.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

First Edition, December 2000

© 2000 Computer Associates International, Inc.
One Computer Associates Plaza, Islandia, NY 11749
All rights reserved.

All trademarks, trade names, service marks, or logos referenced herein belong to their respective companies.

Contents

| | |
|---|------|
| How to Use This Manual | v |
| Chapter 1. Introduction | 1-1 |
| 1.1 About the Dictionary Loader | 1-3 |
| 1.2 System Overview | 1-4 |
| 1.3 Dictionary Loader Capabilities | 1-7 |
| 1.4 Dictionary Loader Reports | 1-9 |
| Chapter 2. Program Processor | 2-1 |
| 2.1 About the Program Processor | 2-3 |
| 2.2 Input Requirements | 2-4 |
| 2.3 Output | 2-6 |
| 2.3.1 Management Summary Report | 2-6 |
| 2.3.2 Diagnostic Report | 2-7 |
| 2.3.3 File and Record Layouts Report | 2-10 |
| 2.3.4 DATA DIVISION Cross-Reference Report | 2-13 |
| 2.4 Parameter Statement | 2-15 |
| 2.5 Executing the Program Processor | 2-18 |
| Chapter 3. Cross Reference Processor | 3-1 |
| 3.1 About the Cross Reference Processor | 3-3 |
| 3.2 Overview | 3-4 |
| 3.3 Developing a File of Control Statements | 3-7 |
| 3.4 Filling in Worksheets | 3-10 |
| 3.5 Parameter Statement | 3-13 |
| 3.6 Title Statement | 3-17 |
| 3.7 Selection Statement | 3-18 |
| 3.8 Sample Control File | 3-19 |
| 3.9 System Data Cross-Reference Report | 3-20 |
| 3.10 Dictionary of Data Names Report | 3-23 |
| 3.11 Executing the Cross Reference Processor | 3-24 |
| Chapter 4. DDDL Generator | 4-1 |
| 4.1 About the DDDL Generator | 4-3 |
| 4.2 Overview | 4-4 |
| 4.3 Developing a File of Control Statements | 4-7 |
| 4.4 Parameter Statement | 4-9 |
| 4.5 VERSION Statement | 4-11 |
| 4.6 Grouping Statement | 4-12 |
| 4.7 Using the Grouping Statement | 4-14 |
| 4.8 Editing Generated DDDL Statements | 4-17 |
| 4.9 Executing the DDDL Compiler | 4-20 |
| Appendix A. Sample COBOL Input and DDDL Output | A-1 |
| A.1 About this Appendix | A-3 |
| Appendix B. Runtime Error Messages | B-1 |

| | |
|---|------------|
| B.1 About Runtime Error Messages | B-3 |
| B.2 Runtime Messages Issued by the Program Processor | B-4 |
| B.3 Runtime Message Issued by the Cross Reference Processor | B-6 |
| B.4 Runtime Messages Issued by the DDDL Generator | B-8 |
| Index | X-1 |

How to Use This Manual

What this Manual Contains

This manual provides the conceptual and operational information necessary to use the Dictionary Loader including:

- Syntax and job control language
- Considerations relating to using the Dictionary Loader effectively

How this Manual is Organized

This manual is divided into four chapters and two appendices:

- **Introduction** (Chapter 1)

An overview of the system architecture, system operation, and system capabilities.

- **Program Processor** (Chapter 2)

Details on the input, output, and operation of the Program Processor. This section includes syntax, sample reports, and execution JCL for operating the Program Processor under OS/390 and VSE/ESA.

- **Cross Reference Processor** (Chapter 3)

Details on the input, output, and operation of the Cross Reference Processor. This section includes syntax, sample reports, and execution JCL for operating the Cross Reference Processor under OS/390 and VSE/ESA.

- **DDDL Generator** (Chapter 4) Details on the input, output, and operation of the DDDL Generator. This section includes syntax, sample generated DDDL statements, a summary of the DDDL clauses generated, and execution JCL for operating the DDDL Generator under OS/390 and VSE/ESA.

- **Sample COBOL input and DDDL Output** (Appendix A)

Listing of three related COBOL programs and a listing of the DDDL statements generated by processing these programs.

- **Runtime Error Messages** (Appendix B)

Details on run-time messages issued by the Dictionary Loader.

Related Manuals

The following documents provide additional information relating to the information contained in this manual:

- *IDD DDDL Reference*
- *CA-IDMS Dictionary Structure Reference*

Chapter 1. Introduction

- 1.1 About the Dictionary Loader 1-3
- 1.2 System Overview 1-4
- 1.3 Dictionary Loader Capabilities 1-7
- 1.4 Dictionary Loader Reports 1-9

1.1 About the Dictionary Loader

Dictionary Loader populates the dictionary: The Dictionary Loader is a syntax converter used in conjunction with the Integrated Data Dictionary (IDD) to simplify the task of populating the dictionary with information contained in COBOL source programs. The Dictionary Loader processes a system of programs (that is, programs that process common files and records) individually and then collectively. This processing yields a collection of useful reports and the Data Dictionary Definition Language (DDDL) source statements (that is, ADD PROGRAM, ADD RECORD, and ADD FILE) needed to populate the dictionary with information about the programs.

What follows: To acquaint you with the Dictionary Loader, this chapter presents a system overview, a list of system capabilities, and a description of the reports the Dictionary Loader generates. Sections 2, 3, and 4 discuss the input, output, and operation of each of the three Dictionary Loader components separately.

1.2 System Overview

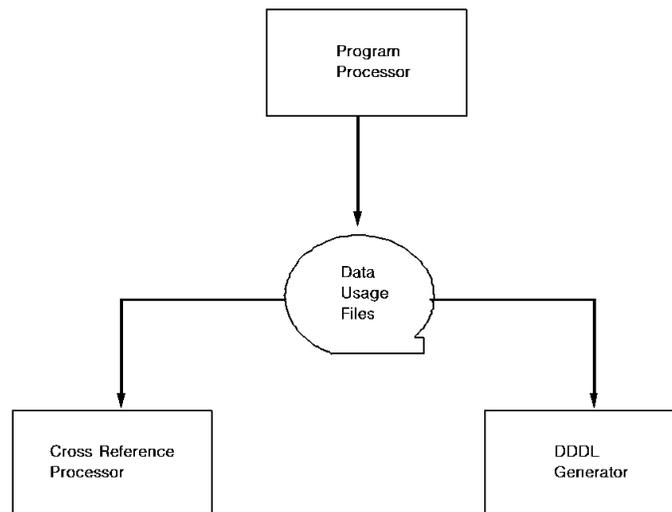
Dictionary Loader components: The Dictionary Loader consists of three components:

Program Processor: The Program Processor analyzes a single COBOL program and produces an intermediate file (data usage file) containing information about data usage within the program (for example, an element name and the source lines that refer to the name). A collection of data usage files (that is, one file for each COBOL program in a system of programs) is input to the DDDL Generator and optionally to the Cross Reference Processor.

Cross Reference Processor: The optional Cross Reference Processor analyzes a collection of data usage files and produces reports that aid in developing the file of control statements for running the DDDL Generator. Generally, the Cross Reference Processor is executed for a system of programs (for example, several programs that process the same file).

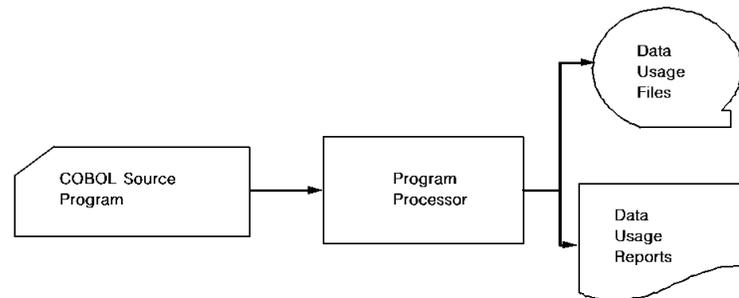
DDDL Generator: The DDDL Generator reads data usage files produced by the Program Processor and generates the appropriate DDDL source statements for subsequent input to the DDDL compiler.

Illustration of the components: The figure below illustrates how the three Dictionary Loader components are related:

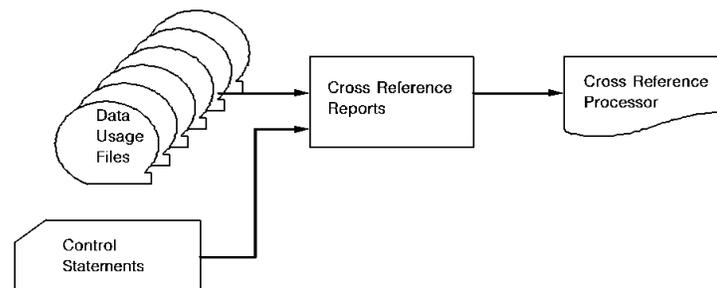


Function of the Program Processor: The Program Processor (PRANCOB) analyzes a single COBOL program. Output from this program is a set of reports and a data usage file. The reports and the file contain information about the way that the program uses data. The data usage file is used as input to the Cross Reference

Processor and the DDDL Generator. Note that the Program Processor is executed separately for each COBOL program in the system of programs to be processed. The functioning of the Program Processor is illustrated in the figure below:

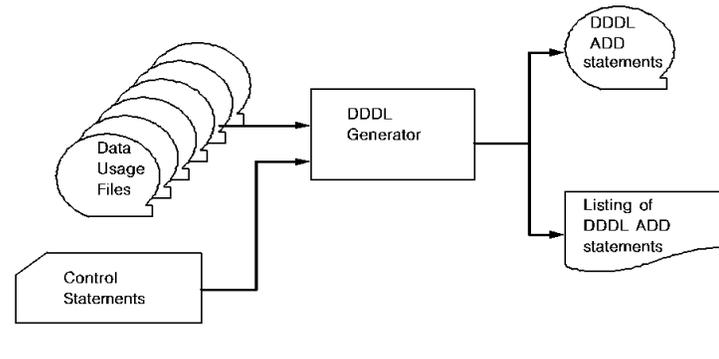


Function of the Cross Reference Processor: The Cross Reference Processor (PRANXREF) analyzes a collection of data usage files to track all references to data elements throughout a system of programs. Output from this component are reports that provide extensive cross-reference information (for example, data items and the source lines that refer to each item) about the system of programs being analyzed. The reports also aid in developing the control statements for running the DDDL Generator. You can bypass the Cross Reference Processor if you want to. The figure below illustrates the functioning of the Cross Reference Processor:



Function of the DDDL Generator: The DDDL Generator (PRANIDDG) reads a collection of data usage files and generates the appropriate DDDL source statements for input to the IDD DDDL compiler. Optional control statements can be used to specify a VERSION clause to be added to generated statements and to identify synonymous and nonunique names (that is, multiple names used to refer to the same file or record or single names used to refer to two or more different files or records). This module generates a file containing all DDDL ADD PROGRAM, ADD FILE, and ADD RECORD statements associated with the system of programs processed and produces a listing of all generated statements. The functioning of the DDDL Generator is illustrated in the figure below:

1.2 System Overview



1.3 Dictionary Loader Capabilities

The Dictionary Loader has the capabilities described below.

Generates DDDL statements: The Dictionary Loader can process a system of up to 99 COBOL programs to generate a file of DDDL statements that describe the programs and the files, records, and elements that the programs use. This file can be submitted to the DDDL compiler to populate the data dictionary.

Generates VERSION clauses: The Dictionary Loader adds VERSION clauses to all generated statements. If directed by a control statement, the DDDL Generator includes a user-specified VERSION clause in each generated statement; otherwise, the DDDL Generator includes a VERSION 01 clause in each statement.

Processes synonyms: The Dictionary Loader can identify synonyms within generated ADD statements. When a single file or record is referred to by many different names throughout the system of programs, the DDDL Generator can be directed to generate a SYNONYM clause within each ADD statement to identify all other names used to refer to the file or record.

Processes nonunique names: The Dictionary Loader can differentiate between multiple uses of the same name. When multiple files or records are referred to by a single name, the DDDL Generator can be directed to generate an ADD statement for each unique file or record, assigning each occurrence of the name of a separate version number (NEXT HIGHEST/NEXT LOWEST) or assigning all occurrences the same version.

Using NEXT HIGHEST/LOWEST: If NEXT HIGHEST/NEXT LOWEST is used in generating the DDDL statements with the DDDL Generator, the DDDL compiler will add all of the entities to the data dictionary, using the same name and differentiating one from another by the version number.

Using explicit version numbers: If all entities are assigned an explicit version number (that is, the same version number) during DDDL Generator processing, the DDDL compiler will process the statements in one of two ways depending on the setting of the DDDL compiler option DEFAULT IS ON/OFF:

| | |
|----------------|--|
| DEFAULT IS ON | The DDDL compiler will process the first ADD statement containing the nonunique entity-occurrence name and change subsequent ADD statements that use the name to MODIFY statements. |
| DEFAULT IS OFF | The DDDL compiler will process only the first ADD statement that contains the nonunique entity-occurrence name and will flag as erroneous all subsequent ADD statements that use the name. |

Editing the generated statements: You can edit the generated DDDL statements to eliminate unwanted ADDs, to establish different version numbers, or to merge several ADD statements that describe the same record or file into a single ADD statement.

1.4 Dictionary Loader Reports

Program Processor reports: The Program Processor produces four reports that are useful in analyzing the program, as follows:

| | |
|--------------------------------------|--|
| Management Summary Report | Lists the number of source lines in each division of the program, the number of diagnostic messages issued, and file usage information. The report aids in a quick assessment of the program's complexity, conformance to standard and file usage. |
| Diagnostic Report | Lists all source program lines found to contain a potential error condition. The report aids in identifying COBOL syntax errors, non-conformance to ANS standards, and logical errors that could not be detected by a COBOL compiler. |
| File and Record Layouts Report | Lists information about the attributes of each file and detail information about the data items within each record. The report aids in finding information about files and data items without having to refer to the source listing. |
| DATA DIVISION Cross-Reference Report | Lists all data items used in the program and all references to the data items made in the PROCEDURE DIVISION of the program. The report allows comprehensive tracking of the use of data items within the program. |

Cross-Reference Processor reports: The Cross-Reference Processor produces two reports that are useful in analyzing a collection of related programs as follows:

| | |
|------------------------------------|--|
| System Data Cross-Reference Report | Lists data items and references to the items for a system of programs. The report allows comprehensive tracking of the use of data items within the entire system of programs. |
| Dictionary of Data Names | Lists alphabetically, all data element and record names used in a system of programs together with extensive information about each item listed. This report aids in tracking the use of data names. |

Chapter 2. Program Processor

- 2.1 About the Program Processor 2-3
- 2.2 Input Requirements 2-4
- 2.3 Output 2-6
 - 2.3.1 Management Summary Report 2-6
 - 2.3.2 Diagnostic Report 2-7
 - 2.3.3 File and Record Layouts Report 2-10
 - 2.3.4 DATA DIVISION Cross-Reference Report 2-13
- 2.4 Parameter Statement 2-15
- 2.5 Executing the Program Processor 2-18

2.1 About the Program Processor

Description: The Program Processor processes a single COBOL source program and produces a data usage file and reports. This component is a full COBOL parser; it includes functional phases for reading, scanning, parsing, analyzing, sorting, and reporting on a COBOL source program. The Program Processor produces the following reports:

- The Management Summary Report
- The Diagnostic Report
- The File and Records Layout Report
- The DATA DIVISION Cross-Reference Report.

Data usage file: The data usage file produced is the input required for the Cross Reference Processor and the DDDL Generator. The Program Processor must be executed once for each program in the system of programs being processed.

What follows: This chapter describes the input requirements and the reports associated with the Program Processor and provides instructions for executing this component under OS/390 and VSE/ESA.

| Parameter | Default Option | Override Option |
|-----------------------|---|---|
| SYSREF/NOSYSREF | SYSREF — The data usage file is to be produced | NOSYSREF — The data usage file is not to be produced |
| SOURCE/NOSOURCE | NOSOURCE — The COBOL source program is not to be listed | SOURCE — The COBOL source program is to be listed |
| SUMM/NOSUMM | SUMM — The Management Summary Report is to be printed | NOSUMM — The Management Summary Report is not to be printed |
| DMAP/NODMAP | DMAP — The File and Record Layouts Report is to be printed | NODMAP — The File and Records Layout Report is not to be printed |
| DXREF/NODXREF | DXREF — The DATA DIVISION Cross-Reference Report is to be printed | NODXREF — The DATA DIVISION Cross-Reference Report is not to be printed |
| DIAG/NODIAG | DIAG — The Diagnostic Report is to be printed | NODIAG — The Diagnostic Report is not to be printed |
| ANS/ANS68/ANS74/NOANS | NOANS — ANS diagnostics are not to be included in the Diagnostic Report | ANS — All diagnostic messages are to be included in the Diagnostic Report ANS68 — Only ANS 1968 diagnostic messages are to be included in the Diagnostic Report ANS74 — Only ANS 1974 diagnostic messages are to be included in the Diagnostic Report |
| FLO/NOFLO | FLO — FLO diagnostic messages are to be included in the Diagnostic Report | NOFLO — FLO diagnostic messages are not to be included in the Diagnostic Report |
| NUM/NUM | NUM — The line numbers present in the source program are to be used for referencing | NONUM — Line numbers are to be assigned sequentially to all lines in the source program for referencing |

2.3 Output

Types of output: The Program Processor automatically produces the following output:

- Data Usage File
- Management Summary Report
- Diagnostic Report
- File and Record Layouts Report
- DATA DIVISION Cross-Reference Report

Overrides: Note that override processing options are available to suppress the output of the data usage file and any of the reports, and to request the inclusion of a source program listing (see 2.4, “Parameter Statement” on page 2-15 below).

Title page: Output from the Program Processor begins with a title page. The title page identifies the program and the date of the run, and supplies a table of contents listing all reports produced for the run. If a program listing has been requested, it appears after the Management Summary Report. Program Processor reports are discussed separately below.

2.3.1 Management Summary Report

Source program information: The Management Summary Report provides the following information about the source program:

- The number of source lines in each division of the program
- The number of diagnostic messages issued for each type of error
- File usage information for each file associated with the program

Sample report: This report aids in an overall assessment of the source program's complexity, conformance to standards, and file usage. A sample Management Summary Report appears below:

| FILE NAME | DEVICE | OPENED FOR: IN OU IO EX | RECORD LENGTH | BLOCK SIZE |
|---------------|------------------|----------------------------|------------------|---------------|
| CUSTOMER-FILE | UT-2400-S-CUSTIN | X | 104 | UNBLOCKED |
| RPTFILE | UT-S-SYSLST | X | 133 | UNBLOCKED |

2.3.2 Diagnostic Report

Lists incorrect source: The Diagnostic Report lists all source program lines found to contain a potential error condition. Each line listed is followed by a diagnostic message. The message identifies the problem portion of the COBOL statement with an asterisk (*), indicates the type of condition detected with a keyword indicator, and briefly describes the condition.

Sample report

| GEN-LN | SOURCE CARD | REMARKS |
|--------|---------------------------------|-----------------------|
| 130000 | MOVE SPACE TOO DETAIL-REC. * | (\$\$\$) SYNTAX ERROR |

Diagnostic Report messages: The Diagnostic Report lists three types of diagnostic messages:

Syntax (\$\$\$): One of the following three messages appears following the \$\$\$ indicator:

1. 'Character-string' NOT ALLOWED

The character string reported is a valid COBOL keyword or expression, but it cannot be used where it appears.

2. PROCEDURE NOT FOUND

The operand of the PERFORM statement is undefined.

3. SYNTAX ERROR

The word or construction does not conform to COBOL syntax rules.

ANS, ANS68, ANS74: The appropriate form of the following diagnostic message appears following the ANS, ANS68, or ANS74 indicators:

```
ANS/ANS-68/ANS-74 DOES NOT ALLOW 'keyword'
```

The keyword reported violates ANS 1968 standards for COBOL (ANS-68), 1974 standards (ANS-74), or both 1968 and 1974 standards (ANS).

