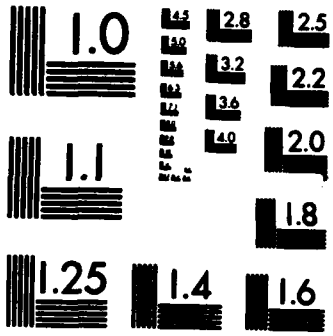
AD-A101 955 INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 5 1/2
COMMON DATA MODEL S..(U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONSU..
UNCLASSIFIED J L ALTHOFF ET AL. 01 NOV 85 UM-620141100 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A181 955

(2)

AFWAL-TR-86-4006
Volume V
Part 7

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 7 - NDDL User's Guide

General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345

Final Report for Period 22 September 1980 - 31 July 1985

November 1985

Approved for public release; distribution is unlimited.

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533

NOTICE

When Government drawings specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the government may have formulated, furnished, or in any way supplied the said drawings specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

DAVID L. JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986
DATE

FOR THE COMMANDER:

GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86
DATE

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MLTC, W-PAFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

AD A181 955

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-4006 Vol V, Part 7
6a. NAME OF PERFORMING ORGANIZATION General Electric Company Production Resources Consulting 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFWAL/MLTC
6c. ADDRESS (City, State and ZIP Code) 1 River Road Schenectady, NY 12345	7b. ADDRESS (City, State and ZIP Code) WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Materials Laboratory Air Force Systems Command, USAF 8b. OFFICE SYMBOL (If applicable) AFWAL/MLTC	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-80-C-5155
8c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433	10. SOURCE OF FUNDING NOS PROGRAM ELEMENT NO. 78011F; PROJECT NO. 7500; TASK NO. 62; WORK UNIT NO. 01
11. TITLE (Include Security Classification) (See Reverse)	

12. PERSONAL AUTHOR(S) Althoff, J. Apicella, M. Bernier, M. Bradley, W. Singh, S. Thompson, D. Hogan, J. Leifeste, B.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final Technical Report	22 Sept 1980 - 31 July 1985	1985 November	160

16. SUPPLEMENTARY NOTATION

ICAM Project Priority 6201

The computer software contained herein are theoretical and/or references that in no way reflect Air Force-owned or -developed computer software.

17. COSATI CODES

FIELD	GROUP	SUB GR
1308	0905	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Common Data Model Processor (CDMP) is a mechanism by which application programs can retrieve and update data without knowing where or how the data are stored. The CDMP relies on three types of schemas (conceptual, external, and internal) and on two types of schema mappings (conceptual-to-external and conceptual-to-internal). A schema is a description of data from a particular viewpoint. A mapping is a description of the correspondence between elements of one schema and those of another. The schemas and mappings are stored in a database, called the Common Data Model (CDM), where they are available to the CDMP software. The Neutral Data Definition Language (NDDL) is the means for storing and maintaining the schemas and mappings in the CDM database. This manual explains the syntax and semantics of each NDDL command.

ICAM (Integrated computer aided manufacturing)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NUMBER (Include Area Code) 513-255-6976 22c. OFFICE SYMBOL AFWAL/MLTC

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

11. Title

Integrated Information Support System (IISS)
Vol V - Common Data Model Subsystem
Part 7 - NDDL User's Guide

A S D 86 1477
17 Jul 1986

PREFACE

This user's manual covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Reviewer.
D. Appleton Company (DACOM)	Responsible for IDEF support, state-of-the-art literature search.
General Dynamics/ Ft. Worth	Responsible for factory view function and information models.

Subcontractors	Role
Illinois Institute of Technology	Responsible for factory view function research (IITRI) and information models of small and medium-size business.
North American Rockwell	Reviewer.
Northrop Corporation	Responsible for factory view function and information models.
Pritsker and Associates	Responsible for IDEF2 support.
SofTech	Responsible for IDEF0 support.

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Responsible for consultation on applications of the technology and on IBM computer technology.
Computer Technology Associates (CTA)	Assisted in the areas of communications systems, system design and integration methodology, and design of the Network Transaction Manager.
Control Data Corporation (CDC)	Responsible for the Common Data Model (CDM) implementation and part of the CDM design (shared with DACOM).
D. Appleton Company (DACOM)	Responsible for the overall CDM Subsystem design integration and test plan, as well as part of the design of the CDM (shared with CDC). DACOM also developed the Integration Methodology and did the schema mappings for the Application Subsystems.

Subcontractors	Role
Digital Equipment Corporation (DEC)	Consulting and support of the performance testing and on DEC software and computer systems operation.
McDonnell Douglas Automation Company (McAuto)	Responsible for the support and enhancements to the Network Transaction Manager Subsystem during 1984/1985 period.
On-Line Software International (OSI)	Responsible for programming the Communications Subsystem on the IBM and for consulting on the IBM.
Rath and Strong Systems Products (RSSP) (In 1985 became McCormack & Dodge)	Responsible for assistance in the implementation and use of the MRP II package (PIOS) that they supplied.
SofTech, Inc.	Responsible for the design and implementation of the Network Transaction Manager (NTM) in 1981/1984 period.
Software Performance Engineering (SPE)	Responsible for directing the work on performance evaluation and analysis.
Structural Dynamics Research Corporation (SDRC)	Responsible for the User Interface and Virtual Terminal Interface Subsystems.

Other prime contractors under other projects who have contributed to Test Bed Technology, their contributing activities and responsible projects are as follows:

Contractors	ICAM Project	Contributing Activities
Boeing Military Aircraft Company (BMAC)	1701, 2201, 2202	Enhancements for IBM node use. Technology Transfer to Integrated Sheet Metal Center (ISMC).

Contractors	ICAM Project	Contributing Activities
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP).
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology.
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements.
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI).
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

INTRODUCTION

The Neutral Data Definition Language, hereafter NDDL, is an interpretive language that was developed to populate and maintain the Common Data Model database (CDM1 Doc. Control No.-CCS 620141000). As the NDDL is intended to be used by data processing professionals who are experts with various data base management systems, a discussion of data base management systems is not included in this document (see CDM Administrators Manual Doc. Control No. UM 620141000). Furthermore, an understanding of the CDM database is necessary.

The CDM Database uses a Three-Schema approach based upon "The ANSI/X3/SPARC DBMS Framework: Report of the Study Group on Data Base Management Systems". In the IISS, the Three-Schema Architecture is implemented through the CDM facilities to store each of the three types of schemas and the interschema mappings. The NDDL supports the population and maintenance of appropriate representations for each of the three types of schemas.

The conceptual schema is represented by an IDEF-1 model. The CDM stores this model in terms of entity classes, attribute classes, and relation classes.

The external schemas are represented by tables. The mappings between these tables and the IDEF-1 model of the conceptual schema are part of the CDM database.

The internal schemas are represented in terms of physical database components, including record types and inter-record relationships.

Section 2 of this manual is a discussion of how to interface with the NDDL processor and the NDDL syntax which is similar to that of the Neutral Data Manipulator Language (see NDML Guide PRM 620141200).

Actual NDDL commands are explained in Section 3 which is followed by NDDL Command Error Messages which are listed in Appendix A.

Appendix B is a Glossary of the more important terms used in this guide and is followed by a list of references.

SECTION 2

OVERVIEW

2.1 NDDL ENVIRONMENT

NDDL can be executed from three environments within IISS:

- batch environment
- interactive environment without the IISS User Interface/Virtual Terminal Interface (UI/VTI)
- interactive environment with the IISS User Interface/Virtual Terminal Interface (UI/VTI)

The environment is determined from the users response to the "args" prompt upon NDDL activation as depicted below.

2.1.1 Batch Environment

args: ‹ file specification › file specification

‹ file specification - redirect the input for NDDL to come from the file specified

› file specification - redirect the output for NDDL to be written to the file specified

file specification - DISK_NAME /UFD/SFD/FILE.EXTENSION

Example:

args: ‹NDDL.INP ›NDDL.OUT

NDDL.INP is a file containing NDDL commands.

Note:
This file name must follow the Unix standard for naming files, not that of the host operating system.

2.1.2 Interactive Environment (without IISS UI/VTI)

args: (carriage return)

The NDDL "NEXT COMMAND" prompt will be displayed, at which time

the user may enter any valid NDDL command.

Example:

```
args:     (carriage return)

NEXT COMMAND--

CREATE MODEL test_model;

--COMMAND SUCCESSFUL--

NEXT COMMAND--

HALT;
```

2.1.3 Interactive Environment (with IISS UI/VTI)

args:-I

The NDDL form will be displayed, at which time the user may enter NDDL commands. A description of the form and how it is used for NDDL is contained in the next section of this document "Using the NDDL Form."

2.2 Using the NDDL Form

This section describes the use of the NDDL form. The form manipulations are supported by the IISS User Interface/Virtual Terminal Interface (UI/VTI) software. This section will also describe the detailed forms procedures unique to the NDDL application. For general information regarding forms use, refer to the IISS UI/VTI Terminal Operator's Guide.

2.2.1 NDDL Form Description

The NDDL form appears immediately following NDDL activiation with the -I parameter as explained in the "NDDL Environment" section. A description of the individual fields follows.

Field 1 - Current Model Field

This field displays the current model name. The current model field is modified each time a CREATE MODEL or ALTER MODEL command is successfully executed. If neither

command has been executed in a session, the "NOT YET SPECIFIED" message will appear in this field.

Field 2 - Current Database Field

This field displays the current database name. The current database field is modified each time a DEFINE DATABASE, DEFINE SET, or DEFINE RECORD command is successfully executed. If none of the commands have been executed in a session, the "NOT YET SPECIFIED" message will appear in this field.

Field 3 - NDDL Command Field

This is the field in which NDDL commands are entered. This field is 78 columns wide and 100 lines long and is viewed 3 lines at a time. This field may be paged and scrolled as described in the IISS UI/VTI Terminal Operator's Guide.

Field 4 - Message Area

This field displays NDDL error and informative messages. The message area is 30 lines long and is viewed 3 lines at a time. This field may be paged and scrolled as described in the IISS UI/VTI Terminal Operator's Guide.

Field 5 - Message Line Number Field

The message line number field serves two purposes. It is used to display the number of the message displayed in field 6, the message line. The message line number is also used to select a particular message in field 6 for viewing. The user can position the cursor to this field, enter a number from 01 to 99 and read the corresponding message, assuming one exists.

Field 6 - Message Line

The message line is 99 lines long, viewed one line at a time. This field may contain informative and error messages which overflow field 4. In addition, this field will contain summary messages indicating whether or not errors have occurred and the number of generated messages.

2.2.2 Form Use

VT100 Keypad

The VT100 keypad is depicted in Figure 2-1. The keys which apply uniquely to NDDL are:

pf4 - Press pf4 to terminate NDDL. This key forces an NDDL HALT command, freeing the user from explicitly entering the HALT.

enter - Press enter to initiate execution of the entered NDDL command(s). After the commands are executed or syntax checked, the form reappears. If an error was encountered, all commands reappear along with their error messages. If, however, no errors are encountered, the commands do not reappear when the form is repainted.

0 (pf16) - Press 0 to initiate execution of the entered NDDL command(s). After the commands are executed or syntax checked, the form reappears with the previously entered commands showing whether or not error occurred. This feature is intended to be used when the user has a number of similar commands to execute. The user need only modify that dynamic portion of the command for each subsequent execution.

All other keypad keys operate as described in the UI/VTI Terminal Operator's Guide. In addition, there are a number of non-keypad keys that manipulate the form and are described in the same document.

2.2.3 NDDL Command Entry

All NDDL commands must be entered in the NDDL command field with syntax as described in the "NDDL Commands" section of this document. The terminating semicolons are mandatory for all commands. The user may enter as many complete commands as may be contained in 100 lines. After execution has started, the form will not reappear until all commands have executed or had their errors diagnosed.

pf1 GOLD	pf2 help	pf3 debug	pf4 quit HALT
7 (pf5 scroll left	8 (pf6) scroll right	9 (pf7) scroll up	- (pf8) scroll down
4 (pf9) page left	5 (pf10) page right	6 (pf11) page up	, (pf12) page down
1(pf13)	2 (pf14)	3 (pf15)	enter Execute NDDL commands - redisplay the commands only if errors encountered
0 (pf16) Execute NDDL commands redisplay the commands unconditionally		. (pf17)	

Figure 2-1. VT100 Function Keypad

2.2.4 Error Message Reporting

After a form of commands has executed, error messages may or may not appear. If no errors have occurred, a "no errors encountered" message will appear in the message line of the form. If errors have occurred, a "command(s) completed - n error messages" message will appear with n equal to the number of messages generated. Note that a particular command may generate many error messages. The message area may contain up to 30 messages, displayed 3 messages at a time. Should there be more than 30 messages generated, the final line of the message area will contain the following message: PLEASE REFER TO THE MESSAGE LINE FOR REMAINING MESSAGES. Since the message line indicates that there were n messages generated and since 30 messages are contained in the message area, (n-30) messages remain to be viewed in the message line. Position the cursor to the message line number field (field 4), key in 01 and press enter for the first message. For each subsequent message, increment field 4 by one and press enter.

If many commands have been entered and errors were encountered, all commands preceding the erroneous command will have executed, applying their specified database modifications. The erroneous command and all subsequent commands in the same form will not execute. The commands following the erroneous command on the same form will be syntax checked only.

To assist in matching the error messages with the proper NDDL commands, the following messages are generated:

--COMMAND SUCCESSFUL--	- The command has successfully executed.
--NO CHANGE MADE--	- Errors were encountered. The detailed error messages which apply to this message precede it.
--COMMAND SYNTAX CORRECT--	- An error in a previous NDDL command caused all subsequent commands in the form to be syntax checked only.

--INVALID SYNTAX-- - The command syntax is incorrect.

For example, if five "--COMMAND SUCCESSFUL--" messages are generated before the first error message appears, the offending NDDL command is the sixth one on the form.

2.3 NDDL COMMAND SYNTAX

In the syntactic description of the NDDL commands, the following symbols are used:

[ ] indicates an optional word or phrase
{ } indicates a choice of only one word or phrase
... indicates repetition of the last element
; indicates the end of a command and must be entered by the user
| indicates a choice of options

All uppercase words must be specified as indicated; lowercase words indicate a user-defined variable which the user may fill in with uppercase or lowercase letters. Lowercase words are case sensitive, e.g., variable XYZ is not the same as variable xyz. The user may enter any string of up to 30 characters for lowercase words. These characters may be any combination of letter, digit, dash, or underscore

Comments may be embedded by surrounding the comment with /* and */.

Most repetitions are separated by a single blank space; however, a few commands must be separated by commas.

2.4 NDDL RESERVED WORDS

Following is a list of NDDL reserved words. These words cannot be used as a user-defined variable in any NDDL command.

ACCESSED	FILES	PCB
ADD	FLOAT	POSITION
ALIAS	FROM	PRIMARY
ALTER	HALT	PSB
AND	HL6	RECORD
AREAS	HOST	RELATING
AS	IBM	RELATION
ATTRIBUTE	IDMS	RENAME
BY	IDS_II	REQUIRED

CHARACTER
CHECK
CLASS
COLUMNS
COMBINE
COMMIT
COMPARE
COPY
CREATE
DATA
DATABASE
DATAITEM
DATAFIELD
DECIMAL
DEFINE
DESCRIBE
DESCRIPTION
DOMAIN
DROP
ELEMENTS
ENTITY
EXCEPT
EXIT
FEEDBACK
FIELD
FIELDS
FILE_NAME
IMS
IN
INTEGER
INTO
IS
ITEM
ITEMS
KEY
KEYWORD
LINKED
LENGTH
MANY
MAP
MERGE
MIGRATES
MODEL
NAMED
NUMERIC
OF
ON
OPTIONAL
ORACLE
OWNED
P
PACKED
PATH
PASSWORD
ROLLBACK
S
SCHEMA
SECOND
SEGMENT
SELECT
SET
SIGNED
SIZE
STANDARD
START
STRUCTURE
SUBSCHEMA
TABLE
TO
TOTAL
U
UNKNOWN
UNSIGNED
TYPE
VALUE
VAX_11
VAX
VIEW
WHERE
WITH

A cross reference between NDDL object types and NDDL command is contained in Table 2-1.

