CA-IDMS®

DML Reference — COBOL
with FORTRAN and RPG II Supplements
15.0

Computer Associates™

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

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation “as is” without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

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

Provided with “Restricted Rights” as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

Second Edition, October 2001

© 2001 Computer Associates International, Inc.
All rights reserved.

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

Contents

How to Use This Manual xvii

Chapter 1. Introduction to CA-IDMS Data Manipulation Language 1-1
1.1 About this chapter 1-3
1.2 Programming in the CA-IDMS environment 1-5
1.2.1 Accessing the database 1-5
1.2.1.1 Navigational DML statements 1-5
1.2.1.2 LRF DML statements 1-6
1.2.1.3 SQL DML statements 1-7
1.2.2 Programming in the online environment 1-7
1.3 Compiling and executing CA-IDMS programs 1-9
1.3.1 Compiling programs 1-9
1.3.2 Executing programs 1-10

Chapter 2. Precompiler Options 2-1
2.1 About this chapter 2-3
2.2 Dictionary ready override 2-4
2.2.1 Syntax 2-4
2.2.2 Parameters 2-4
2.3 Comment generation 2-5
2.3.1 Syntax 2-5
2.4 List generation 2-6
2.4.1 Syntax 2-6
2.4.2 Parameters 2-6
2.5 Log suppression 2-7
2.5.1 Syntax 2-7

Chapter 3. Communications Blocks and Error Detection 3-1
3.1 About this chapter 3-3
3.2 Communications blocks 3-4
3.2.1 IDMS communications block 3-4
3.2.2 LRC block 3-10
3.2.3 IDMS-DC communications block 3-12
3.3 ERROR-STATUS field and codes 3-18
3.3.1 DB status codes 3-18
3.3.2 DC status codes 3-23
3.4 Error detection 3-29
3.4.1 IDMS-STATUS routine 3-29
3.5 IDMS-STATUS routine used under DC 3-30

3.5.1 AUTOSTATUS protocols 3-33
3.5.1.1 Syntax 3-33
3.5.1.2 Parameters 3-33
3.5.2 USER-DEFINED protocols 3-34

Chapter 4. Precompiler-Directive Statements 4-1
4.1 About this chapter 4-3
4.2 IDENTIFICATION DIVISION 4-4
4.2.1 Syntax 4-4
4.2.2 Parameters 4-4
4.3 ENVIRONMENT DIVISION 4-5
4.3.1 Syntax 4-5
4.3.2 Parameters 4-5
4.4 DATA DIVISION 4-9
4.4.1 FILE SECTION 4-9
4.4.1.1 Syntax 4-9
4.4.1.2 Parameters 4-9
4.4.2 SCHEMA SECTION 4-10
4.4.2.1 Syntax 4-10
4.4.2.2 Parameters 4-10
4.4.3 MAP SECTION 4-10
4.4.3.1 Syntax 4-11
4.4.3.2 Parameters 4-11
4.4.4 WORKING-STORAGE and LINKAGE SECTIONs 4-12
4.4.4.1 Syntax 4-14
4.4.4.2 Parameters 4-15
4.5 PROCEDURE DIVISION 4-22
4.5.1 Syntax 4-22
4.5.2 Parameters 4-22

Chapter 5. Data Manipulation Language Statements 5-1
5.1 About data manipulation language (DML) 5-11
5.2 ABEND 5-19
5.2.1 Syntax 5-19
5.2.2 Parameters 5-19
5.2.3 Example 5-19
5.2.4 Status codes 5-19
5.3 ACCEPT 5-20
5.3.1 Syntax 5-20
5.3.2 Parameters 5-20
5.3.3 Example 5-21
5.3.4 Status codes 5-21
5.4 ACCEPT BIND ADDRESS 5-22
5.4.1 Syntax 5-22
5.4.2 Parameters 5-22
5.4.3 Example 5-22
5.4.4 Status codes 5-22
5.5 ACCEPT DATABASE STATISTICS 5-23
5.5.1 Syntax 5-23
5.5.2 Parameters 5-23
5.5.3 Example 5-24