Logical flow (FLO): One of the following messages appears following the FLO indicator:

1.

```
ALTER TO procedure-name IN PROCEDURE  
PERFORM procedure-name-1 THRU procedure-name-n
```

The ALTER statement causes the altered paragraph to transfer into the THRU range of a PERFORM procedure that does not contain the altered paragraph.

2.

```
ALTER TO procedure-name OUT OF PROCEDURE  
PERFORM procedure-name-1 THRU procedure-name-n
```

The ALTER statement sets the altered paragraph so that it will transfer out of the THRU range of the PERFORM procedure in which the altered paragraph resides.

3.

```
ALTERED PARAGRAPH NEVER REACHED
```

This paragraph is never reached when the program is executed. The paragraph is altered however, by a statement that can be reached.

4.

```
END OF PROC DIV REACHED
```

Program flow can fall through the end of the last paragraph of the PROCEDURE DIVISION. Program flow, should be ended by a STOP RUN statement.

5.

```
GO TO procedure-name IN PROCEDURE  
PERFORM procedure-name-1 THRU procedure-name-n
```

The GO TO statement resides outside the THRU range of the PERFORM procedure and transfers control to a paragraph inside the PERFORM procedure.

6.

```
GO TO procedure-name OUT OF PROCEDURE  
PERFORM procedure-name-1 THRU procedure-name-n
```

The GO TO statement transfers control out of the THRU range of the PERFORM in which the GO TO resides.

7.

PARAGRAPH NEVER REACHED

Program flow cannot reach this paragraph during execution of the program

8.

PERFORM EXIT BEFORE ENTRY

A statement of the form PERFORM procedure-name-1 THRU procedure-name-n has been found where the procedure-name-n precedes procedure-name-1 in the program.

9.

PERFORM RANGE OVERLAPS
PERFORM procedure-name-1 THRU procedure-name-n

The range of this PERFORM statement overlaps the range of PERFORM procedure-name-1 THRU procedure-name-n. Either the two names have a common entry or exit, or one range is not completely nested in the other.

10.

PROCEDURE EXIT NEVER REACHED

The procedure name in the statement flagged can never be reached at execution time. The name is referred to, however, by a statement of the form PERFORM procedure-name-1 THRU procedure-name-n. This message is also issued for a paragraph referred to by an ALTER statement of the form ALTER procedure-name-1 to procedure-name-2, where either procedure-name-1 or procedure-name-2 cannot be reached.

11.

REACHED FROM LAST PARA/SECT AND
PERFORM procedure name-1 THRU procedure-name-2

Program flow can reach this statement in either of the following ways:

- From the end of the preceding paragraph or as the first paragraph of a performed chapter.
- From a PERFORM statement that refers to this paragraph as the entry point of the performed procedure.

12.

SENTENCE NEVER REACHED

This sentence will never be reached during program execution.

13.

STATEMENT NEVER REACHED

This statement (within a sentence) will never be reached during program execution.

Types of problems flagged: Note that with the exception of two of the three syntax messages, Diagnostic Report messages identify problems that normal COBOL compilation might not flag. These problems fall into two categories as follows:

| | |
|---------------|---|
| Compatibility | ANS messages flag areas of potential compatibility in successfully compiled programs that might be run through another compiler. |
| Logical flow | FLO messages flag potential flaws in logic that could not be detected by the COBOL compiler. For example, FLO diagnostics can aid in identifying statements that can never be reached during execution. |

Syntax errors in compiled programs: Note that messages identifying syntax errors may be issued for programs that have compiled successfully. Such error messages usually identify minor differences in the syntax requirements enforced by the user's compiler and the Program Processor. For example, some compilers do not flag as erroneous COBOL statements that begin in column 8 instead of 12. The Program Processor flags such statements. If these syntax errors are not important to the user, they can be ignored.

2.3.3 File and Record Layouts Report

Describe file and record layouts: The File and Record Layouts Report is a six part report that provides information about the attributes of each file and specific details about the data items within each record:

- The first five parts of the report describe the five sections contained within the DATA DIVISION of a COBOL source program (that is, the FILE, WORKING-STORAGE, LINKAGE, COMMUNICATION, and REPORT sections).
- The sixth section of the report lists source statement references to all ACCEPT, DISPLAY, STOP, and CALL statements used the PROCEDURE DIVISION of the source program. This report allows quick access to information about files and data items without having to refer to the source listing.

Sample report: The figure below shows the first page of a sample File and Record Layouts Report.

| PRANDEM2 | FILE AND RECORD LAYOUTS (FILE SECTION) | DICTIONARY LOADER | 28 JAN 99 1425 | PAGE | 1 | |
|---------------------|--|-------------------|----------------|-------|-----|-------|
| FILE NAME: | CUSTOMER-FILE | | | | | |
| DEVICE NAME: | UT-2400-S-CUSTIN | | | | | |
| LABEL: | OMITTED | | | | | |
| BLOCK SIZE: | UNBLOCKED | | | | | |
| RECORD SIZE: | 104 CHARACTERS | | | | | |
| RECORD FORMAT: | FIXED | | | | | |
| | 079000 OPEN INPUT CUSTOMER-FILE | | | | | |
| | 084000 READ CUSTOMER-FILE RECORD | | | | | |
| | 125000 CLOSE CUSTOMER-FILE | | | | | |
| LV-DAT NAME | SRC LN | POS | SIZE | USAGE | OCC | VALUE |
| FD CUSTOMER-FILE | 037000 | | | | | |
| 01 CUSTOMER | 043000 | 1 | (104) | GROUP | | |
| 03 CUST-NUM | 044000 | 1 | 10 | DISP | | |
| 03 CUST-NAME | 045000 | 11 | 20 | DISP | | |
| 03 CUST-ADDRESS | 046000 | 31 | (40) | GROUP | | |
| 05 CUST-ADDR1 | 047000 | 31 | 20 | DISP | | |
| 05 CUST-ADDR2 | 048000 | 51 | (20) | GROUP | | |
| 06 CUST-CITY | 049000 | 51 | 15 | DISP | | |
| 06 CUST-ZIP-CODE | 050000 | 66 | 5 | DISP | | |
| 03 CUST-CREDIT | 051000 | 71 | 3 | DISP | | |
| 88 CUST-CREDIT-EXEC | 052000 | | | | | 'AAA' |
| 88 CUST-GOOD | 053000 | | | | | ' ' |
| 88 CUST-POOR | 054000 | | | | | 'XXX' |
| 03 FILLER | 055000 | 74 | 31 | DISP | | |

Field descriptions

FILE NAME

The file name.

DEVICE NAME

The device name assigned to the file.

LABEL

Information about LABEL records. The report displays the keywords OMITTED or STANDARD, or the name of a user LABEL record.

BLOCK SIZE

The size of the physical block, if blocked.

RECORD SIZE

The size of the file's data records.

RECORD FORMAT

The RECORDING MODE of the record. The report displays FIXED, VARIABLE, UNDEFINED, or SPANNED.

LV

The level number of the data item. For items for which level number is not applicable, codes provide information about the item:

- FD — File description
- SD — Sort description
- DC — Communication description
- RD — Report description

No level number is provided for definitions of index names used by the INDEXED BY clause.

DATA NAME

Name of the data item. DATA NAME can be a file name, record name, or an element name.

SRC LN

The line number of the source line where the data item is defined.

POS

Starting position associated with the data item.

SIZE

The size of the data item. Parentheses enclose a size reported for a group item.

USAGE

The form in which the data item is to be stored as the result of the source program's specifications:

- GROUP — The data item contains subordinate items.
- DISP — The data item is stored in character form.
- DISP-NM — The data item is stored one digit per character position. The PIC contains only S, 9, and V.
- NM-EDIT — The data item is a numeric item stored in character format. The PIC contains some or all of the editing characters +, -, z, \$, comma, B, CR, DB, ., or 0.

The following report writing specifications can also appear in this column:

- RH — Report heading
- RF — Report footing
- PH — Page heading
- PF — Page footing
- CH — Control heading
- CF — Control footing
- DE — Detail

OCC

The number of occurrences of the data item if the definition of the item uses an OCCURS clause.

VALUE

The value assigned to the data item if the definition of the item uses a VALUE clause.

2.3.4 DATA DIVISION Cross-Reference Report

Lists all program fields: The DATA DIVISION Cross-Reference Report provides an alphabetic listing of each data item included in the program and all references to the item in the PROCEDURE DIVISION of the program. The data item name is listed together with its attributes and the number of each source line that refers to the data name. This report allows comprehensive tracking of the use of data items.

Sample report

| PRANDEM2 DATA DIVISION CROSS REFERENCE | | | | DICTIONARY LOADER | | 28 JAN 99 1425 | PAGE 6 | |
|--|-------------------|--------|------|-------------------|--|----------------|---|---------------|
| LV | DATA-NAME | SRC-LN | SIZE | OCC | QUALIFICATION | REF-LN | STATEMENT | REF-LINE-NBRS |
| 03 | CUST-ADDRESS | 046000 | 40 | | (CUSTOMER-FILE) CUSTOMER | | | |
| 05 | CUST-ADDR1 | 047000 | 20 | | (CUSTOMER-FILE) CUSTOMER CUST-ADDRESS | 092000 | MOVE CUST-ADDR1 TO RPT-ADDR1 | 047000 030000 |
| 05 | CUST-ADDR2 | 048000 | 20 | | (CUSTOMER-FILE) CUSTOMER CUST-ADDRESS | 093000 | MOVE CUST-ADDR2 TO RPT-ADDR2 | 048000 032000 |
| 06 | CUST-CITY | 049000 | 15 | | (CUSTOMER-FILE) CUSTOMER CUST-ADDRESS | | | |
| 03 | CUST-CREDIT | 051000 | 3 | | (CUSTOMER-FILE) CUSTOMER | | | |
| 88 | CUST-CREDIT-EXEC | 052000 | | | (CUSTOMER-FILE) CUSTOMER CUST-CREDIT | 087000 | IF NOT CUST-CREDIT-EXEC | 052000 |
| 88 | CUST-CREDIT-GOOD | 053000 | | | (CUSTOMER-FILE) CUSTOMER CUST-CREDIT | | | |
| 88 | CUST-CREDIT-POOR | 054000 | | | (CUSTOMER-FILE) CUSTOMER CUST-CREDIT | | | |
| 03 | CUST-NAME | 045000 | 20 | | (CUSTOMER-FILE) CUSTOMER | 091000 | MOVE CUST-NAME TO RPT-CUST-NAME | 045000 028000 |
| 03 | CUST-NUM | 044000 | 10 | | (CUSTOMER-FILE) CUSTOMER | 090000 | MOVE CUST-NUM TO RPT-CUST-NO | 044000 026000 |
| 06 | CUST-ZIP-CODE | 050000 | 5 | | (CUSTOMER-FILE) CUSTOMER CUST-ADDRESS CUST-ADDR2 | 094000 | MOVE CUST-ZIP-CODE TO RPT-ZIP | 005000 034000 |
| 01 | CUSTOMER | 043000 | 104 | | | | | |
| FD | CUSTOMER-FILE | 037000 | | | | 079000 | OPEN INPUT CUSTOMER-FILE | 037000 |
| | | | | | | 084000 | READ CUSTOMER-FILE RECORD | 037000 |
| | | | | | | 125000 | CLOSE CUSTOMER-FILE | 037000 |
| 01 | DETAIL-REC | 024000 | 133 | | | 081000 | MOVE SPACES TO DETAIL-REC | 024000 |
| | | | | | | 105000 | WRITE DETAIL-REC AFTER POSITIONING POSITION-IND-WS | 024000 061000 |
| | | | | | | 106000 | MOVE SPACES TO DETAIL-REC | 024000 |
| 01 | PAGE-COUNT-WS | 060000 | 2 | | | 107000 | ADD PAGE-INCREMENT-WS TO PAGE-COUNT-WS | 062000 060000 |
| | | | | | | 108000 | IF PAGE-COUNT GREATER THAN +58 | 060000 |
| | | | | | | 118000 | MOVE +4 TO PAGE-COUNT-WS | 060000 |
| 01 | PAGE-INCREMENT-WS | 062000 | 1 | | | 104000 | MOVE 1 TO PAGE-INCREMENT-WS | 062000 |
| | | | | | | 107000 | ADD PAGE-INCREMENT-WS TO PAGE-COUNT-WS | 062000 060000 |

Field descriptions

LV

The level number of the data item. For items for which level number is not applicable, codes provide information about the item:

- FD — File description
- SD — Sort description
- DC — Communication description
- RD — Report description

No level number is provided for definitions of index names used by the INDEXED BY clause.

DATA-NAME

Name of the data item. DATA NAME can be a file name, record name, or an element name.

SRC-LN

The line number of the source line where the data item is defined.

SIZE

The size of the data item. Parentheses enclose a size reported for a group item.

OCC

The number of occurrences of the data item if the definition of the item uses an OCCURS clause.

QUALIFICATION

The name(s) of other data item(s) to which the subject data item is subordinate. The file name is enclosed by parentheses. Highest level qualifiers (for example, files) are listed first, followed by record names. The minimum qualification needed to make the name unique is flagged with an asterisk (*). If there are two identical data names at the same level in the same structure, those data names cannot be uniquely identified; a *\$\$\$ diagnostic will appear in the listing.

REF-LN STATEMENT

A list of statement (including starting source line numbers) that refer to the data item.

REF-LN-NBRS

The source line number where each data item in the REF-LN STATEMENT entry is defined. REF-LN-NBRS are reported for all data items (including the subject item) in order of occurrence in the statement.

2.4 Parameter Statement

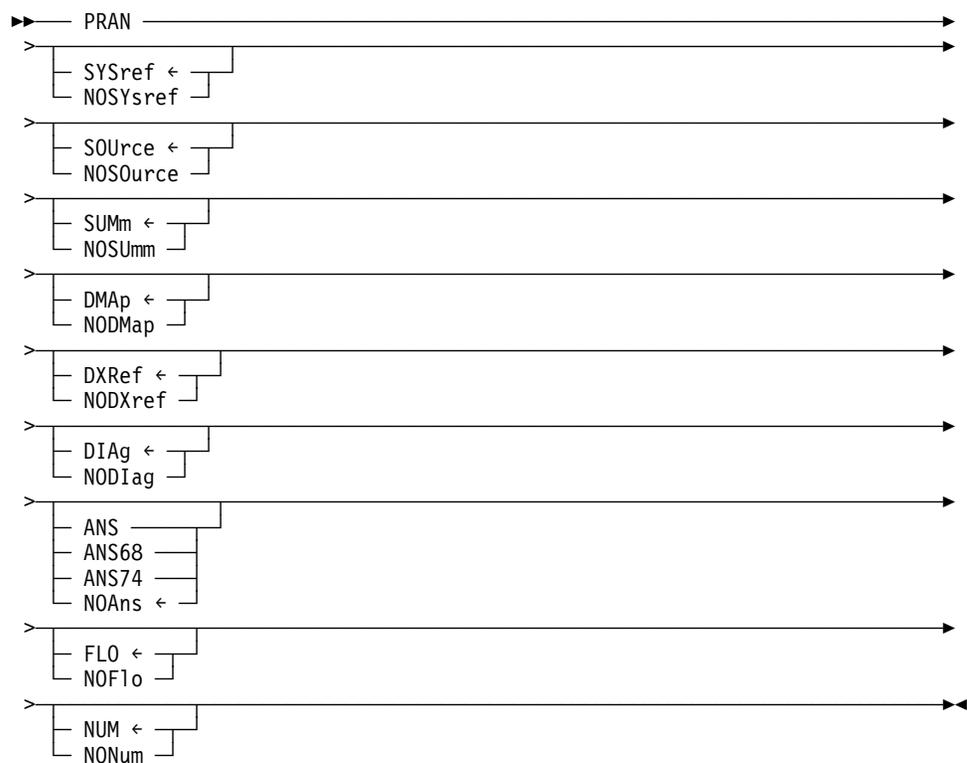
Specifies overrides to Program Processor

The parameter statement specifies override processing options for the Program Processor. Under VSE/ESA, this statement must be used to specify options; under OS/390, this statement can be used but it is usually more convenient to specify options in the JCL in the PARM clause of the EXEC statement.

Coding rules: The following rules apply to coding parameter statements for the Program Processor:

- Parameter statements, if used, must be included at the beginning of the COBOL source program.
- Multiple statements can be entered.
- Statements can be coded in positions 1 through 72.
- Options can be specified in any order, with one or more options per statement and at least one blank or comma between specifications.

Syntax



Parameter list

PRAN

Identifies the statement. Note that this keyword must be used to distinguish this statement from COBOL source statements.

SYSref/NOSYsref

Specifies whether the data usage file is to be produced as follows:

- SYSREF (default) — The file is to be produced.
- NOSYSREF — The file is not to be produced.

SOUrce/NOSOUrce

Specifies whether the COBOL source program is to be listed in the output, as follows:

- SOURCE — The source program is to be listed.
- NOSOURCE (default) — The source program is not to be listed.

SUMm/NOSUMm

Specifies whether the Management Summary Report is to be printed, as follows:

- SUMM (default) — The report is to be printed.
- NOSUMM — The report is not to be printed.

DMAp/NODMAp

Specifies whether the File and Record Layouts Report is to be printed, as follows:

- DMAP (default) — The report is to be printed.
- NODMAP — The report is not to be printed.

DXRef/NODXref

Specifies whether the DATA DIVISION Cross-Reference Report is to be printed, as follows:

- DXREF (default) — The report is to be printed.
- NODXREF — The report is not to be printed.

DIAg/NODIag

Specifies whether the Diagnostic Report is to be printed, as follows:

- DIAG (default) — The report is to be printed.
- NODIAG — The report is not to be printed.

ANS/ANS68/ANS74/NOAns

Specifies the type of errors to be reported in the Diagnostic Report, as follows:

- ANS — Violations of both the 1968 and 1974 ANS standards are to be reported.
- ANS68 — Only violations of the 1968 ANS standards are to be reported.
- ANS74 — Only violations of the 1974 ANS standards are to be reported.
- NOANS (default) — No ANS violations are to be reported.

FLO/NOFlo

Specifies whether FLO diagnostics are to be reported in the Diagnostic Report, as follows:

- FLO (default) — FLO diagnostics are to be reported.
- NOFLO — FLO diagnostics are not to be reported.

NUM/NONum

Specifies whether the original line numbers present in the COBOL source program are to be used in reports to refer to source statements, as follows:

- NUM (default) — The line numbers already associated with source statements are to be used in reports to refer to source statements
- NONUM — Line numbers are to be assigned sequentially to all source statements, and these new line numbers are to be used in reports to refer to source statements.

2.5 Executing the Program Processor

JCL for executing the Program Processor under OS/390 and VSE/ESA is shown below. Under VSE/ESA, processing options must be specified with the parameter statement. Under OS/390, although the parameter statement can be used, it is usually easier to specify options by using the PARM clause of the EXEC statement.

OS/390 JCL — PRANCOB

```
//PRANCOB EXEC PGM=PRANCOB,REGION=1024K,PARM='parameter options'
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//PRANLIB DD DSN=user.copylib,DISP=SHR < Include only if program contains COBOL
// COPY statements
//PRANREF DD DSN=reflib(member-name),DISP=OLD < Include only if using LIBRARY option
//PRANREF DD DSN=sysref,DISP=((NEW,catlg), < Include only if using DISK option
// UNIT=disk,VOL=SER=nnnnnn,
// SPACE=(trk,(10,10),rlse),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120
//PRANWRK DD UNIT=disk,SPACE=(cyl,(5,5))
//dcmsg DD DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrn DD *
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
dmc1=dmc1-name
Insert other SYSIDMS parameters as appropriate
//SYSIPT DD *
Insert COBOL source statements
```

| | |
|---------------------|--|
| idms.dba.loadlib | Data set name of the load library containing the DMCL and database name table load modules |
| idms.loadlib | Data set name of the load library containing CA-IDMS system software modules |
| BLKSIZE=3120 | block size of data usage file; must be multiple of 80 |
| catlg | disposition of new file: CATLG, PASS or KEEP |
| cyl,(5,5) | file space allocation of work file |
| disk | symbolic device name of disk file |
| nnnnnn | serial number of disk volume |
| parameter options | options associated with the Parameter statement for the Program Processor. Multiple options can be specified; keywords must be separated by blanks or commas; the entire entry must be enclosed in single quotes. Note that the keyword PRAN shown in the syntax for the parameter statement must not be included with options specified here. |
| reflib(member-name) | data set name of data usage file |
| sysref | data set name of data usage file |

| | |
|----------------------|--|
| trk,(10,10),rlse | file space allocation of data usage file |
| user.copylib | data set name of COBOL copy book library |
| dcmsg | DDname of the system message (DDLDCMSG) area |
| idms.sysmsg.ddldcmsg | Data set name of system message (DDLDCMSG) area |
| SYSIDMS | DDname of the CA-IDMS parameter file specifying runtime directives and operating system-dependent parameters. ▶▶For a description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> . |

Note: Note that the larger the value specified in the REGION parameter, the more efficiently the Program Processor will run.

