UNIVAC 1100 SERIES

DATA MANAGEMENT SYSTEM (DMS 1100)

AMERICAN NATIONAL STANDARD COBOL (ASCII) DATA MANIPULATION LANGUAGE

PROGRAMMER REFERENCE

SPERRY UNIVAC

This document contains the latest information available at the time of publication. However, the Univac Division reserves the right to modify or revise its contents. To ensure that you have the most recent information contact your local Univac Representative.

UNIVAC is a registered trademark of the Sperry Rand Corporation.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP–7992

Section	Page Number	Update Level
Cover/Disclaimer		Orig.
PSS	1	Orig.
Acknowledgment	1	Orig.
Preface	1	Orig.
Contents	1 thru 3	Orig.
1	1	Orig.
2	1 thru 5	Orig.
3	1 thru 66	Orig.
4	1 thru 16	Orig.
Appendix A	1 thru 18	Orig.
Appendix B	1 thru 5	Orig.
Appendix C	1 thru 3	Orig.
Appendix D	1 thru 3	Orig.
Appendix E	1 thru 3	Orig.
Appendix F	1 thru 3	Orig.
Appendix G	1	Orig.
UCS		
TOTAL	Covers and 132 Pages	

Section	Page Number	Update Level

Section	Page Number	Update Level

ACKNOWLEDGMENT

Univac wishes to acknowledge the efforts of the CODASYL Programming Language Committee (PLC) Data Base Task Group (DBTG). The DBTG produced two reports containing specifications for a standardized Data Management facility containing a DATA DESCRIPTION LANGUAGE and a DATA MANIPULATION LANGUAGE. The DBTG issued its first report in October 1969, the second, revised and expanded, in April 1971. Univac is a member of the DBTG and participated in the development of the Data Management specifications. Univac's implementation of the 1100 Series Data Management System (DMS 1100) is based upon the DBTG specifications.

PREFACE

This document is the Programmer Reference Manual for the UNIVAC 1100 Series Data Management System (DMS 1100) AMERICAN NATIONAL STANDARD COBOL (ASCII) DATA MANIPULATION LANGUAGE. It is one of a series of manuals covering DMS 1100. It is intended that the Reference Manual be used to provide AMERICAN NATIONAL STANDARD COBOL (ASCII) manipulation of databases defined using the *DMS 1100 Schema Definition Data Administrator Reference, UP-7907* (current version).

CONTENTS

APPENDICES

1. INTRODUCTION

1.1. GENERAL

This document describes the Data Management System (DMS) 1100 Series reference manual for the American National Standard COBOL (ASCII) Data Manipulation Language (DML). The DML is not a standalone language (such as the Data Definition Language); the DML is dependent upon *UNIVAC 1100 Series American National Standard COBOL (ASCII) Programmer Reference, UP-7923* (current version) to serve as a host language.

1.2. COBOL DATA MANIPULATION LANGUAGE (DML)

The DML is an extended version of the *COBOL Processor*. Its function is to generate the object elements which can utilize a DMS 1100 database.

Any program accessing a database must register as a run unit with the DMR. This registration is accomplished by the IMPART command. When a run unit has no further need to access the database, it terminates its registration using the DEPART command.

A DML program is written following standard COBOL rules. There are four divisions as follows:

- IDENTIFICATION,
- ENVIRONMENT,
- DATA, and
- PROCEDURE.

In the Data Division DML adds a SCHEMA Section. In this section, the program INVOKES the SCHEMA which it is to use, and names certain 01 and 77 level items which correspond to data-base-data-names appearing in the SCHEMA.

In the Procedure Division, the program can use DML commands such as FIND, STORE, MODIFY, etc., in addition to the standard COBOL commands.

The method used in DMS 1100 to provide an extended COBOL Processor is that of a DML preprocessor to the standard COBOL compiler. Where necessary to understanding and use by the programmer, the actions of the Preprocessor and the COBOL compiler are detailed in the following sections.

2. DATA MANIPULATION LANGUAGE

2.1. GENERAL

The DML syntax consists of basic syntax clauses and combinations of these clauses. These combinations, through definition and convention, are used to communicate meaning or content. The DML syntax must be considered within the framework of COBOL syntax. This section does not discuss the entire COBOL syntax, but only describes the Data Manipulation syntax added to the language.

This section introduces the components of these clauses and the notation used to indicate allowable syntactical combinations of these clauses. The clauses themselves are introduced in Sections 3 and 4.

The DML Preprocessor examines all clauses for syntactical completeness and prints a syntax error message should a violation occur. (See Appendix E for a list of DML PREPROCESSOR ERROR MESSAGES.)

2.2. DML SYNTAX COMPONENTS

This section starts at the most basic component, that of character set, and builds up through the clause and on to the most complex structure, the source program.

2.2.1. Character Set

The DML character set contains the following 51 characters:

0, 1, ..., 9	digits
A, B, ..., Z	letters
	space (blank)
+	plus
–	minus or hyphen
*	asterisk
/	stroke (slash)
=	equal to
$	currency sign
,	comma
;	semicolon
.	period (decimal point)
'	quotation mark
(	left parenthesis

)	right parenthesis
>	greater than
<	less than

2.2.2. Character String

A character string is a set of contiguous characters that form a literal, or a PICTURE in the Data Division.

2.2.3. Word

A word is a character string of not more than 30 characters from the set A, B, ..., Z, 0, 1, ..., 9, and –. The – may not appear as the first or last characters in a word. The following are words:

'HERE–AFTER', '22', 'AB98X' and 'A'.

The following are not words:

'A,BC', '–AT–THE',' 'and 'ABC–'.

2.2.4. Name

A DML name is a word that names or identifies an entity in a source program. The use of a name and its meaning is a function of its syntactical position.

2.2.5. Reserved Word

Any DML or COBOL word whose syntactical usage is restricted and cannot be supplied as a name by the Application Programmer is a DML Reserved Word. A complete list of DML Reserved Words is given in Appendix A of this reference manual.

2.2.6. Literal

A literal is a string of characters whose value is that of the characters comprising the string. There are two types of literals as follows:

- Numeric, and
- Non-numeric.

2.2.6.1. Numeric Literals

A numeric literal is a literal composed of one or more numeric characters. For example '2' represents the value two.

2.2.6.2. Non-Numeric Literals

A non-numeric literal is a string of characters bounded by quotation marks. The string may include any character in the DML character set. Two consecutive quotation marks are necessary to represent a single occurrrence of the symbol in a character string. For example, 'ABD''$'' ' is a non-numeric literal.

2.2.7. Clause

A DML clause is an ordered sequence of words and characters. Clauses are the basic unit of DML syntax that communicate functional pieces of information. Clauses may contain subordinate clauses.

2.2.8. Entry

A DML entry is a descriptive set of consecutive syntactically correct clauses terminated by a period. DML entries occur only in the Data Division and are two types as follows:

(1) SCHEMA SECTION entry contains information necessary to provide a proper interface with the Data Management Routine.

(2) Data Description entry contains a level number and a data-name followed by a set of clauses.

2.2.9. Command

A DML command is a syntactically valid combination of words and clauses beginning with a DML verb. A command occurs only in the Procedure Division.

2.2.10. Source Program

The Data Manipulation Language Preprocessor (DMLP) source program is a syntactically complete set of written entries and commands prepared by the programmer designed to perform the following functions *(in addition to the normal functions of a COBOL source program)*:

- To provide for interface with the Data Management routine.
- To give a set of procedural steps on how to manipulate the data-base to achieve the desired effect on the data-base.

2.3. DML SYNTAX NOTATION

This section supplies the notational symbols and rules used to guide the construction of syntactically correct DML clauses. These conventions apply to all DML syntax skeletons specified in Section 3.

- The elements which make up a syntax skeleton are upper case words, lower case words, characters (spaces included), notational symbols, and discussion comments. Their skeleton order, left to right, next line, must be reflected in DML source language.

- All upper case words represent DML reserved words. Underlined upper case words are required in each specification of the syntax component. Non-underlined upper case words are individually optional. Each may be included or omitted in source language specifications.

  Example:

  DIVISION **DELIVERY-AREA-IS**

- Lower case words *(in italics)* are generic terms, names, or literal types, which must be replaced in the actual source specification with a name or literal. Generic terms always have integers appended; this avoids type, for example, appearance of *data-base-data-name–1* and *data-base-data-name–2* in one skeleton means two *data-base-data-names* must be supplied.

- The use of semi-colons and periods in a skeleton is optional in source specifications. However, when used they must be positioned as in the skeleton. The commas and spaces in a skeleton must be used as indicated by a skeleton.

- When one or more syntax components are enclosed by special notational symbols, the meaning is as follows:

  [a / b / c] — A source specification does not require a component in this position, but can include either a, b, or c.

  { a / b / c } — A source specification must include exactly one of the components a, b, or c.

  ‖ a / b / c ‖ — A source specification must include one of a, b, or c, but can include up to one of each.

  ... — Repeated choices may be made and included in a source specification, one after the other.

- When prose surrounded by double quotes (" ") is in a skeleton, the prose contains comments to clarify the skeleton; they can never appear in a source schema.

- All underlined (**LARGE BOLD FACE**) upper case words (see following example) must begin in a new card, with their first character between columns 8 and 11 inclusive. All other words must begin and end between card columns 12 and 72. More than one card can be used to specify the words between two large bold face underlined words (see following example). The above rules must still be followed and the indicated word order preserved.

  Example:

  IDENTIFICATION .... ENVIRONMENT

The following syntax skeleton has no meaning except to illustrate notation. Examples of possible specifications are as follows:

- Syntax Skeleton:

NAME IS { STANDARD / *data-name–1* } [, *data-name–2*] . . . ;

HOWEVER ALWAYS ‖ ZIG / ZAG ‖ .

- Syntax Examples:

(1) NAME IS STANDARD, DN–A, DN–B; HOWEVER
ALWAYS ZIG ZAG

(2) NAME DN–C HOWEVER ZIG

(3) NAME IS DN–D, DN–E, DN–F;
HOWEVER ALWAYS ZAG.

(4) NAME STANDARD
; HOWEVER ZIG ZAG

(5) NAME IS
STANDARD
DN–G
DN–1
HOWEVER ALWAYS
ZIG.

3. DATA MANIPULATION LANGUAGE

3.1. GENERAL

This section discusses the COBOL DML syntax and the rules governing its use. It first discusses the overall makeup of a COBOL DML program. It then divides the COBOL DML into two logical categories and discusses each in turn. The two categories are:

- SCHEMA SECTION SYNTAX
- DML COMMANDS

Each category is first discussed in an informal manner intended to provide a conceptual understanding and overview of what follows. A complete syntax skeleton is also included. The specific DML syntax clauses in each category are then formally discussed on an individual basis.

The formal discussion of each clause is accomplished under five headings; these headings and their intent are as follows:

Function:

A brief definition of the functional purpose of the clause.

Syntax Skeleton:

A highly notational view of the allowable source syntax combinations which may be constructed from the clause. The Syntax Skeleton Notation is explained in Section 2.3. Violation of the Syntax Skeleton normally results in a DML Preprocessor error message.

Syntactical Constraints:

The rules which must be followed to produce source clauses from the Syntax Skeleton. They generally reflect options and restrictions not inherent in the Syntax Skeleton representation. Violations of a Syntactical Constraint normally result in a DML Preprocessor error message.

Data Manipulation Guides:

The detailed guidelines explaining the preparation, the execution, and the results of a DML operation. They also detail the possible error conditions encountered and returned by the DMR. In the section dealing with

DML commands, the last Data Manipulation Guide is always a summary of the Error Status Conditions which can be encountered during an execution of this command. A complete summary for all commands is contained in Section 4, ERROR RECOVERY TECHNIQUES.

Examples and Discussion:

Examples of clauses and the resulting DML/DMR operation are given under this heading. Also included is any "nice-to-know" information which may aid in data manipulation.

3.2. COBOL DML PROGRAM

The composition of a COBOL DML program is very similar to that of a regular COBOL program. The PROGRAM-ID of the IDENTIFICATION DIVISION is used to name the relocatable element produced by the COBOL compiler. The ENVIRONMENT DIVISION is not changed. The DATA DIVISION includes one additional SECTION, the SCHEMA SECTION, and additional description in the WORKING STORAGE SECTION and/or COMMON STORAGE SECTION. The PROCEDURE DIVISION includes DML commands intermixed with the COBOL verbs. The COBOL DML program thus appears as such:

- IDENTIFICATION DIVISION
- ENVIRONMENT DIVISION
- DATA DIVISION
  - SCHEMA SECTION
  - FILE SECTION
  - COMMUNICATION SECTION
  - WORKING-STORAGE SECTION
  - COMMON-STORAGE SECTION
  - LINKAGE SECTION
  - REPORT SECTION
- PROCEDURE DIVISION

The various sections of the DATA DIVISION are optional. However, when present they must be in the respective order precedingly shown.

Within the ENVIRONMENT DIVISION the PROGRAM-ID is used by the DML Preprocessor as the element name of the COBOL compiler relocatable output if the DML Preprocessor is instructed to make an internal call upon the COBOL compiler (see Appendix C, DML PREPROCESSOR CONTROL CARD, for further details).

Within the DATA DIVISION, the SCHEMA SECTION is required. The clauses in the SCHEMA SECTION identify the schema defining the database to be accessed, identify the run unit, and control the DML Preprocessor generation of record descriptions within the WORKING-STORAGE SECTION, COMMON STORAGE SECTION, and/or LINKAGE SECTION. These clauses are discussed further in Section 3.3, SCHEMA SECTION SYNTAX.

As directed by SCHEMA SECTION clauses, the DML Preprocessor enters, in addition to any 77-level or 01-level descriptions already in the WORKING-STORAGE SECTION, the:

- 01-level data-management-communications-area description
- 01-level record-delivery-area description
- 01-level record descriptions from the schema

These entries are individually discussed in Section 3.3, SCHEMA SECTION SYNTAX. The resulting WORKING-STORAGE, COMMON-STORAGE SECTION or LINKAGE SECTION appears as follows:

- 77-level Programmer entered descriptions including those data-base-data-names defined in the schema and required by the run unit; these may have a USAGE is AREA-NAME, USAGE IS DATABASE-KEY or some other USAGE, but not USAGE IS AREA-KEY.

  .
  .
  .

- 01-level Programmer entered descriptions for those data-base-data-names, defined in the schema and required by the run unit, including those which have a USAGE IS AREA-KEY.

  .
  .
  .

- 01-level Programmer entered descriptions including elementary items having a USAGE IS AREA-NAME, AREA-KEY, or DATABASE-KEY which are not defined in the schema and not required by the run unit.

  .
  .
  .

- 01-level DML Preprocessor entered Data-Management-Communication-Area description.

  .
  .
  .

- 01-level DML Preprocessor entered record-delivery-area description.

  .
  .
  .

- 01-level DML Preprocessor entered record discriptions from the schema.

The DML Preprocessor entered descriptions may be selectively placed in either WORKING STORAGE, COMMON-STORAGE, or LINKAGE SECTION. However, the relative ordering remains the same.

Some of the Programmer entered descriptions at either the 77 or elementary item level may have a USAGE IS AREA-NAME clause or USAGE IS DATABASE-KEY clause (*both with no PICTURE clause*). The DML Preprocessor replaces the USAGE IS AREA-NAME clause with USAGE IS DISPLAY-1 (*or DISPLAY if T option is specified*) and adds a PIC X(12). It replaces the USAGE IS DATABASE-KEY clause with USAGE IS COMP and adds a PIC 9(10).

The 01-level Programmer entered descriptions, having a USAGE IS AREA-KEY and no PICTURE clause, which correspond to data-base-data-names defined in the schema and required by the run unit, must be placed before all other Programmer entered 01-level descriptions. Mispositioning of the above data-base-data-names causes an error message "DATA-BASE-DATA-NAME-MISPOSITIONED" by DML Preprocessor.

NOTE:

Within any subsequent Programmer entered 01-level descriptions which are not required because of data-base-data-names in the schema, elementary items may be defined with a USAGE IS AREA-KEY, AREA-NAME or DATABASE-KEY and no PICTURE clause at discretion.

The DML Preprocessor changes the USAGE IS AREA-KEY to USAGE IS COMP and enters the following item descriptions:

```
02 PAGE-NUM PIC 9(5).
02 RECORD-NUM PIC 9(5).
```

NOTE:

When referenced in the PROCEDURE DIVISION, PAGE-NUM and RECORD-NUM must be qualified to ensure uniqueness.

Finally, within the PROCEDURE DIVISION, the DML commands and the COBOL verbs may be freely intermixed. Each DML command encountered by the DML Preprocessor is examined to determine if it is syntactically correct and then, converted into a COBOL

CALL 'DMRA' USING *literal–1* [*literal–2*] . . .

format which is completely COBOL compatible. The original DML clause is flagged as a comment with an asterisk (*) in card column 7. The individual DML commands are discussed further in Section 3.4, PROCEDURE DIVISION SYNTAX.

3.3. SCHEMA SECTION SYNTAX

The purpose of the SCHEMA SECTION is to provide the DML Preprocessor with the information needed to convert a DML source program into a COBOL source program capable of interfacing with the DMR as a run unit. In a general sense the information required identifies:

- The schema being used.
- The run unit being created.
- The records being used.
- Error handling procedures.

The entire SCHEMA SECTION is flagged as comments with an asterisk (*) in card column 7 by the DML Preprocessor after the above information has been called out.

More specifically, the INVOKE SCHEMA clause, which is the first clause of the SCHEMA SECTION, identifies the schema. The record descriptions in the schema are then brought into the COBOL source program, unless excluded by use of the COPYING clause. Further, the record descriptions can be allocated to WORKING-STORAGE, COMMON-STORAGE or LINKAGE through use of the WORKING/COMMON/LINKAGE option.

After identification of the schema, the run unit may be uniquely identified using the RUN-UNIT-ID clause. The purpose of establishing a run unit DMR interface is to transfer records between the run unit and the database. The record-delivery-area for such transfers is identified by the RECORD clause, and optionally, its size by the LENGTH clause. As above, the record-delivery-area may be allocated to WORKING-STORAGE, COMMON-STORAGE, or LINKAGE through the use of a WORKING/COMMON/LINKAGE option.

Usually, mutually exclusive use is made of the record-delivery-area or the main storage allocated on the basis of the record descriptions. However, the optional OVERLAY clause may be used to reduce the amount of main storage allocated, by redefining one record description over the top of the preceding one.

To provide the necessary run unit/DMR interface, a Data Management Communications Area (DMCA) is required. Normally this is allocated to WORKING-STORAGE, however, the optional DMCA clause may be used to allocate it to COMMON-STORAGE or LINKAGE.

In the event of an unsuccessful execution of a DML command, the error processing to be done may be specified at the command level. However, if more generalized error processing is desired, the optional ERROR clause may be used to specify the paragraph containing the error processing logic.

The following section formally treats the complete SCHEMA SECTION entry. The remaining sections formally treat the individual clauses in the order shown in the complete entry.

3.3.1. Complete Entry Skeleton

Function:

To identify and supply the characteristics of the run unit, and to specify the schema to be used.

Syntax Skeleton:

```
SCHEMA SECTION.
INVOKE SCHEMA schema-name
    [COPYING [WORKING | COMMON | LINKAGE] {ALL | record-name–1 [,record-name–2] . . .}]
    [RUN-UNIT-ID IS run-unit-identification]
    [PRIORITY IS integer–1]
     RECORD DELIVERY-AREA IS record-delivery-area [WORKING | COMMON | LINKAGE]
         [LENGTH IS integer–2 WORDS]
    [OVERLAY {record-delivery-area | record-name–3} WITH {ALL | record-name–4 [,record-name–5] . . .}]
         [OVERLAY record-name–6 WITH record-name–7 [,record-name–8] . . . ] . . .
    [SAVE DATA INCLUDES {COMMAND | RUN-UNIT} QUICK-BEFORE-LOOKS]
```

```
[DMCA [AND RUN-UNIT-STATISTICS] {ARE} {WORKING}  ]
                                {IS } {COMMON }
                                      {LINKAGE}

[ERROR RECOVERY IS general-error paragraph]

[ROLLBACK IS rollback-error-paragraph].
```

Syntactical Constraints:

- The SCHEMA SECTION entry must be present and must be placed in the COBOL DATA DIVISION as the first SECTION.
- The INVOKE clause must be the first clause following the SCHEMA SECTION clause, the remaining clauses may be specified in any order, except the COPYING clause which is part of the INVOKE clause, and the OVERLAY clause which must follow the RECORD clause.
- If the LINKAGE option is specified in any one of the COPYING, RECORD, or DMCA clauses, the WORKING option may not be specified in any of these clauses.
- If the LINKAGE option is specified, an IMPART cannot appear in the compilation.
- The only punctuation allowed is a period following the last specified clause.

Data Manipulation Guides:

- The optional general-error-paragraph, specified in the ERROR RECOVERY clause, is executed whenever a DML function cannot be successfully completed provided the ON-ERROR phrase is not appended to the command and run-unit-rollback is not required.
- If the LINKAGE option is specified on any one of the COPYING, RECORD or DMCA clauses, and, either directly or by default, WORKING is specified on another of these clauses, then an appropriate error message results and the corresponding items are generated in COMMON storage.
- The rollback-error-paragraph is executed under the condition that run-unit-rollback is to be performed on the occurrence of an error. The error status is returned in the DMCA location RB-ERROR-CODE. The rollback-error-paragraph must be specified if an IMPART command appears in this compilation.

  The rollback-error-paragraph is not entered on a DEPART WITH ROLLBACK as successful completion of this command is ROLLBACK. In this case, control is returned to the statement of the run-unit following the DEPART WITH ROLLBACK command.

Examples and Discussion:

The following is an example of a complete schema section entry.

```
SCHEMA SECTION.

INVOKE SCHEMA TESTSCH
    COPYING COMMON ALL
    RECORD DELIVERY-AREA  TRDA
    LENGTH 25
    OVERLAY AREC1 WITH AREC2
    OVERLAY BREC1 BREC2
    SAVE DATA INCLUDES COMMAND QUICK-BEFORE-
    LOOKS
    ERROR T-ERR-PROC
    ROLLBACK IS RB-ERR.
```

In this example, the SCHEMA being INVOKED is in the file TESTSCH, and all records in the SCHEMA are INVOKED. The record descriptions are generated in COMMON STORAGE.

The record-delivery-area, TRDA, is generated in WORKING STORAGE. It is 25 words long.

The OVERLAY clauses cause AREC2 to REDEFINE AREC1 and BREC2 to REDEFINE BREC1.

The SAVE clause specifies that command quick-before-looks are to be taken for all AREAS for which QUICK-BEFORE-LOOKS have been specified in the AREA section of the schema definition.

If an error occurs during the execution of a DML command, control is given to the COBOL paragraph T-ERR-PROC (*unless the optional ON ERROR or AT END clause is present in the command*).

The ROLLBACK clause specifies the COBOL paragraph RB-ERR to which control is returned when the run unit rollback is initiated (*other than by a DEPART WITH ROLLBACK command*).

The RUN-UNIT-ID clause is not specified and therefore, the LINKER acquires RUN-ID from the programs PCT and it is established as the RUN-UNIT-ID.

```
WORKING-STORAGE

77 user entries (if any)
      :
      :
01 user entries (if any)
      :
      :
01 DMCA.
     (description of DMCA)          (generated by DMLP)

01 TRDA PIC X(100).                 (generated by DMLP)

COMMON-STORAGE.

77 user entries (if any)
      :
      :
01 user entries (if any)            (all succeeding entries are generated by DMLP)
01 AREC1.
     (description of AREC1).
01 AREC2 REDEFINES AREC1.
     (description of AREC2)
01 BREC1.
     (description of BREC1)
01 BREC2 REDEFINES BREC1.
     (description of BREC2)
```

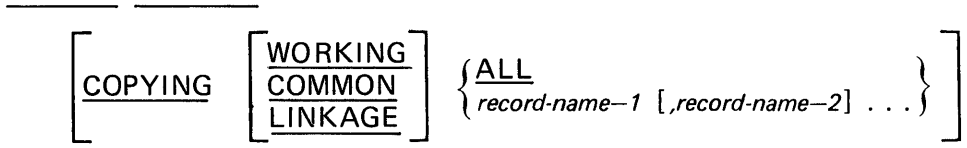
The Application Programmer specified LINKAGE SECTION or REPORT SECTION, if present, or the PROCEDURE DIVISION follows.

3.3.2. INVOKE Clause

Function:

- To specify the schema definition used by the run unit.
- To specify the record definitions to be included in the run unit.

Syntax Skeleton:



Syntactical Constraints:

- The schema specified must be previously defined through the DDL and presently available to the DML Preprocessor. (See the following Data Manipulation Guide.)

- If the COPYING option is used and record-name–1, record-name–2... are specified, they must be the names of records defined in the schema specified.

- All record-names, either directly specified by the COPYING option, or indirectly specified by the ALL option or absence of the COPYING option, must not be duplicated by programmer entered record-names.

Data Manipulation Guides:

- If the schema file, schema-name, which contains the object schema, SCHEMA, is not @ASGed to the run stream at the time the DML Preprocessor is called, the Preprocessor will attempt an @ASG,A schema-name. If the file is not catalogued, the DML Preprocessor terminates on the INVOKE clause without processing the remaining source input.

- The optional COPYING clause is used to control the DML Preprocessor generation of record descriptions, thereby controlling the amount of core allocated to the run unit. Absence of the optional COPYING clause is equivalent to specifying COPYING ALL.

- If the optional WORKING/COMMON/LINKAGE clause is used record descriptions are placed in the run units WORKING-STORAGE, COMMON-STORAGE or LINKAGE following any 01-level programmer entered descriptions or the record-delivery area (if present in the same section) or the DMCA (if present in the same section). If the optional WORKING/COMMON/LINKAGE clause is not specified, 'WORKING' is assumed.

- If the optional ALL is used, the record descriptions of all records defined in the specified schema are placed in WORKING-STORAGE, COMMON-STORAGE, or LINKAGE.

- If the optional record-name–1, record-name–2 is used, only the record descriptions of the specified records are placed in WORKING-STORAGE, COMMON-STORAGE, or LINKAGE.