TABLE 2-1

CROSS REFERENCE OF OBJECT TYPES AND NDDL COMMANDS

OBJECT TYPE	NDDL COMMAND
ALIAS	ALTER ALIAS COMBINE ENTITY COPY ATTRIBUTE COPY ENTITY COPY MODEL CREATE ALAIS DROP ALIAS MERGE MODEL
AREA	DEFINE DATABASE DEFINE RECORD
ATTRIBUTE CLASS	ALTER ATTRIBUTE ALTER ENTITY ALTER MAP COPY ATTRIBUTE COPY DESCRIPTION CREATE ALIAS CREATE ATTRIBUTE CREATE ENTITY DESCRIBE ATTRIBUTE DROP ALIAS DROP ATTRIBUTE RENAME ATTRIBUTE
CARDINALITY	ALTER RELATION CREATE RELATION
DATA ITEM	CREATE VIEW

TABLE 2-1 (Continued)

CROSS REFERENCE OF OBJECT TYPES AND NDDL COMMANDS

OBJECT TYPE	NDDL COMMAND
DATABASE	ALTER MAP
	CREATE MAP
	DEFINE DATABASE
	DEFINE RECORD
	DEFINE SET
	DROP DATABASE
	DROP FIELD
	DROP RECORD
	DROP SET
DATA TYPE	ALTER DOMAIN
	ALTER MAP
	CREATE DOMAIN
	CREATE MAP
	CREATE VIEW
DESCRIPTION	COMBINE ENTITY
	COPY ATTRIBUTE
	COPY DESCRIPTION
	COPY ENTITY
	COPY MODEL
	DESCRIBE ATTRIBUTE
	DESCRIBE ENTITY
	DESCRIBE RELATION
	MERGE MODEL
DOMAIN	ALTER DOMAIN
	CREATE ATTRIBUTE
	CREATE DOMAIN
	DROP DOMAIN
	RENAME DOMAIN

TABLE 2-1 (Continued)

CROSS REFERENCE OF OBJECT TYPES AND NDDL COMMANDS

OBJECT TYPE	NDDL COMMAND
ENTITY CLASS	ALTER ENTITY
	ALTER MAP
	ALTER RELATION
	COMBINE ENTITY
	COPY DESCRIPTION
	COPY ENTITY
	CREATE ALIAS
	CREATE ENTITY
	CREATE MAP
	CREATE RELATION
	CREATE VIEW
	DESCRIBE ENTITY
	DROP ALIAS
	DROP ENTITY
	DROP MAP
	DROP RELATION
	RENAME ENTITY
	ALTER MAP
	CREATE MAP
	DEFINE RECORD
	DEFINE SET
	DROP FIELD
FILE	COMBINE ENTITY
	COPY ATTRIBUTE
	COPY ENTITY
	COPY MODEL
	DEFINE DATABASE
	DESCRIBE ATTRIBUTE
	DESCRIBE ENTITY
	DESCRIBE RELATION
	MERGE MODEL
KEYCLASS	ALTER ENTITY
	ALTER RELATION
	CREATE ENTITY
	CREATE RELATION
KEY FIELD	DEFINE RECORD

TABLE 2-1 (Continued)

CROSS REFERENCE OF OBJECT TYPES AND NDDL COMMANDS

OBJECT TYPE	NDDL COMMAND
KEYWORD	ALTER ATTRIBUTE
	ALTER ENTITY
	ALTER RELATION
	COMBINE ENTITY
	COPY ATTRIBUTE
	COPY ENTITY
	COPY MODEL
	CREATE ATTRIBUTE
	CREATE ENTITY
	CREATE RELATION
	DROP KEYWORD
	MERGE MODEL
	RENAME KEYWORD
MAP	ALTER MAP
	DROP MAP
MODEL	ALTER MODEL
	CHECK MODEL
	COMBINE ENTITY
	COMPARE MODEL
	COPY ATTRIBUTE
	COPY DESCRIPTION
	COPY ENTITY
	COPY MODEL
	CREATE MODEL
	DROP MODEL
	MERGE MODEL
	RENAME MODEL
PASSWORD	DEFINE DATABASE
PATH	DEFINE SET
	DROP SET

TABLE 2-1 (Continued)

CROSS REFERENCE OF OBJECT TYPES AND NDDL COMMANDS

OBJECT TYPE	NDDL COMMAND
PCB	DEFINE DATABASE DEFINE RECORD DEFINE SET DROP DATABASE DROP FIELD DROP RECORD DROP SET
PSB	DEFINE DATABASE
RECORD	ALTER MAP CREATE MAP DEFINE RECORD DEFINE FIELD DEFINE SET DROP FIELD DROP RECORD
RELATION CLASS	ALTER MAP ALTER RELATION COPY DESCRIPTION CREATE MAP CREATE RELATION CREATE VIEW DESCRIBE RELATION DROP MAP DROP RELATION RENAME RELATION
SCHEMA	DEFINE DATABASE
SEGMENT	DEFINE RECORD DROP FIELD DROP RECORD

TABLE 2-1 (Continued)

CROSS REFERENCE OF OBJECT TYPES AND NDDL COMMANDS

OBJECT TYPE	NDDL COMMAND
SET	ALTER MAP
	ALTER RELATION
	CREATE MAP
	CREATE RELATION
	DEFINE SET
	DROP SET
SUBSCHEMA	DEFINE DATABASE
TABLE	DEFINE RECORD
	DROP FIELD
TAG NAME	ALTER MAP
	ALTER RELATION
	CREATE MAP
	CREATE RELATION
	CREATE VIEW
	DROP MAP
VIEW	CREATE VIEW
	DROP VIEW
	RENAME VIEW

SECTION 3

NDDL COMMANDS

This section describes each NDDL command. The syntax of each command is given followed by command semantics and command examples.

3.1 ALTER ALIAS

Syntax:

```
ALTER ALIAS { ENTITY ec_name1      [IS] ec_name2
            | ATTRIBUTE ac_name1 [IS] ac_name2  };
```

Comments:

- The following elements must exist in the Common Data Model:

  entity or attribute with both primary and alias names

- The command switches the current primary and the alias names.

- The current primary names are ec_name1 and ac_name1; the current aliases are ec_name2 and ac_name2.

Examples:

```
ALTER ALIAS ENTITY CUSTOMER IS CUST ;
ALTER ALIAS ATTRIBUTE ORDER_NUMBER  ORDNO ;
```

3.2 ALTER ATTRIBUTE

Syntax:

```
ALTER ATTRIBUTE [CLASS] ac_name [DOMAIN domain_name]
     [DROP KEYWORD keyword...]
     [ADD KEYWORD keyword...];
```

Comments:

- The following elements must exist in the Common Data Model:

  model
  attribute

- ALTER ATTRIBUTE [CLASS] ac_name

  The attribute is altered with one of the options specified.

- [DOMAIN domain_name]

  When a domain_name is specified the validity of the Domain is checked and processing will halt if it is invalid.

  If the domain_name is valid, ATTRIBUTE_CLASS ac_name will be updated to indicate the new domain_no.

- DROP KEYWORD keyword...

  Keyword references are deleted.

- ADD KEYWORD keyword...

  The keyword(s) is created if it does not already exist.

  The attribute keyword reference(s) is created.

Examples:

```
ALTER ATTRIBUTE CLASS AC_ORDER_INFO DOMAIN ADDRESS
     ADD KEYWORD ZIP
     DROP KEYWORD COUNTY;
```

3.3 ALTER DOMAIN

Syntax:

```
ALTER DOMAIN domain_name
     [ ADD [ DATA ] TYPE
       data_type_name / INTEGER   integer1[:integer2] \  ]
                      | CHARACTER                     |
                    <   SIGNED                         >
                      | FLOAT                         |
                      | UNSIGNED                      |
                      \ PACKED                        /
     [DROP [ DATA ] TYPE data_type_name  ...]
     [ ALTER [ DATA ] TYPE
       data_type_name / INTEGER   integer1[:integer2] \ ]
                      | CHARACTER                     |
                    <   SIGNED                         >
                      | FLOAT                         |
                      | UNSIGNED                      |
                      \ PACKED                        /
          [ TO STANDARD ]];
```

Comments:

- The following elements must exist in the Common Data Model:

  domain
  data type

- Integer1 [integer2]... states a range of permissable values where integer1 is the ending value and integer2 is the beginning value (integer2 cannot be greater than integer1). Integer 1 can be used to specify a single value for a data type.

- The NDDL data types correspond to the following COBOL/FORTRAN data types:

NDDL Data Type	COBOL/FORTRAN Data Type
INTEGER	FORTRAN binary integer
CHARACTER	X(n)
SIGNED	S99V99
FLOAT	FORTRAN floating point
UNSIGNED	99V99
PACKED	COMP-3

- ADD [DATA] TYPE
  data_type_name / INTEGER integer1[:integer2] \
  | CHARACTER |
  < SIGNED >
  | FLOAT |
  | UNSIGNED |
  \ PACKED /

  The data type will be inserted as user defined.

- DROP {[DATA] TYPE data_type_name}...

  If it is determined that the data type is associated with a data field, data item or attribute class the data type will not be dropped. All attribute classes that use the data type are reported to the user.

  A standard data type cannot be dropped.

- ALTER [DATA] TYPE
  data_type_name / INTEGER integer1[:integer2] \
  | CHARACTER |
  < SIGNED >
  | FLOAT |
  | UNSIGNED |
  \ PACKED /
  [TO STANDARD]

  The data type is altered to "standard" (it will become the current standard), and the previous standard will become a user defined data type for the same domain.

  The data type may also be altered to another legal type with a new size and decimal specifications.

Examples:

```
ALTER DOMAIN ADDRESS
     ADD DATA TYPE ALPHA_NUMERC integer 5:2
     DROP DATA TYPE NUMERIC
     ALTER TYPE ALPHA character 30 to standard;
```

3.4 ALTER ENTITY

Syntax:

```
ALTER ENTITY [CLASS] ec_name
     [ADD [KEY [CLASS] kc_name[= ac_name ...]]...
       [[OWNED] ATTRIBUTE [CLASS] ac_name ...]
       [KEYWORD keyword ...]]
     [DROP [KEY [CLASS] kc_name ] ...
       [[OWNED] ATTRIBUTE [CLASS] ac_name ...]
       [KEYWORD keyword ...]];
```

Comments:

- The following elements must exist in the Common Data Model:

  entity
  attribute to be dropped or added
  key class to be dropped

- ADD KEY [CLASS] kc_name [=ac_name ...] ...

  An occurrence of attribute for ac_name must exist.

  A new occurrence of attribute use class is created for ac_name if one does not already exist. If the attribute use class does already exist, ac_name may not be owned by any other entity.

  A new occurrence of key class is created for the entity using kc_name.

  A new occurrence of key class member is created for the entity for each ac_name. If ac_name is omitted, a key class member occurrence is created using kc_name as the name of the attribute.

- ADD [OWNED] ATTRIBUTE [CLASS] ac_name ...

  New attributes for the entity being created are added.

  Each attribute is created as an owned attribute class for the entity.

  A new occurrence of attribute use class is created.

- ADD KEYWORD keyword ...

  The keyword references are created.

  The keyword will be created if it does not already exist.

- DROP KEY [CLASS] kc_name ...

  The key class for the entity is deleted.

  All key class members for the key class are deleted.

  If the key class being dropped is from a complete relation, the complete relation is deleted.

  All attribute use classes which were formed from a migration of that key class member and any migration of those attribute use classes are deleted.

  The inherited attribute use class occurrence of a migrated attribute use class is deleted.

- DROP [OWNED] ATTRIBUTE [CLASS] ac_name ...

  The owned attribute occurrence and the attribute use class for ac_name is deleted.

- DROP KEYWORD keyword ...

  The keyword reference is deleted.

Examples:

```
ALTER ENTITY CLASS EC_CUSTOMER
     ADD KEY CLASS KC_CUST_INFO = AC_CUST_NAME
       KEYWORD CUSTOMER;
```

3.5 ALTER MAP

Syntax:

```
ALTER MAP ec-name.tag-name

      [ ADD { TO FIELD database-name.record-name.datafield-name
        [ TYPE datatype-name ] } ...
            | { TO SET database-name.set-name VALUE
              'string'}...]
      [ DROP { TO FIELD database-name.record-name.datafield-
        name}..
            | { TO SET database-name.set-name}...]
      [ ALTER { TO FIELD database-name.record-name.datafield-
        name [(p)] [TYPE datatype-name]}...
              | { TO SET database-name.set-name VALUE
                'string'}...];

                                --or--

ALTER MAP ec-name rc-name ec-name

      [ ADD { TO SET database-name.set-name [.member-record-
        name]}..]

      [ DROP { TO SET database-name.set-name [.member-record-
        name]}...];
```

Comments:

- The following elements must exist in the Common Data Model:

  map to be altered
  datafield to be mapped to
  set to be mapped to

- Tag-name is a unique name for an attribute use class within an entity class. The following rules apply when altering Attribute Use Class (AUC) to datafield mappings:

  a. ALTER ADD rules are the following:

     - The AUC must not have previously been mapped to a set.
     - The AUC must not have been previously mapped to a data field.

- If the data type name is not entered, the standard data type for the AUC's domain is used.
- Only one primary mapping may exist for an AUC.
- Multiple secondary mappings may exist if there is a pre-existing primary mapping.
- If the primary or secondary mapping is not specified, the default is primary.

b. ALTER DROP will not drop a primary datafield map if secondary maps exist for a particular AUC.

c. ALTER ALTER will modify the primary-secondary indicator and/or the datatype name. To change a secondary map to primary and to change the previous primary map to secondary, include the (P) option. To change the datatype name, include the datatype name in the ALTER ALTER command.

- The following rules apply when altering AUC to set mappings:

a. ALTER ADD rules are the following:

- A data field mapping must not exist for the AUC.
- The set to be mapped to must have a single record type for its members.
- The member record name is not used in AUC to set mappings.
- The set to be mapped to must not previously have been mapped from a relation class or another AUC.
- All AUC to set maps must map to the same database for a particular AUC.
- All AUC to set maps must contain a value that must be unique for a particular AUC.
- Each set mapped to from the AUC must have the same record type as its owner.

b. No special rules apply to ALTER DROP.

c. ALTER ALTER will modify the AUC value for a particular AUC to set map. The new AUC value must be entered and must be unique for that particular AUC.

- The following rules apply when altering Relation Class (RC) to set member mapping:

a. ALTER ADD rules are the following:

- The member record name may be omitted if the set to be mapped to is a single member record type set; otherwise the member record name is required.
- There must be no previous mappings to the set.
- The value is not used in Relation Class to set mappings.

b. ALTER DROP requires a member record name entry if, by omitting it, the "to set" specification would be ambiguous.

c. ALTER ALTER is invalid for RC to set maps.

Examples:

```
ALTER MAP DEPARTMENT.EMPLOYEE
      ADD  TO  FIELD  EMPDB.EREC1.RATE
        TO  FIELD  EMPDB.EREC2.SSNO  TYPE SS
      DROP TO  FIELD  EMPDB.AREC.DEPTNO
      ALTER TO FIELD  EMPDB.EREC1.EMPNO (P)
        TO FIELD  EMPDB.EREC2.TITLE TYPE JOBTITLE;

ALTER MAP PART.COMPOSITION
      ALTER TO SET  MATERIAL.COMPOSITE VALUE 'COMPOSITE';

ALTER MAP INVENTORY HAS SHELFLIFE
      DROP TO SET  SLDB.BATTERY.NICAD;
```

3.6 ALTER MODEL

Syntax:

```
ALTER MODEL model_name;
```

Comments:

- Required existence in the Common Data Model:

  model class

- The model is updated and marked as unchecked.

- The model becomes the "current" model for all other modeling commands and remains current until another CREATE MODEL or ALTER MODEL command is entered.

Examples:

```
ALTER MODEL ABC_COMPANY;
```

3.7 ALTER RELATION

Syntax:

```
ALTER RELATION [CLASS] [{INTEGER1|MANY}] ind_ec_name1
  rc_name [{INTEGER2: INTEGER3|MANY} dep_ec_name2
     [ADD  [MIGRATES {kc_name [SET {tag_name1 =
       tag_name2}...]}] [KEYWORD keyword...]]
     [DROP [MIGRATES kc_name ]... [KEYWORD keyword...]];
```

Comments:

- The above command the following elements must exist in Common Data Model:

  model
  relation class
  independent entity (ind_ec_name1)
  dependent entity (dep_ec_name2)

- ALTER RELATION [CLASS] [INTEGER1|MANY] ind_ec_name1
  rc_name [INTEGER2: {INTEGER3|MANY}] dep_ec_name2

  The relation is altered if any cardinality is altered.

  Integer1 or many indicates the cardinality for the independent entity class. Integer2 and integer3 indicate the cardinality for the dependent entity class.