▶▶The DISK option and LIBRARY option are documented in Chapter 3, “Cross Reference Processor” on page 3-1.

VSE/ESA JCL — PRANCOB

```
// DLBL    SSLn,'user.srclib'
// EXTENT  ,nnnnnn
// LIBDEF  SL,SEARCH=SSLn,TEMP
// DLBL    PRANREF,'sysref',2099/365,SD
// EXTENT  SYS010,nnnnnn,1,,ssss,200
// ASSGN   SYS010,DISK,VOL=nnnnnn,SHR
// DLBL    PRANWRK,'pranwork',0,SD
// EXTENT  SYS011,nnnnnn,1,ssss,300
// ASSGN   SYS011,DISK,VOL=nnnnnn,SHR
// EXEC    PRANCOB,SIZE=512K
parameter statements(s)
=COPY IDMS member statement or COBOL source statements
/*
```

| | |
|-------------|---|
| nnnnnn | serial number of disk volume |
| pranwork | file-id for work file |
| ssss | starting track (CKD) or block (FBA) of disk extent |
| sysref | file-id for sequential file containing data usage file |
| SYS010 | logical unit assignment for data usage file (SYS010 required) |
| SSLn | filename of source statement library |
| SYS011 | logical unit assignment for work file (SYS011 required) |
| user.srclib | source statement library containing data usage files |

Note: The keyword PRAN must appear at the beginning of each parameter statement. PRAN is only used in the parameter statement for this component.

Note: The Program Processor must run in a partition that is at least 512 K. The larger the partition size, the more efficiently the Program Processor will run.

JCL for VSE/ESA source statement library: The optional JCL shown below places the data usage file generated by the Program Processor into a source statement library. From the source statement library, data usage files can be accessed by the Cross Reference Processor and the DDDL Generator.

If the source statement library option is to be used, add this JCL to the JCL for executing the Program Processor, shown above.

```
// DLBL    IJSYSIN,'sysref'  
// EXTENT  SYSIPT,nnnnnn  
// ASSGN   SYSIPT,DISK,VOL=nnnnnn,SHR  
// DLBL    SSLn,'user.src1ib'  
// EXTENT  ,nnnnnn  
// LIBDEF  SL,T0=SSLn,TEMP  
// EXEC    LIBR  
// CLOSE   SYSIPT,SYSRDR
```

Note that the output is placed in the X. sublibrary.

| | |
|------|--------------------------------------|
| SSLn | filename of source statement library |
|------|--------------------------------------|

Chapter 3. Cross Reference Processor

- 3.1 About the Cross Reference Processor 3-3
- 3.2 Overview 3-4
- 3.3 Developing a File of Control Statements 3-7
- 3.4 Filling in Worksheets 3-10
- 3.5 Parameter Statement 3-13
- 3.6 Title Statement 3-17
- 3.7 Selection Statement 3-18
- 3.8 Sample Control File 3-19
- 3.9 System Data Cross-Reference Report 3-20
- 3.10 Dictionary of Data Names Report 3-23
- 3.11 Executing the Cross Reference Processor 3-24

3.1 About the Cross Reference Processor

Tracks all references to data items: The Cross Reference Processor analyzes a collection of data usage files to track all references to data elements throughout a system of COBOL programs. Control statements assign a descriptive title to each subset of records to be reported together (most commonly a file), specify the 01-level records that are to be associated with each title, and specify processing options.

Output: Output from this module are two reports that provide extensive cross-reference information about the system of programs: the System Data Cross-Reference Report and the Dictionary of Data Names Report. These reports aid in developing control statements for the DDDL Generator.

What follows: This chapter presents an overview of the Cross Reference Processor, describes its control statements and reports, and provides instructions for executing the Cross Reference Processor under OS/390 and VSE/ESA.

3.2 Overview

Purpose of the processor: The main purpose of the Cross-Reference Processor is to produce the System Data Cross-Reference Report. The control statements associated with running this component allow the user to specify the organization of the information to be included in this report as follows:

- **Group information about a file that has many different names.**

Information about a file that has many different names can be grouped under one descriptive title. A single file (for example, a transaction file) may be named differently (for example, TRANSFILE, TRANS-IN, TRANS-OUT) in the system of programs. Control statements can be used to assign a descriptive name to such a file and to connect the appropriate descriptions from specific programs to that name.

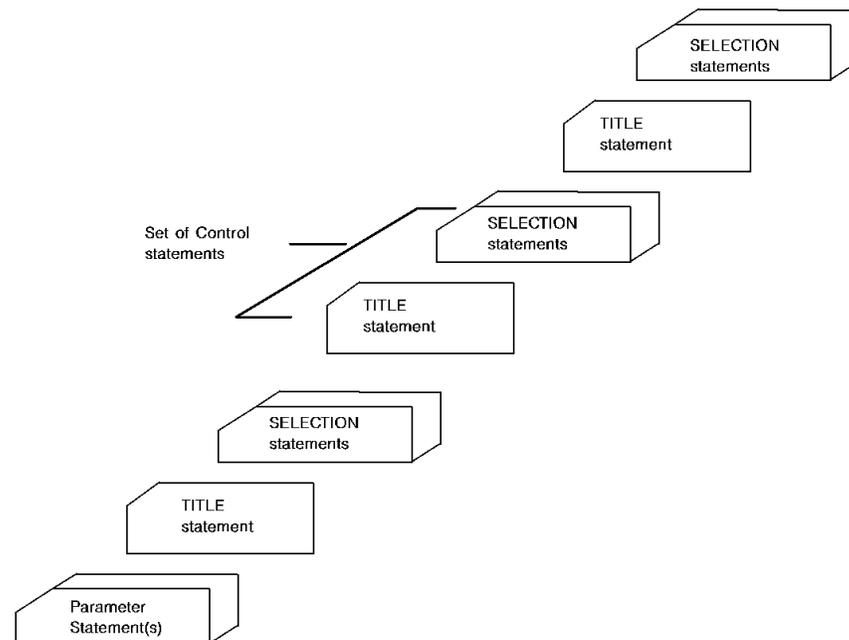
- **Associate record descriptions with a specific file.**

Record descriptions can be associated with a specific file. Within the DATA DIVISION of each program, any number of record descriptions that apply to the same file can exist. Control statements can be used to specify which record descriptions apply to a specific file.

- **Associate record descriptions with a specific program section.**

Record descriptions can be associated with a specific section of the program. Record descriptions can be present in the FILE, WORKING-STORAGE, or LINKAGE sections of programs. Control statements can be used to designate the appropriate section if necessary.

File of control statements: A file of control statements is illustrated in the figure below. The parameter statement specifies processing options for the run. The rest of the file consists of sets of control statements (one set for each subset of records for which cross referencing is desired). Each set contains a title statement and one or more selection statements. Syntax and rules for control statements are presented later in this chapter.



Establishing processing options: The parameter statement establishes processing options for the run. Each set of control statements identifies a group of records (most commonly a file) for which an individual cross-reference report is to be produced.

Assigning a title to the report: A set of control statements assigns a descriptive title to the report on the subset of records with the title statement and specifies, with selection statements, the 01-level records that are to be included in the report. Typically, many sets of control statements are specified in the file of control statements.

System Data Cross-Reference report: During execution, the Cross Reference Processor cross references data elements throughout the system of programs, as directed by the control file, and produces a series of reports (one for each set of control statements). These reports are known collectively as the System Data Cross-Reference Report. In the reports, all PROCEDURE DIVISION statements using a specific data element are listed below the element. Additionally, all data elements are identified by their data names and associated with their program names and records names. Source line numbers for each data name and PROCEDURE DIVISION statement are also supplied.

Sample report: Because the System Data Cross-Reference Report lists data elements in order by starting columns, synonymous elements are grouped together and overlapping data fields are close to one another in the report. Thus, all uses of any column or range of columns is easy to research, as shown below:

3.2 Overview

| SYSTEM DATA CROSS REFERENCE | | | | FOR REPORT: CUSTOMER RECORD | | | DICTIONARY LOADER | | 28 JAN 99 1425 | PAGE 2 |
|-----------------------------|----|----|-------------|-----------------------------|----------|----------|-------------------|-------|----------------------|---|
| FROM | TO | LV | DATA NAME | SRC LN | PROG ID | REC NAME | SIZE | USAGE | OCCURS | QUALIFIER |
| | | | | | | | | | REF LN REF STATEMENT | |
| 1 | 10 | 03 | CUST-NUMBER | 047000 | PRANDEM1 | CUSTOMER | 10 | DISP | | CUSTOMER-FILE 131000 MOVE SPACES TO CUST-NUMBER 138000 IF ORD-CUST-NUMBER = CUST-NUMBER 144000 MOVE CUST-NUMBER TO RPT-CUST-NO |
| | | 03 | CUST-NUM | 044000 | PRANDEM2 | CUSTOMER | 10 | DISP | | CUSTOMER-FILE 190000 MOVE CUST-NUM TO RPT-CUST-NO |
| 11 | 30 | 03 | CUST-NAME | 048000 | PRANDEM1 | CUSTOMER | 20 | DISP | | CUSTOMER-FILE 145000 MOVE CUST-NAME TO RPT-NAME |
| | | 03 | CUST-NAME | 045000 | PRANDEM2 | CUSTOMER | 20 | DISP | | CUSTOMER-FILE 091000 MOVE CUST-NAME TO RPT-CUST-NAME |
| | | 03 | CUST-NAME | 041000 | PRANDEM3 | CUST | 20 | DISP | | CUSTFILE 064000 MOVE CUST-NAME TO MAIL-LINE-1 |

Dictionary of Data Names reports: The Dictionary of Data Names Report is an optional report that can also be produced by a Cross Reference Processor run. This report lists all data elements alphabetically with additional information that points to the definitions of data items in the source code. Thus, this report can be used to control changes in programs, files, records, or data elements.

3.3 Developing a File of Control Statements

Control file specifies report organization: To direct the operation of the Cross Reference Processor, a file of control statements must be developed. The control file specifies the organization of information to be reported in the System Data Cross-Reference Report by identifying groups of records (most commonly files) for which individual cross-reference reports are needed. The control file uses three types of statements:

- The parameter statement (to specify processing options)
- The title statement (to identify a group of records)
- The selection statement (to specify selection criteria for records in a group)

Worksheets: To aid in developing a file of control statements, a worksheet is provided. Information found in the File and Record Layouts Reports and the DATA DIVISION Cross Reference Reports for the system of programs aids in filling out the worksheets.

Control file optional, but recommended: Note that the purpose of the control file is to limit the amount of information cross referenced together so that the report can be used to research various descriptions of the same records easily. The control file can be omitted, in which case all records from all programs and files will be reported together. But the value of the System Data Cross-Reference Report depends upon its organization. A carefully planned control file results in a more useful report.

Steps: To develop a file of control statements, follow the four steps outlined below:

Step 1 — Specify processing options: Refer to the following table and determine whether the default processing options in effect are acceptable. Select any override processing options needed for the run. Specify the override options with a parameter statement. This statement, if used, must be the first statement in the control file. For syntax and rules, refer to 3.5, “Parameter Statement” on page 3-13 later in this chapter.

| Parameter | Default Option | Override Option |
|-------------------|---|---|
| FILLER/NOFILLER | NOFILLER — Data elements named FILLER are not to be included in the System Data Cross-Reference Report. | FILLER — Data elements named FILLER are to be included in the System Data Cross-Reference Report. |
| REFONLY/NOREFONLY | REFONLY — Only data items referred to by a PROCEDURE DIVISION statement are to be included in the System Data Cross-Reference Report. | NOREFONLY — All data items are to be included in the System Data Cross-Reference Report. |

| Parameter | Default Option | Override Option |
|---|---|--|
| DICTIONARY/ NODICTIONARY | NODICTIONARY — The Dictionary of Data Names Report is not to be printed. | DICTIONARY — The Dictionary of Data Names Report is to be printed. |
| LIBRARY/NOLIBRARY | NOLIBRARY — Data usage files are not to be read from a library. The default DISK (see below) must be taken with NOLIBRARY. | LIBRARY — Data usage files are to be read from a partitioned data set (OS/390) or source statement library (VSE/ESA). |
| DISK/NODISK | DISK — Data usage files are to be read from a sequential data set. | NODISK — Data usage files are not to be read from a sequential data set. LIBRARY (see above) must be specified with NODISK. |
| MEMBER-NAME-IS-ID/ NOMEMBER-NAME-IS-ID | MEMBER-NAME-IS-ID — All of the member names supplied with the LIBRARY parameter are to be used as the program IDs on the reports. | NOMEMBER-NAME-IS-ID — The program names in the source programs are to be used as the program IDs on the reports. |
| PROGRAM-ID | -- | PROGRAM-ID — The source program identified by <i>source-program-name</i> is to be identified on reports by the new name specified. |
| LIMIT/NOLIMIT | LIMIT — Complete reference statements for each data item up to the limit specified are to be listed. 10 is the default limit. | NOLIMIT — An unlimited number of complete reference statements are to be listed for each data item. |

Step 2 — Identify groups of records: Determine the groups of records for which cross referencing is desired and assign a descriptive title to each group. Any group of records can be cross referenced, but the most common group is the file. Therefore, consider first the files common to multiple programs in the system of programs being processed and give each file a descriptive title. Then, identify any other group of records for which cross referencing would be useful. For example, defining a group of records to be all records from working storage from all programs yields a cross-reference report that allows extensive analysis of the use of work records for the system of programs.

Step 3 — Fill in worksheets: Determine which records are to be included in each group and identify these records by filling in worksheets. Completed worksheets will be used to code title and selection statements. A sample worksheet is shown below. Instructions for filling in worksheets are presented later in this session.

Step 4 — Create the control file: When the worksheets are complete, create the control file by generating one statement for each line on each worksheet. If used, the parameter statement must be first, followed by the title statement and its selection statements. Continue to code a title statement and selection statements for all of the remaining worksheets. For syntax and rules for coding title statements and selection statements, refer to 3.6, “Title Statement” on page 3-17 and 3.7, “Selection Statement” on page 3-18 later in this chapter.

| CROSS REFERENCE PROCESSOR CONTROL FILE WORKSHEET | | | |
|--|---|----------------------|------------------------------|
| FILE/REPORT = 2 *** MASTER PROFILE FILE *** | | | |
| PROGRAM-ID | | RECORD NAME | IN or QUALIFICATION OF |
| | : | <i>TAPE-IN</i> | |
| | : | <i>MPF-REC</i> | |
| | : | <i>MPF-RECORD</i> | |
| <i>WRITREP</i> | : | <i>MAST-REC</i> | |
| | : | <i>MAST-PROF-REC</i> | |
| | : | <i>MAST-REC</i> | <i>IN MASTER-FILE</i> |
| | : | <i>NEW-PROF-REC</i> | |
| | : | | |
| | : | | |
| | : | | |
| | : | | |
| | : | | |
| | : | | |
| | : | | |
| | : | | |

3.4 Filling in Worksheets

Write in the title first: Start a worksheet for each group of records, as shown in the figure above by writing the descriptive title (that is, file or other group identifier) after the header REPORT=. The descriptive title clearly identifies the group of records, most commonly a particular file that may be known by many different names in the system of programs. Next, enter from one to three of the following variables, line by line, on each worksheet:

1. Program ID
2. Record name
3. Qualification

Each line represents one selection statement: Each line on the worksheet represents one selection statement. The variable(s) specified on each line causes the Cross Reference Processor to select the defined subset of records. For example, supplying a program ID only specifies that all records from the named program are to be included in the report, supplying a record name only specifies that the record associated with that name is to be included. Often, a single record from a file is called by many different names in a system of programs. In this case, many separate names are needed to specify the selection of all copies of the record. Each line contains a different name for the record. Guidelines for specifying various combinations of the three variables are presented below.

Use Program Processor reports to fill in worksheets: The reports produced by the Program Processor can be helpful in filling in the worksheets:

- The File and Record Layouts Report can be used to find file names and record names without having to search through the COBOL source code for all of the programs. This report can also be used to research READ INTO and WRITE FROM statements to locate the resultant copies of records that may reside in the WORKING STORAGE or LINKAGE sections under different names.
- The DATA DIVISION Cross Reference Report can be used to track MOVE statements that move 01-level records from the FILE section to the WORKING STORAGE section or the reverse. This tracking aids in locating copies of records.

Guidelines for specifying selection variables: The record name is the key variable in specifying selection criteria. Most commonly, the record name alone is used to identify a member of the group of records to be reported on. However, it may be advantageous to further qualify record name (because, for example, the name is not unique) or to request the inclusion of records without regard to record name (because, for example, the objective of the report is to look at all records in the LINKAGE section of all programs). All possible combinations of program id, record name, and qualification are valid. Listed below are guidelines for supplying the program id, the record name, and/or a qualification. Note that the qualification can be an FD file name or keywords to indicate the WORKING STORAGE or LINKAGE sections.

| | |
|---|---|
| Record name only | If the same record name is used in different programs and always exclusively for the file under consideration, supply only the record name. |
| Record name and FD file name (that is, qualification) | If the same record name is used in a single program for multiple files, supply the record name and the FD file name. Program ID can be left blank unless the record name is used in other ways in the system of programs being processed. |
| Record name and program ID | If the same record name is used for different files in different programs, supply the record name and program ID for each record that applies to the file under consideration. Qualification can be left blank unless the record name is also used for multiple files in the program. |
| FD file name (that is, qualification) | If all record descriptions for an FD are to be included, supply the FD file name under qualification. If, throughout the system of programs, the FD file name is used only to refer to the file to be cross referenced under the specified title, leave the program ID and record name blank. All record descriptions for the FD file name from any program in the system will be cross referenced and reported. However, if the FD file name is used for different files in different programs, a line must be completed for each program. Each line must supply the FD file name, under qualification, as well as the program id. All record descriptions for the FD file name in the specified programs will be cross referenced and reported. |
| WORKING STORAGE or LINKAGE (qualification) | If all record descriptions from the WORKING STORAGE or LINKAGE sections are to be included, supply the appropriate keyword under qualification. |
| None of the three variables | If all records from all programs are to be cross referenced together, omit selection statements altogether. |

Summary table: The table below summarizes the subsets of records selected based on the combination of variables specified.

| Variables in the selection statement | | | |
|---|-----------|-----------|--|
| Blank | Specified | Blank | The named record from all programs with no qualification (that is, from all FD files and from all sections). |
| Blank | Specified | Specified | The named record from all programs as qualified (that is, from the FD file specified or from the working storage or linkage sections). |
| Blank | Blank | Specified | All records from all programs as qualified. |
| Specified | Blank | Blank | All records from the named program (with no qualification). |
| Specified | Blank | Specified | All records from the named program as specified. |
| Specified | Specified | Blank | The named record from the named program (with no qualification) |
| Specified | Specified | Specified | The named record from the named program as qualified. |
| Blank | Blank | Blank | All records from all programs (with no qualification). |

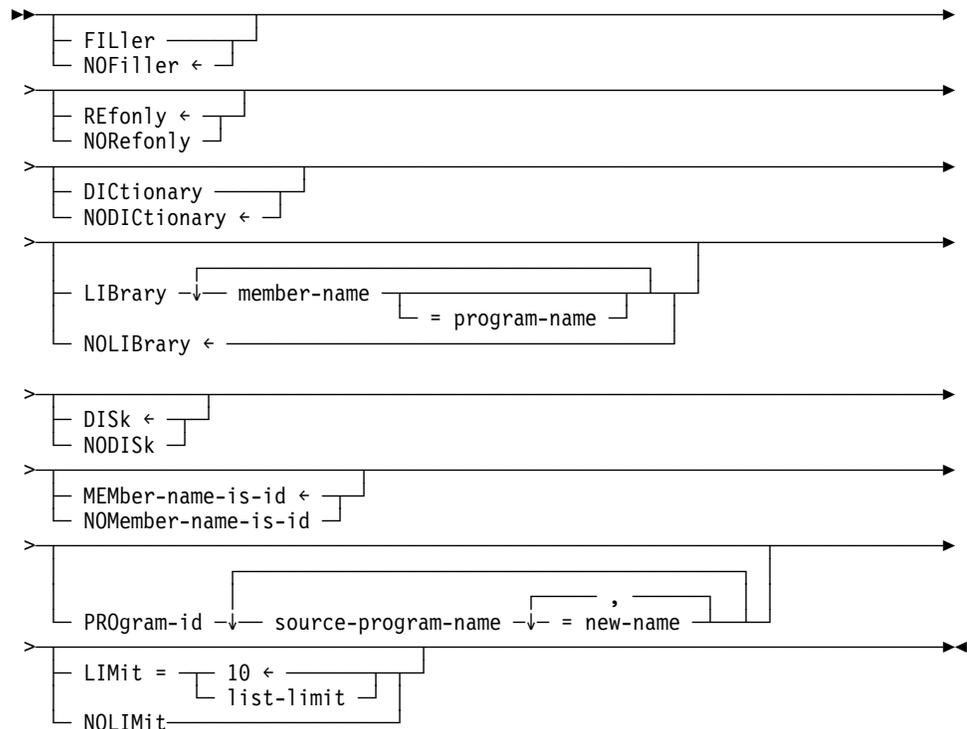
3.5 Parameter Statement

Specifies overrides: The parameter statement specifies override processing options for the Cross-Reference Processor.