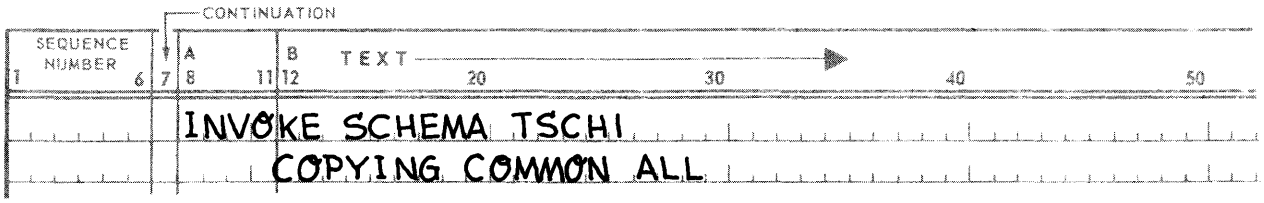
Examples and Discussion:

- **Example 1:**

```
          CONTINUATION
SEQUENCE |A    |B   TEXT ------------------>
NUMBER   |     |
1      6 |7|8 11|12    20        30        40        50
         | INVOKE SCHEMA TSTSCHEMA
```

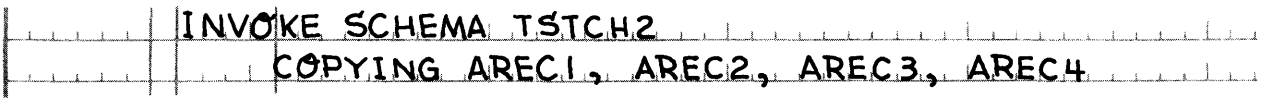
In this example, all of the records in the schema TSTSCHEMA are invoked, and their descriptions generated in WORKING STORAGE.

- **Example 2:**



All of the records in the schema TSCH1 are invoked and their descriptions generated in COMMON WORKING STORAGE.

- **Example 3:**



Only the four records listed are invoked and thus have their descriptions generated in WORKING STORAGE.

3.3.3. RUN-UNIT-ID Clause

Function:

To establish the identity of the run unit.

Syntax Skeleton:

RUN-UNIT-ID **IS** *run-unit-identifier*

Syntactical Constraints:

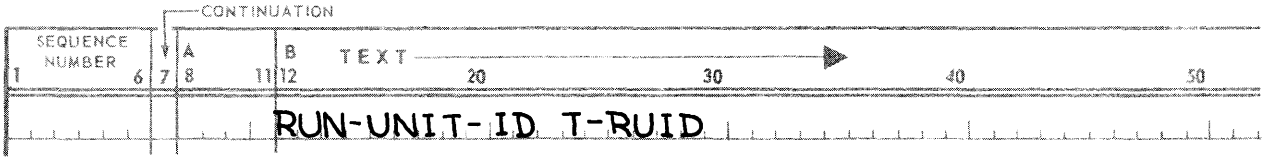
- The run-unit-identifier must be six characters or less.

Data Manipulation Guides:

- If the optional RUN-UNIT-ID clause is omitted, the LINKER acquires the run identity from the PCT and assigns it as the run-unit-identifier.

- No two run-units can operate concurrently with the same run-unit identifier. The DMR prevents this by using the run identity from the PCT. If the user assigns run-unit-identifier, he must control all run units accessing the DMR and must give each run unit a unique identity.

Examples and Discussion:



The word 'T-RUID' is placed in the DMCA for access by the DMR or the run unit.

3.3.4. PRIORITY Clause

Function:

To inform the Data Management Routine of the priority with which this run unit is to be processed.

Syntax Skeleton:

PRIORITY **IS** *integer–1*

Syntactical Constraints:

Integer–1 must be an unsigned positive integer of 10 digits or less.

Data Manipulation Guide:

- This mechanism is used to override the DMR priority scheme for resolving Deadlock and activating queued run units if the systems generation parameter QDLR = 1.
- This priority can be established by the PRIORITY clause and also can be updated on each entry to the DMR.
- If this priority scheme is used:
  (a) On deadlock detection DMR will rollback the run unit with the lowest priority to resolve the deadlock,
  (b) On activation of queued run units, the run unit with the highest priority queued for the facility is activated.
- If this optional clause is omitted the default priority scheme used at run time is based on the number of pages a run unit has altered in the data base.
- If the program wants to change its priority dynamically, it can do so through the use of a standard COBOL MOVE command (*e.g., MOVE 10 TO PRIORITY*).

Examples and Discussion: None

3.3.5. RECORD Clause

Function:

To identify the record-delivery-area to be used for the transfer of records and data for the LOG command between the run-unit and the system buffer.

Syntax Skeleton:

RECORD **DELIVERY-AREA IS** *record-delivery-area* [WORKING / COMMON / LINKAGE]

[LENGTH **IS** *integer–2* **WORDS**]

Syntactical Constraints:

- The RECORD clause is required, its absence preventing generation of the COBOL compatible source output element.
- The specified record-delivery-area name must be unique among all 77-level or 01-level names in the DATA DIVISION.
- Integer–2 if present, must be an unsigned positive integer.
- If integer–2 is not present, the size of the largest INVOKED record is used as the record-delivery-area size.

Data Manipulation Guides:

- A record is the physical unit of transfer between the run unit and the DMR. The record-delivery-area serves as the point of transfer. Upon a FETCH or GET command, the DMR will deposit the desired record in the record-delivery-area. Upon a STORE or MODIFY command, the DMR expects to find the record going to the database in the record-delivery-area.
- The length of the record-delivery-area, as specified in integer–2, must be at least as great as the maximum size of any record description invoked or the maximum size required for a LOG command.
- The DML Preprocessor will replace the record clause with the entry:

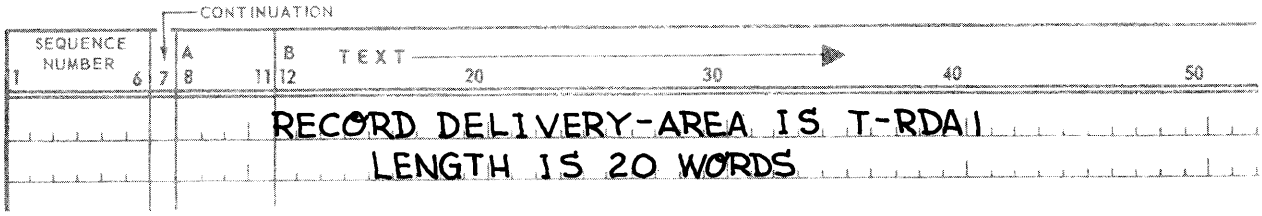
  01 record-delivery-area PIC X(4*(integer–2)).

  following any 01-level programmer entered descriptions, or the DMCA. If the T option is specified on the DMLP call card, the record clause is replaced by:

  01 record-delivery-area PIC X(6*(integer–2)).

Examples and Discussion:

- **Example 1:**



The following entry is generated in WORKING STORAGE:

```
01 T–RDA1 PIC X(80).
```

- **Example 2:**

The following entry is generated in LINKAGE:

```
01 T–RDA2 PIC X(60).
```

No IMPARTs may be specified in this compilation, and the WORKING option may not be specified in the COPYING and DMCA clauses.

3.3.6. OVERLAY Clause

Function:

To conserve on space allocated to the run unit by causing the same space to be used by different record types.

Syntax Skeleton:

OVERLAY { *record-delivery-area* / *record-name–3* } **WITH** { ALL / *record-name–4* [,*record-name–5*] . . . }

[OVERLAY *record-name–6* **WITH** *record-name–7* [,*record-name–8*] . . .] . . .

Syntactical Constraints:

- The OVERLAY clause must immediately follow the RECORD clause.
- The record descriptions of the specified record-names must have been previously included in the DATA DIVISION by either the absence of the COPYING clause, the presence of the ALL option on the COPYING clause, or the presence of the respective record-name on the COPYING clause.

- If the record-delivery-area name is specified, it may be only in the first OVERLAY clause.

- If the ALL option is used, then only the record-delivery-area may be specified as the overlaid record and only one OVERLAY clause is permitted.

- If the record-delivery-area is to be overlaid, both it and the overlaying records must be all allocated to either WORKING-STORAGE or COMMON-STORAGE, but may not be mixed.

- A record name may not appear in more than one OVERLAY clause.

Data Manipulation Guides:

- If the record-delivery-area is overlaid, then those record types which overlay the record-delivery-area may be processed without the necessity of accomplishing a MOVE from the record-delivery-area after a FETCH or GET command, or a MOVE into the record-delivery-area before a STORE or MODIFY command.

- Extreme care must be used when determining which record types may overlay other record types. Generally, record types defined in the DDL as part of the same hierarchical structure should not overlay each other.

Examples and Discussion:

- The effect of an OVERLAY clause depends not only on the clause itself, but also on the INVOKE and/or RECORD DELIVERY clauses.

- **Example 1:**

```
INVOKE SCHEMA NAMEA COPYING COMMON ALL

    RECORD COMMON T-RDA1
           LENGTH IS 20
    OVERLAY T-RDA1 AREC1, AREC2
    :
```

The following entries are generated in COMMON STORAGE:

```
01 T-RDA1  PIC X(80).
01 AREC1 REDEFINES T-RDA1.
    (DESCRIPTION OF AREC1)
01 AREC2 REDEFINES AREC1.
    (DESCRIPTION OF AREC2)
01 (ALL OTHER RECORDS IN THE SCHEMA)
    :
```

■ **Example 2:**

```
INVOKE SCHEMA NAMEB
    RECORD COMMON T-RDA2
    OVERLAY T-RDA2 ALL
```

This situation is an error (see Syntactical Constraints).

3.3.7. SAVE Clause

Function:

To allow the run unit to specify that command quick-before-looks are required.

Syntax Skeleton:

SAVE DATA INCLUDES { COMMAND | RUN-UNIT } QUICK-BEFORE-LOOKS

Syntactical Constraints: None

Data Manipulation Guides:

- The SAVE DATA clause allows the run unit to specify, that command quick-before-looks are to be taken for all AREAs for which QUICK-BEFORE-LOOKS have been specified in the AREA section of the schema definition.
- Command quick-before-looks and run unit quick before looks are written to the quick-look file which must be assigned to the run unit.
- The COMMAND QUICK-BEFORE-LOOKS clause specifies, that quick-before-looks are to be written to the quick-look file for each page prior to the first time each command of the run unit alters the page.
- The RUN-UNIT QUICK-BEFORE-LOOK clause specifies that quick-before-looks are to be written to the quick-look file for each page prior to the first time the run unit alters the page.

- The default value for the SAVE DATA clause is RUN-UNIT QUICK-BEFORE-LOOKS.

- If COMMAND QUICK-BEFORE-LOOKS are specified, more looks are written to the quick-look file but the database is restored to the condition it was in prior to the execution of the command (*if an error occurs during execution of the command, and the command had altered the database*).

Examples and Discussion: None

3.3.8. DMCA Clause

Function:

To direct the allocation of the Data Management Communication Area (DMCA) to the desired COBOL DATA DIVISION section, and optionally to request the collection of run-unit statistics.

Syntax Skeleton:

DMCA [AND RUN-UNIT-STATISTICS] { ARE / IS } { WORKING / COMMON / LINKAGE }

Syntactical Constraints:

- If the LINKAGE option is specified, no IMPART command can appear in the PROCEDURE DIVISION of this compilation.

Data Management Guides

- The DMCA is a data record used for interface between the run unit and the DMR. It is generated as an 01 level entry in the COBOL DATA DIVISION. If the optional DMCA clause is not specified, the DMCA is allocated in the WORKING-STORAGE SECTION. If the DMCA clause is specified, the DMCA is allocated in the WORKING-STORAGE, COMMON-STORAGE or LINKAGE section, according to the DMCA clause specification.

- Within the DMCA record several data-items are always generated. Some of these items are used exclusively by the DMR, and are value assigned FILLER items. The remainder of these items are available to the run unit, for examination and/or initialization. The data-items available to the run unit are:

  (1) RUN-UNIT-ID USAGE IS DISPLAY–1 PIC X(6) VALUE IS 'XXXXXX' .

  This data-item is initialized by the DML Preprocessor with the RUN-UNIT-ID specified in the Schema Section Entry if the RUN-UNIT-ID clause is present, otherwise the LINKER acquires the RUN-ID from the programs PCT. The RUN-UNIT-ID is available for run unit reference, however, it must not be altered following the execution of an IMPART command, or prior to the execution of a DEPART command.

  (2) IMPART-DEPART USAGE IS DISP–1 PIC X VALUE '0'

  This item contains an indicator which the run unit may reference to determine if either an IMPART or DEPART command has been successfully completed. It is initialized by the LINKER and can contain the following possible values.

0: = no IMPART or DEPART command has been issued
1: = an IMPART command has successfully completed
2: = a DEPART command has successfully completed

This indicator is available as a debug aid in addition to the standard ERROR-STATUS returned after the execution of each individual command. Also, a DEPART must be performed for all run units and any error situation encountered in the run unit must check the IMPART-DEPART indicator, to determine whether or not a DEPART has been performed.

(3) AREA-NAME USAGE IS DISPLAY–1 PIC X(12).

This data-item must, in certain instances, be initialized with an area-name by the run unit prior to executing a command. In addition, it will contain, upon the successful execution of any FIND/FETCH or STORE COMMAND, the area-name in which the object record is located.

(4) AREA-KEY USAGE IS COMP.

This group data-item contains, upon the successful execution of a FIND/FETCH or STORE command the page and record number of the object record. It consists of the following data-items:

PAGE-NUM PIC 9(5).
RECORD-NUM PIC 9(5).

(5) RECORD-NAME USAGE IS DISPLAY–1 PIC X(30).

This data-item contains, upon the successful execution of a FIND/FETCH or STORE command, the record-name of the object record.

(6) ERROR-AREA USAGE IS DISPLAY–1 PIC X(12).

This data-item contains, upon the unsuccessful execution of any command, the error-area-name if the error is area related and the error-area may be ascertained.

(7) ERROR-RECORD USAGE IS DISPLAY–1 PIC X(30).

This data-item contains, upon the unsuccessful execution of any command, the error record-name if the error is record related and the error may be ascertained.

(8) ERROR-SET USAGE IS DISPLAY–1 PIC X(30).

This data-item contains, upon the unsuccessful execution of any command, the error set-name if the error is set related and the error set may be ascertained.

(9) ERROR-STATUS USAGE IS DISPLAY–1.

This group of data-items contains zeros upon the successful execution of any command. In the event the execution was unsuccessful, it contains a code identifying the type of command being executed and the type of error encountered.

The data-items are:

```
RB-ERROR-CODE     PIC X(2).
ERROR-FUNCTION    PIC X(2).
ERROR-CODE        PIC X(2).
```

The RB-ERROR-CODE is set when an error requires the run-unit to be rolled back without the run unit requesting rollback. A full definition of the error code returned is given in the error descriptions of each command.

(10) PRIORITY PIC 9(10) USAGE IS COMP VALUE 0000000000.

This item is established by the priority clause and controls the priority of this run-unit relative to current run units for reactivation after queueing, or rollback to break deadlock.

(11) DATABASE-KEY PIC 9(10) USAGE IS COMP.

This data-item is used in DMLP produced procedures to pass database keys to the DMR for execution of certain commands.

(12) CURRENT-AREA-KEY USAGE IS COMP.

This group data-item is used in DMLP produced procedures to pass area keys to the DMR for the execution of certain commands. It consists of data-items identical to those of the AREA-KEY data-item.

(13) CURRENT-AREA-NAME PIC X(12) USAGE IS DISPLAY–1.

This data-item is used in conjunction with CURRENT-AREA-KEY. It is used to pass area-names to the DMR for the execution of certain commands.

The proper use of the DMCA Reserved Words are discussed in Section 3.4, PROCEDURE DIVISION SYNTAX, whenever they become involved in the execution of a DML command.

NOTE:

Both AREA-KEY and CURRENT-AREA-KEY consist of data-items PAGE-NUM and RECORD-NUM. Hence, any reference to PAGE-NUM and/or RECORD-NUM require qualification to ensure uniqueness.

If the DMCA clause is specified, and the optional RUN-UNIT-STATISTICS specification is used, several additional data-items are generated in the DMCA. When these items are generated the DMR uses them to collect run unit statistics; during execution of a run unit these data-items may be examined, but should not be altered. The data-items used for collection of run unit statistics are:

(1) COMMAND-STATISTICS USAGE IS COMP.

This group data-item is used to count the number of times each DML command type is executed within the run unit. One sub-item is used for each command type; sub-items have names equal to command names with a '-COUNT' suffix, c.f. STORE-COUNT.

(2) QUEUE-STATISTICS USAGE IS COMP.

This group data-item is used to count the number of times the run unit is queued, in the DMR, for each of the various reasons that this can happen.

The queuing reasons are described following their respective sub-items. All the following sub-items have a picture of 9(5).

(3) TABLE-COUNT

Count of the times queued waiting for space in the table region of the DMR buffer.

(4) PAGE-COUNT

Count of the times queued waiting for space in the page region of the DMR buffer.

(5) AREA-COUNT

Count of the times queued waiting for the opening of an area that has been prevented by the concurrent usage of that area by other run units.

(6) LRI-COUNT

Count of the times queued waiting to use the DMR last record interval table.

(7) TRL-COUNT

Count of the times queued waiting to use a DMR table reference list. This is usually caused by an attempt to have concurrent execution of an IMPART with another run unit.

(8) USAGE-LOCK-COUNT

Count of the times queued waiting for a usage lock on a database page that another run unit has a lock on.

(9) I-O-COUNT

Count of the times queued waiting for the DMR to complete an input/output function.

(10) CORE-LOCK-COUNT

Count of the times queued waiting for a core lock on a database page that another run unit has a usage lock on.

(11) CURRENT-LOCK-COUNT

Count of the times queued waiting for a currency lock on a page that another run unit has a currency lock on.

(12) CORE-ALLOCATION-COUNT

Count of the times queued waiting for a database page that is unavailable, because it is currently involved in an allocation configuration in the DMR buffer page region.

(13) ROLLBACK-BUFFER-COUNT

Count of the times queued waiting for the allocation of a buffer required to accomplish rollback.

(14) TIP-PAGE-COUNT

Count of the times queued waiting for space in the TIP page region of the DMR buffer.

- The DMCA is generated as an 01-level record entry following the last COBOL 01-level entry in WORKING-STORAGE, COMMON-STORAGE, or LINKAGE.

Examples and Discussion:

- DMCA IS COMMON

  Causes the DMCA to be generated in the COMMON-STORAGE SECTION; run-unit-statistics data-items are not generated and therefore the DMR does not collect run-unit-statistics.

- DMCA AND RUN-UNIT-STATISTICS ARE WORKING

  This specification would probably be used only if run-unit-statistics are desired; otherwise the clause probably would not be used since WORKING is the assumed specification when the DMCA clause is not specified.

3.3.9. ERROR Clause

Function:

To specify a general-error-paragraph if error recovery is not to be handled at the point of error.

Syntax Skeleton:

ERROR **RECOVERY IS** *general-error-paragraph*

Syntactical Constraints:

- General-error-paragraph must be defined in the non-DECLARATIVES portion of the PROCEDURE DIVISION.
- General-error-paragraph may not be defined in an independent COBOL SEGMENT (*i.e., SECTION number 50-99*).

Data Manipulation Guides:

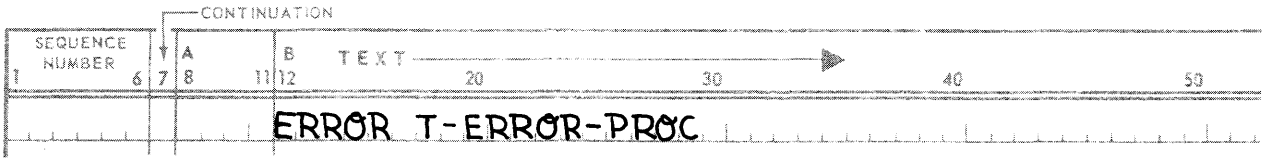
- If the optional ERROR clause is specified, control will be returned to the general-error-paragraph in the event an error is encountered during the execution of any DML command. However, if the optional ON ERROR clause is specified as part of the erring command, then it serves to override the ERROR clause.
- If the optional ERROR clause is not specified, control will always be returned following the DML command. However, if the optional ON ERROR clause is specified as part of the erring command, in the event of an error, control is returned at that error paragraph.
- If the error requires run unit rollback, control is given to the rollback-error-paragraph.

- In all cases, if the SAVE-DATA clause specifies Command-Quick-Before-Looks, the data base will be restored to its condition prior to the execution of the command. This procedure is known as "command roll back". If the SAVE-DATA clause does not include COMMAND QUICK-BEFORE-LOOKS and the data base has been altered by the command, run-unit rollback will be performed and return is made to the rollback-error-paragraph.

- The general error paragraph must verify if either an IMPART has been successfully completed, or if a DEPART has successfully completed by checking the IMPART-DEPART indicator in the DMCA. If an IMPART has not been completed or if the DEPART has completed, the DMR must not be entered except to do an IMPART. Also, if the run unit has IMPARTed, a DEPART must be performed prior to termination of the run unit.

- Section 4, ERROR RECOVERY TECHNIQUES, contains additional information concerning the proper use of the ERROR clause.

Examples and Discussion:



If an error occurs during the execution of any DML command, control is given to the COBOL paragraph T-ERROR-PROC, unless an optional ON ERROR clause is appended to that command.

3.3.10. ROLLBACK Clause

Function:

Establishes a point to which the DMR can return if the run unit was non-voluntarily rolled back. The rollback routine established by the user is responsible for all non-DMR files and reversal of any action the run unit has performed outside DMR control.

Syntax Skeleton:

ROLLBACK **IS** *rollback-error-paragraph*

Syntactical Constraints:

- Rollback-error-paragraph must be defined in the non-DECLARATIVES SECTION of the PROCEDURE DIVISION.

- Rollback-error-paragraph may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

- The ROLLBACK clause is required if an IMPART command appears in this compilation.

Data Manipulation Guides:

- This clause defines the point to which the DMR returns when the DMR initiates run unit rollback. RB-ERROR-CODE contains the reason why DMR rolled back the run unit.
- In the process of executing run unit rollback, the DMR performs a DEPART. Execution of any command other than an IMPART after run unit rollback results in an error return to the run unit's rollback-error-paragraph indicating that an IMPART has not been performed.
- The rollback paragraph is not given control on command rollback, rather a return is made to the command error return address (see ERROR clause of the SCHEMA SECTION).

Examples and Discussion: None

3.4. PROCEDURE DIVISION SYNTAX

The DML commands are specified in the PROCEDURE DIVISION of the COBOL run unit intermixed with the COBOL verbs. Although no physical restrictions are upon the arrangement of DML commands and COBOL verbs, there are certain logical restrictions. Thus, in general, when executed, the logic would appear similar to the following description.

The first DML command executed by a run unit must always be an IMPART command, thus registering the run unit with the DMR. After registration, the OPEN command must be used to open those areas for access.

After the opening of the required areas, a FIND/FETCH or STORE command usually follows, as most other commands implictly refer to a current record of run unit or of area/record/set. The FIND/FETCH and STORE are the only DML commands which establish currency. Once a current record of the run unit is established, it may variously be the object of a GET command which transfers it into the record-delivery area; a MODIFY command which returns it to the database in an altered form; or a DELETE command to remove it from the database. If the current record is defined to be manual member of a set, an INSERT command may be used to include it as an actual member of that set, or conversely, a REMOVE command may be used to discontinue its membership. The conditional IF command may be used to ascertain the set status of the current record.

After all accesses to an area have been completed, the CLOSE command must be exercised to relinquish use of the area. After all accesses to the database have been completed, the DEPART command must be executed to deregister the run unit with the DMR.

A complete entry skeleton is presented in the following section which shows a functional breakdown of the DML commands. The subsequent sections formally present the DML commands in alphabetical order.

3.4.1. Complete Entry Skeleton

Function:

To initiate the specified operation within the DMR.

Syntax Skeleton:

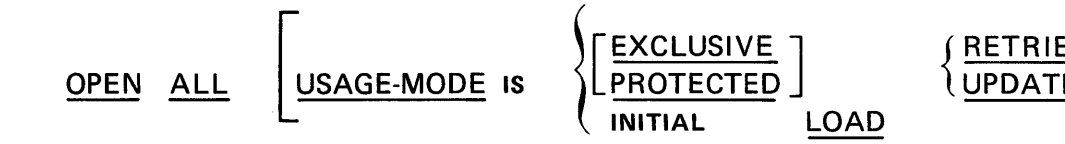
"Run-Unit Oriented Commands"

IMPART [ON ERROR GO TO *error-paragraph*] .

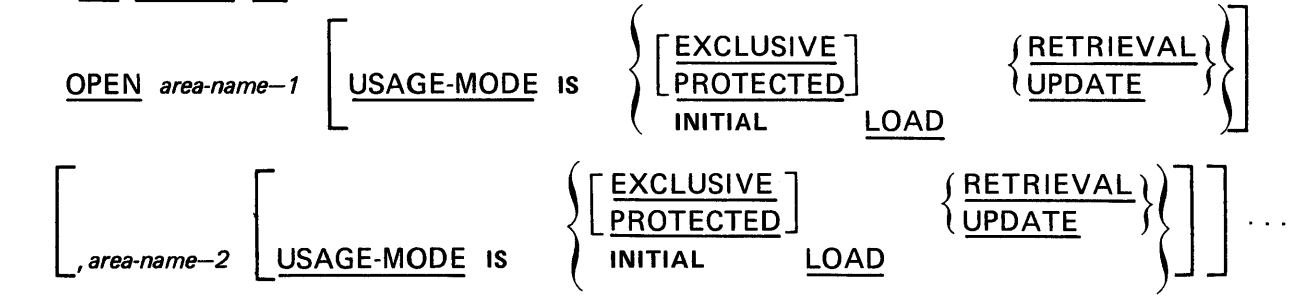
DEPART [WITH ROLLBACK] [ON ERROR GO TO *error-paragraph*] .

FREE [ON ERROR GO TO *error-paragraph*] .

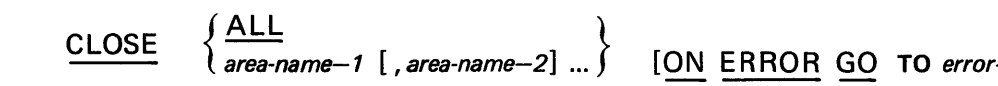
"Area Oriented Commands"



[ON ERROR GO TO *error-paragraph*] .