  Cardinality values default to the current cardinality for the Relation.

  A warning message may be generated to indicate either left or right dependent cardinality too large. This will not halt processing but defaults the value to the current cardinality of the relation class. The right dependent cardinality (integer2) cannot be less than the left dependent cardinality (integer3).

  Individual cardinalities may be changed without changing all the cardinalities.

  The right dependent (integer3) cardinality cannot be zero.

- [ADD [MIGRATES {kc_name [SET {tag_name1 = tag_name2}...]}
  [KEYWORD keyword...]]]

The following elements must exist in the Common Data Model:

key class for independent entity
key class members for the independent entity
attribute use class for each key class member associated with the independent entity

The key class cannot have been previously migrated to the dependent entity.

An attribute use and an inherited attribute use class for the dependent entity is created for each key class member of the independent entity migrated to the dependent entity.

If the set phrase is specified, tag_name1 (the independent entity's tag name) is migrated with the new name of tag_name1.

A complete relation class occurrence is created.

The keyword is created in the Common Data Model if it does not already exist.

A relation class keyword reference is created.

- [DROP [MIGRATES kc_name]... [KEYWORD keyword...]];

The key class migration will be dropped from each relation and entity in the model.

The relation class keyword reference is deleted.

Examples:

```
ALTER RELATION CLASS IND_EC_STORE RC_INVOICING
  DEP_EC_CUSTOMER
     ADD MIGRATES KC_ORDER SET AC_INVOICE = AC_FORM
     DROP KEYWORD KC_ORDER;

ALTER RELATION CLASS IND_EC_DEPARTMENT HAS
  DEP_EC_EMPLOYEE;
```

3.8 CHECK MODEL

Syntax:

```
CHECK MODEL model_name;
```

Comments:

- model_name must exist.
- If the model meets all rules, the model will be marked as checked.
- The following rules are checked for the model:

  a. no non-specific relations are allowed (independent cardinality greater than one)
  b. no incomplete relations (key has not been migrated)
  c. each entity has at least one attribute use class
  d. each owned attribute has a domain and that domain has a standard data type
  e. a key is defined for each entity
  f. multiple key classes of an entity are not subsets of one another
  g. no one to one relations
  h. no dependency loops, e.g., A->B->C->A.
  i. at least one entity exists in the model

- The following rules cannot be checked for the model:

  a. one to none or one relationships implying identical keys
  b. key uniqueness throughout the model is not checked, i.e., no two entities may have the same key unless they are related to each other with a one to none or one relation

Examples:

```
CHECK MODEL A;

CHECK MODEL INTEGRATED_MODEL;
```

3.9 COMBINE ENTITY

Syntax:

```
COMBINE ENTITY ec_name_1 [FROM MODEL model_name] INTO
  ec_name_2 ON FILE 'file_name'
    [EXCEPT [DESCRIPTION] [ALIAS] [KEYWORD]];
```

Comments:

- If model_name is not entered, an intra-model combine is assumed and ec_name_1 must not be the same as ec_name_2. The current model will be used.

- The NDDL statements necessary to physically combine the two entities are generated on the file named. If the named file does not exist, it is created and opened. If the named file does exist, the generated NDDL is appended to the file.

- On an intra-model combine, any relations between ec_name_1 and ec_name_2 will be dropped and ec_name_1 will be dropped.

- All owned attributes of ec_name_1 and their aliases, keywords, and descriptions are generated for ec_name_2, or the corresponding keyword appears in the EXCEPT clause.

- All entity name aliases, keywords, and descriptions of ec_name_1 are generated as aliases, keywords, and descriptions of ec_name_2 unless they already exist for ec_name_2, or the corresponding keyword appears in the EXCEPT clause.

- All relations in which ec_name_1 is the dependent entity are generated for ec_name_2, provided the independent entities in the relations already exist in the current model and the relation names are not already associated with ec_name_2. The key classes of the independent entities are migrated to ec_name_2 and all key classes of ec_name_1 are generated for ec_name_2.

- All relations in which ec_name_1 is the independent entity are generated for ec_name_2, provided the dependent entities already exist in the current model and the relation names are not already associated with ec_name_2. The key class of ec_name_1 is migrated to the dependent

entities.

- If the EXCEPT clause is entered, at least one option must be specified. If used, the options must appear in the order indicated. If DESCRIPTION is entered, no descriptions for entities, attributes, and relations will be generated or copied. If ALIAS is entered, no aliases for entities and attributes will be generated or copied. If KEYWORD is specified, no keywords for entities, attributes, and relations will be generated or copied.

- Generated Commands:

  The generated NDDL commands should be examined for potential run-time errors.

  a. If the entity being combined has inherited attributes, then the generated NDDL must be changed either to add the inherited attributes to the new entity as owned attributes, or all references to the inherited attributes must be deleted from KEY CLASS and MIGRATES clauses.

  b. A create/alter entity command may attempt to add an owned attribute when the attribute is already owned by another entity in the target model. The modeler must decide which entity should own the attribute and change the NDDL accordingly.

     NOTE:
     When an attribute is added to an entity and the attribute already exists in the target model, a comment is generated in the NDDL command following the attribute. The comment is:

     /* ATTRIBUTE MAY BE OWNED IN TARGET MODEL */

Examples:

```
COMBINE ENTITY ENT8 INTO ENT_B ON FILE 'ENTB.DAT' EXCEPT
  DESCRIPTION;

COMBINE ENTITY ENT_B FROM MODEL SPARKY INTO ENT_B ON FILE
  'CMBENT.ZZZ';
```

3.10 COMPARE MODEL

Syntax:

COMPARE MODEL model_name_1 WITH model_name_2;

Comments:

- model_name_1 and model_name_2 must both exist.
- Similarities, or points of correspondence of the two models will be reported:

  a. entity names correspond, (match identically) either through primary name or alias
  b. attribute names correspond, either through primary name or alias
  c. entity keywords correspond
  d. attribute keywords correspond
  e. relation keywords correspond

Examples:

COMPARE MODEL A WITH MODEL B;

3.11 COPY ATTRIBUTE

Syntax:

```
COPY ATTRIBUTE attr_name_1 [FROM MODEL model_name]
  [TO attr_name_2] [ON FILE 'file_name']
     [EXCEPT [DESCRIPTION] [ALIAS] [KEYWORD]];
```

Comments:

- If from model is used, attr_name_1 must exist in that model.
- If attr_name_2 is not entered, the copy attribute must be an inter-model copy and attr_name_2 will be the same as attr_name_1.
- The attribute will always be copied to the current model unless the FILE clause is specified.
- If from model has been omitted, copy attribute will be an intra-model copy and attr_name_2 must be entered.
- All of the attr_name_1's descriptions, aliases, and keywords will be copied, unless the corresponding appears in the EXCEPT clause.
- If the FILE clause is specified, NDDL commands will be generated/appended on the file specified.
- Refer to COMBINE ENTITY for a complete description of the EXCEPT clause.

Examples:

```
COPY ATTRIBUTE A1 FROM M1 TO A4;

COPY ATTRIBUTE A3 FROM M2 ON FILE 'A3.DAT';

COPY ATTRIBUTE A5 TO A6
     EXCEPT DESCRIPTION ALIAS;
```

3.12 COPY DESCRIPTION

Syntax:

```
COPY desc_type OF object_type object_id_1
  [FROM MODEL model_name] TO object_id_2;
```

Comments:

- Object_id can be an Attribute Use Class, Entity Use Class, or a Relation Class.
- This is a partial description copy. Only the description lines of the identified description type of the given object will be copied, rather than all description types.
- object_type and desc_type are defined in the DESCRIBE command.
- If model_name is omitted, the current model will be used when looking for object_id_1 and object_id_2 must not be the same as object_id_1.
- A description cannot be copied from an occurrence of one object type to a different object type.
- Object id's can consist of multiple words for relations, (ref DESCRIBE command).
- object_id_2 must exist in the current model.

Examples:

```
COPY DEFINITION OF ENTITY E1 TO E2;

COPY USAGE OF ENTITY E2 FROM MODEL M1 TO E2;

COPY DEFINITION OF RELATION E1 USES E3 TO E6 OWNS E7;
```

3.13 COPY ENTITY

Syntax:

```
COPY ENTITY [CLASS] ec_name_1 [FROM MODEL model_name]

  [TO ec_name_2] [WITH { STRUCTURE } ON FILE 'file_name']
                       { RELATION  }


      [EXCEPT [DESCRIPTION] [ALIAS] [KEYWORD]];
```

Comments:

- The following elements must exist in the Common Data Model:

  entity being copied
  current model

- If the FROM clause is omitted, an intra-model copy is assumed and the following rules apply:

  a. ec_name_2 must be entered and may not be the same as ec_name_1.
  b. The WITH clause must be omitted.
  c. A new entity is built for the current model.
  d. All keywords, aliases, and descriptions for the entity being copied are created for the new entity unless excepted.
  e. All key classes and key class members for the entity being copied are created for the new entity.
  f. Attributes associated with the entity are not copied.

- If the FROM clause is entered, an inter-model copy is assumed and the following rules apply:

  a. The entity being copied must not exist in the current model.
  b. If ec_name_2 is omitted the name of the new entity will be the same as ec_name_1.
  c. The WITH clause is required.
  d. The entity being copied and its relations or structures are not physically added to the current model. Instead, the NDDL commands necessary to add the entity to the current model are generated and written to the file named on the 'FILE' clause. If

the named file does not exist, a new file is created and opened. If the named file does exist, the generated commands are appended to the file.

e. A new entity is generated for the current model.

f. The following items are generated for the new entity unless they already exist in the current model or have been excepted:

- All keywords, aliases, and descriptions associated with the entity being copied.
- All the attributes owned by the entity being copied. Note that attributes inherited by the entity being copied are not generated.
- All key classes and key class members belonging to the entity being copied.

- If the RELATION option is chosen on an inter-model copy, the following applies:

  a. All relations in which the entity being copied is the dependent entity are generated for the current model, provided the independent entities in the relations exist in the current model.

  b. All relations in which the entity being copied is the independent entity are generated for the current model, provided the dependent entities in the relations exist in the current model.

  c. All key classes that were migrated in the relations being copied are migrated in the current model. The same names used in the relation being copied are generated for the current model.

- If the STRUCTURE option is chosen on an inter-model copy, the following applies:

  a. The tree structure dependent on the entity being copied is generated for the current model. This includes the entities, their keywords, aliases, descriptions, owned attributes, key classes and key class members; the relations, their migrated keys and set names. Note that everything associated with the structure is generated for the current model, whether or not it exists in the current model.

- Refer to COMBINE ENTITY for complete description of EXCEPT clause.

- **Generated Commands:**

  The generated NDDL commands should be examined for potential run-time errors.

  a. If the entity being copied has inherited attributes, then the generated NDDL must be changed either to add the inherited attributes to the new entity as owned attributes, or all references to the inherited attributes must be deleted from KEY CLASS and MIGRATES clauses.

  b. If a tree structure is being copied, then the create commands for any entities, attributes, aliases, and relations that already exist in the current model must be deleted.

  c. A create/alter entity command may attempt to add an owned attribute to an entity when the attribute is already owned by another entity in the target model. The modeler must decide which entity should own the attribute and change the NDDL accordingly.

     NOTE:
     When an attribute is added to an entity and the attribute already exists in the target model, a comment is generated in the NDDL command following the attribute. The comment is:

     /* ATTRIBUTE MAY BE OWNED IN TARGET MODEL */

Examples:

```
COPY ENTITY INVOICE TO NEWINVOICE
     EXCEPT DESCRIPTION ALIAS KEYWORD;

COPY ENTITY INVOICE FROM MODEL ABC_COMPANY WITH STRUCTURE
  ON FILE 'EXAMINE.NDL'
     EXCEPT ALIAS;
```

3.14 COPY MODEL

Syntax:

```
COPY MODEL [FROM MODEL model_name] TO new_model ON FILE
  'file_name' [EXCEPT [DESCRIPTION] [ALIAS] [KEYWORD]];
```

Comments:

- If the FROM clause is entered, model_name must exist. If it is omitted, the current model will be copied.
- new_model must not exist.
- The model being copied may not contain any dependency loops.
- NDDL commands are generated in the proper sequence to create a new model containing all the entities, owned attributes, inherited attributes, key classes, key class members, relations, aliases, keywords, and descriptions in the model being copied.
- Aliases, keywords, and/or descriptions will be excluded from the generated NDDL if the corresponding option appears in the EXCEPT clause. Refer to COMBINE ENTITY for a complete description of the EXCEPT clause.
- If the named file does not exist, a new file is created and opened. If the file does exist, the generated commands are appended to the file.

Examples:

```
COPY MODEL FROM MODEL M1 TO M2 ON FILE 'M2.DAT';
```

3.15 CREATE ALIAS

Syntax:

```
CREATE ALIAS { ENTITY ec_name1 [IS] ec_name2
             | ATTRIBUTE ac_name1 [IS] ac_name2 } ;
```

Comments:

- The following elements must exist in the Common Data Model:

  entity
  attribute

- ec_name2 and ac_name2 are the names to be created as aliases for entity and attribute respectively.

Examples:

```
CREATE ALIAS ENTITY CUSTOMER IS CUST;

CREATE ALIAS ATTRIBUTE ORDER_NUMBER IS ORDNO;
```

3.16 CREATE ATTRIBUTE

Syntax:

```
CREATE ATTRIBUTE [CLASS] ac_name [DOMAIN domain_name]
  [KEYWORD keyword ...];
```

Comments:

- CREATE ATTRIBUTE [CLASS] ac_name

  Required existence in the Common Data Model:

  the current model

  A new attribute is created for the current model.

- [DOMAIN domain_name]

  The domain_name must name an existing domain.

  If domain_name is not specified, the domain_no for the attribute_class will be undefined.

- KEYWORD keyword ...

  The keyword will be created in the Common Data Model if it does not already exist.

  Keyword references are created for the attribute.

Examples:

```
CREATE ATTRIBUTE ORDER_DATE DOMAIN DATE
  KEYWORD KEY_DATE;
```

3.17 CREATE DOMAIN

Syntax:

```
CREATE DOMAIN domain_name STANDARD
     [DATA] TYPE data_type_name / CHARACTER \
                                | PACKED    |
                               <  SIGNED    >
                                | UNSIGNED  |
                                \ INTEGER   /
        integer1[: integer2] ...;
```

Comments:

- A new domain is created within the system.
- All specified data types are added for the domain.
- A standard data type must be specified. Remaining data types for that domain will be user defined data types.
- Integer1 [integer2]... states a range of permissable values where integer1 is the ending value and integer2 is the beginning value (integer2 cannot be greater than integer1. Integer1 can be used to specify a single value for a data type.
- The NDDL data types correspond to the following COBOL/FORTRAN data types:

NDDL Data Type	COBOL/FORTRAN Data Type
CHARACTER	X(n)
PACKED	COMP-3
SIGNED	S99V99
UNSIGNED	99V99
INTEGER	FORTRAN binary integer
FLOAT	FORTRAN floating point

Examples:

```
CREATE DOMAIN ZIP_CODE standard
     TYPE ABC character 30
     TYPE XY2 integer 6:2;
```

3.18 CREATE ENTITY

Syntax:

```
CREATE ENTITY [CLASS] ec_name
     [KEY [CLASS] kc_name[= ac_name ...]]...
          [[OWNED] ATTRIBUTE [CLASS] ac_name ...]
               [KEYWORD keyword ...];
```

Comments:

- CREATE ENTITY [CLASS] ec_name

  A new entity is created for the current model.

- KEY [CLASS] kc_name [= ac_name ...]

  Required existence in the Common Data Model:

  attribute

  A new occurrence of attribute use class and owned attribute is created for ac_name if one does not already exist. If the attribute use class does not already exist, ac_name may not be owned by any other entity.

  A new occurrence of key class is created for the entity using kc_name.

  A new occurrence of key class member is created for the entity for each ac_name. If ac_name is omitted, a key class member occurrence is created using kc_name.

- [OWNED] ATTRIBUTE [CLASS] ac_name ...

  Required existence in the Common Data Model:

  attribute

  New attributes for the entity being created are added.

  Each attribute class is created as an owned attribute class for the entity.

  A new occurrence of an attribute use class is created.

- KEYWORD keyword...

Keyword references are created for the entity.

The keyword will be created if it does not already exist.