Coding rules: The following rules apply to coding parameter statements.

- Parameter statements, if used, must be included at the beginning of the file of control statements.
- Multiple statements can be entered.
- Statements can be coded in positions 1 through 72.
- Options can be specified in any order, with one or more options per statement and at least one blank or comma between specifications.
- If an option requires a list of information, the list must follow the option keyword immediately on the same statement. If the list must be continued to a new line, the option keyword must be repeated. For the PROGRAM-ID option, *source-program-name* (see syntax below) must also be repeated when a list of new names is being continued.

Syntax



Parameter list

FILLer/NOFiller

Specifies whether the System Data Cross-Reference Report is to include data elements named FILLER, as follows:

- FILLER — Data elements named FILLER are to be included in the System Data Cross-Reference Report.
- NOFILLER (default) — Data elements named FILLER are not to be included in the System Data Cross-Reference Report.

REfonly/NORefonly

Specifies whether the System Data Cross-Reference Report is to include only the data items referred to by a PROCEDURE DIVISION statement, as follows:

- REFONLY (default) — Only those data items referred to by a PROCEDURE DIVISION statement are to be included in the System Data Cross-Reference Report.
- NOREFONLY — All data items are to be included in the System Data Cross-Reference Report. Note that this parameter does not affect the inclusion of data items named FILLER.

DICTIONary/NODICTIONary

Specifies whether to print the Dictionary of Data Names Report, as follows:

- DICTIONARY — The Dictionary of Data Names Report is to be printed.
- NODICTIONARY (default) — The Dictionary of Data Names Reports is not to be printed.

LIBrary/NOLIBrary

Specifies information about the data usage files to be input to the DDDL Generator, as follows:

- LIBRARY identifies the data usage file. Each occurrence of *member-name* identifies a data usage file. All member names specified must be members of the same partitioned data set (OS/390) or source statement library (VSE/ESA). The optional entry, *program-name*, can be specified for any member name and overrides the use of the member name as the program ID on the generated ADD PROGRAM syntax.

LIBRARY must always be specified with NODISK (see below) if all of the data usage files are stored in a partitioned data set (OS/390) or source statement library (VSE/ESA). It can be specified with DISK if data usage files are to be read from both a sequential data set, and partitioned data set (OS/390) or a source statement library (VSE/ESA).

- NOLIBRARY (default) specifies that data usage files are not to be read from a partitioned data set (OS/390) or source statement library (VSE/ESA). If the default of NOLIBRARY is taken, then the default of DISK (see below) must also be taken.

DISK/NODISK

DISK/NODISK are options that are used with LIBRARY/NOLIBRARY, as follows:

- **DISK** (default) specifies that data usage files are to be read from a sequential data set. **DISK** must always be specified with **NOLIBRARY**. **DISK** can be specified with **LIBRARY** if the data usage files are to be read from both a sequential data set and a partitioned data set (OS/390) or source statement library (VSE/ESA).
- **NODISK** specifies that data usage files are not to be read from a sequential data set. **LIBRARY** (see above) must be specified with **NODISK** if all of the data usage files are stored in a partitioned data set (OS/390) or source statement library (VSE/ESA).

MEMber/NOMember

Specifies the source of the program IDs to be used on reports, as follows:

- **MEMBER-NAME-IS-ID** (default) — All of the member names supplied with the **LIBRARY** parameter are to be used as program IDs on the reports. Note that once member names are assigned as program IDs with this parameter, member names must also be used for program IDs on selection statements.
- **NOMEMBER-NAME-IS-ID** — The program ID specified in the **PROGRAM-ID** paragraph in the COBOL source program is to be used as the program id on the report.

Note: To guarantee unique identification of all programs whose data usage files are stored in a partitioned data set or source statement library, operate under the default **MEMBER-NAME-IS-ID** and specify the **LIBRARY** and **NODISK** parameters. To guarantee unique identification of all programs whose data usage files are stored in sequential data sets, use the **PROGRAM-ID** parameter described below, as needed.

PROgram-id

Provides unique program IDs for source programs that have the same name (that is, duplicate names in their internal **PROGRAM-ID** paragraphs) or changes an internal **PROGRAM-ID** name to another name for printing in the reports. This parameter is only used with data usage files that are stored in sequential data sets.

Source-program-name specifies the source **PROGRAM-ID** name that is to be changed. Occurrences of *new-name* specify the names that will be assigned sequentially whenever the common **PROGRAM-ID** name (that is, *source-program-name*) appears in the input data usage files. *Source-program-name = new-name* can be repeated to name other **PROGRAM-ID** names and their associated new names.

Note: Whenever internal **PROGRAM-ID** names are changed in this way, the new names must be used for specifying program-ID on selection statements.

LIMit/NOLIMit

Establishes the maximum number of reference statements per data item to be listed completely in the System Data Cross-Reference Report, as follows:

- **LIMIT=10** (default)/*list-limit* — To be listed are complete reference statements including line number and text for each data item up to the default limit taken (10) or the limit specified. When the limit is reached, only line

numbers are listed for the remaining references to the data item. Limit=O specifies that only line numbers are to be listed for all references to the item.

- NOLIMIT — To be listed are the line numbers and statements for all references to all data items.

3.6 Title Statement

Purpose: The title statement assigns a descriptive title to the report pages related to a specific group of records (for example, a file) and marks the beginning of a new set of control statements.

Specify for each set of control statements: A title page must be specified for each set of control statements. The title specified is printed on the first line of every page associated with the set of control statements. To avoid printing a title, the title statement supplied can specify only the keyword identifier and an equal (=), omitting the descriptive text.

Note: If the title statement is omitted, the following text will be printed as the title "No Report Title" The Cross Reference Processor will assume that all subsequent selection statements pertain to the same group of records until it finds another title statement. The title statement can only be omitted for the first set of selection statements. The title statement can be coded anywhere using positions 1 through 72.

Syntax

Parameter list

REPort/FILE

Identifies the statement as a title statement. One of these keywords followed by an equal sign (=) must be specified. The keyword specified, the equal sign, and *report-title* will appear on the report.

Report-title supplies a descriptive report title. It must be a 1- to 30-character alphanumeric value. Quotes are not required and, if used, become a part of the title printed on the report.

Example: A sample title statement is shown below:

```
FILE=1 *** TRAFFIC FILE ***
```

3.8 Sample Control File

A sample control file is shown below. A parameter statement is shown first, followed by two sets of control statements pertaining to two files.

```
DICTIONARY
FILE=1 *** TRAFFIC FILE ***
      :TRF-IN-REC
      :TRF-IN-RECORD
      :TRF-OUT-REC
      :TRF-OUT-RECORD
ESTIMATE :WORK-TRF      IN 'WORKING-STORAGE'
      :WRK-TRF
FILE=2 *** MASTER PROFILE FILE ***
      :TAPE-IN
      :MPF-REC
      :MPF-RECORD
WRITREP :MAST-REC
      :MAST-PROF-RECORD
      :MASTER-PROF-REC
      :MAST-REC      IN MASTER-FILE
      :NEW-PROF-REC
```

3.9 System Data Cross-Reference Report

Description: The System Data Cross-Reference Report provides extensive information about the use of data items throughout a system of COBOL programs. The report begins with a header page that provides a formatted listing of the file of control statements and a count of records found for each selection statement specified. Each subsequent page identifies the subset of records being cross referenced (using the title form the title statement) and provides detail information about data elements within the records.

Sample report: In the report sample below, the header page appears first, followed by the first page in the main body of the report.

| SYSTEM DATA CROSS REFERENCE | | *LIST OF REQUESTED RECORDS* | | | DICTIONARY LOADER | 28 JAN 99 1425 | PAGE | 1 | | |
|-----------------------------|----|-----------------------------|-----------------|-----------|-------------------|----------------|------|-------|--------|----------------------------------|
| REPORT TITLE | | PROGRAM-ID | 01-LEVEL RECORD | QUALIFIER | COUNT | | | | | |
| CUSTOMER RECORD | | *ANY PROGRAM* | CUSTOMER | | | | | 2 | | |
| | | *ANY PROGRAM* | CUST | | | | | 1 | | |
| ORDOR RECORD | | *ANY PROGRAM* | ORDOR | | | | | 1 | | |
| SYSTEM DATA CROSS REFERENCE | | FOR REPORT: CUSTOMER RECORD | | | DICTIONARY LOADER | 28 JAN 99 1425 | PAGE | 2 | | |
| FROM | TO | LV | DATA NAME | SRC LN | PROG ID | REC NAME | SIZE | USAGE | OCCURS | QUALIFIER |
| | | | | | | | | | REF LN | REF STATEMENT |
| 1 | 10 | 03 | CUST-NUMBER | 047000 | PRANDEM1 | CUSTOMER | 10 | DISP | | CUSTOMER-FILE |
| | | | | | | | | | 131000 | MOVE SPACES TO CUST-NUMBER |
| | | | | | | | | | 138000 | IF ORD-CUST-NUMBER = CUST-NUMBER |
| | | | | | | | | | 144000 | MOVE CUST-NUMBER TO RPT-CUST-NO |
| | | | | | | | | | | CUSTOMER-FILE |
| | | | | | | | | | 190000 | MOVE CUST-NUM TO RPT-CUST-NO |
| 11 | 30 | 03 | CUST-NAME | 048000 | PRANDEM1 | CUSTOMER | 20 | DISP | | CUSTOMER-FILE |
| | | | | | | | | | 145000 | MOVE CUST-NAME TO RPT-NAME |
| | | | | | | | | | | CUSTOMER-FILE |
| | | | | | | | | | 091000 | MOVE CUST-NAME TO RPT-CUST-NAME |
| | | | | | | | | | | CUSTFILE |
| | | | | | | | | | 064000 | MOVE CUST-NAME TO MAIL-LINE-1 |

Field descriptions

REPORT TITLE

The descriptive title used to identify the group of records and taken from the title statement.

PROGRAM-ID

The PROGRAM-ID from the selection statement or, if PROGRAM-ID was blank, the entry *ANY PROGRAM*.

01-LEVEL RECORD

The 01-level record name from the selection statement, or if record name was blank, the entry *ANY RECORD*.

QUALIFIER

The FD file name, the keywords WORKING STORAGE or LINKAGE, or blank as specified on the selection statement.

COUNT

A count of the 01-level records selected as a result of the specifications on the selection statement.

FOR REPORT

The descriptive title used to identify the group of records and taken from the title statement.

FROM

The starting position of the data element.

TO

The ending position of the data element.

LV

The level number from the data item description entry.

DATA NAME

The data name from the data item description entry.

SRC LN

The line number of the data item description entry in the source program.

PROG ID

The program ID being used to identify the source program. The program ID may be the internal PROGRAM-ID from the COBOL source program or a library member name, depending on the user-defined options in effect from the run.

REC NAME

The 01-level record name from the record description entry where the data element was found.

SIZE

The size of the data item field.

USAGE

The form in which the data item is to be stored as the result of the source program's specifications:

- GROUP — The data item contains subordinate items.
- DISP — The data item is stored in character form.
- DISP-NM — The data item is stored one digit per character position. The PIC contains only S, 9, and V.
- COMP — The data item is stored as computational (1,2,3, or 4). The PIC entry contains only S, 9 and V.
- NM-EDIT — The data item is a numeric item stored in character format. The PIC contains some or all of the editing characters +, -, z, \$, comma, B, CR, DB, ., or 0.

The following report writing specifications can also appear in this column:

- RH — Report heading
- RF — Report footing

- PH — Page heading
- PF — Page footing
- CH — Control heading
- CF — Control footing
- DE — Detail

OCCURS

The number of times the data item is repeated as the result of an OCCURS clause.

QUALIFIER

The FD file name or the keywords WORKING STORAGE or LINKAGE to indicate where the data element was found.

REF LN

The line number in the source program from the COBOL statement that follows.

REF STATEMENT

A COBOL statement that refers in any way to the data element.

3.10 Dictionary of Data Names Report

Description: The Dictionary of Data Names Report lists all record and element names alphabetically, together with the following information about each item listed:

- Its position in the record
- Size
- Usage
- Level
- Source line number
- Program ID
- Member name
- Internal program ID
- Record name

This report aids in tracking the use of data elements throughout the system of programs.

Purpose: The purpose of this report is to aid in controlling change. The information supplied indicates the exact line in the appropriate COBOL source program where any data item used throughout the system of programs is defined.

Note: This report is optional and is not produced automatically. To obtain this report, specify the option `DICTIONARY` on a parameter control statement for the run.

Sample report

| SYSTEM DATA CROSS REFERENCE | | | | | *DICTIONARY OF DATA NAMES* | DICTIONARY LOADER | | | 28 JAN 99 1425 | PAGE 1 |
|-----------------------------|-----|------|-------|-----|----------------------------|-------------------|------------|-------------|------------------|----------------|
| FROM | TO | SIZE | USAGE | LVL | D A T A N A M E | SRCLN | PROGRAM ID | MEMBER NAME | INTERNAL PROG-ID | 01-RECORD-NAME |
| 1 | 104 | 104 | GROUP | 01 | CUST | 039000 | PRANDEM3 | (DISK) | PRANDEM3 | CUST |
| 31 | 70 | 40 | GROUP | 03 | CUST-ADDRESS | 049000 | PRANDEM1 | (DISK) | PRANDEM1 | CUSTOMER |
| 31 | 70 | 40 | GROUP | 03 | CUST-ADDRESS | 046000 | PRANDEM2 | (DISK) | PRANDEM2 | CUSTOMER |
| 31 | 70 | 40 | GROUP | 03 | CUST-ADDRESS | 042000 | PRANDEM3 | (DISK) | PRANDEM3 | CUST |

3.11 Executing the Cross Reference Processor

Job Control Language (JCL) for executing the Cross Reference Processor under OS/390 and VSE/ESA is shown below. Under both OS/390 and VSE/ESA, processing options must be specified with the parameter statement.

Note: (OS/390 users only) — The PARM clause of the EXEC statement cannot be used to specify options when executing this component.

OS/390 JCL — PRANXREF

```
//PRANXREF EXEC PGM=PRANXREF,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//SYSLST DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SORTMSG DD SYSOUT=A
//SORTLIB DD DSN=SYS1,SORTLIB,DISP=SHR
//SORTWK01 DD UNIT=disk,SPACE=(cyl,(5,5))
//SORTWK02 DD UNIT=disk,SPACE=(cyl,(5,5))
//SORTWK03 DD UNIT=disk,SPACE=(cyl,(5,5)) Include only if using
//PRANLIB DD DSN=reflib,DISP=SHR ← LIBRARY option.
//PRANREF DD DSN=sysref1,DISP=SHR
// DD DSN=sysref2,DISP=SHR Include only
. ← if using DISK option
.
.
// DD DSN=sysrefn,DISP=SHR
//dcmmsg DD DSN=idms.sysmsg.ddldcmmsg,DISP=SHR
//sysjrn1 DD *
//SYSOUT DD SYSOUT=a
//SYSIDMS DD *
dmc1=dmc1-name
Insert other SYSIDMS parameters as appropriate
//SYSIPT DD *
Insert optional control statements here
```

| | |
|---------------------------|---|
| idms.dba.loadlib | Data set name of the load library containing the DMCL and database name table load modules |
| idms.loadlib | Data set name of the load library containing CA-IDMS system software modules |
| cyl,(5,5) | file space allocation of work file |
| disk | symbolic device name of disk file |
| reflib | data set name of partitioned data set containing data usage files |
| sysref1, sysref2, sysrefn | data set names of sequential data sets containing data usage files |
| dcmsg | DDname of the system message (DDLDCMSG) area |
| idms.sysmsg.ddldcmsg | Data set name of the system message (DDLDCMSG) area |
| SYSIDMS | DDname of the CA-IDMS parameter file specifying runtime directives and operating system-dependent parameters. ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> . |

VSE/ESA JCL — PRANXREF

```
// DLBL    SS1n,'user.srclib'
// EXTENT  ,nnnnn
// LIBDEF  SL,TO=SS1n,TEMP
// DLBL    PRANREF,'sysref',,SD  ← Include only if using DISK option
// EXTENT  SYS010,nnnnn
// ASSGN   SYS010,DISK,VOL=nnnnn,SHR
// DLBL    SORTWK1,'WORK1',0,SD
// EXTENT  SYS001,nnnnn,1,,ssss,200
// ASSGN   SYS001,DISK,VOL=nnnnn,SHR
// EXEC    PRANXREF,SIZE=128K
optional control statements
/*
```

| | |
|-------------|---|
| nnnnn | serial number of disk volume |
| ssss | starting track (CKD) or block (FBA) of disk extent |
| sysref | file-id for sequential file containing data usage file |
| SYS001 | logical unit assignment for sort work file |
| SYS010 | logical unit assignment for data usage file (SYS010 required) |
| user.srclib | source statement library containing data usage files |
| SS1n | filename of source statement library |

Chapter 4. DDDL Generator

| | |
|---|------|
| 4.1 About the DDDL Generator | 4-3 |
| 4.2 Overview | 4-4 |
| 4.3 Developing a File of Control Statements | 4-7 |
| 4.4 Parameter Statement | 4-9 |
| 4.5 VERSION Statement | 4-11 |
| 4.6 Grouping Statement | 4-12 |
| 4.7 Using the Grouping Statement | 4-14 |
| 4.8 Editing Generated DDDL Statements | 4-17 |
| 4.9 Executing the DDDL Compiler | 4-20 |

4.1 About the DDDL Generator

Purpose: The DDDL Generator reads data usage files and generates the appropriate DDDL source statements for input to the IDD DDDL compiler. Statements generated include ADD, PROGRAM, ADD RECORD, and ADD FILE. COBOL substatements of the RECORD statement are generated for defining elements.

DDDL Generator control statements: Control statements can be used to control the operation of the DDDL Generator:

- **Grouping-control statements** specify to the DDDL Generator those file (or record) definitions that describe the same file (or record) but have different names and those file (or record) definitions that have the same name but do not define the same file (or record).
- The **VERSION statement** specifies a VERSION clause, causing the DDDL Generator to include the specified VERSION clause (instead of the default of VERSION 01) in every ADD statement generated.