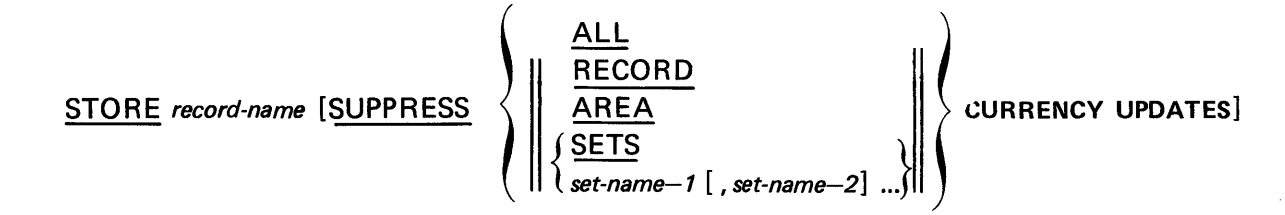


[ON ERROR GO TO *error-paragraph*] .

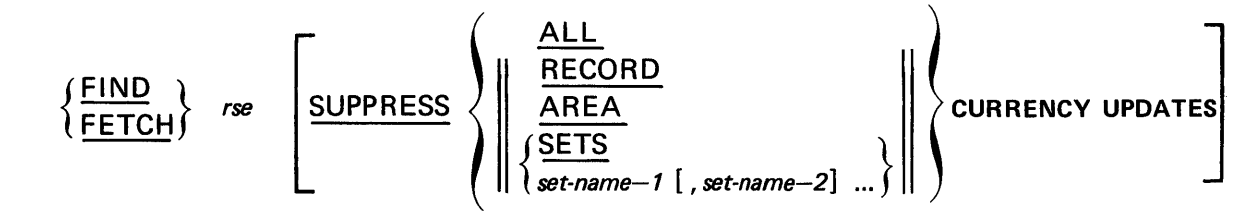


[ON ERROR GO TO *error-paragraph*] .

"Record Oriented Commands"



[ON ERROR GO TO *error-paragraph*] .



[AT END GO TO *end-paragraph*] [ON ERROR GO TO *error-paragraph*] .

"Where the rse is one of the following"

(1) *Identifier–1* [, *identifier–2*]

(2) CURRENT **RECORD WITHIN** *record-name–1* RECORD

(3) { OWNER | CURRENT | NEXT | PRIOR | FIRST | LAST | *identifier–2* } **RECORD WITHIN** { *set-name–3* SET | *area-name–1* AREA }

(4) { NEXT | PRIOR | FIRST | LAST | *identifier–3* } *record-name–2* **WITHIN** { *set-name–4* SET | *area-name–2* AREA }

(5) [NEXT DUPLICATE **WITHIN**] *record-name–3* RECORD

(6) *record-name–4* VIA *set-name–5* [USING *data-base-identifier–1*
[*data-base-identifier–2*] ...]

(7) NEXT DUPLICATE **WITHIN** *set-name–6* USING *data-base-identifier–3*
[, *data-base-identifier–4*] ...

GET [ON ERROR GO **TO** *error-paragraph*] .

MODIFY [*data-base-identifier–1* [, *data-base-identifier–2*] ...]
[ON ERROR GO **TO** *error-paragraph*] .

DELETE [ONLY | ALL] [ON ERROR GO **TO** *error-paragraph*] .

KEEP [ON ERROR GO **TO** *error-paragraph*] .

"Set Oriented Commands"

INSERT **INTO** *set-name–1* [,**INTO** *set-name–2*] ... [ON ERROR GO **TO** *error-paragraph*] .

INSERT **INTO** ALL **SETS** [ON ERROR GO **TO** *error-paragraph*] .

REMOVE **FROM** *set-name–1* [,**FROM** *set-name–2*] ... [ON ERROR GO **TO** *error-paragraph*] .

REMOVE **FROM** ALL **SETS** [ON ERROR GO **TO** *error-paragraph*] .

"Conditional Commands"

IF *set-name–1* SET [NOT] **EMPTY** ;

{*statement–1* / NEXT SENTENCE} [ELSE {*statement–2* / NEXT SENTENCE}] .

IF RECORD [NOT] {OWNER / MEMBER} **OF** {*set-name–2* / ANY} SET ;

{*statement–3* / NEXT SENTENCE} [ELSE {*statement–4* / NEXT SENTENCE}] .

"Supporting Commands"

LOG *data-name–1* **WORDS** [**FOR** RECOVERY **POINT**] [ON ERROR GO **TO** *error-paragraph*] .

MOVE **CURRENCY** STATUS **FOR** { RUN-UNIT / *record-name* RECORD / *area-name* AREA / *set-name* SET }

TO *identifier–1* [ON ERROR GO **TO** *error-paragraph*] .

MOVE ‖ AREA-KEY / AREA-NAME ‖ **FOR** { RUN-UNIT / *record-name* RECORD / *area-names* AREA / *set-name* SET / *identifier–2* } **TO**

‖ *identifier–3* / *identifier–4* ‖ [ON ERROR GO **TO** *error-paragraph*] .

Syntactical Constraints: None

Data Manipulation Guides: None

Examples and Discussion: None

3.4.2. CLOSE Command

Function:

To indicate the completion of access into a specified area.

Syntax Skeleton:

CLOSE { ALL / *area-name–1*[*,area-name–2*] ... } [ON ERROR GO TO *error-paragraph*].

Syntactical Constraints:

- Area-name–1, area-name–2... must be the names of areas defined in the schema named by the INVOKE clause of the Schema Section.
- No more than 47 area-names may be specified on a single CLOSE command.
- File-names specified on the COBOL OPEN verb and area-names specified on the DML OPEN command cannot be intermixed on the same CLOSE.
- Error-paragraph must be a Paragraph Name or Section Name defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

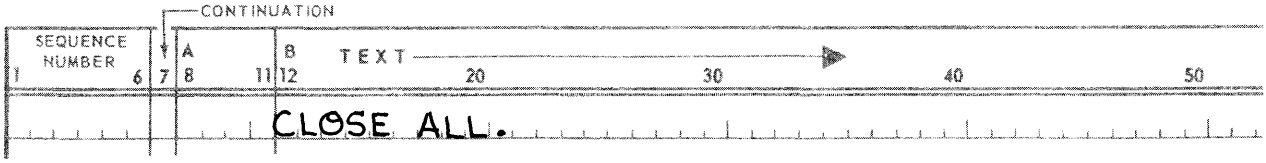
Data Manipulation Guides:

- If the ALL option is specified, any areas already closed are ignored and no Error Status Condition results. However, if an area-name is specified and the area is already closed, then Error Status Condition 000101 results.
- After execution of a CLOSE command for a given area, any subsequent attempt by the run unit to directly or indirectly access that area, other than with an OPEN command, will result in Error Status Condition 00NN01.
- After execution of a CLOSE command for ALL areas, any subsequent attempt by the run unit to access the database, other than with an OPEN command, will result in Error Status Condition 00NN01.
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.
- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
000101	area already closed

Examples and Discussion:

- **Example 1:**



All areas still OPEN to the run-unit are closed.

- **Example 2:**

Areas 'AREA1', 'AREA2', and 'AREA3' are closed.

If any of the areas are already closed, then the command is not executed, and an error status '000101' is returned to the general-error-paragraph specified in the Schema Section, or to the next sentence if no general-error-paragraph is specified.

3.4.3. DELETE Command

Function:

- To surrender the space in the database and the database-key used by the object record occurrence and make them available for subsequent re-use.
- To remove the object record from all set occurrences in which it is a member.
- To delete all record occurrences which are automatic members, and optionally, manual members of set occurrences owned by the object record.
- To remove from those set occurrences all record occurrences which are manual members of set occurrences owned by the object record.

Syntax Skeleton:

DELETE [ONLY / ALL] [ON ERROR GO TO *error-paragraph*].

Syntactical Constraints:

Error-paragraph must be a Paragraph Name or Section Name defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

Data Manipulation Guides:

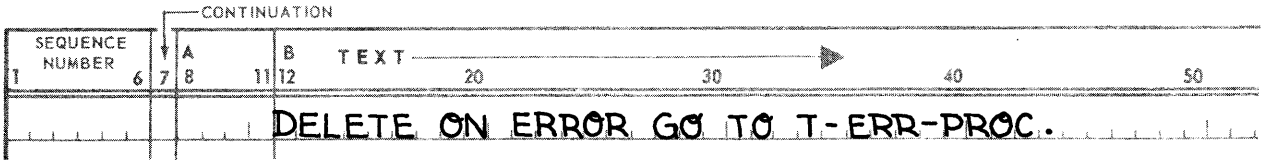
- The object record of the DELETE command is the current record of the run unit. If the current record of the run unit is not known, then Error Status Condition 0213 results.
- The object record is removed from all set occurrences in which it is a member, and is then deleted, that is, made unavailable for further processing. The space and the database-key used by the object record are made available for re-use.
- Use of the DELETE command without either the ONLY or ALL options causes the object record to be deleted. If the object record is the owner of automatic members, they are likewise deleted. If the object record is the owner of manual members, they are removed from the set occurrences owned by the object record. In the event any deleted automatic members are (themselves) the owners of any set occurrences, then the delete command is executed on such records as if they were the object record of the DELETE command. Thus, all automatic members of such sets are also deleted, which in turn causes this process to continue down the hierarchy.
- Use of the DELETE command with the ONLY option causes the object record to be deleted only if automatic or manual members are not present. If the object record is the owner of a non-empty set occurrence, then Error Status Condition 0215 results.
- Use of the DELETE command with the ALL option causes the object record to be deleted. If the object record is the owner of any members, they are deleted, regardless of whether they are automatic or manual members. As with the unqualified DELETE command, the delete process continues down the hierarchy with the difference that all deleted records, which are themselves owners of any set occurrences, are treated as if they were the object record of the DELETE command.
- The current record of the run unit becomes null. No other currency status information is altered. Thus, the object record and all other records deleted or removed remain as current of area-name, record-name and of all set-names in which they were current prior to the execution of the DELETE command.
- The system locations AREA-NAME, AREA-KEY, and RECORD-NAME become null.
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
000213	no current of run unit
000215	owner of a non-empty set occurrence

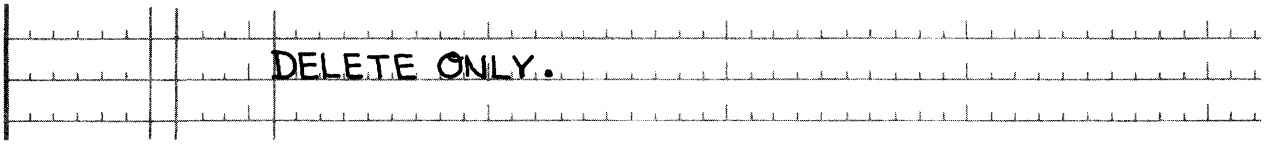
Examples and Discussion:

- **Example 1:**



The current record of run-unit is deleted, in addition to any automatic members of which it is owner. Likewise, any automatic members of sets owned by these deleted records are also deleted, and so on down the hierarchy. If an error occurs, control is returned to T-ERR-PROC.

- **Example 2:**



The current record of run-unit is deleted if no automatic or manual members are present. If the record is the owner of a non-empty set, then an Error Status of 0215 is returned and control is returned to the general-error-paragraph specified in the Schema Section, or to the next sentence, if general-error-paragraph is not specified.

3.4.4. DEPART Command

Function:

To inform the DMR of the termination of the run unit and optionally, to cancel any updates to the database performed by the run unit.

Syntax Skeleton:

DEPART [WITH ROLLBACK] [ON ERROR GO TO *error-paragraph*].

Syntactical Constraints:

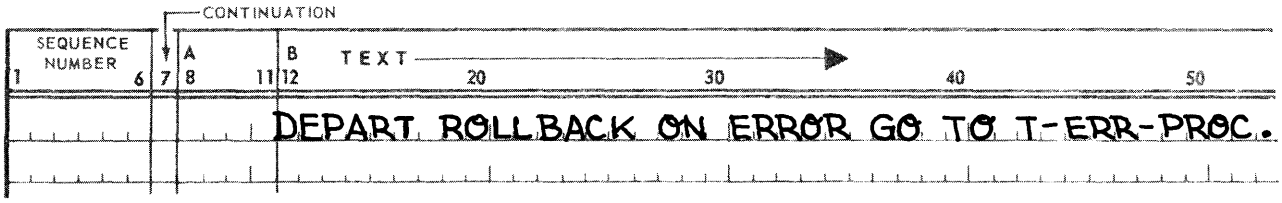
Error-paragraph must be a Paragraph Name or Section Name, defined in the non-declarative portion of the Procedure Division. It must not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

Data Manipulation Guides:

- The run unit will be terminated. An internal close will be done on any open areas.
- All DMR internal tables and space are released. The DMR will make appropriate final entries to the audit tape.
- Any subsequent DML command executed, other than an IMPART, results in Error Status Condition 00NN21.
- If the optional ROLLBACK clause is specified, the database is returned to the state existing prior to the execution of the IMPART or most recent FREE command.
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.
- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
000021	DEPART already done or IMPART not yet issued.

Examples and Discussion:



The database is restored to the state existing prior to the IMPART or most recent FREE command.

If a DEPART has already been done, or if no IMPART has been performed, control is returned to T-ERR-PROC with an ERROR STATUS of 000021.

3.4.5. FIND/FETCH Command

Function:

- To establish a record occurrence specified by a record-selection-expression (rse) as:

  (1) The current record of the run unit.

  (2) The current record of the area in which it is stored.

(3) The current record of record-name.

(4) The current record of set for all set-names in which it currently participates as an owner or member.

- To optionally transfer the contents of the object record occurrence into the record-delivery-area.
- To optionally suppress the establishment of the object record occurrence as:

(1) The current record of the area in which it is stored.

(2) The current record of record-name.

(3) The current record of set for all, or specified sets in which it currently participates as an owner or member.

Syntax Skeleton:

{ FIND | FETCH } *rse* [SUPPRESS { ALL | RECORD | AREA | { SETS | *set-name–1* [, *set-name–2*] . . . } } CURRENCY UPDATES]

[AT END GO TO *end-paragraph*] [ON ERROR GO TO *error-paragraph*].

Syntax Sekeleton of record-selection expressions (rse's):

Format–1

Identifier–1 [, *identifier–2*]

Format–2

CURRENT RECORD WITHIN *record-name–1* RECORD

Format–3

{ OWNER | CURRENT | NEXT | PRIOR | FIRST | LAST | *identifier–3* } RECORD WITHIN { *set-name–3* SET | *area-name–1* AREA }

Format–4

{ NEXT | PRIOR | FIRST | LAST | *identifier–4* } *record-name–2* WITHIN { *set-name–4* SET | *area-name–2* AREA }

Format–5

[NEXT DUPLICATE **WITHIN**] *record-name–3* RECORD

Format–6

record-name–4 VIA *set-name–5* [USING *data-base-identifier–1*[, *data-base-identifier–2*] . . .]

Format–7

NEXT DUPLICATE **WITHIN** *set-name–6* USING *data-base-identifier–3*[, *data-base-identifier–4*] . . .

Syntactical Constraints:

- In *Format–1* if identifier–1 is defined with a USAGE of AREA-KEY, then identifier–2, if present, must be defined with a USAGE of AREA-NAME.
- In *Format–1* if identifier–1 is defined with a usage of DATABASE-KEY, then identifier–2 may not be used.
- In *Format–3* the OWNER option may not be used if area-name–1 is specified.
- In *Format–3* and *Format–4*, identifier–3 and identifier–4, respectively, must be 77-level entries and must have a PICTURE IS 9(9) clause and a USAGE IS COMP clause.
- In *Format–4* record-name–2 must be defined as a member record of set-name–4.
- The AT END option must only be used with *Format–3*, however, not with the OWNER and CURRENT options, and with *Format 4.*
- In *Format–6* data-base-identifier–1, data-base-identifier–2... must be defined as data-items within record-name–4. Record-name–4 must be defined as a member of set-name–5.
- In *Format–7* data-base-identifier–3, data-base-identifier–4... must be defined as data-items within a record defined as a member of set-name–6.
- All data-base-identifiers, record-names, area-names and set-names must be defined in the schema named on the INVOKE clause in the Schema Section.
- Error-paragraph and end-paragraph must be a Paragraph Name or Section Name, defined in the non-declarative portion of the Procedure Division. They may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*). This restriction also applies to end-paragraph.

Data Manipulation Guides:

- The object record of a FIND/FETCH command is specified by the rse.
- The object record of the FIND/FETCH command and all records implicitly referenced during execution of the FIND/FETCH operation, that is, those records included in the search patch used to obtain the object record, must be in areas which are open, otherwise Error Status Condition 0301 results.

- Use of the FIND option causes the DMR to determine the existence or non-existence of the object record. If the object record is non-existent, that is, no record occurrence meets whatever selection criteria specified, then an Error Status Condition results. Existence of the object record does not mean it is available to the run unit, unless a GET command is executed.

  Use of the FETCH option is identical in result to use of the FIND option except that a GET command is implied, therefore, the object record occurrence is available to the run unit in the record-delivery-area, unless an Error Status Condition results.

- If identifier–1 is defined with a usage of AREA-KEY, then *Format–1* of the rse selects a record using the area-key contained in identifier–1 and the area-name contained in identifier–2 if identifier–2 is present. Identifier–1 must have been defined having a USAGE IS AREA-KEY and no PICTURE clause. The resulting items:

  PAGE-NUM of identifier–1
  RECORD-NUM of identifer–1

  both having a PICTURE IS 9(5) and a USAGE IS COMP, must be initialized respectively with the page number and record number of the sought record.

  Identifier–2 must be defined with a USAGE IS AREA-NAME if it is used. The resulting item has a PICTURE IS X(12). It must be initialized with the area-name of the record sought. If identifier–2 is not used, the area-name of the record sought must be initialized in reserved word AREA-NAME.

  If identifier–1 is defined with a usage of DATABASE-KEY, then the object record is found directly using the database-key supplied.

- For *Format–1* COBOL MOVE commands are generated immediately before the FIND/FETCH to MOVE the value of identifier–1 and identifier–2 into system reserved locations.

- *Format–2* of the rse selects the record occurrence that is the current record of the specified record-name. If there is no current record of record-name, then Error Status Condition 0306 results.

- In *Format–3* of the rse, if a set-name is specified, the set occurrence from which the object record is to be selected is identified by the current record of the specified set. If there is no current record of named set, then Error Status Condition 0306 results.
  If an area-name is specified, the object record is selected from the named area. If the NEXT or PRIOR clause is used, and there is no current record of the named area, Error Status Condition 0306 results.

  (1) NEXT RECORD OF area-name AREA means the record with the next higher database-key relative to the current record of the named area. If there is no record in the area named with a higher database-key, Error Status Condition 0307, that is, an End of Area Condition, results.

  (2) PRIOR RECORD OF area-name AREA means the record with the next lower database-key relative to the current record of the named area. If there is no record in the area named with a lower database-key, Error Status Condition 0307, that is, an End of Area Condition, results.

  (3) NEXT RECORD OF set-name SET means the subsequent record relative to the current record of the named set in the logical order of the set regardless of the database-key sequence. If the set is empty, that is, no member record occurrences participate in the set, Error Status Condition 0313 results. If the current record is the last record in the set, Error Status Condition 0307, that is, an End-of-Set Condition, results.

(4) PRIOR RECORD OF set-name SET means the previous record relative to the current record of the named set in the logical order of the set regardless of database-key sequence. If the set is empty, that is, no member record occurrences participate in the set, Error Status Condition 0313 results. If the current record is the first record in the set, Error Status Condition 0307, that is, an End of Set Condition, results.

(5) FIRST RECORD OF area-name AREA is the record occurrence with the lowest database-key in the named area. If there are no records in the named area, Error Status Condition 0307, that is, an End of Area Condition, results.

(6) LAST RECORD OF area-name AREA is the record occurrence with the highest database-key in the named area. If there are no records in the named area, Error Status Condition 0307, that is, an End of Area Condition, results.

(7) FIRST RECORD OF set-name SET is the first member occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current record of the set is the owner record and the NEXT RECORD OF set-name SET rse is used. If the set occurrence is empty, that is, no member record occurrences participate in the set, Error Status Condition 0313 results.

(8) LAST RECORD OF set-name SET is the last member record occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current record of the set is the owner record and the PRIOR RECORD OF set-name SET rse is used. If the set occurrence is empty, that is, no member record occurrences participate in the set, Error Status Condition 0307 results.

(9) Identifier–3 must be initialized with an integer prior to execution of the FIND command. Identifier–3 represents the ordinal count of the object record occurrence relative to the beginning, if positive, or ending, if negative, of the set occurrence or the area. A negative value selects in the PRIOR direction and a positive value in the NEXT direction of the set occurrence or area. If the contents of identifier–3 are greater than the number of record occurrences in the set occurrence or area specified, then Error Status Condition 0307, that is, an End of Set or Area Condition results.

Whenever an Error Status Condition results which reflects an End of Area or Set Condition, then control is returned at the end-paragraph specified on the AT END clause, if present. If the AT END clause is not present, an End of Area or Set Condition is treated identically to any other Error Status Condition.

- *Format–4* of the rse is equivalent to *Format–3* of the rse except that only records of type record-name–2 are considered. In addition, the OWNER and CURRENT options discussed under *Format–3* of the rse do not apply as the former is meaningless and the latter is satisfied by *Format–2* of the rse. The same Error Status Conditions and End of Area or Set Conditions apply.

- *Format–5* of the rse, without the optional NEXT DUPLICATE clause, selects the object record on the basis of the LOCATION MODE clause defined for record-name–3 in the schema.

  If the LOCATION MODE OF record-name–3 is DIRECT:

  (1) The 77-level entry data-base-data-name–2 must be initialized with the area-name of the area in which the object record is stored.

(2) The 01-level entry data-base-data-name–1 must be initialized with a page number and a record number in the DML Preprocessor supplied

PAGE-NUM OF data-base-data-name–1
RECORD-NUM OF data-base-data-name–1

respectively.

If the LOCATION MODE OF THE RECORD IS CALC:

(1) Data-base-procedure–1 must have been included in the absolute element containing the run unit through the use of the @MAP Processor (see Appendix D).

(2) The 77-level entry data-base-data-name–3 must be initialized with the area-name of the area in which the object record is stored. This initialization may be done by either the run unit or data-base-procedure–1.

(3) The input data-items to data-base-procedure–1, that is, data-base-identifier–1, data-base-identifier–2... must be initialized with the argument values required to develop the proper page number and chain number.

If the LOCATION MODE of the record is VIA set-name–1 SET:

(1) The area-name and area-key of the object record is determined by the DMR.

(2) The selection of the object record occurrence is based upon its participation in set-name–1. The selection of the owner of the proper occurrence of set-name–1 is based upon the SET OCCURRENCE SELECTION clause for record-name–3 as a member of set-name–1. The selection of the object record occurrence from the selected set-name–1 set occurrence is based upon the ORDER clause of set-name–1.

(3) The selection of the proper occurrence of set-name–1 may or may not involve additional sets at higher levels of a hierarchy. While the Application Programmer need not be aware of the structure of the data-base, he must be aware of and initialize whatever key data items are required. The Data Administrator is aware of, and must furnish the Application Programmer with the names and values of the key items requiring initialization. See *UNIVAC 1100 Series Data Management System (DMS 1100) Schema Definition Data Administrator Reference, UP-7907* (current version) for complete details.

- *Format–5* of the rse, with the optional NEXT DUPLICATE clause may be used only when the LOCATION MODE clause for record-name–3 is CALC. The object record is the next record of type record-name–3 found on the same chain as the current record of record-name–3 having the same values for data-base-identifier–1, data-base-identifier–2... as those present in the run unit and in the current record of record-name–3. If record-name–3 does not have a LOCATION MODE IS CALC, then Error Status Condition 0313 results. If no Duplicate record occurrence is found, then Error Status Condition 0307 results. If no current record of record-name–3 exists, then Error Status Condition 0306 results.

- *Format–6* of the rse causes the object record to be selected based upon its participation as a member of set-name–5. The selection of the owner of the proper occurrence of set-name–5 is based upon the SET OCCURRENCE SELECTION clause for record-name–4 as a member of set-name–5.

  If the USING option is not present, selection of the object record occurrence from the selected set-name–5 set occurrence is based upon the ORDER clause of set-name–5. If the USING option is present, then

selection of the object record occurrence from the selected set-name–5 set occurrence is based upon the values initialized in data-base-identifier–1, data-base-identifier–2... The object record occurrence selected is the first record, in terms of scanning the set in the next direction, which has values equal to data-base-identifier–1, data-base-identifier–2. If no record is found, then Error Status Condition 0313 results.

The selection of the proper occurrence of set-name–5 may or may not involve additional sets at higher levels of a hierarchy. As in *Format–5,* when the LOCATION MODE IS VIA set-name SET, the Application Programmer need not be aware of the structure of the database, however, he must be aware of and must furnish the Application Programmer with names and values of the key items requiring initialization. See the *UNIVAC 1100 Series Data Management System (DMS 1100) Schema Definition Data Administrator Reference, UP-7907* (current version) for complete details.

- *Format–7* of the rse causes a search of the members of the current set occurrence of set-name–6 for a record which is of the same type, and which has the same values for data-base-identifier–3, data-base-identifier–4..., as the current record of set-name–6. The search is in the next direction and starts from the current record of set-name–6. It continues until either a duplicate is found or until the owner record occurrence is reached. If no duplicate is found, Error Status Condition 0307 results. If there is no current record of set-name–6, Error Status Condition 0306 results.

- A successfully executed FIND/FETCH command results in the record occurrence selected becoming the current record of the run unit, its area-name is placed in the DMCA reserved location AREA-NAME, its page number and record number are placed in the DMCA reserved locations PAGE-NUM and RECORD-NUM of AREA-KEY, and its record name is placed in the DMCA reserved location RECORD-NAME.

- In the absence of the optional SUPPRESS clause, the object record also becomes the current record of its area-name, the current record of its record-name, and the current record of all set-names in which it is defined as an owner or in which it currently participates as a member.

- The SUPPRESS clause specifies the AREA, RECORD and SETS or set-names currency indicators which are to retain their existing status. The SUPPRESS ALL clause prevents the update of all currency indicators except the current of run unit.