Examples:

```
CREATE ENTITY SALES_PERSON
     KEY ORDER = ORDER_NUMBER
          OWNED ATTRIBUTE SALES
               KEYWORD SALES_ID;
```

3.19 CREATE MAP

Syntax:

```
CREATE MAP { ec_name.tag_name
           | ind_ec_name1 rc_name dep_ec_name2}
     {{TO FIELD database_name.record_name.datafield_name
       [{(p) |(s)}] [TYPE datatype_name]} ...
     |{TO SET database_name.set_name [.member_record_name]
       [ VALUE 'string']}...};
```

Comments:

- The following elements must exist in the Common Data Model:

  Integrated Model (model number = 1)
  Attribute Use Class (AUC) to be mapped
  Relation Class to be mapped
  Data field mapped to
  Database mapped to
  Record mapped to
  Set mapped to

- The following rules apply to AUC (ec_name.tag_name) to datafield mappings:

  a. The AUC must not have previously been mapped to a set.
  b. The AUC must not have been previously mapped to a datafield.
  c. If the datatype name is not entered, the standard datatype for the AUC's domain is used.
  d. Only one primary mapping may exist for an AUC.
  e. Multiple secondary mappings may exist if there is a pre-existing primary mapping.
  f. If the primary or secondary mapping is not specified, the default is primary.

- The following rules apply to AUC to set mappings:

  a. A datafield mapping must not exist for the AUC.
  b. The set to be mapped to must have a single record type for its members.

  c. The member record name is not used in AUC to set mappings.

d. The set to be mapped to must not previously have been mapped from a relation class or another AUC.
e. All AUC to set maps must map to the same database for a particular AUC.
f. All AUC to set maps must contain a value which must be unique for a particular AUC.
g. All sets mapped to from the AUC must have the same record type as its owner.

- The following rules apply to Relation Class to set mappings:

  a. The member record name may be omitted if the set to be mapped to is a single member record type set; otherwise the member record name is required.
  b. There must be no previous mappings to the set.
  c. The value is not used in Relation Class to set mappings.

- A Relation Class to datafield map is illegal.

Examples:

```
CREATE MAP PART.SIZE      TO FIELD PARTDB.PREC.PSIZE
                          TO FIELD PARTDB.PREC2.PSIZE2 (S)
                            TYPE METRIC ;

CREATE MAP ORDER.STATUS   TO SET CUSTDB.INPROCESS VALUE '1'
                          TO SET CUSTDB.BAKORDER VALUE '2'
                          TO SET CUSTDB.SHIPPED VALUE '3';

CREATE MAP EMPLOYEE POSSESSES SKILLS
     TO SET SKILLDB.SETA.RECA
     TO SET SKILLDB.SETB ;
```

3.20 CREATE MODEL

Syntax:

```
CREATE MODEL model_name;
```

Comments:

- A new model is created as unchecked in the system.
- The model is the "current" model for following modeling commands until another CREATE MODEL or ALTER MODEL command is entered.

Examples:

```
CREATE MODEL DEF_COMPANY;
```

3.21 CREATE RELATION

Syntax:

```
CREATE RELATION [CLASS] [{INTEGER1|MANY}] ind_ec_name1
   rc_name [{INTEGER2: INTEGER3|MANY} dep_ec_name2
      [MIGRATES {kc_name [SET {tag_name1 = tag_name2}...]}]
      [KEYWORD keyword ...];
```

Comments:

- CREATE RELATION [CLASS] [{INTEGER1|MANY}] ind_ec_name1
  rc_name [{INTEGER2: INTEGER3|MANY} dep_ec_name2

  The following elements must exist in the Common Data Model:

  model
  independent entity (ind_ec_name1)
  dependent entity (dep_ec_name2)

  A new relation is inserted.

  Cardinalities for the entities of the relation are inserted. Integer1 or many indicates the cardinality for the independent entity class. Integer2 and integer3 indicate the cardinality for the dependent entity class. If integer1 or many is omitted, the independent cardinality defaults to one. If integer2 is omitted, the left dependent cardinality defaults to zero. If integer3 or many is omitted, the right dependent cardinality defaults to many. For example, if DEPARTMENT (ex_name1) can have 0 or many Employees (ec_name2):

  CREATE RELATION 1 DEPARTMENT HAS 0:MANY EMPLOYEES

  - or -

  CREATE RELATION DEPARTMENT HAS EMPLOYEE

  Integer3 may not be less than integer2.

  Integer3 may not be zero.

- MIGRATES kc_name [SET {tag_name1 = tag_name2} ...]

The following elements must exist in the Common Data Model:

key class for independent entity
key class members for independent entity
attribute use class for each key class member associated with the independent entity

The key class cannot have been previously migrated.

The dependent entity cannot end up with attribute use classes with the same tag name (a unique name for an attribute use class within an entity class).

An attribute use class and an inherited attribute use class for the dependent entity are created for each key class member of the independent entity migrated to the dependent ENTITY.

If the set phrase is specified, tag_name2 (the independent entity's tag name) is migrated with the new name of tag_name1.

A complete relation class occurrence is created.

- KEYWORD keyword ...

  The keyword is created if it does not exist.

  A relation keyword reference is created.

Examples:

```
CREATE RELATION INVOICE SUPPLIES ORDER_INFO
     MIGRATES ORDER SET ORDER_NUMBER = ORDNO
     KEYWORD ORDER_DATA;
```

3.22 CREATE VIEW

Syntax:

```
CREATE VIEW view_name [DATA ITEM data_item_name
  [[DATA] TYPE data_type_name]]..
AS SELECT{|*
     tag_name ...|
     [abbrev.]tag_name...}
FROM{ec_name
    |ec_name.abbrev...}
[WHERE ind_ec_name rc_name dep_ec_name [and...]]
```

Comments:

- To create a VIEW (SURROGATE ENTITY CLASS) using the above command, the following elements must exist in the Common Data Model:

  independent entity
  dependent entity
  relation class
  entity
  data items and attribute use classes from the same domain existence of standard data type

- CREATE VIEW view_name

  A new view (SURROGATE ENTITY CLASS) is created if it does not exist.

  The relationship between view and relation is created.

- DATA_ITEM data_item_name [[DATA] TYPE data_type_name]...

  Must be specified when views are built from more than one entity.

  The data items must correspond in order and number to the attribute use class tags specified in the select clause.

  If data type name is not specified, the data type will become the standard data type for the corresponding attribute use class.

- AS SELECT{*|
     tag_name...|

[abbrev.]tag_name

If an * is specified:

a. The FROM clause may not be omitted.
b. A data item and project data item occurrence are created for each attribute use class associated with the entity named on the FROM clause.

If the tags (unique attribute use classes within an entity class) are selected from a single entity, the ABBREV may be omitted provided a FROM clause is specified without an abbreviation.

If tags are selected from more than one entity, both abbreviations and tag names must be specified and a FROM clause with entity names and abbreviations must be specified.

Data item and project data item occurrences are created for each attribute use class named in the select.

If abbreviations are used, they must be used for all attribute use classes.

- FROM {ec_name
  |ec_name.abbrev}...

  The abbreviation for the entity must be included if the ABBREV is used in the select clause.

- WHERE {ind_ec_name rc_name dep_ec_name [and]}...

  The WHERE clause must be specified when tags are selected from more than one entity.

  Each entity specified in the FROM clause must appear as either the independent entity or dependent entity in a WHERE clause relation.

  The relation class name must be valid.

  Abbreviations are not allowed in the WHERE clause.

  The WHERE clause selects those relation classes that join the entities into a "surrogate entity class" structure. The structure is checked using the rules found in the

CDM-1 model, June 1983.

Examples:

```
CREATE VIEW SUPPLIES DATA ITEM   PONUM PODATE SUPPLIER
  LOCATION DATADEL REQUESTOR

AS SELECT PO.PORDER_NUMBER PO.PORDER_DATE PO.ITEM_SUPPLIER
  DL.DELIVERY_LOCATION DL.DELIVERY_LOCATION
  DL.DATE_OF_DELIVERY MR.REQUESTING_SEC

FROM   PURCHASE_ORDER.PO DELIVERY.DL MATERIAL_REQUEST.MR

WHERE   PURCHASE_ORDER SPECIFIES DELIVERY AND
  MATERIAL_REQUEST REQUESTS PURCHASE_ORDER;
```

3.23 DEFINE DATABASE

Syntax:

```
DEFINE  / ORACLE \   { DATABASE } NAMED db_name ON HOST{ IBM }
        | TOTAL  |   { PCB      }                      { VAX }
       <  IDMS   >                                     { HL6 }
        | IDS-II |
        | VAX-11 |
        \ IMS    /
    WITH{ [PASSWORD  pw_id ]
          [SCHEMA s_name AND SUBSCHEMA ss_name AREAS
             area_name...]
          [FILES  file_name...]
          [POSITION integer1 IN PSB psb_name FEEDBACK
             LENGTH integer2]};
```

Comments:

- The database name must be unique.
- A database is defined for one of the following types: ORACLE, TOTAL, IDMS, IDS-II, VAX-11 or IMS.
- The database created is established as the current database until such time the user defines another database.
- The three choices of host are not meant to be exclusive. The user may enter any valid three character NTM host designator, uppercase. The values are stored and not checked.
- Oracle Database:

```
DEFINE ORACLE DATABASE NAMED db_name ON HOST  { IBM }
                                              { VAX }
                                              { HL6 }

       WITH PASSWORD  pw_id;
```

A password is required.

- Total Database:

```
DEFINE TOTAL DATABASE NAMED db_name ON HOST  { IBM }
                                             { VAX }
                                             { HL6 }

  WITH FILES file_name... ;
```

At least one file is required.

- VAX-11/IDMS/IDS-II database:

```
DEFINE { VAX-11 } DATABASE NAMED db_name ON HOST { IBM }
       { IDMS   }                                { VAX }
       { IDS-II }                                { HL6 }
     WITH SCHEMA s_name AND SUBSCHEMA ss_name AREAS
       area_name...;
```

The schema, subschema, and at least one area are required.

- IMS database:

```
DEFINE IMS PCB NAMED db_name ON HOST  IBM  WITH POSITION
  integer1 IN PSB  psb_name FEEDBACK LENGTH integer2
```

An IMS database requires the term 'pcb'.

The start position and feedback length (the length of the fully concatenated key of the longest path in the PCB) are required.

Examples:

```
DEFINE ORACLE DATABASE NAMED ORC_DB ON HOST VAX WITH
  PASSWORD SOS;

DEFINE TOTAL DATABASE NAMED TOT_DB ON HOST IBM WITH FILES
  ITEM LOCC STOK ;

DEFINE VAX-11 DATABASE NAMED COD_DB ON HOST VAX WITH
  SCHEMA S1 AND SUBSCHEMA SS1 AREAS A1 A2 ;

DEFINE IMS PCB NAMED IMS_DB ON HOST IBM WITH POSITION 1 IN
  PSB SS_PSB FEEDBACK LENGTH 20 ;
```

3.24 DEFINE RECORD

Syntax:

```
DEFINE  {  TABLE   }   rec_name  [ OF { DATABASE }db_name]
        {  RECORD  }                 { PCB      }
        {  SEGMENT }

     { [IN AREAS  area_name ... ]
          [SEGMENT SIZE  integer1 ]
                    /         \
          WITH     |  FIELDS   |    field_name ...
                  <   COLUMNS   >
                   |  ELEMENTS |
                   |  ITEMS    |
                    \         /
          [{START integer2  [UNKNOWN] } ... ]
     [ ACCESSED {  BY KEY   key_name [(u)] key_field_name
       ...} ...]};
```

Comments:

- This command defines a table/record/segment for a previously defined database.
- The "OF DATABASE/PCB db_name" option may be omitted if a current database is established in the session.
- A key_field_name must have been previously entered as a field_name.
- (U) on the key_field_name enforces a unique key. Default is a non-unique key.
- This syntax does not support repeating groups, component data fields, or redefined data fields.
- Oracle Records:

```
DEFINE  TABLE  rec_name [ OF DATABASE db_name]
           /          \
     WITH  | COLUMNS   |  field_name... ;
          <  FIELDS    >
           | ELEMENTS |
           | ITEMS    |
           \          /
```

field_name is the only required option.

- Total Records:

```
DEFINE  RECORD   rec_name [ OF DATABASE db_name]
              /           \
     WITH   | FIELDS    |  field_name ... [ ACCESSED {BY
            <  COLUMNS   >
              | ELEMENTS |
              | ITEMS    |
              \           /

       KEY key_name [(u)] key_field_name ...} ...] ;
```

FIELD_NAME is required.

ACCESSED BY clause is required for "MASTER" records.

- VAX-11/IDMS/IDS-II Records:

```
DEFINE   RECORD   rec_name  [ OF DATABASE db_name]
                                  /           \
     IN AREAS area_name...  WITH  | FIELDS    |
                                 <  COLUMNS    >
                                  | ELEMENTS |
                                  | ITEMS    |
                                  \           /
       field_name... [ACCESSED { BY KEY key_name [(u)]
       key_field_name...}...];
```

At least one area and one field are required.

- IMS Records:

```
DEFINE  SEGMENT  rec_name  [ OF PCB db_name]
     SEGMENT SIZE integer1 WITH  ELEMENTS
       field_name
     START integer2  [UNKNOWN]  ... ;
```

Segment size and start byte information are required.

Examples:

```
DEFINE TABLE TAB_OR OF DATABASE ORC_DB WITH COLUMNS C1 C2
```

```
  C3 C4 ;

DEFINE RECORD REC_TOT WITH FIELDS ITEMCTRL ITEMI111
  ITEMI222 ACCESSED BY KEY ITEMKEY1 (U) ITEMCTRL BY KEY
  ITEMKEY2 ITEMI111 ITEMI222 ;
/* assumes a current TOTAL database has been established
  in the session */

DEFINE SEGMENT SEG1 OF PCB IMS_DB SEGMENT SIZE 20 WITH
  ELEMENTS SI_ID START 1 SI_QTY START 11 ;
DEFINE RECORD REC1 OF DATABASE COD_DB IN AREAS A1 A2
  WITH FIELDS ITEM_ID ITEM_QTY ACCESSED BY KEY REC1KEY1
  ITEM_ID ;
```

3.25 DEFINE SET

Syntax:

```
DEFINE { SET  }  [setname]  OF { DATABASE } data_base_name
       { PATH }                { PCB      }

   {  RELATING }  record_id1 TO {record_id2
   {  FROM     }

     [REQUIRED|OPTIONAL]}...
    [LINKED BY data_field_name] ;
```

Comments:

- The following elements must exist in the Common Data Model:

  database
  record type
  data field

- The above command is illegal for an Oracle database.

- The following rules govern the creation of a set/path for a DBMS type:

  a. Sets may be created for TOTAL, IMS (called paths), and the Codasyl DBMSs: VAX-11, IDMS, and IDS.
  b. Codasyl DBMSs require an entry for required/optional members.
  c. TOTAL DBMSs require a linked by Data Field Name clause.
  d. Single members are allowed in all DBMSs. Multiple members (Record_List_ID) may be used for the CODASYL DBMSs.

- DEFINE { SET } [setname] OF { DATABASE } data_base_name
  { PATH } { PCB }

  The set or path is a required entry for TOTAL and CODASYL DBMSs. Path is used as a more natural entry for an IMS database. For an IMS database, if "path" is used, the set name is derived from combining the owner (Record_ID1) and

member (Record_ID2) names.

The Data_Base_Name or pcb is an optional entry if a Data_Base_Name or pcb has previously been established during the session using a define database, define record, or define set command. The use of pcb is more natural for an IMS database.

- { RELATING } record_id1 TO {record_id2
  { FROM }
  [REQUIRED|OPTIONAL]}...

  The RELATING/FROM clause is a required entry. From is more natural with an IMS database. Record_ID1 denotes the owner of the set and Record_ID2 denotes the member(s) of the set. Record_ID2 is used for Codasyl DBMSs to show multiple members and includes the member Record_IDs separated by a space. The required or optional clause may only be used for the Codasyl DBMSs. All other DBMSs default to required.

- [LINKED BY data_field_name]

  The LINKED BY clause may only be used for a TOTAL DBMS and must include a Data_Field_Name from the variable record (Record_ID2).