Output: The DDDL Generator produces a listing of statements generated and an output file containing the statements. This file can be input to the DDDL compiler directly or edited first and then input to the compiler. The DDDL compiler processes the generated statements to populate the data dictionary.

What follows: This chapter presents an overview of the DDDL Generator and instructions on how to develop a file of control statements, edit the generated DDDL statements, and execute the DDDL Generator under OS/390 and VSE/ESA.

4.2 Overview

Without control statements: When operated without control statements, the DDDL Generator generates DDDL ADD statements for each unique program, file, and record name in the system of programs being processed. An ADD statement is generated for the first occurrence of each program, file, and record name. Subsequent occurrences are considered to be duplicates and are ignored. The version clause VERSION 01 is generated for each ADD statement.

With control statements: The DDDL Generator operates as described above unless the user supplies control statements. These statements alter the operation of the DDDL Generator as follows:

Parameter statement: This statement specifies override processing options such as suppressing a listing of generated statements.

VERSION statement: This statement specifies an alternative VERSION clause. If this statement is used for a run, the DDDL Generator adds the VERSION clause specified (instead of VERSION 01) to all generated ADD PROGRAM, ADD FILE, and ADD RECORD statements. If grouping-control statements (described below) specify synonyms, the VERSION clause specified is also added to generated SYNONYM clauses.

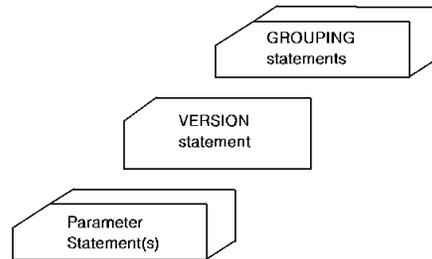
Grouping statement: This statement identifies files or records with synonymous or nonunique names. Synonymous names are different names that refer to definitions of the same file or record; a nonunique name is a single name that refers to the definitions of different files or records.

If a grouping statement identifying synonymous names for the same file or record is used for a run, the DDDL Generator generates an ADD statement for each different name and a SYNONYM clause within each ADD statement. The SYNONYM clause documents all other synonymous entities for which an ADD was generated during the run.

If a grouping statement identifying a nonunique name is used for a run, the DDDL Generator generates an ADD statement for each unique entity referred to by the name (instead of just for the first occurrence of the name).

Note that an ADD statement is always generated for the first occurrence of every file or record name. If the name appears in a grouping statement for the run, an ADD statement will be generated for the first occurrence of the name for each group defined. Additionally, an ADD statement is generated for the first occurrence of the name that is not described by any of the grouping statements.

Sample file of control statements: A file of control statements is illustrated below. The parameter statement is first and specifies override processing options for the run. Next, the VERSION statement specifies a VERSION clause to be added to generated statements. The rest of file consists of grouping statements.



Output from the DDDL Generator: Output from the DDDL Generator consists of a file of DDDL statements and a listing of the generated statements. For a breakdown of the DDDL clauses generated for each entity type, refer to the table below.

Use the output file to populate the dictionary: The output file can be input to the DDDL compiler to populate the data dictionary. Before being input to the compiler, this file can be edited. Editing considerations are presented later in this section.

| DDDL statement | DDDL clauses |
|------------------------------------|--|
| ADD FILE | VERSION LABELS ARE STANDARD/OMITTED RECORD SIZE RECORDING MODE BLOCK SIZE FILE NAME SYNONYM |
| ADD RECORD (COBOL substatement) | VERSION LANGUAGE WITHIN FILE RECORD NAME SYNONYM level-n element-name REDEFINES OCCURS OCCURS DEPENDING ON ASCENDING/DESCENDING KEY INDEXED BY (for one item) PICTURE VALUE SIGN BLANK WHEN ZERO SYNCHRONIZED JUSTIFIED RIGHT |
| ADD PROGRAM | VERSION LANGUAGE ESTIMATED LINES PROGRAM CALLED INPUT/OUTPUT/I-O/EXTEND FILE ENTRY POINT RECORD USED REFERENCED/ MODIFIED ELEMENT REFERENCED/ MODIFIED |

4.3 Developing a File of Control Statements

Types of control statements: The DDDL Generator accepts three types of optional control statements:

- The parameter statement
- The VERSION statement
- The grouping statement

One or more parameter statements, a single VERSION statement, and one or more grouping statements make up the control file.

Steps to develop a file: To develop a file of control statements, follow these steps:

Step 1: Specify the processing options: See the table below and determine whether the default processing options in effect are acceptable. Select any override processing options needed for the run. Specify the override options with a parameter statement. Note that options can be specified in OS/390 execution JCL by using the PARM clause of the EXEC statement. For syntax and rules, see 4.4, “Parameter Statement” on page 4-9 later in this chapter.

| Parameter | Default option | Override option |
|-----------------------|--|--|
| LIBRARY/ NOLIBRARY | NOLIBRARY — Data usage files are not to be read from a library. The default DISK (see below) must be taken with NOLIBRARY. | LIBRARY — Data usage files are to be read from a partitioned data set (OS/390) or source statement library (VSE/ESA). |
| DISK/NODISK | DISK — Data usage files are to be read from a sequential data set. | NODISK — Data usage files are not to be read from a partitioned data set (OS/390) or source statement library (VSE/ESA). |
| LIST/NOLIST | LIST — The file of generated DDDL statements is to be listed. | NOLIST — The file of generated DDDL statements is not to be listed. |

Step 2: Specify a VERSION statement: Determine whether VERSION 01 is the appropriate clause to be added to generated DDDL statements. For considerations relating to the use of the VERSION clause, refer to 4.8, “Editing Generated DDDL Statements” on page 4-17 later in this chapter. Specify a VERSION statement, if appropriate. For syntax and rules, see 4.5, “VERSION Statement” on page 4-11 later in this chapter.

Step 3: Identify file and record names: Identify nonunique or synonymous file and record names. Use the System Data Cross Reference Report and the Dictionary of Data Names Report to research the use of entity names. Find multiple names for the same file or record and instances when a single name is used to refer to different files or records.

Step 4: Specify grouping statements: Using the information gathered in Step 3, create the grouping statements necessary to ensure that an ADD statement will be generated for each unique entity and that SYNONYM clauses will be generated for ADD statements that describe the same file or record using different entity-occurrence names. See 4.6, “Grouping Statement” on page 4-12 and 4.7, “Using the Grouping Statement” on page 4-14 later in this chapter.

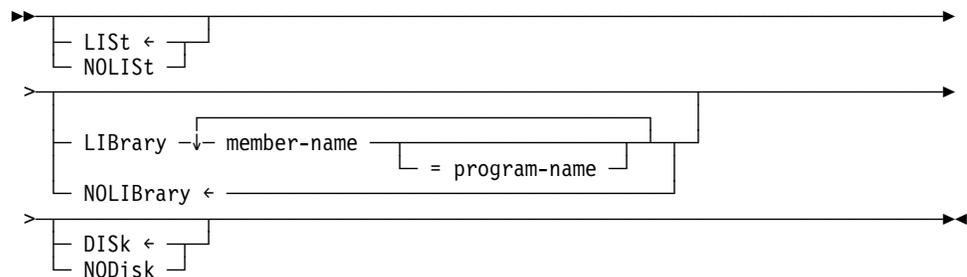
4.4 Parameter Statement

Purpose: The parameter statement specifies override processing options for the DDDL generator. Under VSE/ESA, this statement must be used to specify options; under OS/390, parameters can be specified in the execution JCL by using the PARM clause of the EXEC statement.

Coding rules: The following rules apply to coding a parameter statement for the DDDL Generator:

- Parameter statements, if used, must be input first before the data usage files.
- Multiple parameter statements can be entered.
- Statements can be coded in positions 1 through 72.
- Options can be specified in any order, with one or more options per statement and at least one blank or comma between specifications.

Syntax



Parameter list

LIST/NOLIST

Specifies whether the file of generated DDDL statement is to be listed, as follows:

- LIST (default) — The generated DDDL statements are to be listed.
- NOLIST — The generated DDDL statements are not to be listed.

LIBRARY/NOLIBRARY

Specifies information about the data usage files to be input to the DDDL Generator, as follows:

- LIBRARY — Identifies the data usage files. Each occurrence of *member-name* identifies a data usage file. All files specified must be members of the same partitioned data set (OS/390) or source statement library (VSE/ESA). The optional entry, *program-name*, can be specified for any member and overrides the use of the specified member as the program ID on the generated ADD PROGRAM syntax.

LIBRARY must always be specified with NODISK (see below) and must be specified with DISK if data usage files are to be read from both sequential

data sets and from a partitioned data set (OS/390) or a source statement library (VSE/ESA).

- **NOLIBRARY** — Specifies that data usage files are not to be read from a partitioned data set (OS/390) or source statement library (VSE/ESA). If the default of **NOLIBRARY** is taken, then the default of **DISK** (see below) must also be taken.

DISK/NODisk

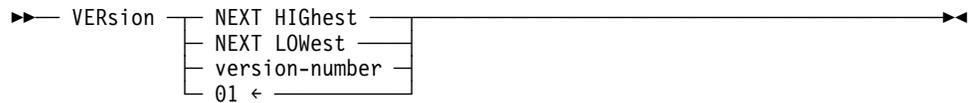
Are options used with **LIBRARY/NOLIBRARY**, as follows:

- **DISK** (default) — Specifies that data usage files are to be read from a sequential data set. **DISK** must always be specified with **NOLIBRARY**. **DISK** can be specified with **LIBRARY** if the data usage files are to be read from both sequential data set and from partitioned data set (OS/390) or source statement library (VSE/ESA).
- **NODISK** — specifies that data usage files are not to be read from a sequential data set. **LIBRARY** (see above) must be specified with **NODISK** if all of the data usage files are stored in a partitioned data set (OS/390) or source statement library (VSE/ESA).

4.5 VERSION Statement

Purpose: The VERSION statement describes the VERSION clause to be added to each generated DDDL statement. This statement is optional; if omitted, the DDDL Generator automatically adds a VERSION 01 clause to each generated ADD statement.

Syntax



Parameter list

VERsion

Identifies the statement and specifies that the clause described is to be added to all generated ADD PROGRAM, ADD FILE, and ADD RECORD statements.

NEXT HIGhest/NEXT LOWest

Specifies the version. *Version-number*, if specified, must be a 1- to 4-digit number in the range 1 through 9999.

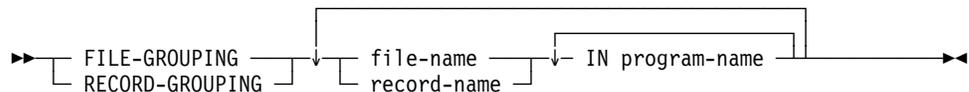
4.6 Grouping Statement

Purpose: Grouping statements uniquely identify to the DDDL Generator the file or record definitions that have nonunique or synonymous names. Synonymous file (or record) definitions describe the same file (or record) but are referred to by different names. Nonunique file (or record) definitions have the same name but do not define the same file (or record).

Coding rules: The following rules apply to coding the grouping statement:

- The keyword identifier must begin in position 1.
- Continuation lines must begin in position 2.
- Grouping statements can be coded in positions 1 or 2 through 72.
- One or more spaces must be included between entries in the statement.
- Punctuation is not allowed.
- Clauses can be specified on the same line as the keyword identifier or on subsequent lines.
- Continuation must occur at a natural space between words.

Syntax



Parameter list

FILE-GROUPING/RECORD-GROUPING

Identifies the statement as a grouping statement and specifies whether the statement applies to files or records.

file-name/record name

Identifies the file or record to be grouped. The name must be specified exactly as it appears in one or more of the programs being processed.

IN program-name

Specifies a program in which *file-name* or *record-name* appears. *Program-name* must be the internal PROGRAM-ID or, if the LIBRARY parameter has been used to rename the program, the *member-name*.

The entry IN *program-name* can be repeated (see note below) to name different programs in which the specified file or record appears. Multiple specifications of IN *program-name* for a single file or record name mean that the file or record uses the same name and is identical in each of the programs named.

Additionally, the entire specification of *file-name/record-name* IN *program-name* can be repeated (see note below). Multiple entities of this specification indicate file or record synonyms. For example, the file name INPFILE in the program TRAN and file

name INPUT in program T2 both refer to the same file; INPFILE and INPUT are file synonyms.

Note: Up to a total of five program names can be specified in a single grouping statement. Each of the program names can be associated with different file or record names (that is, by repeating the entire specification or *file-name/record-name IN program-name*). Alternatively, multiple program names can be associated with the same file or record (that is, by creating only the specification of IN *program-name* for a single file or record).

Sample: The sample grouping statement shown below specifies the maximum allowable number of program names (that is, 5):

```
FILE-GROUPING INPFILE IN PROG1 IN PROG2 IN PROG3
INPUT IN PROG4 TRANFILE IN PROG5
```

The names INPFILE, INPUT, and TRANFILE all refer to the same file, but these names appear in different programs. INPFILE refers to the file in the programs PROG1, PROG2, and PROG3; INPUT refers to the file in PROG4; TRANFILE refers to the file in PROG5.

4.7 Using the Grouping Statement

Use to identify synonyms and nonunique file or record names: Use the grouping statement to identify synonymous and nonunique file or record names to the DDDL Generator:

- Synonym names** — Specify the appropriate keyword identifier (FILE-GROUPING or RECORD-GROUPING). After the keyword identifier, specify a file name (or record name) and its associated program name(s). Repeat the specification of file name (record name) and program name(s) until all synonyms have been identified in the grouping statement.

The statement shown below illustrates grouping for two file names that refer to the same file: FILE-A names the file in PROG-1 and PROG-2, and FILE-B names the file in PROG-3.

```
FILE-GROUPING FILE-A IN PROG-1 IN PROG-2 FILE-B IN PROG-3
```

Assuming the the data usage files are input in the order PROG-1, PROG-2, and PROG-3, the DDDL Generator generates the following statements:

```
(Under PROG-1)  ADD FILE FILE-A
                  FILE NAME SYNONYM IS FILE-B
(Under PROG-2)  (No statements)
(Under PROG-3)  ADD FILE FILE-B
                  FILE NAME SYNONYM IS FILE-A
```

Because a single name cannot be both the primary entity-occurrence name and a synonym, these statements must be edited to designate one name as the primary name and the other name for the file as a synonym. For a complete discussion of synonym usage, see *IDD User Guide*.

- Nonunique name** — Specify the appropriate keyword identifier (FILE-GROUPING or RECORD-GROUPING), followed by the nonunique name and an IN clause for each program in which the name is used to refer to the file or record being grouped by that statement. Repeat this process for each different file or record referred to by the nonunique name.

The statements shown below illustrate file grouping for the name FILE-A, where FILE-A refers to one file in PROG-1 and PROG-2, and to another file in PROG-3 and PROG-4:

```
FILE-GROUPING FILE-A IN PROG-1 IN PROG-2
FILE-GROUPING FILE-A IN PROG-3 IN PROG4
```

Assuming that the data usage files are input in the order PROG-1 though PROG4, the DDDL Generator generates the statements shown below:

```
(Under PROG-1)  ADD FILE FILE-A
(Under PROG-2)  (No statements)
(Under PROG-3)  ADD FILE FILE-A
(Under PROG-4)  (No statements)
```

The two generated ADD FILE FILE-A statements can then be edited to establish different version numbers or to assign a different name to one of the files.

Note that multiple IN clauses cause the DDDL Generator to generate an ADD statement for the first occurrence of the entity description for each group. For a given name, one use of the name can be processed without grouping statement; to

obtain an ADD statement for each distinct IDD entity description, each additional use must be defined by a separate grouping statement.

If grouping statements are omitted, an ADD statement is generated for the first occurrence of a file name or record name. If the file name or record name appears again, no statement is generated; subsequent occurrences of the name are considered to be duplicates.

Example 1: Five programs (PROG-1, PROG-2, PROG-3, PROG-4, PROG-5) are being processed. All five programs access files named MASTER. The name MASTER refers to one file for PROG-1 and PROG-2, to a second file for PROG-3 and PROG-4, and to a third file for PROG-5. The following grouping statements ensure that ADD statements will be generated for each of the three unique files:

```
FILE-GROUPING MASTER IN PROG-1 IN PROG-2
FILE-GROUPING MASTER IN PROG-3 IN PROG-4
```

Note that PROG-5 is not mentioned in these statements; when the DDDL Generator encounters the file name MASTER in PROG-5, it will treat the file as one of the group of all unqualified (that is, not explicitly mentioned in a grouping statement) files named MASTER and automatically generate an ADD statement.

Assuming that the data usage files are input in the order PROG-1 through PROG-5, the DDDL Generator generates the statements shown below. Note that SYNONYM clauses are not generated because all files have the same name.

```
(Under PROG-1)  ADD FILE MASTER...
(Under PROG-2)  (No statements)
(Under PROG-3)  ADD FILE MASTER...
(Under PROG-4)  (No statements)
(Under PROG-5)  ADD FILE MASTER...
```

The three ADD statements that use the file name MASTER can be edited to assure that the three distinct entities are entered into the dictionary. The statement can be distinguished from one another by using different version numbers or by changing the name MASTER for two of the three files.

Example 2: The file name SUM-FILE is used in five programs, PROG-1 through PROG-5. The name SUM-FILE refers to the same file in all five programs but the record description for the file is different in PROG-5. The following grouping statement makes the distinction:

```
FILE-GROUPING SUM-FILE IN PROG-5
```

Assuming that the data usage files are input in the order PROG-1 through PROG-5, the DDDL Generator generates the statements shown below:

```
(Under PROG-1)  ADD FILE SUM-FILE...
(Under PROG-2)  (No statements)
(Under PROG-3)  (No statements)
(Under PROG-4)  (No statements)
(Under PROG-5)  ADD FILE SUM-FILE...
```

These statements can then be edited (that is, versions added or file names changed) to assure that both entities will be added to the dictionary.

Example 3: The file names SUM-FILE in PROG-1 and SUMMARY-IN in PROG-2 both refer to the same file. Each file name has its own record descriptions. The following statement expresses the proper grouping:

```
FILE-GROUPING SUM FILE IN PROG-1 SUMMARY-IN IN PROG-2
```

Assuming that the data usage files are input in the order PROG-1, PROG-2, the DDDL Generator generates the statements shown below:

```
(Under PROG-1)    ADD FILE SUM-FILE...  
                  FILE NAME SYNONYM IS SUMMARY-IN  
(Under PROG-2)    ADD FILE SUMMARY-IN...  
                  FILE NAME SYNONYM IS SUM-FILE
```

Because a single name cannot be both the primary entity-occurrence name and a synonym, these statements must be edited to designate one name as the primary name and all other names for the file as synonyms. For a complete discussion of synonym usage, see *IDD User Guide*

Example 4: One file is named differently in four different programs. The file is named ABC in PROG-1, DEF in PROG-2, GHI in PROG-3, and JKL in PROG-4. The following grouping statement describes this situation:

```
FILE-GROUPING ABC IN PROG-1 DEF IN PROG-2 GHI IN PROG-3 JKL IN PROG-4
```

Assuming that the data usage files are input in the order PROG-1 through PROG-4, the DDDL Generator generates the following statements:

```
(Under PROG-1)  ADD FILE ABC...  
                FILE NAME SYNONYM IS DEF  
                FILE NAME SYNONYM IS GHI  
                FILE NAME SYNONYM IS JKL.  
(Under PROG-2)  ADD FILE DEF...  
                FILE NAME SYNONYM IS ABC  
                FILE NAME SYNONYM IS GHI  
                FILE NAME SYNONYM IS JKL.  
(Under PROG-3)  ADD FILE GHI...  
                FILE NAME SYNONYM IS ABC  
                FILE NAME SYNONYM IS DEF  
                FILE NAME SYNONYM IS JKL.  
(Under PROG-4)  ADD FILE JKL...  
                FILE NAME SYNONYM IS ABC  
                FILE NAME SYNONYM IS DEF  
                FILE NAME SYNONYM IS GHI.
```

These statements must be edited to establish one primary name for the file and to designate all other names for the file as synonyms.