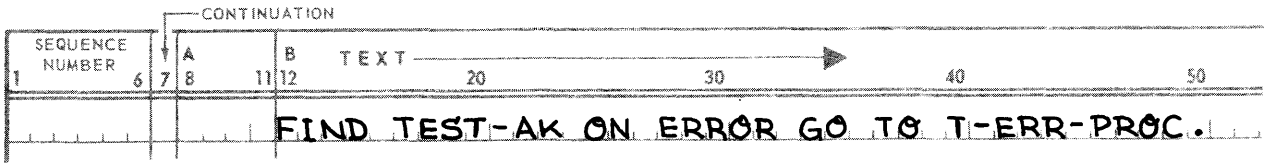
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
000301	area not open
000306	no current of record-name or area-name
000307	end of area or set
000313	no record satisfies criteria

Examples and Discussion:

- **Example 1:**



The data-name 'TEST-AK' must appear as an elementary item with a USAGE IS AREA-KEY in the users run unit and it must be initialized with the area-key of the desired record, in the area whose name is in the reserved data-name "AREA-NAME". This record becomes the current record of run-unit, of the area in which it is stored, of record-name, and of all sets in which it participates as owner or member.

- **Example 2:**

```
FIND CURRENT AREC1 RECORD SUPPRESS ALL.
```

The record which is the current record of type AREC1 becomes the current record of run-unit.

- **Example 3:**

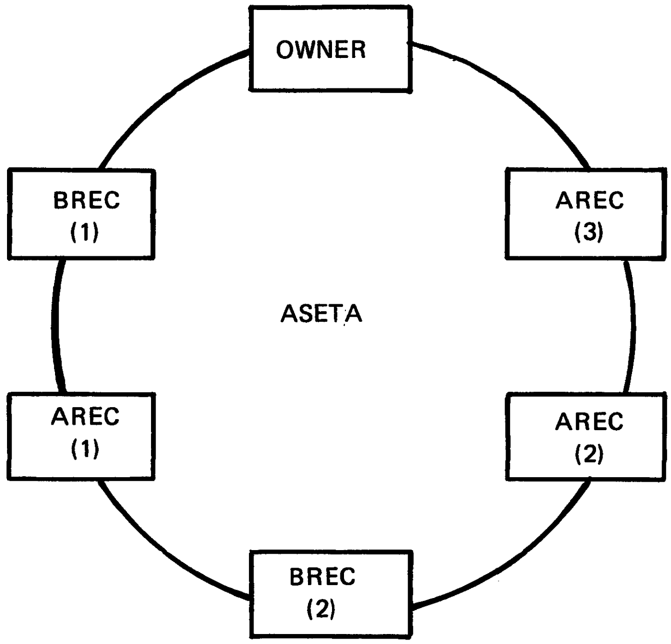
```
FIND NEXT RECORD WITHIN T-AREA1 AREA
SUPPRESS SETS.
```

The record with the next higher database key relative to the current record of area T-AREA1 becomes the current record of run-unit, the current record of record-name, and the current record of area T-AREA1. It does not become the current record of any set in which it participates.

- **Example 4:**

```
FETCH PRIOR AREC ASETA SET SUPPRESS RECORD
AT END GO TO T-AT-END-PROC ON ERROR GO TO
T-ERR-PROC.
```

Assume that the set occurrence with which the programmer is concerned contains the following records:



The record AREC, occurrence 2, is the current record of the set ASETA. After the above command is issued, the record AREC, occurrence 1, is in the record-delivery area. It is also the current record of run unit, the current record of the set ASETA, and the current record of the area in which it is stored.

If the same command is now issued, with AREC, occurrence 1, as current of ASETA, control is given to the COBOL paragraph T-AT-END-PROC.

- **Example 5:**

SEQUENCE NUMBER (1–6)	CONTINUATION (7)	A (8–11)	B (12) TEXT
			FIND CITY RECORD.

Assume that the portion of the data-base with which the programmer is concerned is organized as follows:

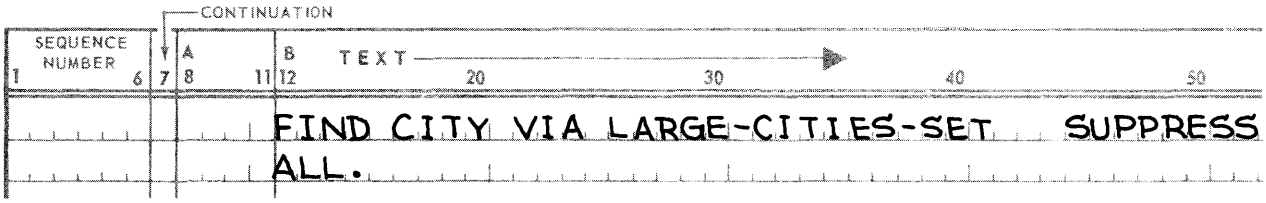
A record, COUNTRY, has a LOCATION MODE OF DIRECT, with COUNTRY-AN and COUNTRY-AK as the area-name and area-key. It is the OWNER of the set, STATES-SET. The members of STATES-SET are of the type STATE, and the set is ordered alphabetically on the key ST-NAME. LOCATION MODE of the record STATE is VIA STATES-SET. SET OCCURRENCE SELECTION is through LOCATION MODE OF OWNER.

Each STATE record is the OWNER of a set, CITIES-SET. The members of the set CITIES-SET are of type CITY. The set is sorted alphabetically on the key CITY-NAME. LOCATION MODE of the record CITY is VIA CITIES-SET, and SET OCCURRENCE SELECTION is through the LOCATION MODE OF OWNER.

Assume that the programmer wishes to access the record for Dallas, Texas, USA. The keys to be initialized are: COUNTRY-AN, COUNTRY-AK (with the area-name and area-key of the COUNTRY record of USA) ST-NAME (with 'Texas'), and CITY-NAME (with 'Dallas').

Note that it is not necessary for the programmer to be aware of the actual structure of the data base. He only needs to know which keys to initialize.

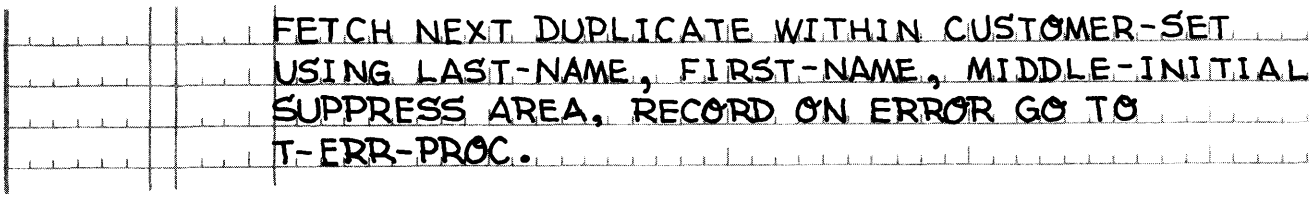
- **Example 6:**



Assume that, in addition to the data-base organization described in Example 5, there is an additional set, LARGE-CITIES-SET, of which the record CITY is also a member. *Format–6* allows the programmer to access the record CITY as a member of this set, i.e., it overrides the LOCATION MODE specified for the record.

For example, the set LARGE-CITIES-SET, whose owner is the STATE record defined in the previous example, is sorted on an ascending range of key, CITY-SIZE, and its SET OCCURRENCE SELECTION is CURRENT OF SET. The programmer must initialize the key CITY-SIZE, to, for example, 100,000. Assume the Texas STATE record is the owner of the current set of type LARGE-CITIES-SET. The record of the city in Texas whose Size is closest to, but not less than, 100,000 becomes the current record of run-unit.

- **Example 7:**



Assume that the data-base contains a set, CUSTOMER-SET, whose member records are of type CUSTOMER-RECORD. The set is sorted alphabetically on three keys, LAST-NAME, FIRST-NAME, and MIDDLE-INITIAL.

DUPLICATES are allowed in the set. Assume further that the current record of the set is a CUSTOMER-RECORD for John J. Jones. If there is another customer whose name is also John J. Jones, his CUSTOMER-RECORD is placed in the record-delivery-area and becomes the current record of run-unit and of the set, CUSTOMER-SET. If no other John J. Jones CUSTOMER-RECORD exists, or if some other error occurs, control is given to T-ERR-PROC.

3.4.6. FREE Command

Function:

The FREE releases all page locks and quick before looks to enable other run units to access these pages.

Syntax Skeleton:

FREE [ON ERROR GO TO *error-paragraph*].

Syntactical Constraints: None

Data Manipulation Guide:

- All alter locks on pages are released.
- All quick looks are released.
- A FREE indication is placed on the Audit file.
- As a result of a FREE all recovery and rollback functions handle the point in time at which the FREE was performed as though a DEPART has been performed. This means that on run unit rollback, the database is restored to the state that existed when the FREE command was issued. Also, on recovery when run units active at a recovery point are backed out of the system, they are backed out to the point of the most immediate FREE in front of the recovery point if any FREE was issued.
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.

Example and Discussion: None

3.4.7. GET Command

Function:

To transfer the contents of the object record into the record-delivery-area.

Syntax Skeleton:

GET [ON ERROR GO TO *error-paragraph*].

Syntactical Constraints:

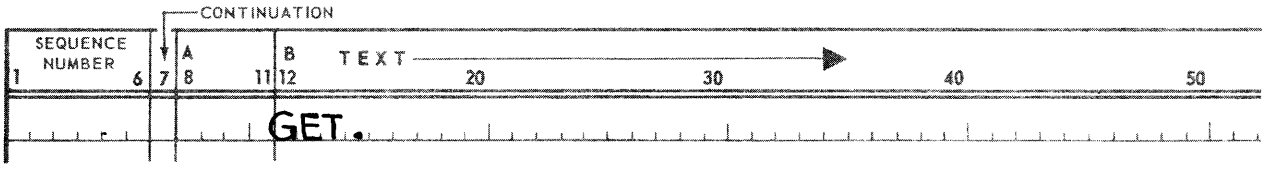
Error-paragraph must be a Paragraph Name or Section Name, defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

Data Manipulation Guides:

- The object record of the GET command is the current record of the run unit.
- If a FIND command was used to establish the object record, then the GET command must be executed before any reference can be made to the data-items of the object record.
- If a FETCH command was used to establish the object record, then the GET command is redundant as the FETCH includes an implicit GET of the object record.
- The GET transfers the data-items of the object record into the record-delivery-area. If the record description for the object record type does not overlay the record-delivery-area, then a COBOL MOVE verb must be used to transfer the data-items from the record-delivery-area to the object record description. This must be done prior to any reference to a data-item, either by the run unit or by the DMR during the execution of a subsequent command.
- If no current record of the run unit is known, then Error Status Condition 0513 results.
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.
- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
000513	no current record of run unit

Examples and Discussion:



The current record of run-unit is retrieved and delivered to the record-delivery-area, which was specified in the Schema Section. If the record-delivery-area is not OVERLAY'd by this record description, the user must use a COBOL MOVE verb to transfer the record from the record-delivery-area to the record-description.

If no current record of run-unit exists, an ERROR STATUS of 0513 is returned to the general-error-paragraph specified in the Schema Section, or to the next sentence if no general-error-paragraph is specified.

3.4.8. IF Command

Function:

The IF command causes a condition to be evaluated. The subsequent action of the run unit depends on whether the value of the condition is true or false.

Syntax Skeleton:

Format–1

```
IF set-name–1 SET [NOT] EMPTY; { statement–1 }
                               { NEXT SENTENCE }

[ELSE { statement–2 }]
      { NEXT SENTENCE }
```

Format–2

```
IF RECORD [NOT] { OWNER  } OF { set-name–2 } SET
                { MEMBER }   {     ANY     }

{ statement–3 }  [ELSE  { statement–4 }]
{ NEXT SENTENCE }       { NEXT SENTENCE }
```

Syntactical Constraints:

Set-name–1 and set-name–2 must be defined in the schema named in the INVOKE clause of the Schema Section.

Data Manipulation Guides:

- *Format–1* of the IF command is designed to determine whether the object set occurrence has any members. The object set occurrence is determined by the current record of set-name–1. If the NOT option is omitted and the set occurrence does not have any member records, the condition is evaluated as true. If the NOT option is omitted and the set occurrence contains member records, the condition is evaluated as false. If the NOT option is stated, the condition is reversed. Regardless of whether the NOT option is stated, the condition is evaluated as false if no current record of set-name–1 exists.

- *Format–2* of the IF command is designed to determine whether the current record of the run unit currently participates as an owner or member, depending on the option specified; in set-name–2 or in any set. To be an owner, for the purposes of the IF command, the current of run unit must have member records currently participating in its set occurrence. If the NOT option is omitted and the record is an owner or member as specified, the condition is evaluated as true. If the NOT option is omitted and the record is not an owner or member as specified, the condition is evaluated as false. If the NOT option is stated, the condition is reversed. Regardless of whether the NOT option is stated, the condition is evaluated as false if no current record of the run unit exists.

- Whenever the condition evaluated in the IF statement is true, control is transferred to the first statement named (*statement–1 or statement–3*) or to the NEXT SENTENCE. If the condition is false and the ELSE

clause is present, then control is transferred to the statement following the ELSE clause (statement–2 or statement–4) or to the NEXT SENTENCE. If the ELSE clause is not present, control is transferred to the next command.

- If the condition is evaluated as true as a result of a successful completion of the IF command, then no Error Status Condition results. If the condition is evaluated as false as a result of a successful completion of the IF command, then Error Status Condition 1401 results. If the condition is evaluated as false as a result of an unsuccessful completion of the IF command, then Error Status Condition 1402 results if the record is not defined to be an owner or member of the named set; Error Status Condition 1406 results if no current of set name exists; or Error Status Condition 1413 results if no current of run unit exists.

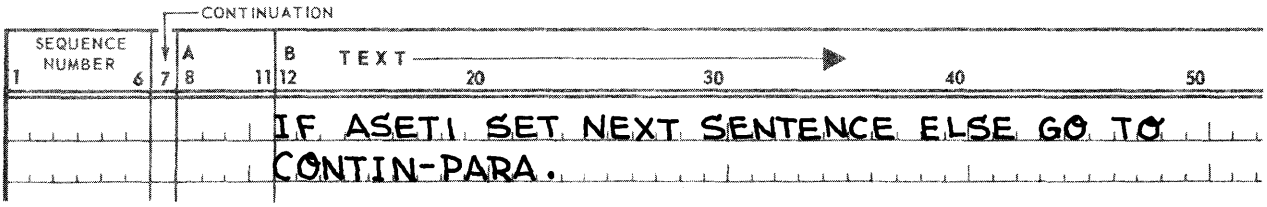
- The ON ERROR clause must not be attached to an IF command. The ERROR clause of the SCHEMA SECTION likewise does not apply to the IF command. Therefore, any error processing logic for Error Status Conditions 1402, 1406, or 1413 must be entered procedurally.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion, condition evaluated as true
001401	successful completion, condition evaluated as false
001402	the object record occurrence is not defined as an owner or member of the named set, condition evaluated as false
001406	no current of named set, condition evaluated as false
001413	no current of run unit, condition evaluated as false

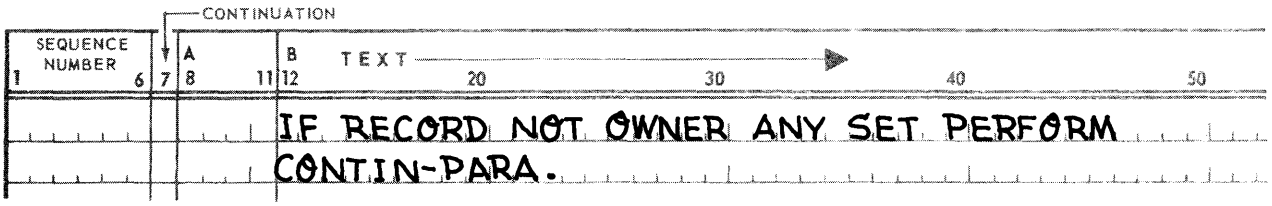
Examples and Discussion:

- **Example 1:**



If the SET ASET1 does not have a current record, or if that current record is the owner of an ASET1 SET and no members of that set exist, control is given to the next sentence.

- **Example 2:**



If the current record of run-unit does not currently participate as an owner of any non-empty set occurrence the procedure 'CONTIN-PARA' is performed, and control is returned to the next sentence. If the current record of run-unit does participate as an owner of any non-empty set, control is given to the next sentence.

3.4.9. IMPART Command

Function:

To register the run unit with the DMR and cause the DMR to create the necessary interface with the object schema.

Snytax Skeleton:

IMPART [ON ERROR GO TO *error-paragraph*].

Syntactical Constraints:

- Error-paragraph must be a Paragraph Name or Section Name defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

- IMPART may not appear in a USE FOR RANDOM PROCESSING procedure.

Data Manipulation Guides:

- The IMPART command must be executed by the run unit prior to the execution of any other DML commands.

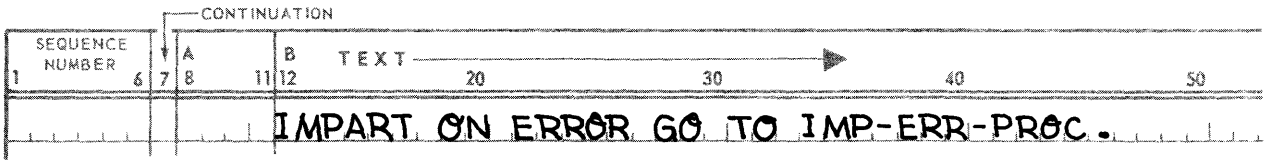
- If the IMPART command is executed by the same run unit a second time (*without an intervening DEPART*), Error Status Condition 1522 occurs.

- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
001522	IMPART already done

Examples and Discussion:



The run-unit may now issue any other DML command. If an IMPART has already been completed, an ERROR-STATUS of 01522 is returned to the COBOL paragraph 'IMP-ERR-PROC'.

3.4.10. INSERT Command

Function:

To make the object record a member of occurrences of the specified set-names, provided that it is defined as a manual member of those sets.

Syntax Skeleton:

Format–1

INSERT **INTO** *set-name–1* [,**INTO** *set-name–2*] ...

[ON ERROR GO **TO** *error-paragraph*] .

Format–2

INSERT **INTO** ALL **SETS** [ON ERROR GO **TO** *error-paragraph*] .

Syntactical Constraints:

- Set-name–1, set-name–2... must be the names of sets defined in the schema named by the INVOKE clause of the Schema Section.
- No more than 36 set-names may be specified in a single INSERT statement.
- Error-paragraph must be a Paragraph Name or Section Name, defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

Data Manipulation Guides:

- The object record occurrence of the INSERT command is the current record of the run unit. If no current record of the run unit exists, then Error Status Condition 0713 results.

- If *Format–1* of the INSERT command is used, then the object record must have been defined in the schema as a manual member of each set named, or Error Status Condition 0714 results. It will be inserted into the object set occurrence of each set-name specified in accordance with the set ordering criteria defined in the schema. For each set named, the object set occurrence is determined by the current record of the named set. If the current record of any set-name specified is not known, then Error Status Condition 0706 results.

- If *Format–2* of the INSERT command is used, the object record is inserted into the appropriate occurrence of each set in which it is defined as a manual member in the schema and in which it is not currently participating. The specific occurrence of each set is determined by the current record of set-name for each of the set-names involved. If the current record of any set-name involved is not known, then Error Status Condition 0706 results. The object record will be inserted into each set occurrence in accordance with the set ordering criteria specified in the schema.

- The number of set occurrences in which a record may participate in simultaneously as a manual member is constrained by the number of pointer locations available to contain whatever required combination of next, prior, or owner pointers. The number of pointer locations avilable is defined on the RESERVE clause of the respective Record Entry in the schema. If an insufficient number of pointer locations are available to accommodate all the concurrent set memberships, then Error Status Condition 0742 results.

- If the set ordering criteria specifies a SET ORDER IS SORTED, and the DUPLICATES ARE NOT ALLOWED clause is present on the ASCENDING/DESCENDING KEY clause, then, if insertion of the object record would violate the DUPLICATES ARE NOT ALLOWED clause, Error Status Condition 0705 results.

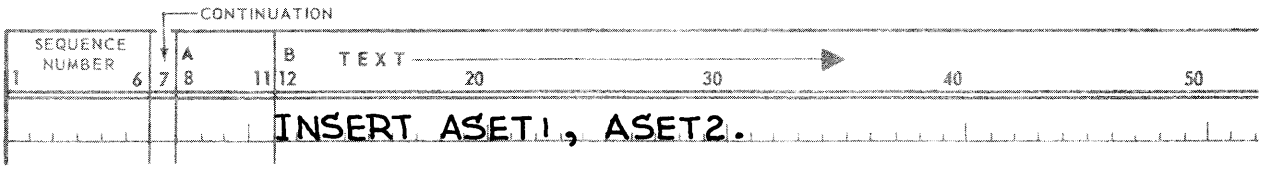
- If *Format–1* of the INSERT command is used, and the object record already participates as a member of any named set, then Error Status Condition 0716 results. If *Format–2* is used, and the object record already participates as a member in any set in which it is defined as a manual member, then the insertion within that set is considered complete, and the insertion operation advances to the next manual set and no Error Status Condition results.

- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
000705	duplicates are not allowed
000706	no current record of set-name
000713	no current record of run unit
000714	object record is not a manual member of the named set
000716	object record is already a member of named set
000742	insufficient pointer locations available

Examples and Discussion:



The record which is current of run-unit is inserted into the current set occurrences of ASET1 and ASET2 as a manual member. If no current of run-unit exists, or if the current of run-unit is not defined in the schema as a manual member of ASET1 and/or ASET2, or if any other error occurs, a non-zero ERROR STATUS is returned to the general-error-paragraph specified in the Schema Section or to the next sentence if no general-error-paragraph is specified.

3.4.11. KEEP Command

Function:

- To advise the System of the run unit's intent to re-access the current record of run unit.
- To apply an alter lock to the page even though no alteration has occurred.

Syntax Skeleton:

KEEP [ON ERROR GO TO *error-paragraph*].

Syntactical Constraints: None

Data Manipulation Guide:

- The object record of the KEEP command is the current record of the run unit.
- Termination of a run unit causes the removal of the alter lock from a.l record occurrences to which it has been applied by the run unit.
- When an AREA is OPEN with a USAGE-MODE of EXCLUSIVE or PROTECTED, the KEEP command need not be used since no update of the AREA can be made by concurrent run units.
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph was specified in the Schema Section, in which case control is returned to it.

Examples and Discussion:

- Assume run unit 1 accesses Record Z through a 'FETCH Z' command moves the record to its working storage, inspects the record, and then wants to 'FIND X' to use some item in Record X in conjunction with items in Record Z. If run unit 1 next issues a 'FETCH X', Record Z is unlocked and could be accessed by another run unit. To prevent another run unit from intervening, run unit 1 would execute a KEEP.

- Assume run unit 1 accesses record Z through a 'FIND Z' to establish Z as current of SET A. If run unit 1 then issues a 'FIND Y', Z is no longer locked and might be altered by a concurrent run unit. If another run unit alters Z and run unit 1 attempts to execute a command, e.g., 'INSERT INTO SET A', dependent on Z being current of SET A; run unit 1 may get incorrect results. In order to prevent interference from concurrent run units, run unit 1 should have the AREA OPEN with PROTECTED or EXCLUSIVE USAGE-MODE; or run unit 1 must perform a 'KEEP' after issuing the 'FIND Z' to prevent a concurrent run unit from altering Z prior to run unit 1's accessing of the record.

3.4.12. LOG Command

Function:

To provide a user with the mechanism necessary to save its non-DMS data on the DMS Audit Trail Tape File.

Syntax Skeleton:

Format–1

LOG *data-name–1* **WORDS** [ON ERROR GO **TO** *error-paragraph*].

Format–2

LOG *data-name–1* **WORDS FOR** RECOVERY **POINT** [ON ERROR GO **TO** *error-paragraph*].

Syntactical Constraints:

- Data-name–1 must be defined as:

  77 level PIC 9(10) USAGE IS COMPUTATIONAL.

- Error-paragraph must be a paragraph name or section defined in the non-declarative portion of the PROCEDURE DIVISION. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

Data Manipulation Guide:

- The information the run unit requests to be placed on the audit file must be located in the run unit's RECORD DELIVERY-AREA. S1 of the first word of the RDA must not be a U, since U indicates DMS data and H2 of the second word in the RDA is overlayed with the page sequence number.

- The contents of data-name–1 must be less than or equal to RDA length.

- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph was specified in the Schema Section, in which case control is returned to it.

- When a recovery point is specified, a recovery point block is written on the audit trail following the run units LOG information in the RDA.

Examples and Discussion: None

3.4.13. MODIFY Command

Function:

- To replace the values of the data-items of the object record occurrence in the database with the values from the record-delivery-area.

- To alter set occurrence membership and intraset position, so as to maintain the database in accordance with the set occurrence selection and ordering criteria specified in the schema.

Syntax Skeleton:

MODIFY [*data-base-identifier–1* [, *data-base-identifier–2*] ...] [ON ERROR GO TO *error-paragraph*] .

Syntactical Constraints:

- Data-base-identifier–1, data-base-identifier–2... must refer to data-items of the immediate owner record(s).

- The record types containing data-base-identifier–1, data-base-identifier–2... must be defined in the schema named on the INVOKE clause in the Schema Section. If the COPYING option is used on the INVOKE clause, the records must be included among the records copied.

- A maximum of 47 data-base-identifiers can be specified on a MODIFY command.

- Error-paragraph must be a Paragraph Name or Section Name defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

Data Manipulation Guides:

- The object record occurrence of the MODIFY command is the current record of the run unit. If there is no current record of the run unit, the MODIFY is not executed and Error Status Condition 0813 results.

- The result of a successful execution of the MODIFY command is that the values of the data-items of the object record occurrence in the database are replaced with the values from the record-delivery-area. Therefore, if the record description of the current of run unit record types does not overlay the record-delivery area, then a COBOL MOVE verb must be used to transfer the data-items from the object record description to the record-delivery-area prior to the execution of the MODIFY command.

- As a result of the execution of a MODIFY command both the intra-set occurrence position and the interset occurrence position may change in those sets in which the object record occurrence currently participates as a member. However, the database-key of the object record occurrence remains unchanged.

  (1) If the data-base-identifier–1, data-base-identifier–2.. option is not used and if any of the modified data items are defined as sort-control items, that is, as ASCENDING/DESCENDING keys, in the object record for and set occurrence in which the object record is currently a member, then its modification causes the intra-set occurrence position of the object record to be examined; and if necessary, the object record is removed and reinserted in the set occurrence to maintain the set order specified in the schema.

  (2) If the data-base-identifier–1, data-base-identifier–2... option is used, the effect is not to change the values of the data-base-identifiers specified, but rather to change the set occurrence, or occurrences, in which the object record participates. The data-base-identifiers specified must refer to data items of an owner record of the object (*member*) record. The data-base-identifiers specified, because they identify the owner record containing them, thus identify the set or sets involved.