Examples:

```
DEFINE SET MAY_HAVE OF DATABASE DB1 RELATING CUSTOMER_REC
  TO CUSTOMER_ACTIVITY ;

DEFINE PATH OF PCB DBIMS2 FROM SUPPLIER_ACCT TO
  SUPPLIER_INVOICE ;

DEFINE SET CONTROLLED_BY OF DATABASE  DBTOTAL14 RELATING
  SHOP_SUPPLY TO SHOP_REQUESTS LINKED BY SHOP_NO ;
```

3.26 DESCRIBE

Syntax:

```
          /                    \      /                          \
DESCRIBE | DEFINITION           | [OF] | ENTITY ec_name           |
         <  DESCRIPTIVE_NAME    >      < ATTRIBUTE ac_name         >
          | EXAMPLE              |      | RELATION ind_ec_name1 |
          |  SOURCE              |      |  rc_name dep_ec_name2 |
          \                    /      \                          /
  [ "DESCRIPTION TEXT"
  | FROM 'file name'  ];
```

Comments:

- The following elements must exist in the Common Data Model:

attribute	database	data field
data item	entity	record set
record type	relation	user datatype
view		

- There are three sources of descriptive text: the command itself, a named file, or the forms text editor. The third option is not implemented and will generate a user warning message if attempted.

- All new descriptive text will replace any preexisting descriptive text for a particular description type and object.

- To delete descriptive text from the model, use the describe command with an empty description text string, e.g.,

  DESCRIBE SOURCE OF ENTITY E3 ";

- No embedded quotation marks may be entered in the description text string.

- When using a file as the source of the descriptive text, each record may be no longer than 80 characters. There is no practical limit on the number of records. Each quotation mark in the descriptive text from a file will be replaced by an apostrophe.

- The description types can be anything preloaded in the CDM.

Examples:

```
DESCRIBE DEFINITION OF ATTRIBUTE customer_address "This is
  the customer bill-to-address.";

DESCRIBE DESCRIPTIVE_NAME ENTITY emp_info FROM
  'empinfo.txt';
```

3.27 DROP ALIAS

Syntax:

```
DROP ALIAS  { ENTITY ec_name
            | ATTRIBUTE ac_name } ;
```

Comments:

- The following elements must exist in the Common Data Model:

  alias for the entity

  - or -

  alias for the attribute

- The ec_name or ac_name specified must be an alias for the entity or attribute.

- The specified alias is deleted.

Examples:

```
DROP ALIAS ENTITY CUST;

DROP ALIAS ATTRIBUTE ORDNO;
```

3.28 DROP ATTRIBUTE

Syntax:

```
DROP ATTRIBUTE attribute_name ...;
```

Comments:

- The following elements must exist in the Common Data Model:

  model
  attribute

- The attribute is deleted; if owned, all occurrences of the attribute are removed from owned attribute, attribute use class, key class member, and inherited attribute use class.

- The attribute occurrence is deleted from the model.

- Those key class occurrences with no remaining key class members are deleted.

- If a key class is deleted, the occurrence of a complete relation is also deleted.

- All keywords associated with the attribute will be dropped.

- The primary name and all aliases for the attribute will be deleted.

- All description texts for the attribute will be deleted.

Examples:

```
DROP ATTRIBUTE PHONE_NO;
```

3.29 DROP DATABASE

Syntax:

```
          /          \
DROP      | VAX-11   |    { DATABASE }    NAMED db_name;
          | IDMS     |    { PCB      }
          | IDS-II  |
          < ORACLE   >
          | TOTAL    |
          | IMS      |
          \          /
```

Comments:

- The following elements must exist in the Common Data Model:

  database/pcb

- An IMS database requires the term 'PCB.'

- This command drops the database, all associated record types, record sets, data fields, and mappings.

- If any of the data fields for this database were mapped to the INTEGRATED_MODEL, their mapping will be dropped. If it was a primary mapping, any secondary mappings, even to other databases, will also be dropped. If it was a secondary mapping, the primary mapping would not be dropped.

- All description texts for the database, all associated sets, fields, or records will be deleted.

Examples:

```
DROP ORACLE DATABASE NAMED ORC_DB;

DROP TOTAL DATABASE NAMED TOT_DB;

DROP VAX-11 DATABASE NAMED COD_DB;

DROP IMS PCB NAMED IMS_DB;
```

3.30 DROP DOMAIN

Syntax:

```
DROP DOMAIN domain_name ...;
```

Comments:

- The following elements must exist in the Common Data Model:

  domain

- If data types of the domain to be dropped are associated with any project data fields, data items, or attribute classes, the usage will be reported and the domain will not be dropped.

- If the above condition (2) is not met, the data types associated with the domain are deleted, and the domain itself is deleted.

- All description texts will be deleted for the Domain.

Examples:

```
DROP DOMAIN ADDRESS;
```

3.31 DROP ENTITY

Syntax:

```
DROP ENTITY ec_name ...;
```

Comments:

- The following elements must exist in the Common Data Model:

  entity
  model

- Any owned attribute class occurrences are deleted. Also removed are attribute use, inherited attribute, key class member, and key classes.

- All relation classes involving the entity are deleted, as are any keywords associated wth the entity.

- The entity is deleted.

- The primary name and all aliases for the entity are deleted.

- All description texts for the entity are deleted.

Examples:

```
DROP ENTITY ORDER_INFO;
```

3.32 DROP FIELD

Syntax:

```
          /            \
    DROP  |  COLUMNS   |  field_name ...  OF { TABLE   }rec_name
          <  FIELDS    >                     { RECORD  }
          |  ELEMENTS  |                     { SEGMENT }
          |  ITEMS     |
          \            /
      OF   { DATABASE }   db_name;
           { PCB      }
```

Comments:

- The following elements must exist in the Common Data Model:

  columns/fields/elements/items
  table/record/segment
  database/pcb

- This command deletes the field(s) specified, all associations, and all associated mappings.

- If the data field(s) being dropped was mapped to the INTEGRATED_MODEL, its mapping would be dropped. If it was a primary mapping, any secondary mapping, even to other databases, would also be dropped. If it is a secondary mapping, the primary mapping would not be dropped.

- All description texts of the data field will be deleted.

Examples:

```
DROP COLUMNS C1 C2 OF TABLE TAB_OR OF DATABASE ORC_DB;

DROP ELEMENTS SI_ID SI_QTY OF PCB IMS_DB;
```

3.33 DROP KEYWORD

Syntax:

```
DROP KEYWORD key_word ...;
```

Comments:

- The following elements must exist in the Common Data Model:

  keyword

- The keyword reference is deleted from any entity, attribute, and relation.

- The keyword is deleted.

- NOTE:
  This command will drop a keyword irrespective of the model.

Examples:

```
DROP KEYWORD KEY_ADDRESS;
```

3.34 DROP MAP

Syntax:

```
DROP MAP {ec_name.tag_name
          |ind_ec_name rc_name dep_ec_name};
```

Comments:

- The following elements must exist in the common data model:

  independent entity
  dependent entity
  attribute use class
  relation class

- The map is deleted.

- Tag_name is a unique name for an attribute use class within an entity class.

Examples:

```
DROP MAP PART.SIZE;

DROP MAP EMPLOYEE POSSESSES SKILL;
```

3.35 DROP MODEL

Syntax:

```
DROP MODEL model_name;
```

Comments:

- Model_name must exist.
- Everything associated with a model will be dropped. This is: all entities, owned attributes, attribute use classes, key classes, key class members, inherited attribute use classes, relation classes and complete relation classes, and all attributes. The descriptions, aliases, and keywords for the entities, attributes, and relations of the model will be dropped.
- The model_name and its descriptions will be deleted.
- The integrated_model cannot be dropped.

Examples:

```
DROP MODEL X;
```

3.36 DROP RECORD

Syntax:

```
DROP  { TABLE  }    rec_name  OF { DATABASE }   db_name ;
      { RECORD }                 { PCB      }
      { SEGMENT }
```

Comments:

- The following elements must exist in the Common Data Model:

  table/record/segment
  database/pcb

- This command deletes the record specified and all associated fields, areas, and sets.

- If any of the datafields for this record were mapped to the integrated_model, their mapping will be dropped. If it was a primary mapping, any secondary mapping, even to other databases, will also be dropped. If it was a secondary mapping, the primary mapping would not be dropped.

- All description texts will be deleted for the record/table/segment and any fields.

Examples:

```
DROP TABLE TAB_OR OF DATABASE ORC_DB;

DROP SEGMENT SEG1 OF PCB IMS_DB;
```

3.37 DROP RELATION

Syntax:

```
DROP RELATION { ind_ec_name_1 rc_name dep_ec_name_2 } ...;
```

Comments:

- The following elements must exist in the Common Data Model:

  relation class
  model

- If a key class had been migrated, it is "unmigrated"; i.e., the attribute use and inherited attributes are removed from the dependent entity and from any other entities to which they have migrated.

- The relation class and complete relation are deleted from the model.

- The keywords associated with the relation are dropped.

- All description texts for the relation are deleted.

Examples:

```
DROP RELATION SALES_PERSON MANAGES ACCOUNT;
```

3.38 DROP SET

Syntax:

```
DROP   { SET  }   set_name  OF { DATABASE }   db_name;
       { PATH }                { PCB      }
```

Comments:

- The following elements must exist in the Common Data Model:

  set/path
  database/pcb

- This command deletes the set specified, and all associated mappings.

- All description texts for the set will be deleted.

Examples:

```
DROP SET TEST1 OF DATABASE TOT_DB;

DROP PATH TEST2 OF PCB IMS_DB;
```

3.39 DROP VIEW

Syntax:

```
DROP VIEW view_name ...;
```

Comments:

- A valid VIEW must exist in the Common Data Model.
- All project data items associated with the view being deleted are deleted.
- The surrogate entity (user view) to relation mappings associated with the view are deleted.
- All data items associated with the view are deleted.
- The view is deleted from the system.
- All description texts for the view and data items belonging to the view are deleted.

Examples:

```
DROP VIEW SUPPLIES MATERIAL_LIST ;
```

3.40 HALT

Syntax:

```
HALT;
```

Comments:

- The current NDDL session will be terminated.

Examples:

```
HALT;
```

3.41 MERGE MODEL

Syntax:

```
MERGE MODEL model_name_1 WITH model_name_2
     [INTO model_name_3] ON FILE 'file_name'
     [EXCEPT [DESCRIPTION] [ALIAS] [KEYWORD]];
```

Comments:

- model_name_1 and model_name_2 must exist.

- The NDDL commands necessary to merge model_name_1 and model_name_2 are generated on the file named. If the named file does not exist, a new file is created and opened. If the named file does exist, the generated commands are appended to the file.

- If model_name_3 is entered, model_name_3 must not exist. The result of the merge will create model_name_3. All attributes, entities, relations, key classes, aliases, keywords, and descriptions of model_name_1 will be generated for model_name_3, unless excepted. The output is the same as output from a COPY MODEL of model_name_1 to model_name_3.

- If model_name_3 is omitted, the result of the merge will alter model_name_1.

- For each entity in model_name_2, if the entity does not exist in model_name_1, the generated commands are the same as the output of a COPY ENTITY from model_name_2 with relation.

- For each entity in model_name_2, if the entity does exist in model_name_1, the generated commands are the same as the output of a COMBINE ENTITY from model_name_2.

- Refer to COMBINE ENTITY for a complete description of the EXCEPT clause.

- Generated Commands:

  The generated NDDL commands should be examined for potential run-time errors.
  A create/alter entity command might attempt to add an owned attribute to an entity when the attribute is already

owned by another entity in the target model. The modeler must decide which entity should own the attribute and change the NDDL accordingly.

NOTE:
When an attribute is added to an entity and the attribute already exists in the target model, a comment is generated in the NDDL command following the attribute. The comment is:

```
/* ATTRIBUTE MAY BE OWNED IN TARGET MODEL */
```

Examples:

```
MERGE MODEL INTEGRATED_MODEL WITH MODEL_A ON FILE
  'INTEGMOD.FIL';

MERGE MODEL MODEL_A WITH MODEL_B INTO NEW_MODEL
  ON FILE 'NEWMODEL.FIL' EXCEPT ALIAS KEYWORD;
```

3.42 RENAME

Syntax:

```
                /                                                    \
RENAME          | ENTITY      ec_name1      TO    ec_name2           |
                | ATTRIBUTE   ac_name1      TO    ac_name2;          |
                | KEYWORD     keyword1      TO    keyword2           |
                < MODEL       model_name1   TO    model_name2        >
                | DOMAIN      dom_name1     TO    dom_name2          |
                | VIEW        view_name1    TO    view_name2         |
                | RELATION    ind_ec_name1 rc_name1 dep_ec_name2     |
                |             TO  ind_ec_name1 rc_name2              |
                |             dep_ec_name2                           |
                \                                                    /
```

Comments:

- The following elements must exist in the Common Data Model:

  entity or attribute or keyword or model or domain or view or relation

- After determining that the above elements exist, and the element to be renamed does not previously exist, the object is updated.

- Rename relation command requires the entity class names for the relation class to be updated.

Examples:

```
NAME ENTITY INVOICE to ORDER;

RENAME RELATION INVOICE SUPPLIES ORDER_INFO  to
  INVOICE DESCRIBES ORDER_INFO;
```

APPENDIX A

NDDL COMMAND ERROR MESSAGES

This section describes NDDL Command Error messages that may be encountered during NDDL execution. These error messages inform the user about error conditions for the object type (Entity, Attribute, Database) being processed. For example, the command "DROP MODEL" may encounter error messages concerning all objects associated with the model (Entities, Attributes, Relations, etc).

The error message report contained in this section is formatted to indicate the type of error (warning, error, fatal), the object type that is associated with the error, and the actual error message (with the object indentification) that will be issued during NDDL execution.

The error types are categorized as:

FATAL - A system or CDM DBMS error that caused the process to fail or be interrupted and stopped.

ERROR - A condition that will not allow the NDDL command to process as expected.

WARNING - An informational message issued to the user based on a nonfatal or nonprocess interrupting condition.

ALTER ALIAS

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	ATTRIBUTE <AC-NAME> DOES NOT EXIST
	ATTRIBUTE CLASS	ALIAS <ALIAS-ID> DOES NOT EXIST FOR ENTITY
	ATTRIBUTE CLASS	ALIAS <ALIAS-ID> DOES NOT EXIST FOR ATTRIBUTE
	ENTITY CLASS	ENTITY <EC-NAME> DOES NOT EXIST
	ENTITY CLASS	ENTITY <EC-NAME> SYSTEM ERROR : UNABLE TO ALTER ALIAS

ALTER ATTRIBUTE

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	<attribute class name> DOES NOT EXIST
	ATTRIBUTE CLASS	UNABLE TO UPDATE ATT CL <attribute class number>
	DOMAIN	DOMAIN NOT CHANGED FOR ATT-CL <attribute class name>
	DOMAIN	UNABLE TO ASSIGN DOMAIN_NO TO <domain name>
	DOMAIN	UNABLE TO INSERT DOMAIN <domain name>
	KEYWORD	KEYWORD NOT DROPPED FOR <attribute class name>
	KEYWORD	KEYWORD NOT ADDED FOR <attribute class name>
	KEYWORD	UNABLE TO INSERT KEYWORD <key-word>
	KEYWORD	ALL KEYWORD NUMBERS ASSIGNED
	KEYWORD	UNABLE TO INSERT KEYWORD <KEY-WORD>
	KEYWORD	AC KEYWORD ALREADY EXISTS
	KEYWORD	UNABLE TO INSERT AC KEYWORD
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE
WARNING	KEYWORD	<KEYWORD <key-word> ALREADY EXISTS
	KEYWORD	KEYWORD <KEY-WORD> ALREADY EXISTS

ALTER DOMAIN

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DATA TYPE	DATA TYPE NAME IS NOT IN DOMAIN
	DATA TYPE	DATA TYPE NAME HAS USAGE <data-type-name>
	DATA TYPE	DATA TYPE COULD NOT BE DELETED <data-type-name>
	DATA TYPE	DATA TYPE COULD NOT BE ADDED FOR <DATA-TYPE-NAME> TYPE : <STD-USER>
	DATA TYPE	A STANDARD DATA TYPE IS ALREADY DEFINED : <DATA-TYPE-NAME>
	DATA TYPE	DATA TYPE NAME ALREADY EXISTS : <DATA-TYPE-NAME>
	DATA TYPE	TYPE IS NOT LEGAL : <TYPE-ID>
	DATA TYPE	DATA TYPE COULD NOT BE INSERTED : <DATA-TYPE-NAME>
	DATA TYPE	DECIMAL SPECIFICATION INCORRECT
	DATA TYPE	DATA TYPE NAME ALREADY EXISTS <data-type-name>
	DATA TYPE	TYPE IS NOT LEGAL <type-id>
	DATA TYPE	DATA TYPE COULD NOT BE INSERTED <data-type-name>
	DATA TYPE	DECIMAL SPECIFICATION INCORRECT FOR <data-type-name>
	DATA TYPE	TYPE ID NOT VALID : <TYPE-ID>
	DATA TYPE	DATA TYPE COULD NOT BE UPDATED FOR : <TYPE-ID>
	DOMAIN	DOMAIN DOES NOT EXIST FOR <dom-name>
WARNING	DATA FIELD	DATA FIELD <df-id> IS ASSOCIATED WITH DATA TYPE