4.8 Editing Generated DDDL Statements

Edit before using as input to DDDL compiler: The output file of generated DDDL statements produced by running the DDDL Generator should be edited before being input to the DDDL compiler. This editing aids in maintaining control of the information entered into the dictionary.

Sample output

```

ADD FILE CUSTOMER-FILE VERSION NEXT HIGHEST
  LABELS ARE OMITTED
  RECORD SIZE IS 104
  RECORDING MODE IS F
  FILE NAME SYNONYM IS CUSTFILE VERSION NEXT HIGHEST.

ADD RECORD CUSTOMER VERSION NEXT HIGHEST
  LANGUAGE IS COBOL
  WITHIN FILE CUSTOMER-FILE VERSION HIGHEST
  RECORD NAME SYNONYM IS CUST VERSION NEXT HIGHEST.
    03 CUST-NUMBER          PIC X(10).
    03 CUST-NAME           PIC X(20).
    03 CUST-ADDRESS.
      05 CUST-ADDR1       PIC X(20).
      05 CUST-ADDR2.
        06 CUST-CITY     PIC X(15).
        06 CUST-ZIP-CODE PIC X(5).
        06 CUST-ZIPCODE REDEFINES CUST-ZIP-CODE
          PIC 9(5).
    03 CUST-CREDIT        PIC XXX.
      88 CUST-CREDIT-EXEC VALUE 'AAA'.
      88 CUST-CREDIT-GOOD VALUE ' '.
      88 CUST-CREDIT-POOR VALUE 'XXX'.

    03 CUST-SALES-INFO.
      05 CUST-SALES-QTR   OCCURS 4.
        06 CUST-NUM-SALES PIC 9(5) COMP-3.
        06 CUST-AMT-SALES PIC S9(7) COMP-3.

    03 FILLER             PIC XXX.

ADD FILE ORDER-FILE VERSION NEXT HIGHEST
  LABELS ARE OMITTED
  RECORD SIZE IS 50
  RECORDING MODE IS F
  BLOCK SIZE IS 5000.

```

Editing functions: You should perform the following editing functions, as needed:

Add comments: Add comments to the descriptions of programs, files, and records to document the function and characteristics of each entity. Comments can be added easily and in an organized way at this point in the process of populating the dictionary.

Eliminate unnecessary entities: Delete the ADD statement for any entity that should not be a part of the dictionary. For example, report title records and report detail records used within a single program generally should not be defined in the dictionary. While important in the context of the specific program in which they are used, such records do not have global applications and tend to clutter the dictionary.

Reconcile nonunique names: If the DDDL output contains multiple ADD statements for the same name, editing may be necessary to ensure that the desired entities reach the dictionary when the ADD statements are processed by the DDDL compiler. Note the following considerations:

- If the multiple ADD statements are associated with the same explicit version number (for example, VERSION 1) and no editing is performed, the DDDL compiler will process these statements in one of the following ways:

| | |
|-------------------|---|
| If DEFAULT IS ON | The DDDL compiler will process the first ADD statement encountered for the nonunique name and change subsequent ADDs to MODIFYs. This means that only the description associated with the last ADD processed will be present in the dictionary. |
| If DEFAULT IS OFF | The DDDL compiler will process only the first ADD statement that refers to the nonunique name and will flag as erroneous subsequent ADD statements for that name. This means that only the description associated with the first ADD statement processed will be present in the dictionary. |

DEFAULT IS ON/OFF can be specified with the SET OPTIONS statement.

►► For more information about this option, see *IDD DDDL Reference*

- If the multiple ADD statements are associated with a VERSION NEXT HIGHEST/LOWEST and no editing is performed, all ADD statements will be processed successfully; each occurrence of the name will be associated with a different version number.

In either case described above, the editing needed depends upon the objectives for the dictionary. Version clauses can be changed, ADD statements can be deleted or combined, or entity names can be changed (in the ADD statements and in the programs that refer to the names).

Note that running the DDDL Generator with the version statement VERSION NEXT HIGHEST and appropriate grouping statements assures that each entity occurrence with a duplicate name will be added to the dictionary when the generated statements are run through the DDDL compiler. Each repetition of the name will be associated with a different version number; the version number uniquely identifies the entity occurrence (for example, CUSTOMER record, version 1; CUSTOMER record, version 2; on so on). This technique should not be used to avoid the thoughtful evaluation of the generated statements and the editing necessary to develop a well organized dictionary.

Reconcile synonyms: Ideally, multiple ADD statements for synonymous file or record descriptions should be merged into a single ADD statement. A single description of a file or record should be entered in the dictionary. This means that all descriptions should be examined and combined. A single name should be chosen for

the entity and associated record and or element names reconciled (that is, one name and description for the element customer name). Subsequently, all programs that use the entity must be changed to use the reconciled entity-occurrence name and to use any other associated reconciled names.

Alternatively, if record and element synonyms are desired, the generated DDDL statements can be edited to include `ELEMENT NAME SYNONYM FOR RECORD NAME SYNONYM` clauses.

►►For additional information on element and record synonyms, refer to *IDD DDDL Reference*

The reconciliation of synonyms is an important user responsibility in building an effective dictionary. Although the DDDL compiler accepts and processes multiple `ADD` statements that essentially define the same entity under different names, the practice of populating the dictionary with such synonymous entities is generally undesirable.

4.9 Executing the DDDL Compiler

JCL for executing the DDDL Generator under OS/390 and VSE/ESA is shown below. Under VSE/ESA, processing options must be specified with the parameter statement. Under OS/390, options can be specified either with the parameter statement or in the PARM clause of the EXEC statement.

OS/390 JCL — PRANIDDG

```
//PRANIDDG EXEC PGM=PRANIDDG,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//SYSLST DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SORTMSG DD SYSOUT=A
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD UNIT=disk,SPACE=(cyl,(5,5))
//SORTWK02 DD UNIT=disk,SPACE=(cyl,(5,5))
//SORTWK03 DD UNIT=disk,SPACE=(cyl,(5,5))
//PRANLIB DD DSN=reflib,DISP=SHR < Include only if using LIBRARY option
//PRANREF DD DSN=sysref1,DISP=SHR < Included only if using DISK option
// DD DSN=sysref2,DISP=SHR
.
.
.
// DD DSN=sysrefn,DISP=SHR
//dmsg DD DSN=idms.sysmsg.ddldmsg,DISP=SHR
//sysjrn1 DD *
//SYSOUT DD SYSOUT=A
//SYSIDMS DD *
dmc1=dmc1-name
Insert additional SYSIDMS parameters as appropriate
//SYSIPT DD *
Insert optional control statements
//SYSPCH DD DSN=ddlstmts,DISP=(NEW,catlg),
SPACE=(trk,(10,10),rlse),UNIT=disk,
VOL=SER=nnnnn,DCB=BLKSIZE=blksize
```

| | |
|----------------------|---|
| idms.dba.loadlib | Data set name of the load library containing the DMCL and database name table load modules |
| idms.loadlib | Data set name of the load library containing CA-IDMS system software modules |
| blksize | block size of DDDL statement file (must be a multiple of 80) |
| catlg | disposition of new file: CATLG, PASS, or KEEP |
| cyl(5,5) | file space allocation of work file |
| dddlstmts | dataset name of file to contain DDDL statements |
| disk | symbolic device name of disk file |
| nnnnnn | serial number of disk volume |
| reflib | data set name of partitioned data set containing data usage files |
| sysref1 | data set names of sequential data sets containing data usage files |
| (trk,(10,10),rlse) | space allocation for DDDL statement file |
| dcmsg | DDname of the system message (DDLDCMSG) area |
| idms.sysmsg.ddldcmsg | Data set name of the system message (DDLDCMSG) area |
| SYSIDMS | DDname of the CA-IDMS parameter file specifying runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> . |

VSE/ESA JCL — PRANIDDG

```
// DLBL    SLLn,'user.srclib'
// EXTENT  ,nnnnnn
// LIBDEF  SL,TO=SSLn,TEMP
// DLBL    PRANREF,'sysref',,SD           ← Include only if using DISK option
// EXTENT  SYS010,nnnnnn
// ASSGN   SYS010,DISK,VOL=nnnnnn,SHR
// DLBL    SORTWK1,'WORK1',0,SD
// EXTENT  SYS001,nnnnnn,1,,ssss,200
// ASSGN   SYS001,DISK,VOL=nnnnnn,SHR
// DLBL    IDMSPCH,'dddl statements',99/365,SD
// EXTENT  SYS020,nnnnnn,1,,ssss,300
// ASSGN   SYS020,DISK,VOL=nnnnnn,SHR
// EXEC    PRANIDDG,SIZE=200K
optional control statements
/*
```

| | |
|-----------------|--|
| dddl statements | file-id of the file to contain DDDL statements |
| nnnnnn | serial number of disk volume |
| ssss | starting track (CKD) or block (FBA) or disk extent |
| sysref | file-id for sequential file containing data usage file |
| user.srclib | source statement library containing data usage files |
| SSLn | filename of source statement library |

Appendix A. Sample COBOL Input and DDDL Output

A.1 About this Appendix A-3

A.1 About this Appendix

This appendix shows sample input to and output from the Dictionary Loader, as follows:

- Input to the Program Processor — Three COBOL source programs
- Input to the DDDL Generator — The control statements used in running the DDDL Generator
- Output from the DDDL Generator — The DDDL statements generated by processing the data usage files associated with the three COBOL programs

Note that the other examples (that is, example reports) shown throughout this manual are all taken from the Dictionary Loader runs made to process the three programs listed below.

Sample COBOL programs

```

001000 ID DIVISION.
002000 PROGRAM-ID.          PRANDEM1.
003000 AUTHOR.              COMPUTER ASSOCIATES INTERNATIONAL, INC.
004000 REMARKS.             SAMPLE PROGRAM CONTAINING FILES
005000                        CUSTOMER-FILE, ORDER-FILE, AND RPTFILE.
006000                        CUSTOMER-FILE AND ORDER-FILE HAVE BEEN
007000                        SORTED ON CUSTOMER NUMBER. THIS
008000                        PROGRAM MATCHES ORDERS TO THE CUSTOMER
009000                        AND PRODUCES A REPORT OF ALL ORDERS
010000                        FOR ALL CUSTOMERS.
011000
012000 ENVIRONMENT DIVISION.
013000 INPUT-OUTPUT SECTION.
014000 FILE-CONTROL.
015000     SELECT CUSTOMER-FILE ASSIGN UT-2400-S-CUSTIN.
016000     SELECT ORDER-FILE   ASSIGN UT-2400-S-ORDERIN.
017000     SELECT RPTFILE     ASSIGN UT-S-SYSLST.
018000
019000 DATA DIVISION.
020000 FILE SECTION.
021000 FD  RPTFILE
022000     RECORDING MODE F
023000     LABEL RECORDS ARE OMITTED
024000     RECORD CONTAINS 133
025000     DATA RECORDS ARE TITLE-REC  DETAIL-REC.
026000

```

```

027000 01 TITLE-REC                PIC X(133).
028000 01 DETAIL-REC.
029000    05 FILLER                PIC X.
030000    05 RPT-CUST-NO           PIC X(10).
031000    05 FILLER                PIC XXX.
032000    05 RPT-NAME             PIC X(20).
033000    05 FILLER                PIC X(5).
034000    05 RPT-ORD-IDENT.
035000        10 RPT-ORD           PIC X(7).
036000        10 FILLER           PIC XXX.
037000    05 RPT-DATE-REQ         PIC X(8).
038000    05 FILLER                PIC X(76).
039000
040000 FD CUSTOMER-FILE
041000    RECORDING MODE F
042000    LABEL RECORDS ARE OMITTED
043000    RECORD CONTAINS 104 CHARACTERS
044000    DATA RECORD IS CUSTOMER.
045000
046000    01 CUSTOMER.
047000        03 CUST-NUMBER       PIC X(10).
048000        03 CUST-NAME        PIC X(20).
049000        03 CUST-ADDRESS.
050000            05 CUST-ADDR1    PIC X(20).
051000            05 CUST-ADDR2.
052000            06 CUST-CITY     PIC X(15).
053000            06 CUST-ZIP-CODE PIC X(5).
054000            06 CUST-ZIPCODE  REDEFINES CUST-ZIP-CODE
055000                                PIC 9(5).
056000        03 CUST-CREDIT      PIC XXX.
057000            88 CUST-CREDIT-EXEC VALUE IS 'AAA'.
058000            88 CUST-CREDIT-GOOD VALUE IS ' '.
059000            88 CUST-CREDIT-POOR VALUE IS 'XXX'.
060000        03 CUST-SALES-INFO.
061000            05 CUST-SALES-QTR OCCURS 4 TIMES.
062000            06 CUST-NUM-SALES PIC 9(5)          COMP-3.
063000            06 CUST-AMT-SALES PIC S9(7)        COMP-3.
064000        03 FILLER           PIC XXX.
065000
066000
067000 FD ORDER-FILE
068000    RECORDING MODE F
069000    LABEL RECORDS ARE OMITTED
070000    RECORD CONTAINS 50 CHARACTERS
071000    BLOCK CONTAINS 100 RECORDS
072000    DATA RECORD IS ORDOR.
073000

```

```

074000 01 ORDOR.
075000 03 ORD-CUST-NUMBER PIC X(10).
076000 03 ORD-NUMBER PIC X(7).
077000 03 ORD-CUST-PO-NUMB PIC X(10).
078000 03 ORD-DATES.
079000 05 ORD-REQ-DATE PIC X(6).
080000 05 ORD-DATE-REQ REDEFINES ORD-REQ-DATE
081000 PIC 9(6).
082000 05 ORD-PROM-DATE PIC X(6).
083000 05 ORD-DATE-PROM REDEFINES ORD-PROM-DATE
084000 PIC 9(6).
085000 05 ORD-SHIPPED-DATE PIC X(6).
086000 05 ORD-DATE-SHIPPED REDEFINES ORD-SHIPPED-DATE
087000 PIC 9(6).
088000 03 ORD-SHIP-CODE PIC XX.
089000 88 ORD-SHIP-ALL VALUE IS 'AS'.
090000 88 ORD-SHIP-PART VALUE IS 'PS'.
091000 03 FILLER PIC XXX.
092000
093000
094000 WORKING-STORAGE SECTION.
095000
096000 01 PAGE-COUNT-WS PIC S99 VALUE +0.
097000 01 POSITION-IND-WS PIC X.
098000 01 PAGE-INCREMENT-WS PIC 9.
099000
100000 01 DATE-AS-INPUT-WS.
101000 05 INPUT-YY-WS PIC 99.
102000 05 INPUT-MM-WS PIC 99.
103000 05 INPUT-DD-WS PIC 99.
104000 01 DATE-FORMATTED-WS.
105000 05 FORMATTED-MM-WS PIC 99.
106000 05 FILLER PIC X VALUE '/'.
107000 05 FORMATTED-DD-WS PIC 99.
108000 05 FILLER PIC X VALUE '/'.
109000 05 FORMATTED-YY-WS PIC 99.
110000
111000 01 TITLE-1-WS.
112000 05 FILLER PIC X(52) VALUE SPACES.
113000 05 FILLER PIC X(29) VALUE
114000 'ORDER INFORMATION BY CUSTOMER'.
115000 05 FILLER PIC X(52) VALUE SPACES.
116000 01 TITLE-2-WS.
117000 05 FILLER PIC X(18) VALUE ' CUSTOMER NO '.
118000 05 FILLER PIC X(22) VALUE 'CUSTOMER NAME '.
119000 05 FILLER PIC X(9) VALUE 'ORDER '.
120000 05 FILLER PIC X(12) VALUE 'DATE REQ '.
121000 05 FILLER PIC X(72) VALUE SPACES.
122000
123000
124000 PROCEDURE DIVISION.
125000
126000 0100-HOUSEKEEPING.
127000 OPEN INPUT CUSTOMER-FILE.
128000 OPEN INPUT ORDER-FILE.
129000 OPEN OUTPUT RPTFILE.
130000 MOVE SPACES TO DETAIL-REC.
131000 MOVE SPACES TO CUST-NUMBER.
132000

```

```
133000 0200-GET-ORDER-INFO.
134000     READ ORDER-FILE RECORD
135000         AT END GO TO 9200-E0J.
136000
137000 0300-GET-CUST-INFO.
138000     IF ORD-CUST-NUMBER = CUST-NUMBER
139000         GO TO 0500-GET-ORDER-INFO.
140000
141000     READ CUSTOMER-FILE RECORD
142000         AT END GO TO 9200-E0J.
143000
144000     MOVE CUST-NUMBER TO RPT-CUST-NO.
145000     MOVE CUST-NAME TO RPT-NAME.
146000
147000 0500-GET-ORDER-INFO.
148000     MOVE ORD-NUMBER TO RPT-ORD.
149000     MOVE ORD-DATE-REQ TO DATE-AS-INPUT-WS.
150000     MOVE INPUT-YY-WS TO FORMATTED-YY-WS.
151000     MOVE INPUT-MM-WS TO FORMATTED-MM-WS.
152000     MOVE INPUT-DD-WS TO FORMATTED-DD-WS.
153000     MOVE DATE-FORMATTED-WS TO RPT-DATE-REQ.
154000
155000     PERFORM 9000-WRITE THRU 9010-EXIT.
156000     GO TO 0300-GET-CUST-INFO.
157000
158000*     THIS PARAGRAPH CAUSES A REPORT FILE RECORD TO BE WRITTEN.
159000*     IT CONTROLS SPACING AND PAGING OF THE REPORT.
160000
161000 9000-WRITE.
162000     MOVE ' ' TO POSITION-IND-WS.
163000     MOVE 1 TO PAGE-INCREMENT-WS.
164000     IF RPT-ORD NOT = SPACES MOVE '0' TO POSITION-IND-WS
165000         MOVE 2 TO PAGE-INCREMENT-WS.
166000     IF RPT-CUST-NO NOT = SPACES MOVE '-' TO POSITION-IND-WS
167000         MOVE 3 TO PAGE-INCREMENT-WS.
168000     WRITE DETAIL-REC AFTER POSITIONING POSITION-IND-WS.
169000     MOVE SPACES TO DETAIL-REC.
170000     ADD PAGE-INCREMENT-WS TO PAGE-COUNT-WS.
171000     IF PAGE-COUNT-WS GREATER THAN +58
172000         PERFORM 9100-NEW-PAGE THRU 9110-EXIT.
173000 9010-EXIT.
174000     EXIT.
175000
176000 9100-NEW-PAGE.
177000     WRITE TITLE-REC FROM TITLE-1-WS AFTER POSITIONING 0.
178000     MOVE SPACES TO TITLE-REC.
179000     WRITE TITLE-REC FROM TITLE-2-WS AFTER POSITIONING 3.
180000     MOVE SPACES TO TITLE-REC.
181000     MOVE +4 TO PAGE-COUNT-WS.
182000 9110-EXIT.
183000     EXIT.
184000
185000*     CLOSE THE FILES AND EXIT FROM THE PROGRAM.
186000
```

```

187000 9200-E0J.
188000     CLOSE CUSTOMER-FILE.
189000     CLOSE ORDER-FILE.
190000     CLOSE RPTFILE.
191000 9210-EXIT.
192000     STOP RUN.
193000
001000 ID DIVISION.
002000 PROGRAM-ID.           PRANDEM2.
003000 AUTHOR.               COMPUTER ASSOCIATES INTERNATIONAL, INC.
004000 REMARKS.              SAMPLE PROGRAM CONTAINING FILE
005000                        CUSTOMER-FILE. THIS PROGRAM PRODUCES
006000                        A REPORT OF ALL CUSTOMERS WITH A
007000                        CREDIT RATING OF EXCELLENT.
008000
009000 ENVIRONMENT DIVISION.
010000 INPUT-OUTPUT SECTION.
011000 FILE-CONTROL.
012000     SELECT CUSTOMER-FILE ASSIGN UT-2400-S-CUSTIN.
013000     SELECT RPTFILE      ASSIGN UT-S-SYSLST.
014000
015000 DATA DIVISION.
016000 FILE SECTION.
017000 FD  RPTFILE
018000     RECORDING MODE F
019000     LABEL RECORDS ARE OMITTED
020000     RECORD CONTAINS 133
021000     DATA RECORDS ARE TITLE-REC  DETAIL-REC.
022000
023000 01  TITLE-REC          PIC X(133).
024000 01  DETAIL-REC.
025000     05  FILLER          PIC X.
026000     05  RPT-CUST-NO     PIC X(10).
027000     05  FILLER          PIC XXX.
028000     05  RPT-CUST-NAME   PIC X(20).
029000     05  FILLER          PIC X(10).
030000     05  RPT-ADDR1      PIC X(20).
031000     05  FILLER          PIC X(5).
032000     05  RPT-ADDR2      PIC X(20).
033000     05  FILLER          PIC X(5).
034000     05  RPT-ZIP        PIC X(20).
035000     05  FILLER          PIC X(19).
036000
037000 FD  CUSTOMER-FILE
038000     RECORDING MODE F
039000     LABEL RECORDS ARE OMITTED
040000     RECORD CONTAINS 104 CHARACTERS
041000     DATA RECORD IS CUSTOMER.
042000
043000 01  CUSTOMER.
044000     03  CUST-NUM        PIC X(10).
045000     03  CUST-NAME       PIC X(20).
046000     03  CUST-ADDRESS.
047000         05  CUST-ADDR1    PIC X(20).
048000         05  CUST-ADDR2.
049000             06  CUST-CITY    PIC X(15).
050000             06  CUST-ZIP-CODE PIC X(5).
051000     03  CUST-CREDIT     PIC XXX.
052000     88  CUST-CREDIT-EXEC VALUE IS 'AAA'.