5.5.4 Status codes 5-24
5.6 ACCEPT DB-KEY FROM CURRENCY 5-25
5.6.1 Syntax 5-25
5.6.2 Parameters 5-25
5.6.3 Example 5-25
5.6.4 Status codes 5-26
5.7 ACCEPT DB-KEY RELATIVE TO CURRENCY 5-27
5.7.1 Syntax 5-27
5.7.2 Parameters 5-27
5.7.3 Example 5-28
5.7.4 Status codes 5-28
5.8 ACCEPT page-info-location 5-29
5.8.1 Syntax 5-29
5.8.2 Parameters 5-29
5.8.3 Example 5-30
5.8.4 Status codes 5-30
5.9 ACCEPT PROCEDURE CONTROL LOCATION 5-31
5.9.1 Syntax 5-31
5.9.2 Parameters 5-31
5.9.3 Example 5-31
5.9.4 Status codes 5-31
5.10 ACCEPT TRANSACTION STATISTICS 5-32
5.10.1 Syntax 5-32
5.10.2 Parameters 5-32
5.10.3 Example 5-34
5.10.4 Status codes 5-34
5.11 ATTACH 5-35
5.11.1 Syntax 5-35
5.11.2 Parameters 5-35
5.11.3 Example 5-36
5.11.4 Status codes 5-36
5.12 BIND MAP 5-37
5.12.1 Syntax 5-37
5.12.2 Parameters 5-37
5.12.3 Example 5-38
5.12.4 Status codes 5-38
5.13 BIND PROCEDURE 5-39
5.13.1 Syntax 5-39
5.13.2 Parameters 5-39
5.13.3 Example 5-39
5.13.4 Status codes 5-39
5.14 BIND RECORD 5-41
5.14.1 Syntax 5-41
5.14.2 Parameters 5-41
5.14.3 Example 5-42
5.14.4 Status codes 5-42
5.15 BIND RUN-UNIT 5-43
5.15.1 Syntax 5-43
5.15.2 Parameters 5-43
5.15.3 Example 5-44

5.15.4 Status codes 5-45
5.16 BIND TASK 5-47
5.16.1 Syntax 5-47
5.16.2 Parameters 5-47
5.16.3 Example 5-47
5.16.4 Status codes 5-47
5.17 BIND TRANSACTION STATISTICS 5-48
5.17.1 Syntax 5-48
5.17.2 Example 5-48
5.17.3 Status codes 5-48
5.18 CHANGE PRIORITY 5-49
5.18.1 Syntax 5-49
5.18.2 Parameters 5-49
5.18.3 Example 5-49
5.18.4 Status codes 5-49
5.19 CHECK TERMINAL 5-50
5.19.1 Syntax 5-50
5.19.2 Parameters 5-50
5.19.3 Example 5-51
5.19.4 Status codes 5-51
5.20 COMMIT 5-53
5.20.1 Syntax 5-53
5.20.2 Parameters 5-53
5.20.3 Example 5-53
5.20.4 Status codes 5-53
5.21 CONNECT 5-55
5.21.1 Syntax 5-56
5.21.2 Parameters 5-56
5.21.3 Example 5-56
5.21.4 Status codes 5-57
5.22 DC RETURN 5-58
5.22.1 Syntax 5-60
5.22.2 Parameters 5-60
5.22.3 Example 5-62
5.22.4 Status codes 5-62
5.23 DELETE QUEUE 5-63
5.23.1 Syntax 5-63
5.23.2 Parameters 5-63
5.23.3 Example 5-63
5.23.4 Status codes 5-63
5.24 DELETE SCRATCH 5-65
5.24.1 Syntax 5-65
5.24.2 Parameters 5-65
5.24.3 Example 5-66
5.24.4 Status codes 5-66
5.25 DELETE TABLE 5-67
5.25.1 Syntax 5-67
5.25.2 Parameters 5-67
5.25.3 Example 5-67
5.25.4 Status codes 5-68
5.26 DEQUEUE 5-69