  If the object record is an AUTOMATIC MEMBER of a set thus identified, the object record is removed from its present set occurrence and inserted into a new set occurrence which is determined by the relevant SET OCCURRENCE SELECTION clause. Data items which control set occurrence selection at levels in the hierarchy higher than the ones in which the object record participates must also be initialized. If such a set occurrence is not found, then Error Status Conditions 0813 results. If the object record is not defined as a member of a set-name implicitly specified by data-base-identifier–1, data-base-identifier–2.., then Error Status Condition 0808 results. If the specific set occurrence is found, then the insertion occurs in accordance with the ordering criteria defined for the object record as a member of the set.

  If the object record is a MANUAL MEMBER of a set thus identified, the object record is removed from its present set occurrence and inserted into the set occurrence which is identifier by the current record of that set-name. This requires that the set occurrence into which the object record is to be inserted must have been procedurally selected by a previous FIND/FETCH or STORE command. If the current record of set-name is null, then Error Status Condition 0806 results. If the object record is not defined as a member of a set-name or is not currently participating as a member of an occurrence of a set-name implicitly specified by data-base-identifier–1, data-base-identifier–2..., then Error Status Condition 0808 results. Insertion into the set occurrence identified by the current of set-name occurs in accordance with the ordering criteria defined for the object record as a member of the set.

- If the insertion of the object record into a set occurrence violates a DUPLICATES NOT ALLOWED clause specified for that set, then Error Status Condition 0805 results.

- If as a result of a MODIFY, the object record has its inter-set or intra-set position altered, and if the object record was the current of that set-name, then the current record of that set-name becomes null. However, while the current record of set-name is no longer accessible, the next, prior, and owner of set-name remain accessible.

- If the LOCATION MODE of the object record IS CALC and any of the input parameters to the CALC procedure are altered, then such alteration causes the DMR to recall the CALC procedure to determine a new page number, chain number and area-name (*therefore, the area-name must have been previously initialized if not automatically done by the CALC procedure*) based upon the altered input values. The

resulting area-name, page number and chain number must meet the same requirements imposed as if the object record was being stored. That is, the resulting area-name and page number must not violate the area and page limits specified on the WITHIN clause for the record, and the calc chain number must not violate the limits specified on the CALC USES clause for the area. In addition, the altered input values to the CALC procedure must not violate the DUPLICATES option on the record LOCATION MODE clause. The new area-name, page number and chain number form the basis for future comparisons for duplicates. Alteration of the area-name, page number and chain number do not physically alter the record's area and page position (see *UNIVAC 1100 Series Data Management System (DMS 1100) Schema Definition Reference, UP-7907* (current version) for further details).

- If the definition of the object record contains an OCCURS...DEPENDING clause, that is, it is a variable length record, then its alteration may affect record size (see *UNIVAC 1100 Series Data Management System (DMS 1100) Schema Definition Reference, UP-7907* (current version), for further details).

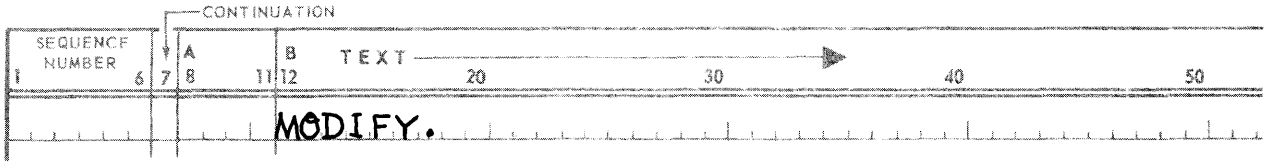
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
000805	violation of DUPLICATES NOT ALLOWED clause
000806	no current record of set-name
000808	record not participating as member of implied set
000813	no current record of run unit

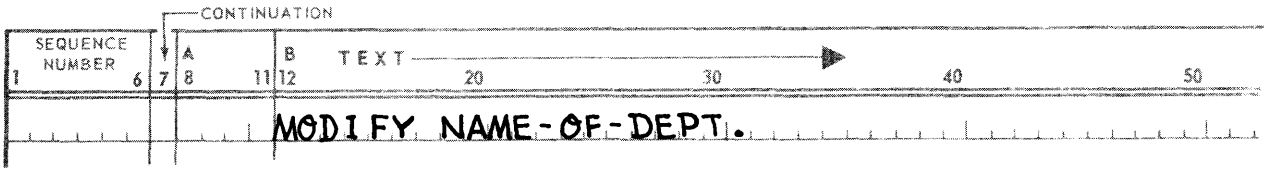
Examples and Discussion:

- **Example 1:**



Assume that the current record of run-unit is a record of type EMPLOYEE and is in the record-delivery-area. It is a member of an occurrence of a set, PERSONNEL. Members of this set are SORTED alphabetically according to the item LAST-NAME in each EMPLOYEE record. Assume that, in the current record of run-unit, the item LAST-NAME is 'Jones', and due to marriage, is now 'Smith'. The issuance of the MODIFY causes this record to be repositioned in the occurrence of the set, PERSONNEL, in which it is currently participating.

■ **Example 2:**



Assume the portion of the database with which the run unit is concerned is defined as follows.

The set, DEPTS, has as owner a record of type COMPANY whose LOCATION MODE IS DIRECT. Its automatic members are of one type, DEPARTMENT. The set is sorted, and the item NAME-OF-DEPT in the DEPARTMENT record is the key on which the set is sorted. The LOCATION MODE of the DEPARTMENT record is VIA DEPTS SET.

The set, PERSONNEL, has as owner a record of type DEPARTMENT. Its automatic members are of one type, EMPLOYEE. Set occurrence selection for EMPLOYEE as a member of the PERSONNEL set is VIA LOCATION MODE of OWNER.

The current record of run-unit is an EMPLOYEE record. It is a member of a set occurrence owned by a DEPARTMENT record whose NAME-OF-DEPT is "Public Relations". The employee has been transferred to the "Payroll" department.

Prior to the issuance of the MODIFY, the run unit, which has already done a FIND and GET or FETCH, must MOVE the new values into the object record (if the object record does not redefine the record delivery-area, it must then be moved back to the record delivery-area).

Because the object record is to change PERSONNEL set occurrences, the item, NAME-OF-DEPT, in its immediate owner (DEPARTMENT) record must be initialized with "Payroll" to cause the DMR to select the proper new set occurrence.

At the highest (COMPANY record) level assume the area-name and area-key data-base-data-names used to select the proper COMPANY record remain initialized from the FIND/FETCH command which obtained the object record. Note that if the employee was being transferred to a different company, say a subsidiary company, the data-base-data-name used to select the proper COMPANY record would have to be initialized to select the new company record. Note further that no additional data-base-identifiers are required on the MODIFY command (this is true no matter how far up the hierarchy the change occurs) because identification of the immediate owner identifies the complete hierarchy.

The run unit now executes the MODIFY command and the object EMPLOYEE record is removed from its position in the "Public Relations" set occurrence and inserted into the "Payroll" set occurrence in accordance with the set ordering criteria specified for the PERSONNEL set.

■ **Example 3:**

Assume that in the description of the database for Example 2, the EMPLOYEE record type was a MANUAL member of the PERSONNEL set instead of an AUTOMATIC member.

Prior to the execution of the FIND/FETCH command for the object EMPLOYEE record, the run unit must execute a FIND/FETCH command for the DEPARTMENT record whose NAME-OF-DEPT is "Payroll" (or for an EMPLOYEE record owned by the "Payroll" DEPARTMENT record occurrence). In either event, the current record of PERSONNEL set-name now identifies the "Payroll" set occurrence, because the currency status (in the absence of the SUPPRESS clause) is updated for every set in which the object record of the command participates.

Once the "Payroll" set occurrence is the current occurrence of set-name, the FIND/FETCH for the object EMPLOYEE record in the "Public Relations" set occurrence may be issued. However, because this FIND/FETCH also updates currency status, a SUPPRESS PERSONNEL CURRENCY UPDATES clause is also required.

The object EMPLOYEE record is now current of run unit and also identifies the "Public Relations" set occurrence. The current of PERSONNEL set-name identifies the "Payroll" set occurrence. Execution of the MODIFY at this point results in the removal of the object EMPLOYEE record from the "Public Relations" set occurrence and its insertion into the "Payroll" set occurrence.

3.4.14. MOVE Command (DML)

Function:

- To pass the contents of the specified currency status indicator to the run unit either as a database-key or as area-name and/or area-key.
- To convert a database-key into an area-name and/or area-key.

Syntax Skeleton:

Format–1

```
MOVE CURRENCY STATUS FOR { RUN-UNIT                  }
                         { record-name RECORD        }
                         { area-name AREA            }
                         { set-name SET              }

    TO identifier–1 [ON ERROR GO TO error-paragraph].
```

Format–2

```
                                              ⎧ RUN-UNIT                   ⎫
MOVE ‖ AREA-KEY  ‖   FOR                      ⎪ record-name RECORD         ⎪
     ‖ AREA-NAME ‖                            ⎨ area-name   AREA           ⎬
                                              ⎪ set-name    SET            ⎪
                                              ⎩ identifier–2               ⎭

   TO ‖ identifier–3 ‖   [ON ERROR GO TO error-paragraph].
      ‖ identifier–4 ‖
```

Syntactical Constraints:

- The specified record-name, area-name, or set-name must be included in the Schema named in the Schema Section of the program.
- Identifier–1 and identifier–2 must be defined with a USAGE IS DATABASE-KEY clause.
- Identifier–3 and identifier–4 must be defined with a USAGE IS AREA-NAME or AREA-KEY clause.

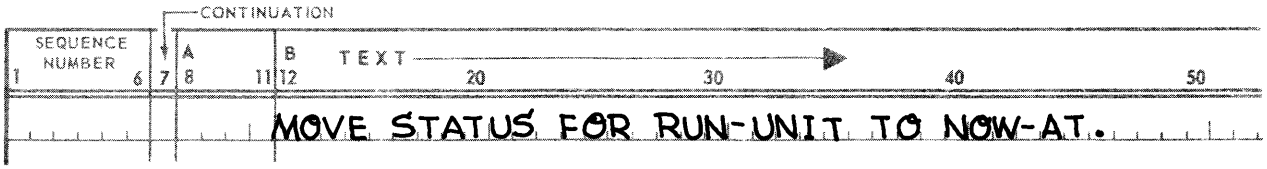
Data Manipulation Guides:

- In *Format–1*, if the RUN-UNIT option is specified, the database-key for the current record of run-unit is placed in identifier–1. The current record of the run-unit is not altered. If a record-name, area-name, or set-name is specified, the data-base-key for the current record of record-name, area-name, or set-name is moved to identifier–1. The current record of record-name, area-name, or set-name is not altered.
- If identifier–2 is specified in *Format–2*, it must contain a database-key. The area-name, area-key or both which correspond to this database-key are stored in identifier–3 and identifier–4. If both AREA-KEY and AREA-NAME are specified in that order, then the area-key goes to identifier–3 and the area-name to identifier–4. If both are specified in the reverse order, they go into identifier–4 and identifier–3, respectively.
- If one of the other forms of *Format–2* is used, the area-name, area-key or both are moved into identifier–3 and identifier–4 following the same rules as in Data Manipulation Guide (*first item*).
- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
001652	area/record/set not invoked
001606	no current record of area/record/set
001613	no current record of run unit

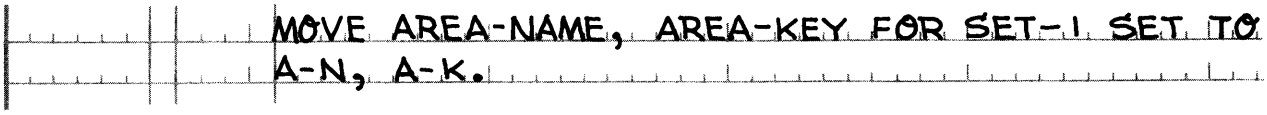
Examples and Discussion:

- **Example 1:**



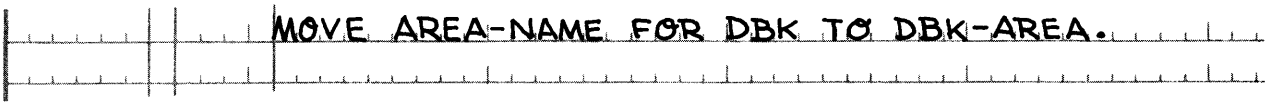
The database-key for the current record of run-unit is stored in the data item NOW-AT. NOW-AT must be defined with USAGE IS DATABASE-KEY.

- **Example 2:**



The area-name represented by the area code present in the current database-key of set SET-1 is stored in A-N. A-N must be defined with a usage of AREA-NAME. The page number and record number of the same database-key are stored in A-K. A-K must be defined with a usage of AREA-KEY.

- **Example 3:**



The area-name represented by the area code present in the database-key stored in DBK is stored in DBK-AREA.

3.4.14-a. MOVE Command (COBOL)

Function:

To cause certain cases of the COBOL MOVE command to be handled by the DMR. These cases are:

- The transferral of a record from its record-description to the record-delivery-area; and,
- The transferral of a record from the record-delivery-area to its record-description.

Syntax Skeleton:

MOVE *record-delivery-area* TO *record-name–1.*

or

MOVE *record-name–2* TO *record-delivery-area.*

Syntactical Restraints:

Multiple receiving fields are not permitted.

Data Manipulation Guides:

The DMR handles the above MOVE's because if the group USAGE's of the record-description and the record-delivery are not the same (*i.e., one is ASCII, the other Fieldata*), an ASCII COBOL 'MOVE' causes a conversion of data to be performed before the MOVE is done.

Examples and Discussion: None

3.4.15. OPEN Command

Function:

To indicate a run unit's intent to access an area and the conditions associated with the access.

Syntax Skeleton:

Format–1

```
OPEN ALL [ USAGE-MODE IS { [ EXCLUSIVE ]          { RETRIEVAL } } ]
                         { [ PROTECTED ]          { UPDATE    } }
                           INITIAL LOAD

    [ON ERROR GO TO error-paragraph].
```

Format–2

```
OPEN area-name–1 [ USAGE-MODE IS { [ EXCLUSIVE ]          { RETRIEVAL } } ]
                                 { [ PROTECTED ]          { UPDATE    } }
                                   INITIAL LOAD

[ , area-name–2 [ USAGE-MODE IS { [ EXCLUSIVE ]          { RETRIEVAL } } ] ] ...
                                { [ PROTECTED ]          { UPDATE    } }
                                  INITIAL LOAD

[ON ERROR GO TO error-paragraph].
```

Syntactical Constraints:

- In *Format–1* ALL refers to every area defined in the schema named by the INVOKE clause of the Schema Section.
- In *Format–2* area-name–1, area-name–2... must be the names of areas defined in the schema named by the INVOKE clause of the Schema Section.
- File-names specified on the COBOL OPEN verb and area-names specified on the DML OPEN command cannot be intermixed on the same OPEN.
- No more than 47 area-names may be specified on a single OPEN command.
- Error-paragraph must be a Paragraph Name or Section Name, defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50-99*).

Data Manipulation Guides:

- If an OPEN command is to be executed for an area, either indirectly through *Format–1* or directly through *Format–2*, the mass storage for the area must have first been assigned to the run stream of which the run unit is a part unless the area is under TIP file control. For an Executive this must be accomplished with the appropriate @ASG or @USE control card using a file name identical to the area name. A TIP file must have been assigned (i.e., mass storage space allocated) via TIP system's generation or a TIP control stream statement.
- A USAGE-MODE of LOAD implies the following:

  (1) A usage of EXCLUSIVE UPDATE.

  (2) The LOAD factor specified in the Schema for the applicable AREAs is used in determining when a page is full.

  (3) Overflow pages will be used for storing CALC records, *not* when storing records DIRECT or VIA SET.

  (4) If 'pre-initialized' pages have been specified in the schema "ALLOCATE" clause, the DMS Utility Processor for initializing pages will have to be run before doing an OPEN for INITIAL LOAD or an OPEN for INITIAL LOAD must previously have been performed for this area when pre-initialized pages were *not* specified. Initialization sets all the pages in an AREA to contain only a page header with the remainder of the page set to zero. If 'Pre-initialized' pages have *not* been specified, the pages will be initialized during the OPEN for INITIAL LOAD. In this case pages are initialized as they are are referenced by DML commands; that is, all pages up to the highest page referenced are initialized. All remaining un-initialized pages in an area OPEN for INITIAL LOAD are initialized by a CLOSE or a DEPART command. If the applicable non pre-initialized areas contain information before the OPEN for INITIAL LOAD, the information will be destroyed during the loading.

  (5) During processing when an AREA is OPENed for INITIAL LOAD special recovery considerations are necessary (see *UNIVAC 1100 Series Data Management System (DMS 1100) System Support Functions Reference, UP-7909* (current version)). Before Looks go to the Audit-Trail tape for areas OPEN for INITIAL LOAD only after a FREE command has occurred and then only for those pages initialized or modified previous to the FREE command. These before-looks are used to re-establish the database to the way it was just previous to the last FREE command should recovery be necessary. However, after looks do go to the audit trail and if COMMAND QUICK-BEFORE-LOOKS are specified, they go to the quick look file.

- Use of the USAGE-MODE IS RETRIEVAL phrase (*without the EXCLUSIVE or PROTECTED phrases*) allows concurrent run units to open the same area with any usage-mode other than one which is exclusive.

- Use of the USAGE-MODE IS UPDATE phrase (*without the EXCLUSIVE or PROTECTED phrases*) allows concurrent run units to open the same area with any usage-mode other than one which is exclusive or protected.

- If the optional USAGE-MODE clause is omitted, then RETRIEVAL is assumed. RETRIEVAL does not interfere with any concurrent run unit.

- Use of the EXCLUSIVE phrase prevents concurrent run units from interacting with the same area in any usage-mode.

- Use of PROTECTED RETRIEVAL prevents concurrent UPDATE, including EXCLUSIVE and PROTECTED, and allows concurrent RETRIEVAL and PROTECTED RETRIEVAL, but not EXCLUSIVE RETRIEVAL.

- Use of PROTECTED UPDATE prevents concurrent UPDATE, including EXCLUSIVE and PROTECTED, and allows RETRIEVAL, but not EXCLUSIVE or PROTECTED.

- When an area is OPEN FOR RETRIEVAL, any STORE, MODIFY, INSERT, REMOVE and DELETE statements on record occurrences in that area will not be executed. Any attempts to do so will result in an error.

- All specified usage-modes remain in effect until the run unit issues a CLOSE statement for the specified areas, or until the run unit DEPARTs.

- To successfully execute a FIND, STORE, DELETE, or CLOSE statement, the run unit must have previously opened (*as relevant to the statement being executed*):

  (1) The area that contains the object record of a FIND statement.

  (2) The area into which a record is to be stored.

  (3) All areas containing any record occurrence which would be deleted or removed as a result of a DELETE statement.

  (4) All areas that are the object of a CLOSE statement.

  Otherwise Error Status Condition nn01 results, where nn indicates the particular DML statement being attempted.

  In addition to the areas containing the object records of the statements cited above, there are additional – that is, implicit areas which can be impacted by DML statements. The impact can be of two forms:

  (1) The DMR requires information contained within the implicit area; in which case the area must be OPENED.

  (2) The DMR must alter the information contained in record occurrences within the implicit area; in which case the area must not only be OPENED but it must permit the necessary alterations (*i.e., OPENED for update*).

- Record occurrences which are in the search path of the sought record of a FIND statement, or in the search path of the sought record of an implicit find which occurs during the execution of a STORE, DELETE, or INSERT statement, need to be in areas which are OPENED.

- To successfully execute an INSERT, REMOVE, STORE, DELETE or MODIFY statement, both the explicit and the altered implicit areas involved must be open with a usage-mode of update. If any of the areas involved are open for retrieval, Error Status Condition nn09 will result, where nn indicates the particular DMLP statement being attempted.

- Any attempt to execute an OPEN statement which would result in usage-mode conflict for an area will result in the run unit attempting to execute the OPEN statement being queued.

  The following table reflects usage-mode conflicts with an 'N' and permitted concurrence with a 'Y'.

FIRST OPEN USAGE MODE / NEXT OPEN USAGE MODE	INITIAL LOAD	RETRIEVAL	UPDATE	PROTECTED RETRIEVAL	PROTECTED UPDATE	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE
INITIAL	N	N	N	N	N	N	N
RETRIEVAL	N	Y	Y	Y	Y	N	N
UPDATE	N	Y	Y	N	N	N	N
PROTECTED RETRIEVAL	N	Y	N	Y	N	N	N
PROTECTED UPDATE	N	Y	N	N	N	N	N
EXCLUSIVE RETRIEVAL	N	N	N	N	N	N	N
EXCLUSIVE UPDATE	N	N	N	N	N	N	N

- An attempt by a run unit to execute an OPEN statement against an area which is already opened for that run unit will result in Error Status Condition 0928.

- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph was specified in the Schema Section, in which case control is returned to it.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

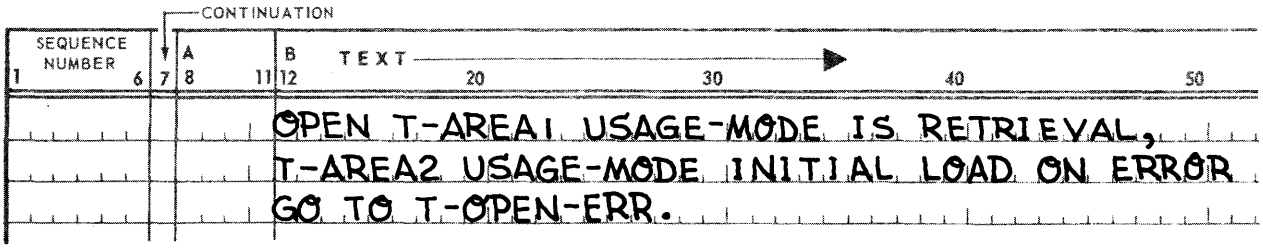
  000000 successful completion

  000934 area is not assigned

  000928 attempt to OPEN an area already open

  000929 Entry in ILLPR table indicating OPEN for INITIAL LOAD not completed, but doing another OPEN (*other than INITIAL LOAD*).

Examples and Discussion:



Assuming that both mass storage files, T-AREA1 and T-AREA2 have been properly assigned in the run stream, area T-AREA1 is OPENED for retrieval of records; area T-AREA2 is OPENED for update. In addition, all the pages in T-AREA2 are constructed and initialized with a page header if not specified as pre-initialized.

If either area is not assigned, then neither area is opened and control is returned to T-OPEN-ERR with ERROR STATUS of 000934.

3.4.16. REMOVE Command

Function:

To cancel the membership of the object record in the occurrences of the specified set-names in which it currently participates as a member, provided that the object record is defined in the schema as a manual member of the sets named.

Syntax Skeleton:

Format–1

REMOVE **FROM** *set-name–1* [,*set-name–2*] ... [ON ERROR GO **TO** *error-paragraph*] .

Format–2

REMOVE **FROM** ALL **SETS** [ON ERROR GO **TO** *error-paragraph*] .

Syntactical Constraints:

- Set-name–1, set-name–2... must be the names of sets defined in the schema named by the INVOKE clause of the Schema Section.
- No more than 36 set-names may be specified in a single REMOVE command.
- Error-paragraph must be a Paragraph Name or Section Name, defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50–99*).

Data Manipulation Guides:

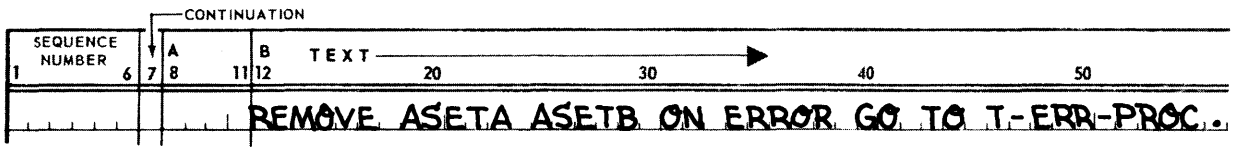
- The object record occurrence of the REMOVE command is the current record of the run unit. If no current record of the run unit exists, then Error Status Condition 1113 results.
- If *Format–1* of the REMOVE command is used, then the object record must have been defined in the schema as a manual member of each set named, or Error Status Condition 1114 results. If the object record occurrence does not currently participate in each set named, then Error Status Condition 1113 results.
- If *Format–2* of the REMOVE command is used, then the object record occurrence is removed from the appropriate occurrence of each set in which it is defined as a manual member in the schema and is currently participating. If the object record occurrence does not currently participate as a member in any set in which it is defined as a manual member, then the removal from that set is considered complete. The removal operation advances to the next manual set and an Error Status Condition results.
- If the object record occurrence is the current record of set-name for any sets, from which the object record is removed, then the current record of set-name becomes null for those sets, otherwise, the current of set-name is not changed. In the event the current record of set name becomes null, it is no longer accessible as current, however, the next, prior, and owner of set-name remain accessible.
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph was specified in the Schema Section, in which case control is returned to it.
- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

  000000 successful completion

  001113 no current record of the run unit or the current record of the run unit does not currently participate in the named set

  001114 object record is not a manual member of the named set

Examples and Discussion:



The current record of run-unit must be participating as a manual member of ASETA and ASETB. If it is not, control is returned to T-ERR-PROC with a non-zero ERROR-STATUS.

Otherwise the current record is removed from ASETA and ASETB – that is, it no longer participates as a member of these SETs.

3.4.17. STORE Command

Function:

- To acquire space and a database-key for a new record occurrence in the database.
- To cause the values of the appropriate data-items in the record-delivery-area to be included in the occurrence of the object record in the database.
- To insert the object record into all sets for which it is defined as an automatic member in the schema.
- To establish a new set occurrence for each set, for which the object record is defined as owner in the schema.
- To establish the object record as follows:

  (1) The current record of the run unit.

  (2) The current record of the area in which it is stored.

  (3) The current record of record-name.

  (4) The current record of set for all set-names in which it is specified as owner or automatic member.