```

A.1 About this Appendix

```

053000      88 CUST-CREDIT-GOOD      VALUE IS ' '.
054000      88 CUST-CREDIT-POOR     VALUE IS 'XXX'.
055000      03 FILLER                PIC X(31).
056000
057000
058000 WORKING-STORAGE SECTION.
059000
060000 01 PAGE-COUNT-WS              PIC S99      VALUE +0.
061000 01 POSITION-IND-WS             PIC X.
062000 01 PAGE-INCREMENT-WS        PIC 9.
063000
064000 01 TITLE-1-WS.
065000      05 FILLER                PIC X(46)      VALUE SPACES.
066000      05 FILLER                PIC X(41) VALUE
067000          'CUSTOMERS WITH AN EXCELLENT CREDIT RATING'.
068000      05 FILLER                PIC X(46)      VALUE SPACES.
069000 01 TITLE-2-WS.
070000      05 FILLER                PIC X(18) VALUE ' CUSTOMER NO      '.
071000      05 FILLER                PIC X(22) VALUE 'CUSTOMER NAME      '.
072000      05 FILLER                PIC X(5)  VALUE SPACES.
073000      05 FILLER                PIC X(9)  VALUE 'ADDRESS '.
074000      05 FILLER                PIC X(79) VALUE SPACES.
075000
076000
077000 PROCEDURE DIVISION.
078000
079000      OPEN INPUT  CUSTOMER-FILE.
080000      OPEN OUTPUT RPTFILE.
081000      MOVE SPACES TO DETAIL-REC.
082000
083000 0300-GET-CUST-INFO.
084000      READ CUSTOMER-FILE RECORD
085000          AT END GO TO 9200-EOJ.
086000
087000      IF NOT CUST-CREDIT-EXEC GO TO 0300-GET-CUST-INFO.
088000
089000
090000      MOVE CUST-NUM      TO RPT-CUST-NO.
091000      MOVE CUST-NAME     TO RPT-CUST-NAME.
092000      MOVE CUST-ADDR1   TO RPT-ADDR1.
093000      MOVE CUST-ADDR2   TO RPT-ADDR2.
094000      MOVE CUST-ZIP-CODE TO RPT-ZIP.
095000
096000      PERFORM 9000-WRITE THRU 9010-EXIT.
097000      GO TO 0300-GET-CUST-INFO.
098000
099000*      THIS PARAGRAPH CAUSES A REPORT FILE RECORD TO BE WRITTEN.
100000*      IT CONTROLS SPACING AND PAGING OF THE REPORT.
101000
102000 9000-WRITE.
103000      MOVE ' ' TO POSITION-IND-WS.
104000      MOVE 1 TO PAGE-INCREMENT-WS.
105000      WRITE DETAIL-REC AFTER POSITIONING POSITION-IND-WS.
106000      MOVE SPACES TO DETAIL-REC.
107000      ADD PAGE-INCREMENT-WS TO PAGE-COUNT-WS.
108000      IF PAGE-COUNT-WS GREATER THAN +58
109000          PERFORM 9100-NEW-PAGE THRU 9110-EXIT.
110000 9010-EXIT.
111000      EXIT.

```

```
112000
113000 9100-NEW-PAGE.
114000 WRITE TITLE-REC FROM TITLE-1-WS AFTER POSITIONING 0.
115000 MOVE SPACES TO TITLE-REC.
116000 WRITE TITLE-REC FROM TITLE-2-WS AFTER POSITIONING 3.
117000 MOVE SPACES TO TITLE-REC.
118000 MOVE +4 TO PAGE-COUNT-WS.
119000 9110-EXIT.
120000 EXIT.
121000
122000* CLOSE THE FILES AND EXIT FROM THE PROGRAM.
123000
124000 9200-E0J.
125000 CLOSE CUSTOMER-FILE.
126000 CLOSE RPTFILE.
127000 9210-EXIT.
128000 STOP RUN.
129000
001000 ID DIVISION.
002000 PROGRAM-ID. PRANDEM2.
003000 AUTHOR. COMPUTER ASSOCIATES INTERNATIONAL, INC.
004000 REMARKS. SAMPLE PROGRAM CONTAINING FILE
005000 CUSTOMER-FILE. THIS PROGRAM PRODUCES
006000 A REPORT OF ALL CUSTOMERS WITH A
007000 CREDIT RATING OF EXCELLENT.
008000
009000 ENVIRONMENT DIVISION.
010000 INPUT-OUTPUT SECTION.
011000 FILE-CONTROL.
012000 SELECT CUSTOMER-FILE ASSIGN UT-2400-S-CUSTIN.
013000 SELECT RPTFILE ASSIGN UT-S-SYSLST.
014000
015000 DATA DIVISION.
016000 FILE SECTION.
017000 FD RPTFILE
018000 RECORDING MODE F
019000 LABEL RECORDS ARE OMITTED
020000 RECORD CONTAINS 133
021000 DATA RECORDS ARE TITLE-REC DETAIL-REC.
022000
023000 01 TITLE-REC PIC X(133).
024000 01 DETAIL-REC.
025000 05 FILLER PIC X.
026000 05 RPT-CUST-NO PIC X(10).
027000 05 FILLER PIC XXX.
028000 05 RPT-CUST-NAME PIC X(20).
029000 05 FILLER PIC X(10).
030000 05 RPT-ADDR1 PIC X(20).
031000 05 FILLER PIC X(5).
032000 05 RPT-ADDR2 PIC X(20).
033000 05 FILLER PIC X(5).
034000 05 RPT-ZIP PIC X(20).
035000 05 FILLER PIC X(19).
036000
037000 FD CUSTOMER-FILE
038000 RECORDING MODE F
039000 LABEL RECORDS ARE OMITTED
040000 RECORD CONTAINS 104 CHARACTERS
041000 DATA RECORD IS CUSTOMER.
```

```
042000
043000 01 CUSTOMER.
044000 03 CUST-NUM          PIC X(10).
045000 03 CUST-NAME        PIC X(20).
046000 03 CUST-ADDRESS.
047000 05 CUST-ADDR1       PIC X(20).
048000 05 CUST-ADDR2.
049000 06 CUST-CITY        PIC X(15).
050000 06 CUST-ZIP-CODE    PIC X(5).
051000 03 CUST-CREDIT      PIC XXX.
052000 88 CUST-CREDIT-EXEC VALUE IS 'AAA'.
053000 88 CUST-CREDIT-GOOD VALUE IS ' '.
054000 88 CUST-CREDIT-POOR VALUE IS 'XXX'.
055000 03 FILLER           PIC X(31).
056000
057000
058000 WORKING-STORAGE SECTION.
059000
060000 01 PAGE-COUNT-WS    PIC S99          VALUE +0.
061000 01 POSITION-IND-WS   PIC X.
062000 01 PAGE-INCREMENT-WS PIC 9.
063000
064000 01 TITLE-1-WS.
065000 05 FILLER           PIC X(46)          VALUE SPACES.
066000 05 FILLER           PIC X(41)          VALUE
067000 'CUSTOMERS WITH AN EXCELLENT CREDIT RATING'.
068000 05 FILLER           PIC X(46)          VALUE SPACES.
069000 01 TITLE-2-WS.
070000 05 FILLER           PIC X(18)          VALUE ' CUSTOMER NO      '.
071000 05 FILLER           PIC X(22)          VALUE 'CUSTOMER NAME      '.
072000 05 FILLER           PIC X(5)           VALUE SPACES.
073000 05 FILLER           PIC X(9)           VALUE 'ADDRESS  '.
074000 05 FILLER           PIC X(79)          VALUE SPACES.
075000
076000
077000 PROCEDURE DIVISION.
078000
079000 OPEN INPUT  CUSTOMER-FILE.
080000 OPEN OUTPUT RPTFILE.
081000 MOVE SPACES TO DETAIL-REC.
082000
083000 0300-GET-CUST-INFO.
084000 READ CUSTOMER-FILE RECORD
085000 AT END GO TO 9200-E0J.
086000
087000 IF NOT CUST-CREDIT-EXEC GO TO 0300-GET-CUST-INFO.
088000
089000
090000 MOVE CUST-NUM      TO RPT-CUST-NO.
091000 MOVE CUST-NAME    TO RPT-CUST-NAME.
092000 MOVE CUST-ADDR1   TO RPT-ADDR1.
093000 MOVE CUST-ADDR2   TO RPT-ADDR2.
094000 MOVE CUST-ZIP-CODE TO RPT-ZIP.
095000
096000 PERFORM 9000-WRITE THRU 9010-EXIT.
097000 GO TO 0300-GET-CUST-INFO.
```

```
098000
099000*   THIS PARAGRAPH CAUSES A REPORT FILE RECORD TO BE WRITTEN.
100000*   IT CONTROLS SPACING AND PAGING OF THE REPORT.
101000
102000 9000-WRITE.
103000   MOVE   ' ' TO POSITION-IND-WS.
104000   MOVE   1 TO PAGE-INCREMENT-WS.
105000   WRITE  DETAIL-REC AFTER POSITIONING POSITION-IND-WS.
106000   MOVE   SPACES TO DETAIL-REC.
107000   ADD    PAGE-INCREMENT-WS TO PAGE-COUNT-WS.
108000   IF     PAGE-COUNT-WS GREATER THAN +58
109000           PERFORM 9100-NEW-PAGE THRU 9110-EXIT.
110000 9010-EXIT.
111000   EXIT.
112000
113000 9100-NEW-PAGE.
114000   WRITE  TITLE-REC FROM TITLE-1-WS AFTER POSITIONING 0.
115000   MOVE  SPACES TO TITLE-REC.
116000   WRITE  TITLE-REC FROM TITLE-2-WS AFTER POSITIONING 3.
117000   MOVE  SPACES TO TITLE-REC.
118000   MOVE   +4 TO PAGE-COUNT-WS.
119000 9110-EXIT.
120000   EXIT.
121000
122000*   CLOSE THE FILES AND EXIT FROM THE PROGRAM.
123000
124000 9200-EQJ.
125000   CLOSE CUSTOMER-FILE.
126000   CLOSE RPTFILE.
127000 9210-EXIT.
128000   STOP RUN.
129000
```

Sample control statements for the DDDL Generator

```
VERSION NEXT HIGHEST
FILE-GROUPING  CUSTOMER-FILE IN PRANDEM1 IN PRANDEM2
CUSTFILE IN PRANDEM3
RECORD-GROUPING CUSTOMER      IN PRANDEM1 IN PRANDEM2
CUST          IN PRANDEM3
```

Sample generated DDDL statements

```
ADD FILE CUSTOMER-FILE VERSION NEXT HIGHEST
LABELS ARE OMITTED
RECORD SIZE IS 104
RECORDING MODE IS F
FILE NAME SYNONYM IS CUSTFILE VERSION NEXT HIGHEST.
```

```
ADD RECORD CUSTOMER VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE CUSTOMER-FILE VERSION HIGHEST
RECORD NAME SYNONYM IS CUST VERSION NEXT HIGHEST.
  03 CUST-NUMBER    PIC X(10).
  03 CUST-NAME     PIC X(20).
  03 CUST-ADDRESS.
    05 CUST-ADDR1  PIC X(20).
    05 CUST-ADDR2.
      06 CUST-CITY
         PIC X(15).
      06 CUST-ZIP-CODE
         PIC X(5).
      06 CUST-ZIPCODE
         REDEFINES CUST-ZIP-CODE
         PIC 9(5).
  03 CUST-CREDIT   PIC XXX.
    88 CUST-CREDIT-EXEC
       VALUE 'AAA'.
    88 CUST-CREDIT-GOOD
       VALUE ' '.
    88 CUST-CREDIT-POOR
       VALUE 'XXX'.
  03 CUST-SALES-INFO.
    05 CUST-SALES-QTR
       OCCURS 4.
      06 CUST-NUM-SALES
         PIC 9(5) COMP-3.
      06 CUST-AMT-SALES
         PIC S9(7) COMP-3.
  03 FILLER       PIC XXX.
```

```
ADD FILE ORDER-FILE VERSION NEXT HIGHEST
LABELS ARE OMITTED
RECORD SIZE IS 50
RECORDING MODE IS F
BLOCK SIZE IS 5000.
```

```
ADD RECORD ORDOR VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE ORDER-FILE VERSION HIGHEST.
  03 ORD-CUST-NUMBER
     PIC X(10).
```

```
03 ORD-NUMBER      PIC X(7).
03 ORD-CUST-PO-NUMB
                   PIC X(10).
03 ORD-DATES.
05 ORD-REQ-DATE    PIC X(6).
05 ORD-DATE-REQ    REDEFINES ORD-REQ-DATE
                   PIC 9(6).
05 ORD-PROM-DATE   PIC X(6).
05 ORD-DATE-PROM   REDEFINES ORD-PROM-DATE
                   PIC 9(6).
05 ORD-SHIPPED-DATE
                   PIC X(6).
05 ORD-DATE-SHIPPED
                   REDEFINES ORD-SHIPPED-DATE
                   PIC 9(6).
03 ORD-SHIP-CODE   PIC XX.
88 ORD-SHIP-ALL    VALUE 'AS'.
88 ORD-SHIP-PART   VALUE 'PS'.
03 FILLER         PIC XXX.

ADD FILE RPTFILE VERSION NEXT HIGHEST
LABELS ARE OMITTED
RECORD SIZE IS 133
RECORDING MODE IS F.

ADD RECORD TITLE-REC VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE RPTFILE VERSION HIGHEST.
02 TITLE-REC      PIC X(133).

ADD RECORD DETAIL-REC VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE RPTFILE VERSION HIGHEST.
05 FILLER         PIC X.
05 RPT-CUST-NO    PIC X(10).
05 FILLER         PIC XXX.
05 RPT-NAME       PIC X(20).
05 FILLER         PIC X(5).
05 RPT-ORD-IDENT.
10 RPT-ORD        PIC X(7).
10 FILLER         PIC XXX.
05 RPT-DATE-REQ   PIC X(8).
05 FILLER         PIC X(76).

ADD RECORD PAGE-COUNT-WS VERSION NEXT HIGHEST
LANGUAGE IS COBOL.
02 PAGE-COUNT-WS  PIC S99
                   VALUE +0.

ADD RECORD POSITION-IND-WS VERSION NEXT HIGHEST
LANGUAGE IS COBOL.
02 POSITION-IND-WS
                   PIC X.
```

```

ADD RECORD PAGE-INCREMENT-WS VERSION NEXT HIGHEST
LANGUAGE IS COBOL.
    02 PAGE-INCREMENT-WS
        PIC 9.

ADD RECORD DATE-AS-INPUT-WS VERSION NEXT HIGHEST
LANGUAGE IS COBOL.
    05 INPUT-YY-WS      PIC 99.
    05 INPUT-MM-WS      PIC 99.
    05 INPUT-DD-WS      PIC 99.

ADD RECORD DATE-FORMATTED-WS VERSION NEXT HIGHEST
LANGUAGE IS COBOL.
    05 FORMATTED-MM-WS
        PIC 99.
    05 FILLER           PIC X
        VALUE '/'.
    05 FORMATTED-DD-WS
        PIC 99.
    05 FILLER           PIC X
        VALUE '/'.
    05 FORMATTED-YY-WS
        PIC 99.

ADD RECORD TITLE-1-WS VERSION NEXT HIGHEST
LANGUAGE IS COBOL.
    05 FILLER           PIC X(52)
        VALUE SPACES.
    05 FILLER           PIC X(29)
        VALUE
            'ORDER INFORMATION BY CUSTOMER'.
    05 FILLER           PIC X(52)
        VALUE SPACES.

ADD RECORD TITLE-2-WS VERSION NEXT HIGHEST
LANGUAGE IS COBOL.
    05 FILLER           PIC X(18)
        VALUE ' CUSTOMER NO      '.
    05 FILLER           PIC X(22)
        VALUE 'CUSTOMER NAME    '.
    05 FILLER           PIC X(9)
        VALUE 'ORDER           '.
    05 FILLER           PIC X(12)
        VALUE 'DATE REQ        '.
    05 FILLER           PIC X(72)
        VALUE SPACES.

ADD PROGRAM PRANDEM1 VERSION NEXT HIGHEST
LANGUAGE IS COBOL
ESTIMATED LINES ARE 195
INPUT FILE IS CUSTOMER-FILE VERSION HIGHEST
INPUT FILE IS ORDER-FILE VERSION HIGHEST
OUTPUT FILE IS RPTFILE VERSION HIGHEST
RECORD USED IS CUSTOMER VERSION HIGHEST
ELEMENT IS CUST-NUMBER
    REFERENCED 2 TIMES
    MODIFIED 1 TIME
ELEMENT IS CUST-NAME