5.26.1 Syntax . . . 5-69
5.26.2 Parameters . . . 5-69
5.26.3 Example . . . 5-69
5.26.4 Status codes . . . 5-69
5.27 DISCONNECT . . . 5-71
5.28 Disconnecting a record from a set . . . 5-72
5.28.1 Syntax . . . 5-73
5.28.2 Parameters . . . 5-73
5.28.3 Example . . . 5-73
5.28.4 Status codes . . . 5-73
5.29 END LINE TERMINAL SESSION . . . 5-74
5.29.1 Syntax . . . 5-74
5.29.2 Example . . . 5-74
5.29.3 Status codes . . . 5-74
5.30 END TRANSACTION STATISTICS . . . 5-75
5.30.1 Syntax . . . 5-75
5.30.2 Parameters . . . 5-75
5.30.3 Example . . . 5-75
5.30.4 Status codes . . . 5-75
5.31 ENDPAGE . . . 5-77
5.31.1 Syntax . . . 5-77
5.31.2 Example . . . 5-77
5.31.3 Status codes . . . 5-77
5.32 ENQUEUE . . . 5-78
5.32.1 Syntax . . . 5-78
5.32.2 Parameters . . . 5-78
5.32.3 Examples . . . 5-79
5.32.4 Status codes . . . 5-79
5.33 ERASE . . . 5-81
5.33.1 Syntax . . . 5-81
5.33.2 Parameters . . . 5-82
5.33.3 Example . . . 5-82
5.33.4 Status codes . . . 5-85
5.34 ERASE (LRF) . . . 5-87
5.34.1 Syntax . . . 5-87
5.34.2 Parameters . . . 5-87
5.34.3 Example . . . 5-88
5.35 FIND/OBTAIN . . . 5-89
5.36 FIND/OBTAIN CALC/DUPLICATE . . . 5-90
5.36.1 Syntax . . . 5-90
5.36.2 Parameters . . . 5-90
5.36.3 Example . . . 5-91
5.36.4 Status codes . . . 5-91
5.37 FIND/OBTAIN CURRENT . . . 5-92
5.37.1 Syntax . . . 5-92
5.37.2 Parameters . . . 5-92
5.37.3 Example . . . 5-92
5.37.4 Status codes . . . 5-94
5.38 FIND/OBTAIN DB-KEY . . . 5-95
5.38.1 Syntax . . . 5-95

5.38.2 Parameters 5-95
5.38.3 Example 5-96
5.38.4 Status codes 5-96
5.39 FIND/OBTAIN OWNER 5-98
5.39.1 Syntax 5-98
5.39.2 Parameters 5-98
5.39.3 Example 5-99
5.39.4 Status codes 5-99
5.40 FIND/OBTAIN WITHIN SET USING SORT KEY 5-101
5.40.1 Syntax 5-101
5.40.2 Parameters 5-101
5.40.3 Example 5-103
5.40.4 Status codes 5-103
5.41 FIND/OBTAIN WITHIN SET/AREA 5-104
5.41.1 Syntax 5-105
5.41.2 Parameters 5-105
5.41.3 Example 5-106
5.41.4 Status codes 5-108
5.42 FINISH 5-110
5.42.1 Syntax 5-110
5.42.2 Parameters 5-110
5.42.3 Example 5-110
5.42.4 Status codes 5-110
5.43 FREE STORAGE 5-112
5.43.1 Syntax 5-112
5.43.2 Parameters 5-112
5.43.3 Example 5-112
5.43.4 Status codes 5-112
5.44 GET 5-114
5.44.1 Syntax 5-114
5.44.2 Parameters 5-114
5.44.3 Example 5-114
5.44.4 Status codes 5-114
5.45 GET QUEUE 5-116
5.45.1 Syntax 5-116
5.45.2 Parameters 5-116
5.45.3 Example 5-118
5.45.4 Status codes 5-118
5.46 GET SCRATCH 5-120
5.46.1 Syntax 5-120
5.46.2 Parameters 5-120
5.46.3 Example 5-122
5.46.4 Status codes 5-122
5.47 GET STORAGE 5-123
5.47.1 Syntax 5-123
5.47.2 Parameters 5-123
5.47.3 Example 5-126
5.47.4 Status codes 5-126
5.48 GET TIME 5-128
5.48.1 Syntax 5-128
5.48.2 Parameters 5-128