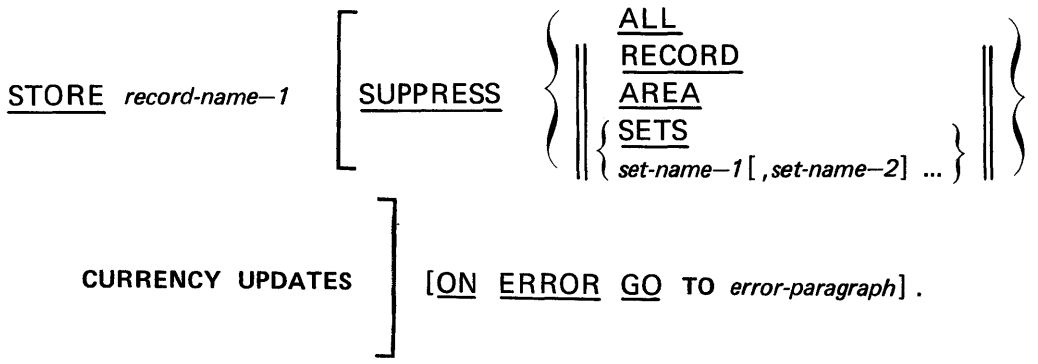
- To optionally suppress the establishment of the object record as follows:

  (1) The current record of the area in which it is stored.

  (2) The current record of record-name.

  (3) The current record of set for all, or specified sets, in which it is described as owner or automatic member.

Syntax Skeleton:



Syntactical Constraints:

- Record-name–1 must be the name of a record included in the schema named on the INVOKE clause in the Schema Section. If the COPYING option was used on the INVOKE clause, record-name–1 must have been included among the records copied.

- Set-name–1, set-name–2... must be the names of sets included in the schema named on the INVOKE clause in the Schema Section.

- No more than 63 set-names may be specified in a single STORE command.

- Error-paragraph must be a Paragraph Name or Section Name, defined in the non-declarative portion of the Procedure Division. It may not be defined in an independent COBOL SEGMENT (*i.e., SECTION numbers 50–99*).

Data Manipulation Guides:

- Prior to the execution of the STORE command, the record-delivery-area must be initialized with the data-items of the record to be stored on the data-base.

- If the LOCATION MODE of the record is DIRECT:

  (1) The 77-level entry data-base-data-name–2, specified on the LOCATION MODE clause must be initialized with the area-name of the area in which the object record is to be stored. If the area-name is not on the list of area-names specified on the WITHIN clause of the record, then Error Status Condition 1208 results. If the area is not OPEN, or is OPEN with a USAGE-MODE of RETRIEVAL, then Error Status Condition 1201 results.

  (2) The 01-level entry data-base-data-name–1, specified on the LOCATION MODE clause, must be initialized with a page number in data item PAGE-NUM and a record number in data item RECORD-NUM. If the page number is beyond the allocated number of pages for the area, then Error Status Condition 1202 results. If the page number is outside the page range allocated for records of this type on the WITHIN clause, then Error Status Condition 1287 results. If the record number on the page is already in use, then Error Status Condition 1212 results.

- If the LOCATION MODE of the record is CALC:

  (1) Data-base-procedure–1, specified on the LOCATION MODE clause, must have been included in the absolute element containing the run unit through the use of the @MAP Processor. (See Appendix D.)

(2) The 77-level entry data-base-data-name–3, specified on the LOCATION MODE clause, must be initialized with the area-name of the area in which the object record is to be stored. This initialization may be accomplished either by the run unit or data-base-procedure–1. If the area-name is not on the list of area-names specified on the WITHIN clause of the record, then Error Status Condition 1208 results. If the area is not OPEN, or is OPEN with a USAGE-MODE of RETRIEVAL, then Error Status Condition 1201 results.

(3) The input data-items to data-base-procedure–1; that is, data-base-identifier–1, data-base-identifier–2..., specified on the LOCATION MODE clause, must be initialized with the proper argument values. If the page number developed by the data-base-procedure is a page number beyond the allocated number of pages for the area, then Error Status Condition 1202 results. If the page number is a page number outside the page range, allocated for records of this type on the WITHIN clause, then Error Status Condition 1287 occurs. If storage of the record would violate the DUPLICATES ARE NOT ALLOWED clause, then Error Status Condition 1205 results.

- If the LOCATION MODE of the record is VIA set-name–1 SET:

(1) The area, page number, and record number are determined by the DMR. This determination is based upon the record placement strategy defined by the Data Administrator. See the *UNIVAC 1100 Series Data Management System (DMS 1100) Schema Definition Data Administrator Reference, UP-7907* (current version), for complete details. If the area is not OPEN, or is OPEN with a USAGE-MODE of RETRIEVAL, then Error Status Condition 1201 results.

(2) The placement of the object record occurrence is based upon its participation in set-name–1. The selection of the owner of the proper occurrence of set-name–1 is based upon the SET OCCURRENCE SELECTION clause for record-name–1 as a member of set-name–1. The selection of the logical insert point for the object record occurrence within the selected set-name–1 set occurrence is based upon the ORDER clause of set-name–1. If the object record violates a DUPLICATES NOT ALLOWED clause, then Error Status Condition 1205 results.

The selection of the proper occurrence of set-name–1 may or may not involve additional sets at higher levels of a hierarchy. While the Application Programmer is not concerned with the structure of the database, he must be aware of and initialize whatever key data items are required. The Data Administrator is aware of and must furnish the Application Programmer with the names and values of the key data items requiring initialization. See the *UNIVAC 1100 Series Data Management System (DMS 1100) Schema Definition Data Administrator Reference, UP-7907* (current version), for complete details.

- The object record is established as the owner of a set occurrence for each set in which it has been defined as an owner. These set occurrences are empty at this time, that is, no member records exist.

- The object record occurrence is inserted into a set occurrence for each set in which it is defined as an automatic member. Data Manipulation (VIA set-name–1 SET) regarding the selection of the object set occurrence, the selection of the logical insert point within the object set occurrence, and if necessary, the selection of additional set occurrences at higher levels in a hierarchy applies for each set-name in which the set is defined as an automatic member. If no set occurrence of the set-name meets the SET OCCURRENCE SELECTION criteria specified, then Error Status Condition 1213 results.

- The successfully stored record occurrence becomes the current record of the run-unit and:

  (1) Its area-name is placed in the DMCA reserved location AREA-NAME.

  (2) Its page number and record number are placed in PAGE-NUM and RECORD-NUM of the DMCA reserved location AREA-KEY.

  (3) Its record-name is placed in the DMCA reserved location RECORD-NAME.

- In the absence of the optional SUPPRESS clause, the object record also becomes:

  (1) The current record of its area-name.

  (2) The current record of its record-name.

  (3) The current record of all set-names in which it is defined as an owner or automatic member.

- The SUPPRESS clause provides the selective facility to prevent the object record from becoming:

  (1) The current record of the area.

  (2) The current record of its record-name.

  (3) The current record of any or all set-names in which it is defined as either an owner or an automatic member.

  When the SUPPRESS clause is used, the specified currency indicators are not affected by the execution of the STORE command. The SUPPRESS ALL clause prevents the update of all currency indicators described in this rule except the current of run-unit.

- Use of the SUPPRESS clause does not prevent the object record from becoming the current record of the run unit. Nor does it prohibit the updating of the DMCA reserved locations AREA-NAME, AREA-KEY, and RECORD-NAME.

- If space is not available for storage of the object record, then Error Status Condition 1211 results.

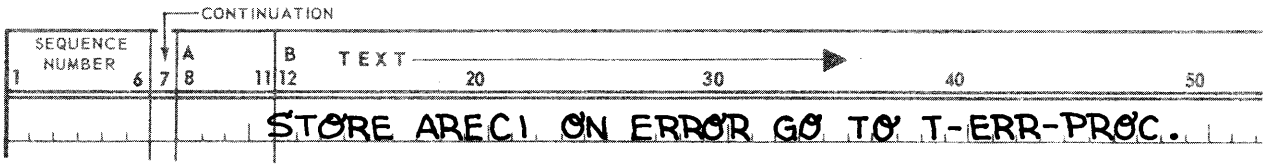
- In the event of an Error Status Condition, if the optional ON ERROR clause is specified, then control is returned to the specified error-paragraph. If the optional ON ERROR clause is not specified, control is returned to the next sentence, unless a general-error-paragraph is specified in the Schema Section, in which case control is returned to it.

- Summary of Error Status Conditions (see Section 4, ERROR RECOVERY TECHNIQUES, for further details).

000000	successful completion
001201	area not open
001202	page number outside allocated pages
001205	violation of DUPLICATES NOT ALLOWED clause
001208	invalid area-name
001211	space not available
001212	database-key not available
001213	no set occurrence meets selection criteria

Examples and Discussion:

- **Example 1:**



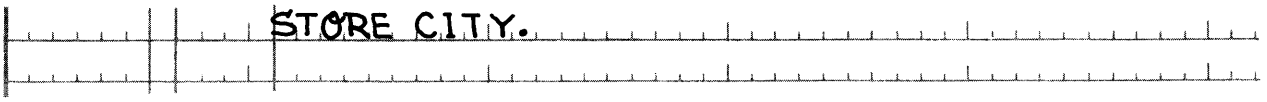
The record AREC1 must be in the record-delivery-area prior to the execution of the command. In addition, the proper keys must be initialized. The keys depend on the LOCATION MODE of the record AREC1.

For example, assume the LOCATION MODE is DIRECT with AREC1-AK as the area-key and AREC1-AN as the area-name. Then, prior to the execution of the STORE command, AREC1-AN must contain the area-name where the record is to be stored, and AREC1-AK must contain the area-key.

- **Example 2:**

Let us assume the LOCATION MODE of the record CITY is CALC, and the procedure to be used is CALC-PROC-1. The area in which the record is to be stored is contained in CALC-REC-AN. The programmer must ensure that the record CITY is in the record-delivery-area. CALC-REC-AN must be initialized with the area-name where the record is to be stored, unless this step is performed by the CALC procedure. The record becomes current record of run-unit on a successful completion of the command.

- **Example 3:**



Assume the portion of the data-base with which the programmer is concerned is organized as described in Section 3.4.5 (FIND/FETCH Command, Examples and Discussion, FIND CITY RECORD).

Let us assume further that the programmer wishes to store a record for Los Angeles, California, USA. He must first initialize COUNTRY-AN, COUNTRY-AK (with the area-name and area-key of the COUNTRY record of USA), ST-NAME (with 'California'), and CITY-NAME (with 'Los Angeles'). On successful completion of the command:

The CITY record for Los Angeles is the:

(1) Current record of run-unit.

(2) Current record of the set CITIES.

(3) Current record of type CITY.

(4) Current record of the area in which it is stored.

4. ERROR RECOVERY TECHNIQUES

4.1. GENERAL

The DMS 1100 contains four methods of returning control to the run unit when it detects an error in the user program. The user must specify:

- A rollback-error-paragraph,

and may specify:

- A general-error-paragraph for the run unit.
- A more specialized error-paragraph for specific instances; that is, at the command level.
- The program to continue.

In all cases, the DMR provides the run unit with a code for the type of error and names the concerned area, record or set where applicable.

Note that error processing centers about the Error Status Condition returned to the run unit in the reserved DMCA location ERROR-STATUS. This Error Status Condition is always returned as an error code to be processed by the run-unit. Frequently, an Error Status Condition is expected and is part of the normal logic for the run unit.

4.2. ERROR NOTIFICATION

The DML Preprocessor generates the following items as part of the DMCA in the user Working Storage Section.

These items are the vehicles used by the DMR to communicate information about a detected error to the run unit. Normally, the item ERROR-STATUS is set to 0. When an error is detected, the ERROR-STATUS is initialized with a code describing the error. A list of these codes is contained in Section 4.7. If the error pertains to a record, set, or area; the appropriate names are passed to the run unit via ERROR-RECORD, ERROR-SET, or ERROR-AREA, respectively.

The DMR will then pass control to a paragraph specified by the user for processing, or if none is specified, to the next Procedure Division statement.

In the case of an RB-ERROR-CODE, return is made to the rollback-error-paragraph.

Example:

CONTINUATION
SEQUENCE NUMBER 1–6 | 7 | A 8–11 | B 12 TEXT → 20 30 40 50

```
       01  DMCA.
            :
           02  RESERVED-WORD-AREA.
            :
               03  ERROR-AREA           PIC X(30).
               03  ERROR-RECORD         PIC X(30).
               03  ERROR-SET            PIC X(30).
               03  ERROR-STATUS
                   04  RB-ERROR-CODE    PIC X(2).
                   04  ERROR-FUNCTION   PIC X(2).
                   04  ERROR-CODE       PIC X(2).
```

4.3. COMPREHENSIVE ERROR HANDLING

The run unit may establish a general return point to be taken, if an error is encountered during the execution of any DML command. To accomplish this, the user specifies a general-error-paragraph on the ERROR RECOVERY IS clause of the Schema Section. The DMR will return control to this paragraph when ERROR-STATUS is non-zero, unless it is over-ridden by a specific paragraph through the ON ERROR clause (as described later), or if RB-ERROR-CODE is non-zero in which case return is made to the rollback-error-paragraph.

4.4. SPECIFIC ERROR HANDLING

Each DML PROCEDURE DIVISION command except IF may include the clause:

ON ERROR GO TO *paragraph-name*

Where:

Paragraph-name is a Paragraph Name or Section Name.

If an error is detected during execution of the command, the DMR will return control to paragraph-name. Use of this optional statement overrides the paragraph named on the ERROR RECOVERY clause of the Schema Section.

The DML Preprocessor generates a GO TO statement following any DML command. The paragraph is the appropriate error-paragraph. If this statement is executed and is appropriate, the information described in 4.2

will be initialized. Note however, that this paragraph is not a closed subroutine and the run-unit must direct control at the end.

One additional specific form of error handling is used in conjunction with *Formats–3* and *–4* of the FIND/FETCH command. This command may include the option AT END GO TO end-paragraph. During execution of the command, if an Error Status Condition results which reflects an End of Area or Set Condition (0307), then control is returned at the end-paragraph specified on the AT END clause (*if present*). If the AT END clause is not present, this condition is treated identically to any other Error Status Condition as previously described.

4.5. DEFAULT ERROR HANDLING

If neither an ERROR Recovery paragraph or an ON ERROR clause is specified when a non-run-unit rollback error is detected, the ERROR-STATUS is initialized and control returned to the next statement in the program. The user may then test procedurally for the error condition. In all cases, when a run-unit rollback error situation arises, return is made to the rollback-error-paragraph with RB-ERROR-CODE containing the reason for the rollback.

4.6. EXAMPLE

Assume that the user has defined an area named HISTORY with 20 pages. He has defined a record MEDICAL whose location mode is direct, using AKEY, ANAME.

Example:

```
SEQUENCE         CONTINUATION
NUMBER      A    B   TEXT
1      6  7 8  11 12       20         30         40         50

                 :
                 ERROR IS ERROR-RECOVERY.
                 :
                 MOVE 'HISTORY' TO ANAME.
                 MOVE 30 TO PAGE-NUM OF AKEY.
                 MOVE 1 TO RECORD-NUM OF AKEY.
                 MOVE MEDICAL TO RDA.
                 STORE MEDICAL.
            BETA.
                 MOVE 31 TO PAGE-NUM OF AKEY.
                 MOVE 5 TO RECORD-NUM OF AKEY.
                 MOVE MEDICAL TO RDA.
                 STORE MEDICAL ON ERROR GO TO OOPS.
            ALPHA.
                 :
```

```
ERROR-RECOVERY.
    MOVE ERROR-STATUS TO PRINT-LINE.
    WRITE PRINT-LINE.
    MOVE ERROR-RECORD TO PRINT-LINE.
    WRITE PRINT-LINE.
    GO TO BETA.
OOPS.
    IF ERROR STATUS EQUALS 1235 GO TO ALPHA
    ELSE STOP RUN.
```

The program would generate the output:

001235 MEDICAL	from first STORE

execution would continue from ALPHA, after execution of second STORE, after going to on-error-paragraph OOPS.

4.7. DMR EXTERNAL ERROR CODES

NN below represents a code for the particular DML command involved and has values:

01	CLOSE
02	DELETE
03	FIND/FETCH
04	FREE
05	GET
06	KEEP
07	INSERT
08	MODIFY
09	OPEN
11	REMOVE
12	STORE
13	DEPART
14	IF
15	IMPART
16	MOVE
17	LOG

4.7.1. DMR Error Codes

NOTE:

**Marks entries for which an AREA, RECORD or SET name is returned to the run unit.*

Error Code		Error Condition and Correction
NN01	*	AREA NOT OPEN OR INCORRECT USAGE MODE.
		The area-name which the run unit has supplied to the DMR denotes an area not yet opened by the run unit, or an update was attempted in an area opened for RETRIEVAL.
NN01		IF CONDITION LEGITIMATELY EVALUATED AS FALSE.
		The condition statement specified on an IF command was meaningful in the database and evaluated as false. This is not an error diagnostic so much as a switch condition for the run unit.
NN02	*	AREA-KEY VIOLATES WITHIN CLAUSE.
		The page number or record number specified in a database-key exceeds the possible range of pages or records specified on the WITHIN clause.
NN02		IF CONDITION ERROR. RECORD NOT DEFINED IN SCHEMA AS OWNER/MEMBER OF SET.
		The condition statement specified on an IF command was invalid, since the record specified was not defined in the schema as either the owner or a member of the set.
NN03		OCCURS DEPENDING ITEM COUNT IS OUT OF RANGE.
		On the OCCURS DEPENDING clause for the ERROR RECORD, the value specified in data-base-identifier-4 is less than integer–7 or greater than integer–8.
NN04		MAXIMUM OF 10 CALC KEYS EXCEEDED.
		The total number of elementary item CALC keys specified on a LOCATION MODE IS CALC clause must not exceed 10.
NN05	*	VIOLATION OF DUPLICATES NOT ALLOWED CLAUSE.
		All calc-keys for a record being stored CALC match those of a record already stored on the chain or an ASCENDING/DESCENDING key matches the key for a record already stored. This contradicts the DDL clause that duplicates should not be allowed.
NN06	*	NO CURRENT RECORD OF AREA/RECORD/SET NAME.
		No current record of AREA/RECORD/SET name was set up by doing a FIND or STORE in a previous command.
NN07	*	END OF SET/AREA.
		The DMR encountered the end of a set or an area while tracing a path to find a record occurrence.

Error Code		Error Condition and Correction
NN08	*	RECORD NOT DEFINED IN SET/AREA. The record type specified or implied to be an owner or member of the set or within an area is not defined as such in the schema.
NN09	*	IMPROPER AREA USAGE. Usage of area violates way in which area has been OPENed.
NN10		INSUFFICIENT SPACE AVAILABLE IN D$WORK. D$WORK created by DMLP is not large enough to hold all invoked area, records, and sets. A common error is that the wrong D$WORK has been collected in the MAP with the R.U., i.e., not the one that was created by the DMLP compilation of the Run Unit.
NN11		CONFLICT IN SCHEMA USAGE. The run unit is IMPARTing a SCHEMA from an EXEC file while another run unit is concurrently using the same schema-name from a TIP file (or vice versa). All run units using a TIP file schema must be created by a DMLP compilation after the TIP FILE CODE is –– clause has been added to the SCHEMA – so that all RU's will use the TIP file SCHEMA at program execution time.
NN12		UNASSIGNED FILE FOR SCHEMA. The Executive file containing the SCHEMA is not assigned or a search of TIP's file directory failed to find the TIP file number assigned for the SCHEMA. The TIP file must be assigned via a TIP system's generation or TIP control stream statement.
NN13	*	NO RECORD OR SET OCCURRENCE SATISFIES CRITERIA. Using the criteria specified or implied by the run unit, the DMR found no appropriate record or set occurrence.
N13		NO CURRENT RECORD OF RUN UNIT. The run unit has issued a command which expects a record to have been established as current of run-unit. However, no record is current of run-unit.
NN14	*	RECORD OCCURRENCE DOES NOT CURRENTLY PARTICIPATE AS MANUAL MEMBER OF SET. The run unit attempted to remove a record occurrence from a set, but the record occurrence was not found.

Error Code		Error Condition and Correction
NN15	*	DELETE WITH ONLY OPTION ATTEMPTED ON OWNER OF NON-EMPTY SET OCCURRENCE. The run unit attempted to delete a record occurrence, which is the owner of a non-empty set using the ONLY option. This is not allowed.
NN16	*	RECORD OCCURRENCE CURRENTLY PARTICIPATES IN SET. The run unit attempted to insert a record occurrence into the database. The record occurrence is already an owner or member of the set.
NN19	*	RECORD NOT DEFINED IN SCHEMA AS MANUAL MEMBER OF THE SET NAMED OR ANY SET. The run unit tried to insert or remove a record occurrence, although the record occurrence was not specified in the schema as a manual member of the set in question.
NN21		RUN UNIT HAS NOT DONE AN IMPART. The run unit issued a procedural command before first issuing an impart command, or after a DEPART, the run unit issued a procedural command which was not an IMPART.
NN22		NEW RUN UNIT ID DUPLICATES ID FOR ACTIVE RUN UNIT. The run unit-identifier specified in the SCHEMA SECTION duplicates that of an active run unit.
NN26		INSUFFICIENT SPACE IN THE AREA/RECORD/SET REFERENCE LIST TO ALLOW ALL AREAS, RECORDS, OR SETS TO BE IMPARTED. The maximum number of areas, records, or sets for one of the respective reference lists has been reached. This maximum is the system's generation parameter ARLLEN, RRLLEN, or SRLLEN respectively. To change see *UNIVAC 1100 Series Data Management System (DMS 1100) System Support Functions Reference, UP-7909* (current version), Section 3.
NN28		AREA ALREADY OPENED BY THIS RUN UNIT.
NN29		AREA OPEN (NOT FOR INITIAL LOAD) ATTEMPTED ON PARTIALLY INITIALIZED AREA. Area must be reopened for INITIAL LOAD.
NN30		NO SPACE IN INITIAL LOAD PAGE RECOVERY TABLE FOR NEW ENTRY. The system's generation parameter NUMBIL (maximum number of areas being initially loaded concurrently) is not large enough.
NN31		INSUFFICIENT ROLLBACK BUFFER SPACE FOR PAGES IN AREA. The rollback buffer size is not large enough to allow rollback for pages in the area. The area may be opened for retrieval but cannot be opened for update until the system's generation parameter QLSIZE is enlarged. To change see *UNIVAC 1100 Series Data Management System (DMS 1100) System Support Functions Reference, UP-7909* (current version), Section 3.

Error Code		Error Condition and Correction
NN32		ENTRY POINT TO CALC PROCEDURE IS INCORRECT. The entry point to the calc procedure found in the segment load table is not within the bounds of the user D-Bank.
NN33		REQUEST FOR TABLE OR PAGE SPACE LARGER THAN DMR SYSTEM BUFFER. The run unit needed more space for a table or page than was in the DMR system buffer.
NN34	*	AREA NOT ASSIGNED. The run unit referenced an area which has not been assigned via an Executive ASG control card.
NN35	*	PAGE OR RECORD NUMBER ZERO OR EXCEEDS MAXIMUM ALLOCATED FOR AREA, WHERE LOCATION MODE IS DIRECT. The page or record number specified in an area-key by the run unit is either zero or greater than the maximum allocated in the schema.
NN36		PAGE OR RECORD NUMBER ZERO OR EXCEEDS MAXIMUM ALLOCATED FOR AREA WHERE LOCATION MODE IS CALC. The page number returned by a CALC Procedure is greater than the maximum pages allocated or is zero.
NN37		DATA NAME FOR UPPER OR LOWER PAGE BOUND OUTSIDE PAGE LIMITS OF AREA. Page boundaries specified in the data-base-data-name-4 or data-base-name-5 of the WITHIN clause contradict page limits specified on the ALLOCATE clause.
NN38		SECOND IMPART MADE WITHOUT DOING A DEPART. The run unit must issue a DEPART to release altered pages, system tables, etc., before a second IMPART can be issued.
NN42	*	TOO FEW POINTERS RESERVED IN RECORD DESCRIPTION. The run unit attempted to insert a record occurrence into a set. However, no unused pointers in the record description existed.
NN51		ITEM NAMES ON A MODIFY COMMAND REFER TO CURRENT RECORD OF THE RUN UNIT. The item names should refer to an owner record(s) of a set(s) in which the current record of the run participates as a member.

Error Code	Error Condition and Correction
NN52 *	AREA/RECORD/SET NOT INVOKED BY RUN UNIT. Needed description table could not be found. This could reflect an internal error.
NN53	INSUFFICIENT SPACE IN PAGE USAGE TABLE TO ALLOW ANOTHER PAGE. The DMR has room in the buffer to allow loading another page, but to do so would extend the page usage table beyond its maximum. This maximum is a system generation parameter, BUTLEN. To change, see *UNIVAC 1100 Series Data Management System (DMS 1100) System Support Functions Programmer Reference, UP-7909* (current version), Section 3.
NN57	DATABASE-DATANAME SPECIFIED IN SCHEMA NOT PRESENT IN RUN UNIT. The schema implies that the run unit will define 77-level or 01-level entries in working storage; for example, datanames for area-names or area-keys. However, the run unit did not define them.
NN63	BAD BLOCK TYPE HEADER FOR LOG COMMAND. The first character of the block to be transferred from the RDA to audit trail contains a 'U'.
NN64	NUMBER OF WORDS TO WRITE IS GREATER THAN RDA LENGTH. The Run Unit request packet indicates that the number of words to write on a LOG command is greater than the RDA length. This is not allowed.
NN65	ATTEMPTING LOG COMMAND WHEN DMR CONFIGURED WITHOUT AN AUDIT-TRAIL. A LOG command is not allowed if the DMR is configured without an AUDIT-TRAIL. See the appropriate DMS Release Bulletin for a complete description of the MDS system's generation parameters.
NN69	SCHEMA-NAME IN SCHEMA IS NOT THE SAME AS SCHEMA-NAME SPECIFIED ON DML INVOKE CLAUSE. The file containing the absolute schema (specified in the DML INVOKE clause) must be the same as file specified in the SCHEMA clause of the Data Definition Language.
NN80	RECOVERABLE I/O ERROR. A user caused I/O error was received and only the affected Run Unit need be rolled back. The Executive I/O error status is printed in octal on the machine console and at the end of the Run Unit's print file along with Initiation and Termination information.