```

```
REFERENCED 1 TIME
ELEMENT IS CUST-ADDRESS
ELEMENT IS CUST-ADDR1
ELEMENT IS CUST-ADDR2
ELEMENT IS CUST-CITY
ELEMENT IS CUST-ZIP-CODE
ELEMENT IS CUST-ZIPCODE
ELEMENT IS CUST-CREDIT
ELEMENT IS CUST-SALES-INFO
ELEMENT IS CUST-SALES-QTR
ELEMENT IS CUST-NUM-SALES
ELEMENT IS CUST-AMT-SALES
RECORD USED IS ORDOR VERSION HIGHEST
ELEMENT IS ORD-CUST-NUMBER
REFERENCED 1 TIME
ELEMENT IS ORD-NUMBER
REFERENCED 1 TIME
ELEMENT IS ORD-CUST-PO-NUMB
ELEMENT IS ORD-DATES
ELEMENT IS ORD-REQ-DATE
ELEMENT IS ORD-DATE-REQ
REFERENCED 1 TIME
ELEMENT IS ORD-PROM-DATE
ELEMENT IS ORD-DATE-PROM
ELEMENT IS ORD-SHIPPED-DATE
ELEMENT IS ORD-DATE-SHIPED
ELEMENT IS ORD-SHIP-CODE
RECORD USED IS TITLE-REC VERSION HIGHEST
MODIFIED 4 TIMES
RECORD USED IS DETAIL-REC VERSION HIGHEST
MODIFIED 3 TIMES
ELEMENT IS RPT-CUST-NO
REFERENCED 1 TIME
MODIFIED 1 TIME
ELEMENT IS RPT-NAME
MODIFIED 1 TIME
ELEMENT IS RPT-ORD-IDENT
ELEMENT IS RPT-ORD
REFERENCED 1 TIME
MODIFIED 1 TIME
ELEMENT IS RPT-DATE-REQ
MODIFIED 1 TIME
RECORD USED IS PAGE-COUNT-WS VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 2 TIMES
RECORD USED IS POSITION-IND-WS VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 3 TIMES
RECORD USED IS PAGE-INCREMENT-WS VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 3 TIMES
RECORD USED IS DATE-AS-INPUT-WS VERSION HIGHEST
MODIFIED 1 TIME
ELEMENT IS INPUT-YY-WS
REFERENCED 1 TIME
ELEMENT IS INPUT-MM-WS
```

```
REFERENCED 1 TIME
ELEMENT IS INPUT-DD-WS
REFERENCED 1 TIME
RECORD USED IS DATE-FORMATTED-WS VERSION HIGHEST
REFERENCED 1 TIME
ELEMENT IS FORMATTED-MM-WS
MODIFIED 1 TIME
ELEMENT IS FORMATTED-DD-WS
MODIFIED 1 TIME
ELEMENT IS FORMATTED-YY-WS
MODIFIED 1 TIME
RECORD USED IS TITLE-1-WS VERSION HIGHEST
REFERENCED 1 TIME
RECORD USED IS TITLE-2-WS VERSION HIGHEST
REFERENCED 1 TIME.

ADD PROGRAM PRANDEM2 VERSION NEXT HIGHEST
LANGUAGE IS COBOL
ESTIMATED LINES ARE 131
INPUT FILE IS CUSTOMER-FILE VERSION HIGHEST
OUTPUT FILE IS RPTFILE VERSION HIGHEST
RECORD USED IS CUSTOMER VERSION HIGHEST
ELEMENT IS CUST-NUM
REFERENCED 1 TIME
ELEMENT IS CUST-NAME
REFERENCED 1 TIME
ELEMENT IS CUST-ADDRESS
ELEMENT IS CUST-ADDR1
REFERENCED 1 TIME
ELEMENT IS CUST-ADDR2
REFERENCED 1 TIME
ELEMENT IS CUST-CITY
ELEMENT IS CUST-ZIP-CODE
REFERENCED 1 TIME
ELEMENT IS CUST-CREDIT
RECORD USED IS TITLE-REC VERSION HIGHEST
MODIFIED 4 TIMES
RECORD USED IS DETAIL-REC VERSION HIGHEST
MODIFIED 3 TIMES
ELEMENT IS RPT-CUST-NO
MODIFIED 1 TIME
ELEMENT IS RPT-CUST-NAME
MODIFIED 1 TIME
ELEMENT IS RPT-ADDR1
MODIFIED 1 TIME
ELEMENT IS RPT-ADDR2
MODIFIED 1 TIME
ELEMENT IS RPT-ZIP
MODIFIED 1 TIME
RECORD USED IS PAGE-COUNT-WS VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 2 TIMES
RECORD USED IS POSITION-IND-WS VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 1 TIME
```

```
RECORD USED IS PAGE-INCREMENT-WS VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 1 TIME
RECORD USED IS TITLE-1-WS VERSION HIGHEST
REFERENCED 1 TIME
RECORD USED IS TITLE-2-WS VERSION HIGHEST
REFERENCED 1 TIME.

ADD FILE CUSTFILE VERSION NEXT HIGHEST
LABELS ARE OMITTED
RECORD SIZE IS 104
RECORDING MODE IS F
FILE NAME SYNONYM IS CUSTOMER-FILE VERSION NEXT HIGHEST.

ADD RECORD CUST VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE CUSTFILE VERSION HIGHEST
RECORD NAME SYNONYM IS CUSTOMER VERSION NEXT HIGHEST.
  03 FILLER          PIC X(10).
  03 CUST-NAME      PIC X(20).
  03 CUST-ADDRESS.
      05 CUST-ADDR1 PIC X(20).
      05 CUST-ADDR2 PIC X(20).
  03 FILLER          PIC X(34).

ADD FILE MAILFILE VERSION NEXT HIGHEST
LABELS ARE OMITTED
RECORD SIZE IS 21
RECORDING MODE IS F.

ADD RECORD MAIL-REC-1 VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE MAILFILE VERSION HIGHEST.
  03 FILLER          PIC X.
  03 MAIL-LINE-1    PIC X(20).

ADD RECORD MAIL-REC-2 VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE MAILFILE VERSION HIGHEST.
  03 FILLER          PIC X.
  03 MAIL-LINE-2    PIC X(20).

ADD RECORD MAIL-REC-3 VERSION NEXT HIGHEST
LANGUAGE IS COBOL
WITHIN FILE MAILFILE VERSION HIGHEST.
  03 FILLER          PIC X.
  03 MAIL-LINE-3    PIC X(20).

ADD PROGRAM PRANDEM3 VERSION NEXT HIGHEST
LANGUAGE IS COBOL
ESTIMATED LINES ARE 81
INPUT FILE IS CUSTFILE VERSION HIGHEST
OUTPUT FILE IS MAILFILE VERSION HIGHEST
RECORD USED IS CUST VERSION HIGHEST
ELEMENT IS CUST-NAME
REFERENCED 1 TIME
ELEMENT IS CUST-ADDRESS
ELEMENT IS CUST-ADDR1
```

REFERENCED 1 TIME
ELEMENT IS CUST-ADDR2
REFERENCED 1 TIME
RECORD USED IS MAIL-REC-1 VERSION HIGHEST
MODIFIED 2 TIMES
ELEMENT IS MAIL-LINE-1
MODIFIED 1 TIME
RECORD USED IS MAIL-REC-2 VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 1 TIME
ELEMENT IS MAIL-LINE-2
MODIFIED 1 TIME
RECORD USED IS MAIL-REC-3 VERSION HIGHEST
REFERENCED 1 TIME
MODIFIED 1 TIME
ELEMENT IS MAIL-LINE-3
MODIFIED 1 TIME.

Appendix B. Runtime Error Messages

| | |
|---|-----|
| B.1 About Runtime Error Messages | B-3 |
| B.2 Runtime Messages Issued by the Program Processor | B-4 |
| B.3 Runtime Message Issued by the Cross Reference Processor | B-6 |
| B.4 Runtime Messages Issued by the DDDL Generator | B-8 |

B.1 About Runtime Error Messages

Where messages appear: This appendix documents the runtime messages issued by the three Dictionary Loader components. These runtime messages can indicate fatal or nonfatal conditions and appear in the console log or the printed output for the run:

| | |
|-----------------------------------|--|
| Program Processor Messages | Both nonfatal and fatal messages appear on the console log |
| Cross Reference Processor Message | Nonfatal messages appear at the beginning of the report output for the run; fatal messages appear on the console log |
| DDDL Generator Messages | Nonfatal messages appear at the beginning of the report output for the run; fatal messages appear on the console log |

Nonfatal messages: The nonfatal messages issued by the Dictionary Loader components mainly identify errors in the control statement information. When one of the components detects a nonfatal error condition, the component issues the appropriate message and continues running.

Fatal messages: The fatal messages issued by the Dictionary Loader components flag two types of error conditions:

- I/O errors (most commonly INPUT/OUTPUT FILE WILL NOT OPEN)
- Internal errors from CA-IDMS utility programs

Consequence of a fatal error: When one of the Dictionary Loader components detects a fatal condition, the component issues a write-to-operator message (which appears on the console log) and terminates the run.

B.2 Runtime Messages Issued by the Program Processor

Nonfatal

1.

'keyword' INVALID PARAMETER

The reported keyword is an invalid specification.

Fatal

1.

BAD IDMSUTIO RETURN CODE - PARMINTF - return-code - CSFCDSP

IDMSUTIO issued the reported return code; the job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

2.

FATAL ERROR - BAD IDMSFLIO RETURN CODE - return-code - SSRFIO

IDMSFLIO issued the reported return code; the job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

3.

FATAL ERROR - BAD IDMSUTIO RETURN CODE - return-code - SSRPIO

IDMSUTIO issued the reported return code; the job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

4.

FATAL ERROR - CBIO - INVALID OPERATION - SSCBIO

The job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

5.

FATAL ERROR - RPIO - INVALID OPERATION -- SSRPIO

The job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

6.

INPUT FILE SYSIPT WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSIPT is specified correctly.

7.

MEMBER NOT FOUND IN LIBRARY (VSE/ESA users only)

The job ended with a user abend code of 100. This message is issued when the =COPY IDMS option is being used to read the input program from a library into the SYSIPT file and the library member cannot be accessed. Check that the correct member-name is specified in the =COPY IDMS statement and that the sublibrary name is specified if necessary (the default is the A. sublibrary).

8.

OUTPUT FILE PRANREF WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if PRANREF is specified correctly.

9.

OUTPUT FILE SYSLST WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSLST is specified correctly.

10.

OUTPUT FILE SYSPCH WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSPCH is specified correctly.

B.3 Runtime Message Issued by the Cross Reference Processor

Nonfatal

1.

ILLEGAL ALIAS FOR PROG-ID

The name following the equal sign (=) in the LIBRARY option in the parameter statement is missing.

2.

ILLEGAL DELIMITER

Statement keywords are not delimited by the required comma or blank.

3.

ILLEGAL MEMBER NAME

Member name is missing as the operand of the LIBRARY option in the parameter statement.

4.

ILLEGAL PROGRAM-ID 'NEW NAME'

The name following the equal sign (=) in the PROGRAM-ID option in the parameter statement is missing.

5.

OPTION/SELECT NOT RECOGNIZED

A statement keyword is not valid as expressed.

6.

TOO MANY LIBRARY NAMES

More than 99 library members are specified.

7.

TOO MANY PROGRAM-IDS

More than 39 source program names are changed to new names in the PROGRAM-ID option.

Fatal

1.

BAD IDMSUTIO RETURN CODE - PARMINTF - return-code - CSFCDSP

IDMSUTIO issued the reported return code; the job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

2.

BAD RETURN CODE - module-name - return-code

The error originated in the named module (either IDMSUTIO or IDMSDLIO). The module issued the listed return code. The job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

3.

INPUT FILE PRANREF WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if PRANREF is specified correctly.

4.

INPUT FILE SYSIPT WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSIPT is specified correctly.

5.

MEMBER NOT FOUND IN LIBRARY (VSE/ESA users only)

The job ended with a user abend code of 100. Check the member name specified in the LIBRARY option.

6.

OUTPUT FILE SYSLST WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSLST is specified correctly.

Other fatal messages: Note that the Cross Reference Processor may issue an additional class of fatal messages. These messages are generated by the utility module IDMSORT and report system internal errors.

IDMSORT messages: IDMSORT error messages are write-to-operator messages that are displayed on the console log. When the Cross Reference Processor transmits an IDMSORT message, the run abends with a user abend code of 3134 and a two-line message appears in the following format:

```
+IDMS 999000L IDMSORT - message-text
+IDMS 208001L 0100
```

Examples: Four examples of message text that can appear in this message are shown below:

```
INVALID KEYWORD PASSED TO IDMSORT
UNSUPPORTED SORT CONTROL STATEMENT PASSED
NO keyword-type KEYWORD ON SORT record-name STATEMENT
NO keyword-length KEYWORD ON SORT record-name STATEMENT
```

Although the user cannot take corrective action to resolve IDMSORT error conditions (because such errors are system internal), the user can retry the run. In some cases, the internal error will disappear. If the error condition persists, consult with the person responsible for tracking system errors.

B.4 Runtime Messages Issued by the DDDL Generator

Nonfatal messages

1.

ILLEGAL ALIAS FOR PROG-ID

The name following the equal sign (=) in the LIBRARY option in the parameter statement is missing.

2.

ILLEGAL DELIMITER

Statement keywords are not delimited by the required comma or blank.

3.

ILLEGAL MEMBER NAME

The member name is missing as the operand of the LIBRARY option.

4.

LINE EXCEEDS 72 CHARACTERS

The length of a generated DDDL statement exceeds 72 characters. The statement must be edited by the user before being input to the DDDL compiler.

5.

MORE THAN 5 PROGRAMS IN GROUPING STATEMENT

A grouping statement specified more than five program names. The DDDL Generator accepted the first five, ignored the additional program names, and continued processing.

6.

MORE THAN 100 DIFFERENT PROGRAMS CALLED - TABLE EXCEEDED

A program being processed by the DDDL Generator called more than 100 other different programs. The DDDL Generator generated ADD PROGRAM statements for the first 100 programs called, ignored additional program calls, and continued processing.

7.

OPTION/SELECT NOT RECOGNIZED

A statement keyword is not valid as expressed.

8.

TOO MANY LIBRARY NAMES

More than 99 library members are specified.

Fatal

1.

BAD RETURN CODE - module-name - return-code

The error originated in the named module (either IDMSUTIO or IDMSFLIO). The module issued the listed return code. The job ended with a user abend code of 100. This message reports a system internal error; rerun the job.

2.

INPUT FILE PRANREF WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if PRANREF is specified correctly.

3.

INPUT FILE SYSIPT WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSIPT is specified correctly.

4.

MEMBER NOT FOUND IN LIBRARY (VSE/ESA users only)

The job ended with a user abend code of 100. Check the member name specified in the LIBRARY option.

5.

OUTPUT FILE SYSLST WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSLST is specified correctly.

6.

OUTPUT FILE SYSPCH WILL NOT OPEN

The job ended with a user abend code of 2000. Check the JCL to see if SYSPCH is specified correctly.

Additional fatal messages: Note that the DDDL Generator may issue an additional class of fatal messages. These messages are generated by the utility module IDMSORT and report system internal errors.

IDMSORT messages: IDMSORT error messages are write-to-operator messages that are displayed on the console and in the JES message listing. When the DDDL Generator transmits an IDMSORT message, the run abends with a user abend code of 3134 and a two-line message appears in the following format:

```
+IDMS 999000L IDMSORT - message-text
+IDMS 208001L 0100
```

Four examples of message text that can appear in the message are:

```
INVALID KEYWORD PASSED TO IDMSORT
UNSUPPORTED SORT CONTROL STATEMENT PASSED
NO keyword-type KEYWORD ON SORT record-name STATEMENT
NO keyword-length KEYWORD ON SORT record-name STATEMENT
```

Although the user cannot take corrective action to resolve IDMSORT error conditions (because such errors are system internal), the user can retry the run. In some cases,

the internal error will disappear. If the error condition persists, consult with the person responsible for tracking system errors.

Index

Special Characters

\$\$\$ diagnostic message 2-7

A

ANS,ANS68,ANS74 diagnostic message 2-8

C

COBOL input, samples A-3—A-11

control file

See also worksheet

cross reference processor, creating 3-7—3-9

cross reference processor, sample 3-18

DDDL generator, creating 4-7—4-8

worksheet, creating 3-8

cross reference processor

control file parameters, table of 3-7

control statement file 3-7

default processing options, table of 3-7

Dictionary of Data Names report 3-6, 3-23

executing 3-24—3-26

fatal runtime messages B-6—B-7

file control statements 3-9

general discussion 3-3—3-6

IDMSSORT runtime messages B-7

nonfatal runtime messages B-6

OS/390 JCL 3-24

output 3-5, 3-6

override processing options, table of 3-7

parameter statement 3-13—3-16

PRANXREF program 3-24

selection statement 3-18

System Data Cross-Reference report 3-5, 3-20

title statement 3-17

VSE/ESA JCL 3-25

worksheet, filling in 3-10—3-12

D

DATA DIVISION Cross-Reference report

and cross reference processor 3-10

field descriptions 2-13

sample 2-13

DDDL compiler 4-5

DDDL generator

clauses, table 4-6

control statement file 4-7

DDDL generator (*continued*)

editing generated statements 4-17—4-19

executing the compiler 4-20—4-22

fatal runtime messages B-8

general discussion 4-3—4-6

grouping statement 4-12—4-16

grouping statement examples 4-15—4-16

identifying nonunique names 4-14

identifying synonyms 4-14

IDMSSORT messages B-9

input 2-3, 4-5

nonfatal runtime messages B-8

operating with control statements 4-4

operating without control statements 4-4

OS/390 JCL 4-20

output 4-5

parameter statement 4-9—4-10

PRANIDDG program 4-20

sample control statements A-11

sample generated DDDL statements A-11—A-18

VERSION clause 4-11

VERSION statement 4-11

VERSION statement syntax 4-11

VSE/ESA JCL 4-21

DDDL output, samples A-3

diagnostic messages

\$\$\$ 2-7

ANS,ANS68,ANS74 2-8

FLO 2-8—2-10

Diagnostic report 2-9

messages 2-7—2-10

sample 2-7

types of problems flagged 2-9

Dictionary of Data Names report, sample 3-23

F

Fatal errors, consequences B-3

File and Record Layouts report

and cross reference processor 3-10

field descriptions 2-10, 2-11—2-12

sample 2-10

FLO diagnostic message 2-8

G

generator.grouping statement

See DDDL

generator.VERSION clause

See DDDL

generator.VERSION statement

See DDDL

grouping statement

coding rules 4-12

examples of usage 4-15—4-16

parameter list 4-12—4-13

sample 4-13

syntax 4-12

I

IDMSDLIO B-7

IDMSFLIO B-4

IDMSSORT runtime messages B-7, B-9

IDMSUTIO B-4, B-6

input program, program processor 2-4

J

JCL

for VSE/ESA source statement library 2-20

OS/390, cross reference processor 3-24—3-25

OS/390, DDDL compiler 4-20

OS/390, program processor 2-18—2-19

VSE/ESA, cross reference processor 3-25

VSE/ESA, DDDL compiler 4-21

VSE/ESA, program processor 2-19

M

Management Summary report, sample 2-6

O

OS/390

cross reference processor JCL 3-24

DDDL generator JCL 4-20

program processor JCL 2-18

program processor overrides 2-15

P

parameter statement

cross reference processor 3-13

DDDL generator 4-9—4-10

program processor 2-15, 2-17

PRANCOB program

for OS/390 2-18

for VSE/ESA 2-19

PRANIDDG program

for OS/390 4-20

for VSE/ESA 4-21

PRANXREF program

for OS/390 3-24

for VSE/ESA 3-25

PROCEDURE DIVISION, tracking use of 2-13

program processor

DATA DIVISION Cross-Reference report 2-13

data usage file 2-3

default runtime options, table of 2-4

diagnostic messages 2-7—2-10

Diagnostic report 2-7

executing 2-18—2-20

fatal runtime messages B-4—B-5

File and Record Layouts report 2-10—2-12

general description of 2-3

input 2-4—2-5

Management Summary report 2-6

nonfatal runtime messages B-4

OS/390 JCL 2-18

output 2-3, 2-6—2-14

override processing options 2-15

override runtime options, table of 2-4

parameter statement 2-15—2-17

PRANCOB program 2-18

reports

VSE/ESA considerations 2-4

VSE/ESA JCL 2-19

VSE/ESA source statement library 2-20

R

reports

DATA DIVISION Cross-Reference report 2-13

Diagnostic report 2-7

Dictionary of Data Names report 3-6, 3-23

Management Summary report 2-6

System Data Cross-Reference report 3-5, 3-20

S

selection statement

coding rules 3-18

parameter list 3-18

syntax 3-18

statement.DDDL generator

See grouping

SYSIPT B-4, B-7

SYSLST B-5, B-7

SYSPCH B-5
System Data Cross-Reference report
 field descriptions 3-20—3-22
 sample 3-5, 3-20

T

title statement, cross reference processor
 syntax 3-17

V

VERSION clause 4-11
VERSION statement 4-11
 parameter list 4-11
 syntax 4-11
VSE/ESA
 =COPY facility 2-4
 and program processor 2-4
 cross reference processor JCL 3-25
 DDDL generator JCL 4-21
 program processor JCL 2-19
 program processor overrides 2-15
 source statement library JCL 2-20

W

worksheet, control file
 and using program processor reports 3-10
 filling in 3-10—3-12
 lines on 3-10
 sample 3-9
 specifying selection criteria 3-10—3-11
 variables in selection statement, table 3-12