5.48.3 Example . . . 5-129
5.48.4 Status codes . . . 5-129
5.49 IF . . . 5-130
5.49.1 Syntax . . . 5-130
5.49.2 Parameters . . . 5-130
5.49.3 Examples . . . 5-131
5.49.4 Status codes . . . 5-131
5.50 INQUIRE MAP . . . 5-133
5.50.1 Syntax . . . 5-133
5.50.2 Parameters . . . 5-133
5.50.3 Example . . . 5-136
5.50.4 Syntax . . . 5-137
5.50.5 Parameters . . . 5-137
5.50.6 Example . . . 5-137
5.50.7 Syntax . . . 5-138
5.50.8 Parameters . . . 5-138
5.50.9 Example . . . 5-138
5.50.10 Syntax . . . 5-139
5.50.11 Parameters . . . 5-139
5.50.12 Examples . . . 5-141
5.50.13 Status codes . . . 5-142
5.51 KEEP CURRENT . . . 5-143
5.51.1 Syntax . . . 5-143
5.51.2 Parameters . . . 5-143
5.51.3 Example . . . 5-143
5.51.4 Status codes . . . 5-143
5.52 KEEP LONGTERM . . . 5-145
5.52.1 Syntax . . . 5-147
5.52.2 Parameters . . . 5-147
5.52.3 Lock-options . . . 5-148
5.52.4 Example . . . 5-149
5.52.5 Status codes . . . 5-149
5.53 LOAD TABLE . . . 5-151
5.53.1 Syntax . . . 5-151
5.53.2 Parameters . . . 5-151
5.53.3 Examples . . . 5-153
5.53.4 Status codes . . . 5-154
5.54 MAP IN . . . 5-155
5.54.1 Syntax . . . 5-155
5.54.2 Parameters . . . 5-155
5.54.3 Examples . . . 5-157
5.54.4 Status codes . . . 5-158
5.55 MAP OUT . . . 5-160
5.55.1 Syntax . . . 5-160
5.55.2 Parameters . . . 5-161
5.55.3 Examples . . . 5-164
5.55.4 Status codes . . . 5-165
5.56 MAP OUTIN . . . 5-167
5.56.1 Syntax . . . 5-167
5.56.2 Parameters . . . 5-167

5.56.3 Example 5-169
5.56.4 Status codes 5-169
5.57 MODIFY 5-171
5.57.1 Syntax 5-172
5.57.2 Parameters 5-172
5.57.3 Example 5-172
5.57.4 Status codes 5-172
5.58 MODIFY (LRF) 5-174
5.58.1 Syntax 5-174
5.58.2 Parameters 5-174
5.58.3 Example 5-175
5.59 MODIFY MAP 5-176
5.59.1 Syntax 5-176
5.59.2 Parameters 5-177
5.59.3 Examples 5-184
5.59.4 Status codes 5-185
5.60 OBTAIN (LRF) 5-186
5.60.1 Syntax 5-186
5.60.2 Parameters 5-186
5.60.3 Example 5-187
5.61 POST 5-189
5.61.1 Syntax 5-189
5.61.2 Parameters 5-189
5.61.3 Example 5-189
5.61.4 Status codes 5-189
5.62 PUT QUEUE 5-190
5.62.1 Syntax 5-190
5.62.2 Parameters 5-190
5.62.3 Example 5-191
5.62.4 Status codes 5-192
5.63 PUT SCRATCH 5-193
5.63.1 Syntax 5-193
5.63.2 Parameters 5-193
5.63.3 Example 5-194
5.63.4 Status codes 5-194
5.64 READ LINE FROM TERMINAL 5-195
5.64.1 Syntax 5-195
5.64.2 Parameters 5-195
5.64.3 Examples 5-196
5.64.4 Status codes 5-196
5.65 READ TERMINAL 5-198
5.65.1 Syntax 5-198
5.65.2 Parameters 5-198
5.65.3 Example 5-199
5.65.4 Status codes 5-200
5.66 READY 5-201
5.66.1 Syntax 5-201
5.66.2 Parameters 5-201
5.66.3 Example 5-202
5.66.4 Status codes 5-203
5.67 RETURN 5-204