ALTER ENTITY

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	INVALID ATTRIBUTE <AC-NAME> FOR THIS MODEL
	ATTRIBUTE USE CLASS	ATTRIBUTE USE CLASS WAS NOT DELETED
	ENTITY CLASS	ENTITY NAME <EC-NAME> DOES NOT EXIST
	INHERITED AUC	INHERITED-ATT-USE NOT DELETED FOR TAG-NO <tag number>
	KEY CLASS	KEY CLASS <KC-NAME> NOT FOUND IN THIS ENTITY
	KEY CLASS	COMPLETE RELATION NOT FOUND FOR KEY CLASS <KC-NAME>
	KEY CLASS	KEY CLASS <KC-NAME> CANNOT BE DELETED
	KEY CLASS	KEY CLASS <KC-NAME> COPY XREF CANNOT BE DELETED
	KEY CLASS	KEY CLASS <KEY-CLASS> MEMBER CANNOT BE DELETED
	KEY CLASS	UNABLE TO DELETE COMPLETE RELATION FOR KEY CLASS <KC-NAME>
	KEY CLASS	ALL KEY CLASS NUMBERS ASSIGNED
	KEY CLASS MEMBER	UNABLE TO DELETE MIGRATING KEY CLASS MEMBER
	KEY CLASS MEMBER	KEY_CLASS_MEMBER NOT DELETED FOR TAG-NO <tag number>
	KEYWORD	UNABLE TO INSERT KEYWORD <key-word>
	KEYWORD	ALL KEYWORD NUMBERS ASSIGNED
	KEYWORD	UNABLE TO INSERT KEYWORD <KEY-WORD>
	KEYWORD	AC KEYWORD ALREADY EXISTS
	KEYWORD	UNABLE TO INSERT AC KEYWORD
	TAG	ATTRIBUTE_USE_CL NOT DELETED FOR TAG-NO-<tag number>
	TAG	AUC_COPY_XREF NOT DELETED FOR TAG-NO <tag number>
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE
	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
WARNING	ATTRIBUTE USE CLASS	<AUC-NAME> IS AN INVALID KEY CLASS MEMBER
	KEY CLASS	KEY CLASS <KC-NAME> ALREADY EXISTS
	KEY CLASS	KEY CLASS NAME OMITTED - ONLY FIRST ATTRIBUTE USE CLASS PROCESSED
	KEYWORD	KEYWORD <key-word> ALREADY EXISTS
	KEYWORD	KEYWORD <KEY-WORD> ALREADY EXISTS

ALTER MAP

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE USE CLASS	TAG NAME <tag name> DOES NOT EXIST
	ATTRIBUTE USE CLASS	A VALUE STRING MUST BE ENTERED WHEN MAPPING AN AUC TO A SET
	DATA FIELD	DATA FIELD <data field name> DOES NOT EXIST
	DATA FIELD	A DATA FIELD MAPPING EXISTS FOR <data field name>
	DATA FIELD	PRIMARY MAPPING ALREADY EXISTS FOR <data field name>
	DATA FIELD	A PRIMARY MAPPING MUST EXIST FOR <data field name>
	DATA FIELD	A DATAFIELD MAPPING EXISTS FOR THIS AUC
	DATA FIELD	<database name><record name><datafield name> DOES NOT EXIST
	DATA FIELD	DATABASE ERROR, <database name><record name><datafield name> NOT D ELETED
	DATABASE	THE DATABASE NAME <database name> DOES NOT EXIST
	DOMAIN	NO DOMAIN FOR ATTRIBUTE NUMBER <attribute class number>
	DOMAIN	DOMAIN <domain number> HAS NO STANDARD DATA TYPE
	DOMAIN	<data type name> IS NOT IN THE DOMAIN
	ENTITY CLASS	INDEPENDENT ENTITY NAME <IND-EC-NAME> IS INVALID
	MAPPING	A DATABASE ERROR HAS OCCURRED - MAPPING NOT PERFORMED
	MAPPING	MAPPING NOT PERFORMED DUE TO A DATABASE ERROR
	MAPPING	PRIMARY MAP DROP IS INVALID WHEN SECONDARY MAPPING EXIST - MAP NAM E IS : <database name><record name><datafield name>
	RELATION CLASS	RELATION CLASS <relation class name> DOES NOT EXIST
	RELATION CLASS	ALTER FUNCTION INVALID FOR RC MAPPING
	RELATION CLASS	A RELATION CLASS - DATA FIELD MAPPING IS INVALID
	SET	TAG NUMBER <tag number> HAS BEEN MAPPED TO A SET
	SET	<set name> DOES NOT EXIST
	SET	<set name> ALREADY MAPS TO A RELATION CLASS
	SET	<set name> HAS ALREADY BEEN AUC MAPPED
	SET	DATABASE ERROR, <database name><set name> NOT DELETED

ALTER MODEL

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	MODEL	MODEL <model-name> TO BE ALTERED DOES NOT EXIST.
	MODEL	MODEL <model-name> CAN NOT BE ALTERED.
WARNING	MODEL	MODEL <model-name> HAS BEEN ALTERED.

ALTER RELATION

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

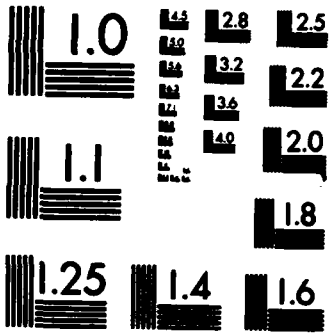
ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	COMPLETE RELATION	UNABLE TO INSERT COMPLETE RELATION
	ENTITY CLASS	INDEPENDENT ENTITY NAME <IND-EC-NAME> IS INVALID
	INHERITED AUC	INHERITED-ATT-USE NOT DELETED FOR TAG-NO <tag number>
	KEY CLASS	KEY-CLASS <KC-NAME> NOT FOUND ON INDEPENDENT ENTITY
	KEY CLASS	KEY CLASS <KC-NAME> NOT MIGRATED BY THIS RELATION CLASS
	KEY CLASS	KEY CLASS <kc-name> IS INVALID
	KEY CLASS MEMBER	UNABLE TO DELETE MIGRATING KEY CLASS MEMBER
	KEY CLASS MEMBER	KEY_CLASS_MEMBER NOT DELETED FOR TAG-NO <tag number>
	KEYWORD	UNABLE TO INSERT KEYWORD <key-word>
	KEYWORD	ALL KEYWORD NUMBERS ASSIGNED
	KEYWORD	UNABLE TO INSERT KEYWORD <KEY-WORD>
	KEYWORD	AC KEYWORD ALREADY EXISTS
	KEYWORD	UNABLE TO INSERT AC KEYWORD
	RELATION CLASS	RELATION CLASS <RC-NAME> DOES NOT EXIST
	RELATION CLASS	ILLEGAL DEPENDENT CARDINALITY
	RELATION CLASS	RELATION HAS MIGRATION
	TAG	ATTRIBUTE_USE_CL NOT DELETED FOR TAG-NO-<tag number>
	TAG	AUC_COPY_XREF NOT DELETED FOR TAG-NO <tag number>
	TAG	TAG-NAME <NEW-TAG-NAME> IS A DUPLICATE NAME
	TAG	NO MORE TAG NUMBERS AVAILABLE
	TAG	UNABLE TO INSERT TAG NAME <NEW-TAG-NAME> AS ATTRIBUTE-USE-CLASS
	TAG	UNABLE TO INSERT TAG NAME <NEW-TAG-NAME> AS INHERITED-ATTRIBUTE-US E-CLASS.
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE
	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
WARNING	KEYWORD	KEYWORD <key-word> ALREADY EXISTS
	KEYWORD	KEYWORD <KEY-WORD> ALREADY EXISTS
	RELATION CLASS	INDEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO 99
	RELATION CLASS	LEFT DEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO ZERO
	RELATION CLASS	RIGHT DEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO 99

CHECK MODEL

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	MODEL	MODEL <MODEL-NAME> DOES NOT EXIST
	MODEL	MODEL <MODEL-NAME> COULD NOT BE MARKED AS CHECKED
FATAL	MODEL	ERROR IN DOPEN
	MODEL	ERROR IN DSQL3
	MODEL	ERROR IN DDFINN1
	MODEL	ERROR IN DDFINN2
	MODEL	ERROR IN DDFINN3
	MODEL	ERROR IN DBINDN
	MODEL	ERROR IN DEXEC
	MODEL	ERROR IN DFETCH
WARNING	ENTITY CLASS	ENTITY <EC-NAME> HAS NO KEY CLASS
	KEY CLASS	<KC-NAME> OF <EC-NAME> IS A DUPLICATE KEY OR SUBSET
	MODEL	1:1 RELATIONSHIP EXISTS BETWEEN <EC-NAME> <RC-NAME> <EC-NAME2>
	MODEL	A NON SPECIFIC RELATIONSHIP EXISTS BETWEEN <EC-ANME1> <RC-NAME> <EC-NAME2>
	MODEL	AN INCOMPLETE RELATIONSHIP EXISTS BETWEEN <EC-ANME1> RC-NAME> <EC-NAME2>

AD-A101 955 INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 5 2/2
COMMON DATA MODEL S..(U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONSU..
UNCLASSIFIED J L ALTHOFF ET AL. 01 NOV 85 UM-620141100 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

COMBINE ENTITY

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ENTITY CLAS	
	ENTITY CLASS	ENTITY NAMES MUST DIFFER ON INTRA-MODEL COMBINE
	ENTITY CLASS	FROM-MODEL <MOD-NAME> DOES NOT EXIST
	ENTITY CLASS	FROM-ENTITY <ENT-NAME> DOES NOT EXIST
	ENTITY CLASS	TO-ENTITY <ENT-NAME> DOES NOT EXIST
	ENTITY CLASS	FROM MODEL <MOD-NAME> DOES NOT EXIST
	MODEL	MODEL <model-name> HAS MORE THAN 400 RELATIONS
	MODEL	TABLE OVERFLOW ON RC-DEPKC-LIST: INCREASE SIZE
	MODEL	AN ENTITY IN MODEL <model-name> HAS MORE THAN 5 KEY CLASSES
	MODEL	TABLE OVERFLOW ON EC-LIST
WARNING	RELATION CLASS	MORE RELATIONS EXIST WITH INDEPENDENT ENTITY: <IND-EC> AND DEPENDANT ENTITY <DEP_EC>
	RELATION CLASS	RCND-LIST OVERFLOWED, MORE RELATIONS EXIST
	RELATION CLASS	MORE RELATIONS EXIST WITH INDEPENDANT ENTITY: <IND-EC> AND DEPENDANT ENTITY <DEP-EC>

COMPARE MODEL .

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	MODEL	MODEL <model-name1> DOES NOT EXIST
	MODEL	MODEL <model-name2> DOES NOT EXIST
WARNING	ATTRIBUTE CLASS	ATTRIBUTE OR ALIAS <AC-NAME> EXIST IN BOTH MODELS
	ENTITY CLASS	ENTITY OR ALIAS <EC-NAME> EXISTS IN BOTH MODELS
	ENTITY CLASS	ENTITY <EC-NAME1> AND ENTITY <EC-NAME2> HAVE MATCHING KEYWORD <KW-NAME>
	ENTITY CLASS	ATTRIBUTE <AC-NAME1> AND ATTRIBUTE <AC-NAME2> AND RELATION <IND-EC-NAME2> <RC-NAME2> <DEP-EC-NAME2> HAVE MATCHING KEYWORD <KW-NAME>

COPY ATTRIBUTE

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> TO BE COPIED DOES NOT EXIST IN MODEL <model-name>
	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> CAN NOT BE COPIED.
	ATTRIBUTE CLASS	ATTRIBUTE NAME TO BE COPIED AS MUST BE ENTERED
	ATTRIBUTE CLASS	MODEL <MODEL-NAME> WHERE ATTRIBUTE IS TO BE COPIED DOES NOT EXIST
	ATTRIBUTE CLASS	ATTRIBUTE <AC-NAME> TO BE COPIED DOES NOT EXIST
	ATTRIBUTE CLASS	ATTRIBUTE <AC-NAME> NOT INSERTED INDT ATTRIBUTE_NAME
	ATTRIBUTE CLASS	ATTRIBUTE DESCRIPTION NOT INSERTED
	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> ALREADY EXISTS IN THE MODEL
	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> CAN NOT BE CREATED
	MODEL	MODEL <model-name> FROM WHICH ATTRIBUTE <attribute-name> IS TO BE COPIED, DOES NOT EXIST.
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED

COPY DESCRIPTION

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DESCRIPTION	DESCRIPTION TYPE <DESC-TYPE> IS INVALID
	DESCRIPTION	ERROR IN COPYING DESCRIPTION TEXT
	MODEL	MODEL <MODEL-NAME> DOES NOT EXIST
	OBJECT	DESCRIPTION OF <OBJ-ID> DOES NOT EXIST
	OBJECT	DESCRIPTION OF <OBJ-ID> ALREADY EXISTS IN THE TO MODEL

COPY ENTITY

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR		EXPECTING STRUCTURE OR RELATIONS
	ATTRIBUTE CLASS	ATTRIBUTE COULD NOT BE COPIED
	ATTRIBUTE CLASS	NEW ATTRIBUTE <tag-name> COULD NOT BE ADDED AS OWNED
	ATTRIBUTE CLASS	ATTRIBUTE CROSS-REFERENCE COULD NOT BE INSERTED
	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> ALREADY EXISTS IN THE MODEL
	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> CAN NOT BE CREATED
	ENTITY CLASS	ENTITY <ec-name> TO BE COPIED DOES NOT EXIST IN MODEL <model-name>
	ENTITY CLASS	TO ENTITY NAME MUST BE ENTERED
	ENTITY CLASS	FROM / TO ENTITY NAMES ARE THE SAME
	ENTITY CLASS	FROM ENTITY <FROM-EC-NAME> DOES NOT EXIST
	ENTITY CLASS	TO ENTITY <TO-EC-NAME> ALREADY EXISTS
	ENTITY CLASS	ON FILE MUST BE SPECIFIED FOR STRUCTURE/RELATION
	ENTITY CLASS	FROM MODEL <FROM-MOD-NAME> DOES NOT EXIST
	ENTITY CLASS	ENTITY <EC-NAME> NOT INSERTED INOT ENTITY-CLASS
	ENTITY CLASS	ENTITY <EC-NAME> NOT INSERTED INTO ENTITY-NAME
	ENTITY CLASS	ENTITY <EC-NAME> CROSS REFERENCE NOT INSERTED
	ENTITY CLASS	ALIASES/INHERITED OR OWNED ATTRIBUTES CAN'T BE COPIED
	ENTITY CLASS	ERROR GENERATING CREATE ENTITY COMMAND
	KEY CLASS	UNABLE TO ASSIGN UNIQUE KC NUMBER
	KEY CLASS	KEY CLASS CAN NOT BE CREATED FOR COPIED ENTITY
	KEY CLASS	KEY CLASS CROSS REF COULD NOT BE CREATED
	KEY CLASS MEMBER	KEY CLASS MEMBER CAN NOT BE CREATED FOR COPIED ENTITY
	MODEL	MODEL <model name> FROM WHICH ENTITY <ec-name> IS TO BE COPIED, DO ES NOT EXIST.
	MODEL	MODEL <model-name> HAS MORE THAN 400 RELATIONS
	MODEL	TABLE OVERFLOW ON RC-DEPKC-LIST: INCREASE SIZE
	MODEL	AN ENTITY IN MODEL <model-name> HAS MORE THAN 5 KEY CLASSES
	MODEL	TABLE OVERFLOW ON EC-LIST
	TAG	TAG CROSS REFERENCE COULD NOT BE CREATED
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED

COPY MODEL

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	MODEL	MODEL <MOD-NAME> DOES NOT EXIST
	MODEL	MODEL <MOD-NAME> ALREADY EXISTS
	MODEL	UNABLE TO OPEN <USER-FILNAM> FILE AS OUTPUT
	MODEL	<MOD-NAME> MODEL STRUCTURE IS INVALID
	MODEL	<MOD-NAME> MODEL CANNOT BE COPIED
	MODEL	MODEL <model-name> HAS MORE THAN 400 RELATIONS
	MODEL	TABLE OVERFLOW ON RC-DEPRC-LIST: INCREASE SIZE
	MODEL	AN ENTITY IN MODEL <model-name> HAS MORE THAN 5 KEY CLASSES
	MODEL	TABLE OVERFLOW ON EC-LIST