Error Code		Error Condition and Correction
NN81		DATABASE-KEY SPECIFIED ON DIRECT STORE ALREADY IN USE. The run unit attempted to store a record occurrence whose location mode was direct, but the area-name and area-key it supplied generated a DATABASE-KEY already in use.
NN84		OVERFLOW PAGES ASSOCIATED WITH DATA PAGES ARE FULL. All overflow pages, as well as the prime data pages, contain too little unused space to allow storing another record occurrence. On initial load, the overflow pages are not used except for the calc chain.
NN85	*	RECORD WAS INVOKED BUT NOT COPIED. NO RECORD DESCRIPTION AVAILABLE. The record type in question is defined in the Schema, but this definition was not copied under the present impart.
NN87	*	AREA-NAME VIOLATES WITHIN CLAUSE. In a direct store, the run unit specified an area name which contradicts the WITHIN clause.
NN88		CALC PROCEDURE NOT IN CLSEG$ TABLE. The externalized label which serves as an entry point to the CALC Procedure was absent from the user generated Entry Point Table when required. (See Appendix D.)
NN89	*	RECORD OCCURRENCE FOUND USING DIRECT LOCATION MODE IS NOT OF EXPECTED TYPE. A record occurrence was found using the AREA NAME/KEY supplied by the run unit. It was not an occurrence of the type expected.
NN92		DATABASE-KEY FOUND VACANCY ENTRY. Using the DATABASE-KEY supplied by the run unit, the DMR found a vacancy entry (*i.e., the record was DELETED*).
NN96		NO SPACE ON PAGE. There is insufficient space remaining on a page to store or insert another record occurrence. No overflow pages were specified.
NN97		RECORD NUMBER SPECIFIED NOT AVAILABLE ON PAGE. The record number specified in a DATABASE-KEY exceeds the largest number representable in the field size allocated for the record number.

Error Code	Error Condition and Correction
NN99	SCHEMA FILE NOT FOUND. The Executive file containing the SCHEMA cannot be found.

4.8. SUMMARY BY COMMAND OF MOST PROBABLE ERRORS

	CLOSE 01	DELETE 02	FIND/ FETCH 03	FREE 04	GET 05	KEEP 06	INSERT 07	MODIFY 08	OPEN 09	REMOVE 11	STORE 12	DEPART 13	IF 14	IMPART 15	MOVE 16	LOG 17
AREA NOT OPEN OR INCORRECT USAGE MODE	01	01	01	–	–	–	01	01	–	01	01	–	–	–	–	–
IF CONDITION LEGITIMATELY EVALUATED AS FALSE	–	–	–	–	–	–	–	–	–	–	–	–	01	–	–	–
AREA-KEY VIOLATES WITHIN CLAUSE	–	–	–	–	–	–	–	02	–	–	02	–	–	–	–	–
IF CONDITION ERROR. RECORD NOT DEFINED IN SCHEMA AS OWNER/MEMBER OF SET	–	–	–	–	–	–	–	–	–	–	–	–	02	–	–	–
OCCURS DEPENDING ITEM COUNT IS OUT OF RANGE	–	–	03	–	03	–	–	03	–	–	03	–	–	–	–	–
MAXIMUM OF 10 CALC KEYS EXCEEDED	–	–	–	–	–	–	–	04	–	–	04	–	–	–	–	–
VIOLATION OF DUPLICATES NOT ALLOWED CLAUSE	–	–	–	–	–	–	05	05	–	–	05	–	–	–	–	–
NO CURRENT RECORD OF AREA/RECORD/SET NAME	–	–	06	–	–	–	06	06	–	06	06	–	06	–	06	–
END OF SET/AREA	–	–	07	–	–	–	–	07	–	–	07	–	–	–	–	–
RECORD NOT DEFINED IN SET/AREA	–	–	08	–	–	–	08	–	–	08	–	–	08	–	–	–
IMPROPER AREA USAGE	–	09	09	–	09	–	09	09	–	09	09	–	–	–	09	–
INSUFFICIENT SPACE AVAILABLE IN D$WORK	–	–	–	–	–	–	–	–	–	–	–	–	–	10	–	–
CONFLICT IN SCHEMA USAGE	–	–	–	–	–	–	–	–	–	–	–	–	–	11	–	–
UNASSIGNED FILE FOR SCHEMA	–	–	–	–	–	–	–	–	–	–	–	–	–	12	–	–
NO RECORD OR SET OCCURRENCE SATISFIES CRITERIA	–	–	13	–	–	–	–	13	–	13	13	–	13	–	–	–
NO CURRENT RECORD OF RUN UNIT	–	13	13	–	13	–	13	13	–	13	–	–	–	–	13	–
RECORD OCCURRENCE DOES NOT CURRENTLY PARTICIPATE AS MANUAL MEMBER OF SET	–	–	–	–	–	–	–	–	–	14	–	–	–	–	–	–
DELETE WITH ONLY OPTION ATTEMPTED ON OWNER OF NON-EMPTY SET OCCURRENCE	–	15	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RECORD OCCURRENCE CURRENTLY PARTICIPATES IN SET	–	–	–	–	–	–	16	–	–	–	–	–	–	–	–	–
RECORD NOT DEFINED IN SCHEMA AS MANUAL MEMBER OF THE SET NAMED OR ANY SET	–	–	–	–	–	–	19	–	–	19	–	–	–	–	–	–
RUN UNIT HAS NOT DONE AN IMPART	21	21	21	21	21	21	21	21	21	21	21	21	21	–	21	21
NEW RUN UNIT ID DUPLICATES ID FOR ACTIVE RUN UNIT	–	–	–	–	–	–	–	–	–	–	–	–	–	22	–	–
INSUFFICIENT SPACE IN AREA/REC/SET REFERENCE LIST TO ALLOW ALL AREA, RECORDS, OR SETS TO BE IMPARTED	–	–	–	–	–	–	–	–	–	–	–	–	–	26	–	–
AREA ALREADY OPENED BY THIS RUN UNIT	–	–	–	–	–	–	–	–	28	–	–	–	–	–	–	–

	CLOSE 01	DELETE 02	FIND/ FETCH 03	FREE 04	GET 05	KEEP 06	INSERT 07	MODIFY 08	OPEN 09	REMOVE 11	STORE 12	DEPART 13	IF 14	IMPART 15	MOVE 16	LOG 17
AREA OPEN (NOT INITIAL LOAD) ATTEMPTED ON PARTIALLY INITIALIZED AREA	–	–	–	–	–	–	–	–	29	–	–	–	–	–	–	–
NO SPACE IN INITIAL LOAD PAGE RECOVERY TABLE FOR NEW ENTRY	–	–	–	–	–	–	–	–	30	–	–	–	–	–	–	–
INSUFFICIENT ROLLBACK BUFFER SPACE FOR PAGES IN AREA	–	–	–	–	–	–	–	–	31	–	–	–	–	–	–	–
ENTRY POINT TO CALC PROCEDURE INCORRECT	–	–	32	–	–	–	–	32	–	–	32	–	–	–	–	–
REQUEST FOR TABLE OR PAGE SPACE LARGER THAN DMR SYSTEM BUFFER	–	33	33	–	33	–	33	33	–	33	33	–	33	–	–	33
AREA NOT ASSIGNED	–	–	–	–	–	–	–	–	34	–	–	–	–	–	–	–
PAGE OR RECORD NUMBER ZERO OR EXCEEDS MAXIMUM ALLOCATED FOR AREA, WHERE LOCATION MODE IS DIRECT	–	–	35	–	–	–	–	35	–	–	35	–	–	–	–	–
PAGE OR RECORD NUMBER ZERO OR EXCEEDS MAXIMUM ALLOCATED FOR AREA WHERE LOCATION MODE IS CALC	–	–	36	–	–	–	–	36	–	–	36	–	–	–	–	–
DATA NAME FOR UPPER OR LOWER PAGE BOUND OUTSIDE PAGE LIMITS OF AREA	–	–	–	–	–	–	–	37	–	–	37	–	–	–	–	–
SECOND IMPART MADE WITHOUT DOING A DEPART	–	–	–	–	–	–	–	–	–	–	–	–	–	38	–	–
TOO FEW POINTERS RESERVED IN RECORD DESCRIPTION	–	–	–	–	–	–	42	–	–	–	–	–	–	–	–	–
ITEM NAMES ON A MODIFY COMMAND REFER TO CURRENT RECORD OF THE RUN UNIT	–	–	–	–	–	–	51	51	–	–	–	–	–	–	–	–
AREA/RECORD/SET NOT INVOKED BY RUN UNIT	52	52	52	–	52	52	52	52	52	52	52	52	52	52	52	–
INSUFFICIENT SPACE IN PAGE USAGE TABLE TO ALLOW ANOTHER PAGE	–	53	53	–	–	–	53	53	–	53	53	–	–	–	–	–
DATABASE-DATANAME SPECIFIED IN SCHEMA NOT PRESENT IN RUN UNIT	–	–	57	–	–	–	–	57	–	–	57	–	–	–	–	–
BAD BLOCK TYPE HEADER FOR LOG COMMAND	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	63
NUMBER OF WORDS TO WRITE GREATER THAN RDA LENGTH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	64
ATTEMPTING LOG COMMAND WHEN DMR CONFIGURED WITHOUT AUDIT TRAIL	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	65
SCHEMA-NAME IN SCHEMA IS NOT THE SAME AS SCHEMA-NAME SPECIFIED ON DML INVOKE CLAUSE	–	–	–	–	–	–	–	–	–	–	–	–	–	69	–	–

	CLOSE 01	DELETE 02	FIND/ FETCH 03	FREE 04	GET 05	KEEP 06	INSERT 07	MODIFY 08	OPEN 09	REMOVE 11	STORE 12	DEPART 13	IF 14	IMPART 15	MOVE 16	LOG 17
RECOVERABLE I/O ERROR	80	80	80	–	80	–	80	80	80	80	80	80	–	80	–	80
DATABASE-KEY SPECIFIED ON DIRECT STORE ALREADY IN USE	–	–	–	–	–	–	–	–	–	–	81	–	–	–	–	–
OVERFLOW PAGES ASSOCIATED WITH DATA PAGES ARE FULL	–	–	–	–	–	–	–	84	–	–	84	–	–	–	–	–
RECORD WAS INVOKED BUT NOT COPIED. NO RECORD DESCRIPTION AVAILABLE	–	–	85	–	–	–	–	85	–	–	85	–	–	–	–	–
AREA-NAME VIOLATES WITHIN CLAUSE	–	–	–	–	–	–	–	–	–	–	87	–	–	–	–	–
CALC PROCEDURE NOT IN CLSEG$ TABLE	–	–	88	–	–	–	–	88	–	–	88	–	–	–	–	–
RECORD OCCURRENCE FOUND USING DIRECT LOCATION MODE IS NOT OF EXPECTED TYPE	–	–	89	–	–	–	–	–	–	–	–	–	–	–	–	–
DATABASE-KEY FOUND VACANCY ENTRY	–	–	92	–	–	–	–	–	–	–	–	–	–	–	–	–
NO SPACE ON PAGE	–	–	–	–	–	–	–	96	–	–	96	–	–	–	–	–
RECORD NUMBER SPECIFIED NOT AVAILABLE ON PAGE	–	–	–	–	–	–	–	–	–	–	97	–	–	–	–	–
SCHEMA FILE NOT FOUND	–	–	–	–	–	–	–	–	–	–	–	–	–	99	–	–

4.9. DMR ROLLBACK ERROR CODES

The following rollback error status codes are stored in RB-ERROR-CODE in the case of rollback errors:

Error Code	Reason
01	R.U. ROLLBACK BECAUSE OF USER ERROR AND COMMAND QUICK BEFORE-LOOKS NOT SPECIFIED IN SAVE DATA CLAUSE The run unit has altered the data base and received an error during a command. Since command rollback cannot be done run unit rollback must be executed.
02	DEADLOCK BETWEEN RUN UNITS. Two or more run units may be deadlocked on facilities which each of them hold. The most common deadlock situations are on AREAS, PAGES, BUFFER SPACE, or combinations of the three.
03	INTERNAL DMR ERROR. All internal error codes (see Appendix A) which cause an ABORT will receive this error code for the run unit which received the internal error and all other run units in the system at the time of the ABORT. Any new run units trying to come into the system during this ABORT and quick recovery will also receive an RB-ERROR-CODE 03. Also, any run unit attempting to enter DMR during a utility routine that locks out access to the DMR receives an RB-ERROR-CODE 03.
04	USER CONTINGENCY ERROR & NO USER CONTINGENCY ROUTINE. A contingency has occurred in the user program between the time of the IMPART & DEPART and the user has no contingency routine address for the DMR to give control. In this case the DMR executes an internal DEPART WITH ROLLBACK from the LINKER and returns control to the user at his rollback address.
05	LONG RECOVERY IN PROGRESS. The user has tried to enter the DMR while long recovery is in progress. This is not allowed.
06	NOT ENOUGH QUICK LOOK FILE SPACE ALLOCATED. The DMR has run out of quick look file space to store quick before looks and must rollback this run unit. (See System's Generation Parameters.)
07	QUICK LOOK FILE NOT ASSIGNED. The quick look file has not been assigned by an Executive control card.
08	CAME INTO TIMER WITHOUT TIMER-ID. The run unit came into the TIMER without the TIMER ID. This could be an internal error.

Error Code	Reason
09	ROLLBACK BECAUSE OF KEYIN FROM CONSOLE WHILE RUN UNIT IN DMR. If Keyin occurred while run unit in COBOL program: (a) If the user registered a contingency routine that routine will get control. (b) If no user contigency routine a DEPART WITH ROLLBACK is executed by the DMR from LINKER and control is returned to the COBOL program at the rollback address with rollback error code 04 (*USER CONTINGENCY ERROR*).
10	NOT ENOUGH SPACE IN AUDIT TRAIL WRITE TABLE. The audit trail write table is not large enough to allow writing to the AUDIT TRAIL. This may occur if more than two run units are issuing LOG RECOVERY POINT commands.
11	THIS RUN UNIT ROLLED BACK BECAUSE ANOTHER RUN UNIT RECEIVED AN I/O ERROR IN WRITING OUT A PAGE THIS RUN UNIT ALTERED. Most common reason is that run unit writing out page did not assign all files (*i.e., area containing page being written*).
12	ERROR IN STARTING TIMER USING CSF$ FOR @START DMS*DMRMT.TIMLT. Most common reason is that file DMS*DMRMT has not been catalogued and freed correctly.
13	QUICK RECOVERY IN PROGRESS. No run units are allowed to IMPART while QUICK RECOVERY is in progress. Run Units can be submitted upon completion of QUICK RECOVERY.
99	NOT ENOUGH SPACE IN D$WORK FOR GS09 SAVE REGISTER AREA.

APPENDIX A. DML RESERVED WORD LIST

A.1. GENERAL

The DMLP recognizes two types of reserved words:

- those which may be used only in a syntactical sense such as IDENTIFICATION, OPEN, MULTIPLY, and
- reserved data-names such as ERROR-AREA and PAGE-NUM.

Words reserved in the syntactical sense may be used only as specified in the syntactical skeletons of Section 3 of this document, or as specified in the *UNIVAC 1100 Series American National Standard COBOL (Fieldata) Programmer Reference, UP-7845* (current version).

Reserved data-names are data-names which are inherent in the structure of the DML language and need no description. When used, they must conform precisely to rules established for their use.

Words reserved in the syntactical sense are denoted by an 'S' in the following list. Reserved data-names are denoted by 'D'. A blank, '', means that the word may not be used in that section.

A.2. RESERVED WORD LIST

Reserved Word	ID-ENV Division	Data Division	Procedure Division
ACCEPT			S
ACCESS	S	S	
ACTUAL	S	S	
ADD			S
ADVANCING			S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
AFTER		S	S
ALL		S	S
ALPHABETIC		S	S
ALTER			S
ALTERNATE	S		
AN		S	
AND	S	S	S
ANY			S
APPLY	S		
ARE	S	S	
AREA	S	S	S
AREAS	S		
AREA-COUNT			D
AREA-KEY		S	D,S
AREA-NAME		S	D,S
ASCENDING		S	S
ASSIGN	S		
AT			S
AUTHOR	S		
BEFORE			S
BEGINNING			S
BLANK		S	
BLOCK	S	S	
BY	S	S	S
CALL			S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
CARD-PUNCH	S		
CARD-READER	S		S
CD		S	
CF		S	
CH		S	
CHANGED		S	
CHARACTER		S	S
CHARACTERS	S	S	
CLOSE			S
CLOSE-COUNT			D
CLUSTER-DUMPS	S	D	D
COBOL			S
COBOL-OPTIONS	S		
CODE		S	
COLUMN		S	
COMMA	S		
COMMAND		S	
COMMAND-STATISTICS			D
COMMON		S	
COMMON-STORAGE		S	
COMMUNICATION		S	
COMP		S	
COMPACT		S	
COMP-1		S	
COMP-2		S	

Reserved Word	ID-ENV Division	Data Division	Procedure Division
COMP-3		S	
COMP-4		S	
COMPUTATIONAL		S	
COMPUTATIONAL-1		S	
COMPUTATIONAL-2		S	
COMPUTATIONAL-3		S	
COMPUTATIONAL-4		S	
COMPUTE			S
CONDITION-WORD-S3	S		
CONDITION-WORD-S4	S		
CONFIGURATION	S	S	
CONSOLE	S		S
CONTAINS	S	S	
CONTROL		S	
CORE-ALLOCATION-COUNT			D
CORE-INDEX	S		
CORE-LOCK-COUNT			D
COPY	S	S	S
COPYING		S	
CORR			S
CORRESPONDING			S
COUNT		S	
CREATION-DATE		S	
CURRENCY	S		S
CURRENT			S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
CURRENT-AREA-KEY			D
CURRENT-AREA-NAME			D
CURRENT-LOCK-COUNT			D
DATA	S	S	S
DATE		S	
DATE-COMPILED	S		
DATE-WRITTEN	S		
DE		S	
DECIMAL-POINT	S		
DECLARATIVES		S	
DELETE			S
DELETE-COUNT			D
DELIMITED			S
DELIMITER			S
DELIVERY-AREA		S	
DEPART			S
DEPART-COUNT			D
DEPENDING		S	S
DEPTH		S	
DESCENDING		S	S
DESTINATION		S	D
DETAIL			S
DIRECT	S		
DISABLE		S	

Reserved Word	ID-ENV Division	Data Division	Procedure Division
DISPLAY		S	S
DISPLAY-1		S	
DISP		S	
DIVIDE			S
DIVISION	S	S	S
DMCA		S	D
DMR			
DMRA			
DOWN			S
DUPLICATE			S
ELSE			S
EMI			S
EMPTY			S
ENABLE			S
END	S	S	S
ENDING			S
ENTER			S
ENTRY			S
ENVIRONMENT	S	S	
EQUAL			S
EQUALS			S
ERROR		S	S
ERROR-AREA			D
ERROR-CODE			D
ERROR-FUNCTION			D

Reserved Word	ID-ENV Division	Data Division	Procedure Division
ERROR-RECORD			D
ERROR-SET			D
ERROR-STATUS			D
ESI			S
ETI			S
EVERY	S		
EXAMINE			S
EXCLUSIVE			S
EXDEF	S		
EXHIBIT			S
EXIT			S
EXREF	S		
FD	S	S	
FETCH			S
FILE		S	S
FILE-ACCESS		S	
FILE-CONTROL	S	S	
FILE-ID		S	
FILE-LIMIT	S		
FILE-LIMITS	S		
FILE-QUALIFIER		S	
FILLER		S	
FINAL		S	
FIND			S
FIND-COUNT			D

Reserved Word	ID-ENV Division	Data Division	Procedure Division
FIRST			S
FOOTING		S	
FOR	S	S	S
FORM-PRINTER	S		
FORTRAN			S
FREE			S
FREE-COUNT			D
FROM		S	S
GENERATE			S
GET			S
GET-COUNT			D
GIVING			S
GO			S
GOBACK			S
GREATER			S
GROUP		S	
HEADING		S	
HIGH-VALUE		S	S
HIGH-VALUES		S	S
IDENTIFICATION	S		
IF			S
IF-COUNT			D
IMPART			S
IMPART-COUNT			D
IN		S	

Reserved Word	ID-ENV Division	Data Division	Procedure Division
INCLUDES		S	
INDEX		S	
INDEXED	S	S	
INDICATE		S	
INITIAL			S
INITIATE			S
INPUT	S	S	S
INPUT-OUT	S	S	S
INSERT			S
INSERT-COUNT			D
INSPECT			S
INSTALLATION	S		
INTERLOCK	S		
INTERNAL		S	
INTO			S
INVALID			S
INVOKE		S	
IS	S	S	S
I-O	S	S	S
I-O-CONTROL	S	S	
I-O-COUNT			D
JUST		S	
JUSTIFIED		S	
KEEP			S
KEEP-COUNT			D

Reserved Word	ID-ENV Division	Data Division	Procedure Division
KEY	S	S	S
LABEL		S	S
LAST			S
LEADING			S
LEFT		S	
LENGTH		S	
LESS			S
LINKAGE		S	
LINE			S
LINES			S
LINES-AT-BOTTOM		S	
LINES-AT-TOP		S	
LINES-PER-PAGE		S	
LINE-SPACING		S	
LINKAGE		S	
LOAD			S
LOCK			S
LOG			S
LOG-COUNT			D
LOW-VALUE		S	S
LOW-VALUES		S	S
MASS-STORAGE	S		
MEMBER			S
MEMORY	S		
MESSAGE		S	S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
MODE	S	S	
MODIFY			S
MODIFY-COUNT			D
MODULES	S		
MONITOR			S
MOVE			S
MOVE-COUNT			D
MULTIPLE	S		
MULTIPLY			S
NAMED		S	
NEGATIVE			S
NEXT			S
NO	S	S	S
NOT			S
NOTE			S
NUMBER		S	
NUMERIC		S	S
OBJECT-COMPUTER	S	S	
OCCURS		S	
OF	S	S	S
OFF	S		
OMITTED		S	
ON	S	S	S
ONLY			S
OPEN		S	S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
OPEN-COUNT			D
OPTIONAL	S		
OR	S		S
ORGANIZATION	S		
OTHERWISE			S
OUTPUT	S	S	
OVERLAY		S	
OWNER			S
PAGE			S
PAGE-COUNT			D
PAGE-NUM			D
PERFORM			S
PF		S	
PH		S	
PIC		S	
PICTURE		S	
PLACE		S	
PLUS		S	
POINTER			S
POINTS	S		S
POSITION	S	S	
POSITIVE			S
PRINTER	S		S
PRINTING			S
PRIOR			S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
PRIORITY		S	
PROCEDURE	S	S	S
PROCEED			S
PROCESS			S
PROCESSING	S		
PROGRAM			S
PROGRAM-ID	S		
PROCTECT	S		
PROTECTED			S
PURGE-DATE		S	
QUEUE		S	
QUEUE-STATISTICS			D
QUICK-BEFORE-LOOKS		S	
QUOTE		S	S
QUOTES		S	S
RANDOM	S	S	S
RANK		S	
RD		S	
READ			S
RECEIVE			S
RECORD	S	S	S
RECORDING		S	
RECORDS	S	S	
RECORD-NAME			S
RECORD-NUM			D

Reserved Word	ID-ENV Division	Data Division	Procedure Division
RECOVERY		S	
REDEFINES		S	
REEL	S		S
REFERENCING			S
RELEASE			S
REMAINDER			S
REMARKS	S		
REMOVE			S
REMOVE-COUNT			D
RENAMES		S	
REPLACING	S	S	S
REPORT		S	
REPORTS		S	
RERUN	S		
RESERVE	S		
RESET			S
RESERVED-WORD-AREA			D
RESOLUTION		S	
RETRIEVAL			S
RETURN			S
REVERSED			S
REWIND			S
REWRITE			S
RF		S	
RH		S	

Reserved Word	ID-ENV Division	Data Division	Procedure Division
RIGHT		S	
ROLLBACK		S	S
ROLLBACK-BUFFER-COUNT			D
ROUNDED			S
RUN		S	S
RUN-UNIT		S	S
RUN-UNIT-ID		S	D
RUN-UNIT-STATISTICS			D
SAME	S		
SAVE		S	
SCHEMA		S	
SD		S	
SEARCH			S
SECTION	S	S	S
SECURITY	S		
SEEK			S
SEGMENT			S
SEGMENT-ID		S	D
SEGMENT-LIMIT	S		
SELECT	S		
SEND			S
SENTENCE			S
SEQUENTIAL	S	S	S
SET			S
SETS			S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
SET-ID		S	
SET-NAME			D
SIGN	S	S	
SIZE	S	S	S
SORT	S		S
SOURCE		S	
SOURCE-COMPUTER	S	S	
SPACE		S	S
SPACES		S	S
SPECIAL-NAMES	S	S	
STANDARD		S	S
STATUS	S	S	S
STOP			S
STORE			S
STORE-COUNT			S
STRING			S
SUBROUTINE			S
SUBSCHEMA		S	
SUBTRACT		S	
SUB-QUEUE-1		S	
SUB-QUEUE-2		S	
SUB-QUEUE-3		S	
SUPPRESS			S
SYMBOLIC	S	S	
SYNC		S	

Reserved Word	ID-ENV Division	Data Division	Procedure Division
SYNCHRONIZED		S	
TABLE		S	
TABLE-COUNT			D
TALLYING			S
TAPE	S	S	
TASK-ID		S	
TERMINAL			S
TERMINATE			S
TEXT		S	
THAN			S
THROUGH		S	S
THRU	S	S	S
TIME		S	S
TIMES		S	S
TO	S	S	S
TRAILING			S
TRANSFORM			S
TRL-COUNT			D
TYPE		S	
UNISERVO	S		
UNIT	S	S	S
UNIVAC-1108	S		
UNLOCK			S
UNSTRUNG			S
UNTIL			S

Reserved Word	ID-ENV Division	Data Division	Procedure Division
UP			S
UPDATE			S
UPDATES			S
UPON			S
USAGE		S	
USAGE-LOCK-COUNT			D
USAGE-MODE			S
USE			S
USING			S
VALUE		S	
VALUES		S	
VARYING			S
VIA			S
WHEN		S	S
WITH	S	S	S
WITHIN			S
WORD	S	S	
WORDS	S	S	S
WORKING		S	
WORKING-STORAGE		S	
WRITE			S
WRITE-ONLY		S	
ZERO		S	S
ZEROES		S	S
ZEROS		S	S

APPENDIX B. DML SYNTAX SKELETON

B.1. GENERAL

The following is the complete syntax skeleton of a DML program.