CREATE ALIAS

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	ATTRIBUTE <obj-id> DOES NOT EXIST
	ATTRIBUTE CLASS	ALIAS <alias-id> ALREADY EXISTS FOR ATTRIBUTE
	ATTRIBUTE CLASS	ALIAS COULD NOT BE CREATED FOR ATTRIBUTE <obj-id>
	ENTITY CLASS	ENTITY <obj-id> DOES NOT EXIST
	ENTITY CLASS	ALIAS <alias-id> ALREADY EXISTS FOR ENTITY
	ENTITY CLASS	ALIAS COULD NOT BE CREATED FOR ENTITY <obj-id>

CREATE ATTRIBUTE

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> ALREADY EXISTS IN THE MODEL
	ATTRIBUTE CLASS	ATTRIBUTE <ac-name> CAN NOT BE CREATED
	DOMAIN	DOMAIN <domain-name> ALREADY EXISTS
	DOMAIN	DOMAIN TABLE FULL
	DOMAIN	DOMAIN <DOMAIN-NAME> ALREADY EXISTS
	KEYWORD	UNABLE TO INSERT KEYWORD <key-word>
	KEYWORD	ALL KEYWORD NUMBERS ASSIGNED
	KEYWORD	UNABLE TO INSERT KEYWORD <KEY-WORD>
	KEYWORD	AC KEYWORD ALREADY EXISTS
	KEYWORD	UNABLE TO INSERT AC KEYWORD
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
WARNING	KEYWORD	KEYWORD <key-word> ALREADY EXISTS
	KEYWORD	KEYWORD <KEY-WORD> ALREADY EXISTS

CREATE DOMAIN

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DATA TYPE	DATA TYPE COULD NOT BE ADDED FOR <DATA-TYPE-NAME> TYPE : <STD-USER>
	DATA TYPE	A STANDARD DATA TYPE IS ALREADY DEFINED : <DATA-TYPE-NAME>
	DATA TYPE	DATA TYPE NAME ALREADY EXISTS : <DATA-TYPE-NAME>
	DATA TYPE	TYPE IS NOT LEGAL : <TYPE-ID>
	DATA TYPE	DATA TYPE COULD NOT BE INSERTED : <DATA-TYPE-NAME>
	DATA TYPE	DECIMAL SPECIFICATION INCORRECT
	DATA TYPE	DATA TYPE NAME ALREADY EXISTS <data-type-name>
	DATA TYPE	TYPE IS NOT LEGAL <type-id>
	DATA TYPE	DATA TYPE COULD NOT BE INSERTED <data-type-name>
	DATA TYPE	DECIMAL SPECIFICATION INCORRECT FOR <data-type-name>
	DOMAIN	DOMAIN ALREADY EXISTS - <dom-name>
	DOMAIN	DOMAIN NUMBER OVERFLOW
	DOMAIN	DOMAIN COULD NOT BE INSERTED - <dom-name>
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED

CREATE ENTITY

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ENTITY CLASS	ENTITY NAME <EC-NAME> EXISTS
	ENTITY CLASS	ALL ENTITY NUMBERS ASSIGNED - <EC-NAME> NOT PROCESSED
	KEY CLASS	ALL KEY CLASS NUMBERS ASSIGNED
	KEYWORD	UNABLE TO INSERT KEYWORD <key-word>
	KEYWORD	ALL KEYWORD NUMBERS ASSIGNED
	KEYWORD	UNABLE TO INSERT KEYWORD <KEY-WORD>
	KEYWORD	AC KEYWORD ALREADY EXISTS
	KEYWORD	UNABLE TO INSERT AC KEYWORD
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
WARNING	ATTRIBUTE USE CLASS	<AUC-NAME> IS AN INVALID KEY CLASS MEMBER
	KEY CLASS	KEY CLASS <KC-NAME> ALREADY EXISTS
	KEY CLASS	KEY CLASS NAME OMITTED - ONLY FIRST ATTRIBUTE USE CLASS PROCESSED
	KEYWORD	KEYWORD <key-word> ALREADY EXISTS
	KEYWORD	KEYWORD <KEY-WORD> ALREADY EXISTS

CREATE MAP

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE USE CLASS	A VALUE STRING MUST BE ENTERED WHEN MAPPING AN AUC TO A SET
	DATA FIELD	DATA FIELD <data field name> DOES NOT EXIST
	DATA FIELD	A DATA FIELD MAPPING EXISTS FOR <data field name>
	DATA FIELD	PRIMARY MAPPING ALREADY EXISTS FOR <data field name>
	DATA FIELD	A PRIMARY MAPPING MUST EXIST FOR <data field name>
	DATA FIELD	A DATAFIELD MAPPING EXISTS FOR THIS AUC
	DATABASE	THE DATABASE NAME <database name> DOES NOT EXIST
	DOMAIN	NO DOMAIN FOR ATTRIBUTE NUMBER <attribute class number>
	DOMAIN	DOMAIN <domain number> HAS NO STANDARD DATA TYPE
	DOMAIN	<data type name> IS NOT IN THE DOMAIN
	ENTITY CLASS	ENTITY CLASS <entity class name> DOES NOT EXIST
	ENTITY CLASS	INDEPENDENT ENTITY NAME <IND-EC-NAME> IS INVALID
	MAPPING	A DATABASE ERROR HAS OCCURRED - MAPPING NOT PERFORMED
	MAPPING	MAPPING NOT PERFORMED DUE TO A DATABASE ERROR
	RELATION CLASS	RELATION CLASS <relation class name> DOES NOT EXIST
	SET	TAG NUMBER <tag number> HAS BEEN MAPPED TO A SET
	SET	<set name> DOES NOT EXIST
	SET	<set name> ALREADY MAPS TO A RELATION CLASS
	SET	<set name> HAS ALREADY BEEN AUC MAPPED
	TAG	TAG NAME <tag name> DOES NOT EXIST

CREATE MODEL

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	MODEL	MODEL <model-name> ALREADY EXISTS.
	MODEL	A UNIQUE NUMBER CAN NOT BE ASSIGNED TO MODEL <model-name> BEING CREATED.
	MODEL	MODEL <model-name> CAN NOT BE CREATED.
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
WARNING	MODEL	MODEL <model-name> HAS BEEN CREATED.

CREATE RELATION

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	COMPLETE RELATION	UNABLE TO INSERT COMPLETE RELATION
	KEY CLASS	KEY CLASS <kc-name> IS INVALID
	KEYWORD	UNABLE TO INSERT KEYWORD <key-word>
	KEYWORD	ALL KEYWORD NUMBERS ASSIGNED
	KEYWORD	UNABLE TO INSERT KEYWORD <KEY-WORD>
	KEYWORD	AC KEYWORD ALREADY EXISTS
	KEYWORD	UNABLE TO INSERT AC KEYWORD
	RELATION CLASS	ALL RELATION CLASS NUMBERS ASSIGNED
	RELATION CLASS	ILLEGAL DEPENDENT CARDINALITY
	RELATION CLASS	RELATION HAS MIGRATION
	TAG	TAG-NAME <NEW-TAG-NAME> IS A DUPLICATE NAME
	TAG	NO MORE TAG NUMBERS AVAILABLE
	TAG	UNABLE TO INSERT TAG NAME <NEW-TAG-NAME> AS ATTRIBUTE-USE-CLASS
	TAG	UNABLE TO INSERT TAG NAME <NEW-TAG-NAME> AS INHERITED-ATTRIBUTE-US E-CLASS.
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
WARNING	KEYWORD	KEYWORD <key-word> ALREADY EXISTS
	KEYWORD	KEYWORD <KEY-WORD> ALREADY EXISTS
	RELATION CLASS	INDEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO 99
	RELATION CLASS	LEFT DEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO ZERO
	RELATION CLASS	RIGHT DEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO 99

CREATE VIEW

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR		VIEW ALREADY EXIST : <view-name>
		VIEW COULD NOT BE CREATED : <view-name>
		INTERNAL TABLES FOR VIEW COMMAND CANNOT BE CREATED
		VIEW <view-name> COULD NOT BE CREATED
	ATTRIBUTE USE CLASS	TAG NAME DOES NOT EXIST : <TAG-NAME>
	DATA ITEM	VALUES NOT INSERTED INTO PROJECT DATA ITEM, DATA ITEM NAME : <DI-N AME>
	DATA ITEM	VALUES NOT INSERTED INTO DATA ITEM, DI-NAME : <NEW-DI-NAME>
	DATA ITEM	VALUES NOT INSERTED INTO PROJECT DATA ITEM, DI-NAME : <DI-NAME>
	DATA ITEM	UNABLE TO INSERT DATA_ITEM: <TAG-NAME>
	DATA ITEM	UNABLE TO INSERT PROJECT_DATA_ITEM_ <TAG-NAME>
	DATA ITEM	UNABLE TO INSERT PROJECT_DATA_ITEM <TAG-NAME>
	DOMAIN	DOMAINS ARE NOT THE SAME FOR AUC- <TAG-NAME> AND DATA ITEM - <DI-N AME>
	DOMAIN	STANDARD DATA TYPE NAME NOT FOUND FOR DOMAIN: <DOM-ND>
	ENTITY CLASS	ENTITY NAME: <EC-NAME> NOT FOUND
	RELATION CLASS	ERROR DURING VERIFYING RELATION, RC-NAME : <RC-NAME>
	RELATION CLASS	SEC/RC ENTITY COULD NOT BE CREATED FOR RELATION <VRCL-RC-NAME>
	SYNTAX	ENTITY NAME REQUIRED IN FROM CLAUSE WHEN * USED IN SELECT
	SYNTAX	NO. OF DATA ITEMS MUST EQUAL NO. OF TAGS SELECTED
	SYNTAX	DATAITEM: <VDIL-DATA-ITEM-NAME> HAS NO MATCH IN SELECT LIST
	SYNTAX	ENTITY NAME: <VRL-ENTITY-NAME> HAS NO MATCH IN WHERE CLAUSE
	SYNTAX	NO. OF DATA ITEMS MUST MATCH NO. OF TAG NAMES
	TAG	TAG NUMBER NOT FOUND FOR TAG: <TAG-NAME>
	TAG	DOMAIN-NUMBER NOT FOUND FOR TAG: <TAG-NAME>
	VIEW	INVALID SEC STRUCTURE
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED

DEFINE DATABASE

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DATABASE	DATA BASE <DB-NAME> ALREADY EXISTS
	DATABASE	DATA BASE <DB-NAME> COULD NOT BE INSERTED
	DATABASE	ERROR IN DEFINING DATA BASE
	DATABASE	PASSWORD CLAUSE REQUIRED FOR ORACLE DBMS
	DATABASE	ORACLE DATA BASE COULD NOT BE INSERTED
	DATABASE	FILE CLAUSE REQUIRED FOR TOTAL DBMS
	DATABASE	FILE <FILE-NAME> ALREADY EXISTS
	DATABASE	TOTAL DATA BASE COULD NOT BE INSERTED
	DATABASE	PSB SEQUENTIAL NUMBER REQUIRED FOR IMS
	DATABASE	KEY FEEDBACK LENGTH REQUIRED FOR IMS
	DATABASE	PSB NAME REQUIRED FOR IMS
	DATABASE	PSB NAME <PSB-NAME> ALREADY EXISTS
	DATABASE	IMS DATA BASE COULD NOT BE INSERTED
	DATABASE	CODASYL REQUIRES SCHEMA NAME
	DATABASE	CODASYL REQUIRES SUB-SCHEMA NAME
	DATABASE	CODASYL REQUIRES AREA NAME
	DATABASE	SCHEMA NAMES COULD NOT BE INSERTED
	DATABASE	CODASYL DATA BASE COULD NOT BE INSERTED
	WARNING	
FATAL	DATABASE	UNABLE TO ASSIGN NEW NUMBER
	DATABASE	UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED
WARNING	DATABASE	AREA <AREA-NAME> ALREADY EXISTS

DEFINE RECORD

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	RECORD	UNABLE TO SET THE DATA BASE ID
	RECORD	RECORD <PT-ID> ALREADY EXISTS

DEFINE SET

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DATA FIELD	MEMBER RECORD: <MEMBER-ID> DOES NOT EXIST
	DATA FIELD	UNABLE TO INSERT SET TYPE MEMBER: <MEMBER-ID>
	DATA FIELD	LINKAGE WAS NOT FOUND FOR <DF-ID>
	DATA FIELD	LINKAGE DATA FIELD <DF-ID> IS NOT DEFINED
	DATA FIELD	LINKAGE DATA FIELD: <DF-ID> WAS INSERTED.
	DATABASE	UNABLE TO RETRIEVE DB SET FOR <DB>
	DATABASE	DEFINE SET IS ILLEGAL FOR <DBMS-NAME>
	DBMS	<DBMS-NAME> IS NOT A LEGAL DBMS
	DBMS	MULTIPLE MEMBERS ARE NOT ALLOWED IN: <DBMS-NAME>
	RECORD	OWNER RECORD: <OWNER-ID> DOES NOT EXIST
	RECORD	UNABLE TO INSERT RECORD SET: <SET-ID>
	SET	<SET-ID> ALREADY EXISTS

DESCRIBE

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR		
	DESCRIPTION	DESCRIPTION TYPE <DESCRIPTION TYPE> DOES NOT EXIST.
	DESCRIPTION	DESCRIPTION TYPE <description type> DOES NOT EXIST
	MODEL	A CURRENT MODEL HAS NOT BEEN ESTABLISHED FOR THE SESSION.
	OBJECT	THE OBJECT TYPE <object type> DOES NOT EXIST

DROP ALIAS

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	ALIAS <AC-NAME> DOES NOT EXIST <ellps-id>
	ATTRIBUTE CLASS	ATTRIBUTE <AC-NAME> COULD NOT BE DELETED
	ENTITY CLASS	ALIAS <ec-name> DOES NOT EXIST
	ENTITY CLASS	ENTITY <EC-NAME> COULD NOT BE DELETED

DROP ATTRIBUTE

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	<attribute class name> DESCRIPTION TEXT NOT DELETED
	ATTRIBUTE CLASS	<attribute class name> NAME NOT DELETED
	ATTRIBUTE CLASS	<attribute class name> cross reference not deleted
	ATTRIBUTE CLASS	<attribute class name> ATTRIBUTE CLASS NOT DELETED
	ATTRIBUTE CLASS	<attribute class name> OWNED ATTRIBUTE NOT DELETED
	ATTRIBUTE CLASS	<attribute class name> DOES NOT EXIST
	ATTRIBUTE USE CLASS	ATTRIBUTE USE CLASS WAS NOT DELETED
	KEY CLASS	KEY CLASS NOT DELETED FOR KEY CLASS <key class number>
	KEY CLASS	KEY CLASS CROSS REFERENCE NOT DELETED FOR KEY CLASS <key class number>
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE

DROP DATABASE

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DATABASE	DATA BASE <DB-ID> IS INVALID
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM DATA-BASE
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM SCHEMA-NAMES
	DATABASE	DATA BASE <DB_ID> NOT DELETED FROM RECORD-TYPES
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM DB-AREA-ASSIGNMENT
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM DATA-FIELD
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM RECORD-SET
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM SET-TYPE-MEMBER
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM RECORD-KEY
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM RECORD-KEY-MEMBER
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM AUC-ST-MAPPING
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM RC-BASED-REC-SET
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM IMS-SEGMENT-SIZE
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM DATA-BASE-AREA
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM DF-SET-LINKAGE
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM SEGMENT-DATA-FIELD
	DATABASE	DATA BASE <DB-ID> NOT DELETED FROM PROJECT-DATA-FIELD

DROP DOMAIN

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DOMAIN	DOMAIN DOES NOT EXIST FOR : <DOM-NAME>
	DOMAIN	DOMAIN IN USE CAN'T BE DROPPED FOR : <DOM-NAME>
	DOMAIN	DATA TYPES COULD NOT BE DELETED FOR DOMAIN : <DOM-NAME>
	DOMAIN	DOMAIN COULD NOT BE DELETED FOR : <DOM-NAME>
WARNING	DATA FIELD	DATA FIELD <df-id> IS ASSOCIATED WITH DATA TYPE