B.2. SYNTAX SKELETON:

```
IDENTIFICATION DIVISION.

    PROGRAM-ID. program-id-name.
    .
    .
    .
ENVIRONMENT DIVISION.
    .
    .
    .
DATA DIVISION.

SCHEMA SECTION.

INVOKE SCHEMA schema-name
    [ COPYING [ WORKING ]  { ALL                                          } ]
              [ COMMON  ]  { record-name–1 [ ,record-name–2] ... }
              [ LINKAGE ]
    [RUN-UNIT-ID IS run-unit-identification]
    [PRIORITY IS integer–2]
    RECORD DELIVERY-AREA IS record-delivery-area [ WORKING ]
                                                 [ COMMON  ]
                                                 [ LINKAGE ]
        [LENGTH IS integer–1 WORDS]
```

```
[ OVERLAY  { record-delivery-area }  WITH  { ALL                                        } ]
           { record-name–3         }       { record-name–4 [,record-name–5] ...      }

    [OVERLAY record-name–6 WITH record-name–7 [,record-name–8] ...] ...

[SAVE DATA INCLUDES { COMMAND  } QUICK-BEFORE-LOOKS]
                    { RUN-UNIT }

[DMCA [AND RUN-UNIT-STATISTICS] { IS  }  { WORKING }  ]
                                { ARE }  { COMMON  }
                                         { LINKAGE }

[ERROR RECOVERY IS general-error-paragraph]

ROLLBACK IS rollback-error-paragraph .

[FILE SECTION.]
      :
      :
[COMMUNICATION SECTION]
      :
      :
[WORKING-STORAGE SECTION.
```

77-level programmer entered descriptions, including descriptions for any data-base-data-names, defined in the SCHEMA, except those with a USAGE IS AREA-KEY.

01-level programmer entered descriptions for those data-base-data-names, defined in the schema and required by run unit, including those which have a USAGE IS AREA-KEY.

01-level programmer entered descriptions.]

```
[COMMON-STORAGE SECTION.
```

77-level programmer entered descriptions, including descriptions for any data-base-data-names, defined in the SCHEMA, except those with a USAGE IS AREA-KEY.

01-level programmer entered descriptions for those data-base-data-names, defined in the schema and required by the run unit, including those whach have a USAGE IS AREA-KEY.

01-level programmer entered descriptions.]

```
[LINKAGE SECTION]
      :
      :
[REPORT SECTION.]
PROCEDURE DIVISION.
    :
    :
```

"Run Unit Oriented Commands"

```
IMPART [ON ERROR GO TO error-paragraph].

DEPART [WITH ROLLBACK] [ON ERROR GO TO error-paragraph].
```

FREE [ON ERROR GO TO *error-paragraph*] .

"Area Oriented Commands"

OPEN ALL [USAGE-MODE IS { [EXCLUSIVE] / [PROTECTED] } { RETRIEVAL / UPDATE }]
INITIAL LOAD

[ON ERROR GO TO *error-paragraph*] .

OPEN *area-name–1* [USAGE-MODE IS { [EXCLUSIVE] / [PROTECTED] } { RETRIEVAL / UPDATE }]
INITIAL LOAD

[*,area-name–2* [USAGE-MODE IS { [EXCLUSIVE] / [PROTECTED] } { RETRIEVAL / UPDATE }]] ...
INITIAL LOAD

[ON ERROR GO TO *error-paragraph*] .

CLOSE { ALL / *area-name–1,area-name–2...* } [ON ERROR GO TO *error-paragraph*] .

"Record Oriented Commands"

STORE *record-name* [SUPPRESS { ALL / RECORD / AREA / SETS / *set-name–1*[*,set-name–2*] ... } CURRENCY UPDATES]

[ON ERROR GO TO *error-paragraph*] .

{ FIND / FETCH } *rse* [SUPPRESS { ALL / RECORD / AREA / SETS / *set-name–1*[*,set-name–2*] ... } CURRENCY UPDATES]

[AT END GO TO *end-paragraph*] [ON ERROR GO TO *error-paragraph*] .

"Where the rse is one of the following"

Identifier–1 [,*identifier–2*]

CURRENT **RECORD WITHIN** *record-name–1* RECORD

(OWNER / CURRENT / NEXT / PRIOR / FIRST / LAST / *identifier–2*) **RECORD WITHIN** { *set-name–3* SET / *area-name–1* AREA }

(NEXT / PRIOR / FIRST / LAST / *identifier–3*) *record-name–2* **WITHIN** { *set-name–4* SET / *area-name–2* AREA }

[NEXT DUPLICATE **WITHIN**] *record-name–3* RECORD

record-name–4 VIA *set-name–5* [USING *data-base-identifier–1* [*data-base-identifier–2*] ...]

NEXT DUPLICATE **WITHIN** *set-name–6* USING *data-base-identifier–3* [,*data-base-identifier–4*] ...

GET [ON ERROR GO **TO** *error-paragraph*] .

MODIFY [*data-base-identifier–1* [,*data-base-identifier–2*] ...]

[ON ERROR GO **TO** *error-paragraph*] .

DELETE [ONLY / ALL] [ON ERROR GO **TO** *error-paragraph*] .

KEEP [ON ERROR GO **TO** *error-paragraph*] .

"Set Oriented Commands"

INSERT **INTO** *set-name–1* [,**INTO** *set-name–2*] ... [ON ERROR GO **TO** *error-paragraph*] .

INSERT **INTO** ALL **SETS** [ON ERROR GO **TO** *error-paragraph*] .

REMOVE **FROM** *set-name–1* [,*set-name–2*] ... [ON ERROR GO **TO** *error-paragraph*] .

REMOVE **FROM** ALL **SETS** [ON ERROR GO **TO** *error-paragraph*] .

"Conditional Commands"

IF *set-name–1* SET [NOT] **EMPTY**;

{*statement–1* / NEXT SENTENCE} [ELSE {*statement–2* / NEXT SENTENCE}].

IF RECORD [NOT] {OWNER / MEMBER} **OF** {*set-name–2* / ANY} SET;

{*statement–3* / NEXT SENTENCE} [ELSE {*statement–4* / NEXT SENTENCE}].

"Supporting Commands"

LOG *data-name–1* **WORDS** [ON ERROR GO **TO** *error-paragraph*].

LOG *data-name–1* **WORDS FOR** RECOVERY **POINT** [ON ERROR GO **TO** *error-paragraph*].

MOVE **CURRENCY** STATUS **FOR** { RUN-UNIT / *record-name* RECORD / *area-name* AREA / *set-name* SET }

TO *identifier–1* [ON ERROR GO **TO** *error-paragraph*].

MOVE ‖ AREA-KEY / AREA-NAME ‖ **FOR** { RUN-UNIT / *record-name* RECORD / *area-name* AREA / *set-name* SET / *identifier–2* }

TO ‖ *identifier–3* / *identifier–4* ‖ [ON ERROR GO **TO** *error-paragraph*].

APPENDIX C. DML PREPROCESSOR CONTROL CARD

C.1. GENERAL

The control card for the DML Preprocessor follows standard conventions established for 1100 Language Processors. (See *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version).) If not part of a system library, the DML Preprocessor may be loaded into a program file from tape using an appropriate FUR/PUR command.

C.2. DML PREPROCESSOR CONTROL CARD FORMAT

The format of the DML Preprocessor control card is:

▽ file.ADMLP,options ELEMENT1,ELEMENT2,ELEMENT3

where:

file – Is the name of the program file containing the absolute DML Preprocessor element. This name and trailing period may be omitted if the program file is TPF$ or RLIB$.

ADMLP Preprocessor – Is the name of the absolute element containing the Data Manipulation Language Preprocessor.

options –

NOTE:

The DML Preprocessor will automatically call the installation-standard COBOL compiler unless specifically instructed not to do so. If the compiler is to be called automatically the user must specify the options for the COBOL compiler on the first card following the DML Preprocessor call card in the following format:

COBOL-OPTIONS options.

where options is a character string of legal COBOL options, with no intervening blanks.

The legal options on the call card are as follows:

A – Accept element even if fatal errors are detected. Overrides X.

E – Output all diagnostic messages.

I – Introduce source language into program file from control.

L – Output all listings (*source input, source output and all diagnostics*). Overrides all listing options except N.

N – Suppress all listings. Overrides all listing options.

O – List Source output.

P – Source output stored in Fieldata – card input in Fieldata.

Q – Source output stored in ASCII – card input in ASCII.

S – List Source input.

T – Reverse the meaning of DISP and DISP-1, and COMP and COMP-4. This option also causes the DML Preprocessor to generate an appropriate description of the DMCA and record-delivery-area. If the T-option is specified only DDL records whose MODE IS FIELDATA may be specified in the COPYING clause. If T-option is used for the preprocessor, it must also be used for COBOL.

U – Update. Produce new cycle of source input program element.

V – Check sequence numbers in columns 1–6.

X – Abort run if fatal errors occur.

Z – Do not call the COBOL compiler automatically.

Each of the three following fields (ELEMENT1, ELEMENT2, ELEMENT3) specify program file elements in the standard format. See *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version).

- ELEMENT1 – Program file element name of symbolic input. If present, and there is no I option, the lines immediately following the control statement are taken to be corrections to the source language element. If an I option is present, then the lines following the control statement are given to the processor and are inserted into the program file as well. Hence, not required if input is from cards.

- ELEMENT2 – Specifies the source COBOL element produced by the DML Preprocessor as output. If omitted, the DML Preprocessor output is inserted into TPF$.DMLOUT.

- ELEMENT3 – Specifies the source language element produced by updating the input source language element. If this field is void, no updated source language element will be produced unless a "U" option is specified. In that case, an updated element is produced with the same name and version as the input element, but with a cycle number one greater. Does not need to be specified when 'I' option is used.

C.3. IMPLICIT COBOL CALL CARD

The COBOL call card implicitly produced by the DML Preprocessor to execute the compiler has the following form:

▽ COB,options ELEMENT1,ELEMENT2

NOTE:

This call card causes execution of the COBOL compiler which the installation has decided to be the system standard. If the system standard is not the American National Standard ASCII compiler, then the user should always suppress this call card by using a 'Z' option on his DML Preprocessor call card.

- ELEMENT1 – Names the input element and is ELEMENT2 from the DML Preprocessor call card.
- ELEMENT2 – Names the program file element which is the relocatable code produced by the COBOL compiler. The name consists of the first set of characters from the program-name in the PROGRAM-ID clause. The qualifier, file, version and cycle is the same as those specified for ELEMENT1 of this card. Up to six characters of the program name are used unless the string is terminated by a space or non-alphanumeric character.
- options – If options are desired, they must appear on the 1st card following the @DML Preprocessor call card. The key-word COBOL-OPTIONS must appear in column 8, and the desired options must follow as a single character string; i.e., there may be no intervening spaces in the option list.

APPENDIX D. MAPPING OF A CALC PROCEDURE

D.1. GENERAL

In practice, a user Data Management program may utilize many CALC Procedures for different types of record placement or retrieval. To minimize the size of the total user program, the DMR expects that each CALC Procedure is positioned by the collector into an individual overlay segment. The DMR utilizes the indirect segment loading feature of the collector to load each segment only when needed.

NOTE:

The System CALC Procedure DMSCALC, if used, need not be mapped.

D.2. TABLE ELEMENT 'CALSEG'

For the multi-thread DMR to find and load the segment the required CALC Procedure is in, the user must assemble an entry point element. This element is a table containing three word entries, one for each entry point, of the form:

'ENTRY POINT'
+ENTRY POINT, SEGMENT NAME

S6 of the first word of the table must contain the number of entry points in this table, and H1 must contain SLT$ which will be replaced by the collector with the segment load table address. The field data entry point name (*first two words in each entry*) can be from 1 to 12 alphanumeric characters long, but two words must be specified.

NOTE:

If the entry point is less than six characters, it will be left justified in the first word of the entry.

If the CALC Procedure(s) are in the main segment, the segment name may be '0'. The element must be in the user's D-bank for the REP to access it, therefore, it must be assembled under an even control counter. The entry point to this element must be CLSEG$.

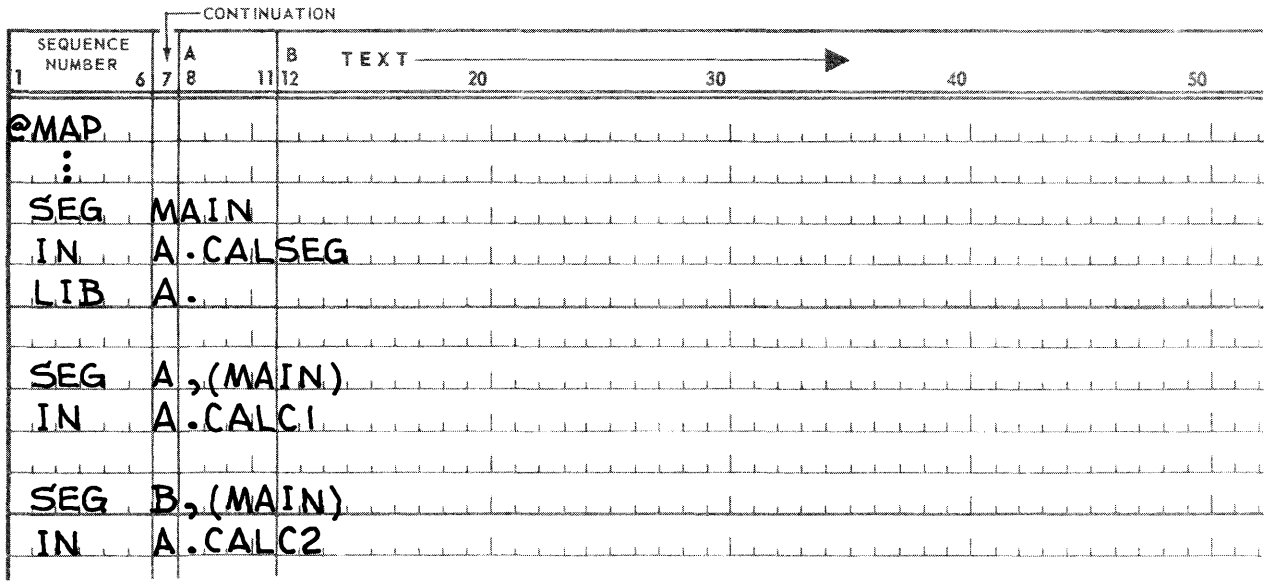
If the CALC Procedures are segmented as in D.4, CALSEG would look like:

```
@ASM,I      A.CALSEG
            $(2),CLSEG$*
            +SLT$,2
            'EPT1  '
            '      '

            +EPT1,A
            'EPT2  '
            '      '

            +EPT2,B
```

Where EPT1 and EPT2 are externalized labels within the respective CALC Procedures where the instruction logic begins.

D.3. EXAMPLE

In the following example assume all elements are in the same file, named A.



Where CALC1 and CALC2 are the names of pre-assembled CALC Procedures.

There is no need to IN A.CALSEG if there is only one element with entry point CLSEG$ in the LIB file.

APPENDIX E. DML PREPROCESSOR ERROR MESSAGES

E.1. GENERAL

Two classes of error messages are issued by the DML Preprocessor:

- Fatal (F)
- Warning (W)

The DML Preprocessor not only indicates if an error is Fatal (*the validity of the output is questionable*) or Warning (*a minor infraction of the syntactical rules occurred*), but also the line number, and whenever possible the word in the line where the error is detected.

Example:

```
ERROR*0080    3F    error-message
```

indicates that a fatal error was detected on line 80, at the third word.

E.2. ERROR MESSAGES

The following section lists the possible error messages which may result.

A DIVISION IS MISSING – – COMPILATION WILL PROCEED
ALPHANUMERIC EDITING MAXIMUM SIZE 255;TRUNCATED.

BAD CHARACTER IN COLUMN 7 – – ACCEPTED AS START OF
BAD COPY STATEMENT
BAD INFORMATION ON LIBRARY

CHARACTER AFTER PUNCTUATION IS BEGINNING OF NEXT WORD
CONFLICTING PICTURE AND USAGE CLAUSES – – USAGE IGNORED
CONTINUATION OF LITERAL PRIOR TO COLUMN 12 – – ACCEPTED
COPYING COMMON AND RDA COMMON DO NOT AGREE
CORRECTION CARDS ARE OUT OF ORDER.

DATA-BASE-DATA-NAME MISPOSITIONED – – SKIPPED.
DECLARATIVES SECTION IS MISSING
DML RESERVED WORD INCORRECTLY USED
DIVISION OR SECTION HEADER NOT RECOGNIZED – – SKIPPED
DUPLICATE AREA/REC/SET OR ITEM NAME – – IGNORED
DUPLICATE DATA DIVISION – SKIPPED
DUPLICATE ENVIRONMENT DIVISION – – SKIPPED
DUPLICATE IDENTIFICATION DIVISION – – SKIPPED
DUPLICATE PROCEDURE DIVISION – – SKIPPED
D$WORK CANNOT BE CONSTRUCTED

END OF PROGRAM IN MIDDLE OF LITERAL – TERMINATED
EXACT BINARY MAXIMUM 72 BITS; EXCESS TRUNCATED.

FILE NAME CONFLICTS WITH SCHEMA-NAME

IF DMCA, REC, OR RDA IS IN LINKAGE, NONE MAY BE IN WORKING
ILLEGAL CHARACTERS IN PICTURE – IGNORED.
ILLEGAL CHARACTERS IN WORD
ILLEGAL ORDER OF CHARACTERS – ILLEGAL ONES IGNORED
ILLEGAL REPEAT COUNT – ILLEGAL CHARACTER(S) IGNORED.
IMPART NOT ALLOWED IF RDA, RECORDS, OR DMCA IN LINKAGE
IMPART NOT ALLOWED WITH RANDOM PROCESSING
IMPROPER PUNCTUATION TO LEFT OF WORD – ACCEPTED
IMPROPER RIGHT PARENTHESIS – IGNORED
INTEGER TOO LARGE – – TRUNCATED
INVALID/OMITTED QUALIFIER
INVALID OR MISSING PROCEDURE-NAME
INTEGER MUST NOT BE SIGNED – – ACCEPTED
INTEGER SHOULD NOT HAVE FRACTIONAL POSITIONS
INVALID SUBSCRIPT
INVALID SUPPRESS CLAUSE – – ACCEPTED
INVALID USAGE ON DATA-NAME IN FIND OR MOVE

LEFT PARENTHESIS FOLLOWED BY SPACE – – ACCEPTED
LIBRARY REQUEST NOT FOUND
LITERAL CONTINUATION UNCLEAR – – TERMINATED AT CARD END
LITERAL HAS INCORRECT DECIMAL POINT – – TREATED AS INTEGER
LITERAL IS TOO LONG IT WILL BE TRUNCATED
LITERAL SHOULD NOT HAVE MINUS SIGN – – TREATED AS PLUS

MAXIMUM DECIMAL SIZE 18; EXCESS TRUNCATED.
MISPLACED DATA DIVISION – – ACCEPTED
MISPLACED ENVIRONMENT DIVISION – – ACCEPTED
MISPLACED IDENTIFICATION DIVISION – – ACCEPTED
MISSING NAME OR RESERVED WORD
MISSING OPTION – – 'ALL' ASSUMED
MISSING OPTION – – SECTION IGNORED
MISSING PERIOD OR INVALID SYNTAX

NAME NOT FOUND IN SCHEMA – – SKIPPED
NUMERIC EDITING MAXIMUM SIZE 37; EXCESS TRUNCATED.

OVER 30 CHARACTERS IN WORD OR NUMERIC LITERAL
OVER 132 CHARACTERS IN LITERAL – – TRUNCATED TO 132
OWNER AND AREA COMBINATION IS NOT ALLOWED
PFS ERROR: SCHEMA CANNOT BE ADDRESSED
PUNCTUATION WAS NOT FOLLOWED BY SPACE – – ACCEPTED.

RECORD ALREADY OVERLAID
RECORD MODE MUST BE ASCII IF T OPTION NOT SET – – RECORD IGNORED.
RECORD-NAME WAS NOT INVOKED – – REJECTED
RECORD CLAUSE NOT FOUND – – SCHEMA SECTION REJECTED
RESERVED WORD INCORRECTLY USED – – ACCEPTED
ROLLBACK CLAUSE REQUIRED FOR IMPART

SCHEMA FILE CANNOT BE ASSIGNED
SCHEMA SEGMENT IS TOO LARGE
SECTION MISPLACED OR ALREADY PROCESSED
SEQUENCE NUMBERS ARE NON-ASCENDING – – ACCEPTED
SIZE OF ITEM MUST BE 9 OR 10 COMPUTATIONAL CHARACTERS
SPACE SHOULD NOT PRECEDE PERIOD – – ACCEPTED
SPECIFIED OPTION IS NOT IMPLEMENTED – – SKIPPED
SPECIFIED USAGE IS INVALID IF INVOKE IS ABSENT
SYNTACTICAL ERROR – – REJECTED

THIS OPTION NOT VALID – – IGNORED
THIS PARAGRAPH NAME MUST BE NON-NUMERIC.
TOO MANY AREA/REC/SET OR ITEM NAMES – TRUNCATED

USAGE NOT VALID AT THIS LEVEL
USER DEFINED NAME MISSING OR OUT OF ORDER

VOID NON-NUMERIC LITERAL – – CHANGED TO 1 SPACE

WORD SHOULD NOT START BEFORE COLUMN 12 – – ACCEPTED
WORD SHOULD START IN COLUMN 8 – – ACCEPTED

APPENDIX F. SAMPLE DML RUN STREAM

F.1. GENERAL

Three sample run streams are given and all examples assume that a previously created SCHEMA exists on a catalogued file.

F.2. EXAMPLES

F.2.1. Example 1:

```
@ASG,A   TSTSCH
@ADMLP,ILZ  .DMLSOR,.COBSOR
        IDENTIFICATION DIVISION.
                  :
        DATA DIVISION
                  :
        SCHEMA SECTION.
        INVOKE SCHEMA TSTSCH
                  :
        PROCEDURE DIVISION.
                  :
@ACOB,LB  .COBSOR,.COBOBJ
@MAP,I   .COBMAP,.COBABS
 IN   .LINKER
 IN   .D$WORKnnnnnn
 IN   .COBOBJ
 IN   .DMR
```

Discussion:

In this example, the previously created SCHEMA resides in the catalogued file, TSTSCH, which must be @ASGed in the run stream. The options indicate that the source program is on the cards which follow the call card, no sequence number check is to be performed, and the COBOL compiler is not to be called by the Preprocessor. The card images are put into a source element called .DMLSOR and put in TPF$. The output of the DMLP is a source element called COBSOR and is put into TPF$.

The ASCII COBOL compiler is in TPF$. The input to the compiler is the DML Preprocessor output COBSOR. The main relocatable output element is called COBOBJ and is placed in TPF$.

The MAP processor collects the relocatable elements LINKER, D$WORKnnnnnn (*where nnnnnn are the first six characters of the PROGRAM-ID in the IDENTIFICATION DIVISION of the COBOL program*) and COBOBJ and other necessary relocatable elements found in TPF$ or the systems library to produce an absolute element, COBABS, which is placed in TPF$. The MAP processor also produces a map element COBMAP, and places it in TPF$.

F.2.2. Example 2:

```
@ASG,CP   TEST.
@ASG,A    TSTSCH.
@ADMLP,IL TEST.DMLSOR,COBSOR
       IDENTIFICATION DIVISION.
       PROGRAM-ID.COBOUT-PROG
           :
       DATA DIVISION
           :
       SCHEMA SECTION
       INVOKE SCHEMA TSTSCH
           :
       PROCEDURE DIVISION.
           :
@MAP,I    TEST.COBMAP,.COBABS
 IN .LINKER
 IN .D$WORKCOBOUT
 IN .COBOUT
 IN .DMR
```

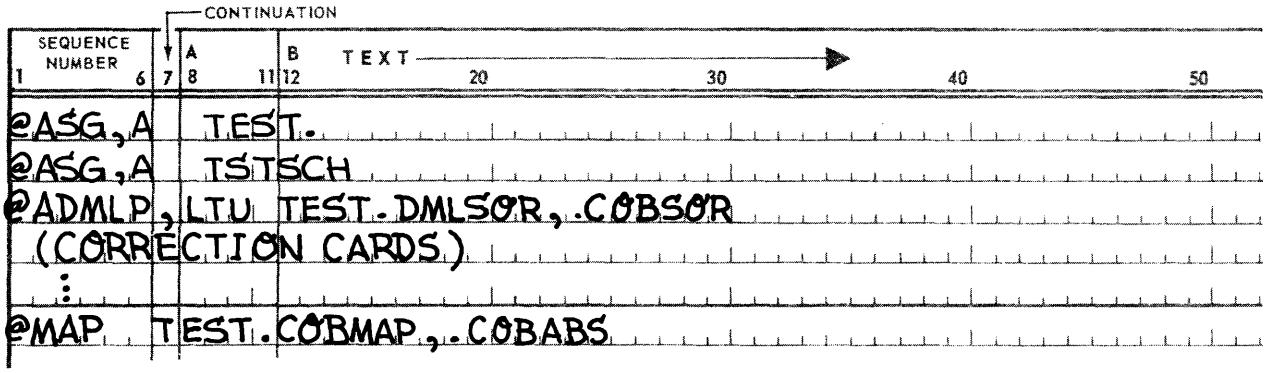
Discussion:

In example F.2.2, the Z option is not present, indicating that the DML Preprocessor is to call the COBOL compiler. The cards following the DML Preprocessor call card are made into a source element, DMLSOR, and put in the file TEST., which was catalogued public by the run stream. The output of the DML Preprocessor is a source element, COBSOR, which is placed in TPF$.

The main relocatable output element of the COBOL compiler is given the name COBOUT (*the first six characters of the PROGRAM-ID*) and placed in TPF$ (*the same file in which the input to the compiler is placed*).

The MAP processor produces an absolute element, COBABS, and puts it in TPF$. The map element, COBMAP, is put in the catalogued file TEST.

F.2.3. Example 3:



Discussion:

In example F.2.3, the DML Preprocessor applies the correction cards following the DML Preprocessor call card to the source element DMLSOR in the catalogued file TEST. (See example F.2.2.) The file TEST. now contains the updated cycle of DMLSOR.

Since the T-option is specified, all the records INVOKED by the run-unit must have been defined to the DDL with a 'RECORD MODE IS FIELDATA' clause.

The MAP processor uses the map element COBMAP in the file TEST. to determine which elements to collect. It produces the absolute element COBABS and places it in TEST.

APPENDIX G. LIST OF ABBREVIATIONS

ATT	–	Audit Trail Tape
AUTORP	–	Automatic Recovery Point
CD	–	Communication Descriptions
DDL	–	Data Definition Language
DMCA	–	Data Management Communications Area
DML	–	Data Manipulation Language
DMR	–	Data Management Routine
DMS	–	Data Management Systems
DMSCALC	–	Data Management System – Supplied Calc
DMSSGP	–	Data Management Systems Support Generation Parameter Element
RDA	–	Record Delivery Area
TIP	–	Transaction Interface Package