DROP ENTITY

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	OWNED ATTRIBUTE CAN NOT BE CREATED
	ATTRIBUTE USE CLASS	ATTRIBUTE USE CLASS WAS NOT DELETED
	ENTITY CLASS	ENTITY <entity name> DOES NOT EXIST AND CAN NOT BE DROPPED.
	ENTITY CLASS	RELATIONS ASSOCIATED WITH ENTITY <ec-name> CAN NOT BE DROPPED.
	ENTITY CLASS	NAMES CAN NOT BE DROPPED FOR ENTITY <ec-name>
	ENTITY CLASS	KEYWORDS CAN NOT BE DROPPED FOR ENTITY <ec-name>
	ENTITY CLASS	DESCRIPTIONS CAN NOT BE DROPPED FOR ENTITY <ec-name>
	ENTITY CLASS	ENTITY <ec-name> COULD NOT BE DROPPED
	ENTITY CLASS	CROSS REFERENCE RECORD CAN NOT BE DROPPED FOR ENTITY <ec-name>
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE

DROP FIELD

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	DATA FIELD	DATA BASE INFORMATION UNAVAILABLE
	DATA FIELD	DATA FIELD <DF-ID> DOES NOT EXIST
	DATA FIELD	DATA FIELD <DF-ID> NOT DELETED FROM DF-SET-LINKAGE
	DATA FIELD	DATA FIELD <DF-ID> NOT DELETED FROM SET-TYPE-MEMBER
	DATA FIELD	DATA FIELD <DF-ID> NOT DELETED FROM RECORD-SET
	DATA FIELD	DATA FIELD <DF-ID> NOT DELETED FROM AUC-ST-MAPPING
	DATA FIELD	DATA FIELD <DF-ID> NOT DELETED FROM DATA-FIELD
	DATA FIELD	DATA FIELD <DF-ID> NOT DELETED FROM RECORD-KEY-MEMBER

DROP KEYWORD

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	KEYWORD	KEYWORD DOES NOT EXIST <key-word>
	KEYWORD	KEYWORD ATTRIBUTE ASSOCIATION NOT DELETED FOR <key-word>
	KEYWORD	KEYWORD-ENTITY ASSOCIATION NOT DELETED FOR <key-word>
	KEYWORD	KEYWORD-RELATION ASSOCIATION NOT DELETED FOR <key-word>
	KEYWORD	KEYWORD COULD NOT BE DELETED <key-word>
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE

DROP MAP

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ENTITY CLASS	ENTITY CLASS <entity class name> DOES NOT EXIST
	ENTITY CLASS	INDEPENDENT ENTITY NAME <IND-EC-NAME> IS INVALID
	MAPPING	DATABASE ERROR - MAP NOT DROPPED
	RELATION CLASS	RELATION CLASS <relation class name> DOES NOT EXIST
	TAG	TAG NAME <tag name> DOES NOT EXIST

DROP MODEL

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ATTRIBUTE CLASS	ERROR WHILE DELETING ATTRIBUTE <AC-NAME> FROM MODEL
	ATTRIBUTE USE CLASS	ERROR WHILE DELETING TAG FOR ENTITY <EC-NAME>
	ENTITY CLASS	ERROR WHILE DELETING ENTITY <EC-NAME> FROM MODEL
	MODEL	NO PRIVELEGE TO DROP INTEGRATED_MODEL
	MODEL	MODEL <MODEL-NAME> DOES NOT EXIST
	MODEL	MODEL <MODEL-NAME> NOT DELETED FROM MODEL-CLASS
	MODEL	MODEL <MODEL-NAME> NOT DELETED FROM DESC-TEXT
	RELATION CLASS	ERROR WHILE DELETING RELATION FOR ENTITY <EC-NAME>
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE

DROP RECORD

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	RECORD	RECORD TYPE: <RT-ID> DOES NOT EXIST
	RECORD	<RT-ID> NOT DELETED FROM RECORD-TYPE
	RECORD	<RT-ID> NOT DELETED FROM OS_AREA_ASSIGNMENT
	RECORD	<RT-ID> NOT DELETED FROM DATA_FIELD
	RECORD	<RT-ID> NOT DELETED FROM PROJECT_DATA_FIELD
	RECORD	<RT-ID> NOT DELETED FROM RECORD_KEY
	RECORD	<RT-ID> NOT DELETED FROM RECORD_KEY_MEMBER
	RECORD	<RT-ID> NOT DELETED FROM IMS_SEGMENT_SIZE
	RECORD	<RT-ID> NOT DELETED FROM SEG_DATA_FIELD
	RECORD	<RT-ID> NOT DELETED FROM LINKAGE_DATA_FIELD
	RECORD	OWNER AND SET MEMBERS NOT DELETED.

DROP RELATION

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ENTITY CLASS	INDEPENDENT ENTITY NAME <IND-EC-NAME> IS INVALID
	RELATION CLASS	RELATION CLASS <RC-NAME> DOES NOT EXIST
	RELATION CLASS	RELATION HAS NO MIGRATION
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE

DROP SET

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	SET	DATA BASE INFORMATION UNAVAILABLE
	SET	SET <SET-ID> DOES NOT EXIST
	SET	SET <SET-ID> NOT DELETED FROM RECORD-SET
	SET	SET <SET-ID> NOT DELETED FROM SET-TYPE-MEMBER
	SET	SET <SET-ID> NOT DELETED FROM AUC-ST-MAPPING
	SET	SET <SET-ID> NOT DELETED FROM RC-BASED-REC-SET
	SET	SET <SET-ID> NOT DELETED FROM DF-SET-LINKAGE

DROP VIEW

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	VIEW	VIEW DOES NOT EXIST FOR : <VIEW-NAME>
	VIEW	PROJECT DATA ITEM COULD NOT BE DELETED FOR : <VIEW-NAME>
	VIEW	SEC-RC-COMPONENT COULD NOT BE DELETED FOR : <VIEW-NAME>
	VIEW	SEC COULD NOT BE DELETED FOR : <VIEW-NAME>
FATAL	DATABASE	REUSABLE NUMBER NOT ADDED TO DATA BASE

HALT

No Error Messages

MERGE MODEL

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	MODEL	MODEL 1 <MOD-NAME> DOES NOT EXIST
	MODEL	MODEL 2 <MOD-NAME> DOES NOT EXIST
	MODEL	MODEL 1 NOT COPIED INTO MODEL 3
	MODEL	ERROR WHILE MERGING ENTITIES OF MODEL 2
	MODEL	NO ERROR MESSAGES
	MODEL	MODEL <model-name> HAS MORE THAN 400 RELATIONS
	MODEL	TABLE OVERFLOW ON RC-DEPKC-LIST: INCREASE SIZE
	MODEL	AN ENTITY IN MODEL <model-name> HAS MORE THAN 5 KEY CLASSES
	MODEL	TABLE OVERFLOW ON EC-LIST

RENAME

ERROR MESSAGE REPORT FOR A SPECIFIED COMMAND

ERROR TYPE	OBJECT TYPE	ERROR MESSAGE
ERROR	ENTITY CLASS	INDEPENDENT ENTITY NAME <IND-EC-NAME> IS INVALID
	OBJECT	OBJECT INVALID FOR RENAMING
	OBJECT	OBJECT TO BE RENAMED DOES NOT EXIST, ID IS : <OLD-ID>
	OBJECT	NEW OBJECT NAME ALREADY EXISTS, ID IS : <NEW-ID>
	OBJECT	OBJECT TYPE NOT UPDATED, OBJECT NUMBER : <OLD-OBJ-NO>

APPENDIX B

GLOSSARY

Alpha-Numeric Data Format

A data format for values that can contain characters other than numerals (0-9). Numerals may be permitted also.

Attribute Class

A collection of all the same kind of attributes, i.e., those that have the same meaning. An attribute is a characteristic or fact about an entity. An attribute consists of a name (e.g., employee hire date) and a value (e.g., 15 August 1980). An attribute value may be:

A. Nondivisible (e.g., state name)

B. Divisible, i.e., a concatenation of two or more other attribute values (e.g., part number formed by concatenating drawing number and material code).

C. Computed from one or more other attribute values (e.g., age computed as current date minus birth date).

Attribute Class Data Description

A generic data description that applies to a particular attribute class.

Attribute Use Class

A model attribute class that appears in a model entity class. Each attribute use class represents either an owned attribute class or an inherited attribute class.

Attribute Use Class/Data Field Mapping

Indicates that an attribute use class corresponds to a data item; i.e. that they have the same meaning and that the data item can be used to access the values for the attribute use class.

Attribute Use Class/Record Set Mapping

Certain attribute use classes can be represented in a database by a group of record sets rather than be a data field. For example, Project:Task record sets called Pending, In-Process, On-Hold, and Completed. An attribute use class/record set mapping indicates that a particular record set corresponds to a particular attribute use class value.

Component Data Field

A data field that is part of another data field; e.g., if data field A is made up of data fields B, C, and D, each of these latter data fields is a component of A. A data field cannot be a component of more than one other data field.

Component Domain

An elementary domain that is part of another domain; e.g., a Date domain might be made up of a Month domain, a Day of Month domain, and a Year domain. Each of these latter domains would be a component of the Date domain. An elementary domain can be a component of several other domains.

DBD

An IMS database is defined by a Database Description (DBD). The DBD consists of statement which map an IMS structure into physical storage.

Database Area

A subdivision of a CODASYL database. This subdivision is a technique for improving the efficiency accessing database record type instances. When a database is subdivided into database areas, some or all of its records types are assigned to particular areas. Instances of these record types are stored only within the assigned areas. Then, these record type instances can be accessed by searching only the appropriate areas rather than the entire database. This access method is only used when the record type instances cannot be located by other means (e.g., by calc keys or record sets).

Database Area Assignment

Indicates that a record type is assigned to a database area.

Database Directory

A software library that must be used when accessing a database.

Database Password

A code that must be supplied when logging on to a DBMS to use a database. The DBMS verifies the password before accepting any other messages.

Data Field

A portion of a record type in which data values can be stored

Data Field/Record Set Linkage

A data field in a variable data set in a TOTAL database that is used as the variable control key for a linkpath from a master data set.

Data Field Redefinition

A data field that occupies the same space in a record type as another data field. A record instance cannot contain values in both data fields. One instance can contain a value in one field while another contains a value in the other.

Data Item

An attribute class as seen by a user in a user view, i.e., a kind of data (e.g., employee hire date), not a particular data value (e.g., 15 August 1980).

Data Type

The combination of a type of values (e.g., alphanumeric, signed numeric, etc.) and a type of storage (e.g., binary, packed, etc.)

Data Type Name

Names of NDDL data types. The NDDL data types correspond to the following COBOL/FORTRAN data types:

NDDL Data Type	COBOL/FORTRAN Data Type
INTEGER	FORTRAN binary integer
CHARACTER	x(n)
SIGNED	S99V99
FLOAT	FORTRAN floating point
UNSIGNED	99V99
PACKED	COMP-3

Description Type

A generic object may have several different kinds or styles of description (short, long, technical, nontechnical, etc.). Each is a description type.

Domain

A set of rules about the values that are allowed for a data item, attribute class, or data field. A domain is either an elementary domain or a group of two or more elementary domains, called component domains.

Domain Range

A series of consecutive values that represent all or part of an elementary domain.

Domain Value

A single value within an elementary domain.

Elementary Data Field

A data field that does not have any component data fields.

Entity Class

A collection of similar entities, i.e., those that have the same kinds of attributes. An entity is a person, place, event, thing, concept, etc.

Entity Class/Record Type Mapping

Indicates that an entity class corresponds to a record type, i.e., that they both have the same meaning and that the record type can be used to store instances of the entity class.

If a record type has more than one EC-RT mapping, some of its instances correspond to instances of one entity class while others correspond to instances of another, i.e., the record type is the relational union of the entity classes. An example is a Replenishment Order record type that maps to both the Purchase Order and Manufacturing Order entity classes. Each record instance represents either a purchase order or a manufacturing order.

Feedback

The length of the key feedback area for an IMS PCB. When IMS retrieved a segment from the database, the requested segment is fetched and a fully concatenated key is placed in the key feedback area. The fully concatenated key consists of the concatenation of the sequence field of values of all segments in the the hierarchical path from the root down to the retrieved segment. They key feedback area must be large enough to accommodate the maximum length for a fully concatenated key and stated in the KEYLEN entry of the PCB macro.

File

A set of stored data that is managed by a file management system (e.g., VSAM).

File/Database

A set of stored data, i.e., either a computer file (e.g., a VSAM or flat file) or a database (e.g., an ORACLE or IMS database).

Generic Data Description

A detailed description of the values for one or more data items, attribute classes, data fields, and/or module parameter. It includes format, measurement, and domain characteristics of the values.

Generic Data Description Domain

A domain that is specified as part of a generic data description.

Generic Data Description Unit of Measure

A unit of measure that is specified as part of a generic

data description.

Host

A computer in the IISS.

IMS Segment

A record type in a database that is controlled by IBM's IMS DBMS.

Inherited Attribute Class

A key class in the independent entity class of a relation class that has migrated to appear in the dependent entity class of that relation class.

Key Class

A group of one or more of an entity's attributes that can be used to uniquely identify the entity within its entity class. An entity can have more than one key. A key class is a collection of the attribute classes whose member attributes comprise the keys for the entities in an entity class. An entity class has the same number of key classes as each of its member entities has keys. For example, if each entity has three keys, the entity class has three key classes.

Key Class Member

An attribute use class that is part of a key class.

Keyword

A word that has been designated as a means of locating a generic object or a number of similar generic objects.

Model

A representation of the information requirements of all or part of an enterprise in terms of entity classes, relation classes, and attribute classes.

Object Type

Sets of attributes are, in relational terms, called objects. Objects participate in relationships with other

objects. Entities within the Common Data Model (see Generic Object in the CDM1 Doc. Control No. CCS620141000) are called OBJECT TYPES for the Integrated Information Support System.

Owned Attribute Class

A model attribute class that appears as an attribute use class in a model entity class and is not an inherited attribute class.

Program Control Block

A portion of a PSB that describes and controls how an IMS database can be accessed.

Program Specification Block

A group of logical views of IMS databases that is used for interacting with the IMS DBMS.

Record Set

An association between one record type, called the owner, and one or more other record types, called the members.

Record Set Member

A record type that is a member of a record set.

Record Type

A group of data values that are stored together as a unit in a computer file or database. A record type is the collection of all the records of the same kind, i.e., all the records that contain the same kind of data values.

Relation Class

An association between an entity in one entity class and one in another. A relationship has a label that describes the association. For example, a customer named ABC Corp. is associated with an order numbered 123 in a manner labeled "placed". A relation class is a collection of the identically labeled relationships between the members of the same two entity classes. Each relation class is either specific or non-specific.

In a specific relation class, one entity class is "independent" while the other is "dependent"; i.e., entities in the first can exist without being associated with any in the second, but those in the second cannot exist without being associated with one in the first. One key class from the independent entity class "migrates" through each specific relation class to appear in the dependent entity class as inherited attribute classes.

In an nonspecific relation class, neither entity class is dependent on the other; i.e., entities in either entity class can exist without being associated with any in the other. For convenience, one entity class is arbitrarily called "independent" and the other is called "dependent".

Segment Data Field

A data field is an IMS segment.

Subschema

The description, in the DDL of a CODASYL DBMS, of all or part of a database. For IISS, only one subschema is needed for a CODASYL database, and it must describe all the common data within the database that is to be accessible with NDML.

Tag Name

A unique name for an attribute use class within an entity class.

Unit of Measure

A standard scale for determining the magnitude of something. Examples include inch, foot, foot-inch, meter, ounce, pound, hour, minute, second, etc.

User View

A group of data items that a user wants to deal with as a group. It is similar to an entity class but does not necessarily meet all the conditions for being one, it can be thought of as an unnormalized entity class. A user view is often the result of combining several entity classes via relational join operations and selecting particular attribute use classes as data items via relational project operations.

APPENDIX C

REFERENCES

Related ICAM Documents included:

UM620141100	CDM Administrator's Manual
TEM620141000	CDM1, An IDEF1 Model of the Common Data Model
PRM620141200	Embedded NDML Programmer's Reference Manual
UM620141002	ICAM Definition Method for Data Modeling (IDEF1 - Extended)
DS620141200	Development Specification for the IISS NDML Precompiler Configuration Item
DS620141320	Development Specification for the IISS Aggregator Configuration Item
DS620141310	Development Specification for the IISS Distributed Request Supervisor Configuration Item

☆ U.S.Government Printing Office: 1987 — 748-061/60875

END

7-87

DTIC