5.67.1 Syntax . . . 5-204
5.67.2 Parameters . . . 5-204
5.67.3 Example . . . 5-205
5.67.4 Status codes . . . 5-206
5.68 ROLLBACK . . . 5-207
5.68.1 Syntax . . . 5-207
5.68.2 Parameters . . . 5-207
5.68.3 Example . . . 5-208
5.68.4 Status codes . . . 5-208
5.69 SEND MESSAGE . . . 5-209
5.69.1 Syntax . . . 5-209
5.69.2 Parameters . . . 5-209
5.69.3 Examples . . . 5-210
5.69.4 Status codes . . . 5-211
5.70 SET ABEND EXIT . . . 5-212
5.70.1 Syntax . . . 5-212
5.70.2 Parameters . . . 5-212
5.70.3 Examples . . . 5-212
5.70.4 Status codes . . . 5-213
5.71 SET TIMER . . . 5-214
5.71.1 Syntax . . . 5-214
5.71.2 Parameters . . . 5-214
5.71.3 Examples . . . 5-216
5.71.4 Status codes . . . 5-217
5.72 SNAP . . . 5-218
5.72.1 Syntax . . . 5-218
5.72.2 Parameters . . . 5-218
5.72.3 Example . . . 5-219
5.72.4 Status codes . . . 5-219
5.73 STARTPAGE . . . 5-220
5.73.1 Syntax . . . 5-220
5.73.2 Parameters . . . 5-220
5.73.3 Examples . . . 5-221
5.73.4 Status codes . . . 5-221
5.74 STORE . . . 5-223
5.74.1 Syntax . . . 5-224
5.74.2 Parameters . . . 5-225
5.74.3 Example . . . 5-225
5.74.4 Status codes . . . 5-226
5.75 STORE (LRF) . . . 5-228
5.75.1 Syntax . . . 5-228
5.75.2 Parameters . . . 5-228
5.75.3 Example . . . 5-229
5.76 TRANSFER CONTROL . . . 5-230
5.76.1 Syntax . . . 5-230
5.76.2 Parameters . . . 5-230
5.76.3 Examples . . . 5-231
5.76.4 Status codes . . . 5-231
5.77 WAIT . . . 5-232
5.77.1 Syntax . . . 5-232

5.77.2 Parameters . . . 5-232
5.77.3 Example . . . 5-233
5.77.4 Status codes . . . 5-233
5.78 WRITE JOURNAL . . . 5-234
5.78.1 Syntax . . . 5-234
5.78.2 Parameters . . . 5-234
5.78.3 Example . . . 5-235
5.78.4 Status codes . . . 5-235
5.79 WRITE LINE TO TERMINAL . . . 5-236
5.79.1 Syntax . . . 5-236
5.79.2 Parameters . . . 5-237
5.79.3 Examples . . . 5-238
5.79.4 Status codes . . . 5-238
5.80 WRITE LOG . . . 5-240
5.80.1 Syntax . . . 5-241
5.80.2 Parameters . . . 5-241
5.80.3 Example . . . 5-244
5.80.4 Status codes . . . 5-244
5.81 WRITE PRINTER . . . 5-246
5.81.1 Syntax . . . 5-247
5.81.2 Parameters . . . 5-247
5.81.3 Examples . . . 5-249
5.81.4 Status codes . . . 5-249
5.82 WRITE TERMINAL . . . 5-251
5.82.1 Syntax . . . 5-251
5.82.2 Parameters . . . 5-251
5.82.3 Example . . . 5-252
5.82.4 Status codes . . . 5-252
5.83 WRITE THEN READ TERMINAL . . . 5-254
5.83.1 Syntax . . . 5-254
5.83.2 Parameters . . . 5-254
5.83.3 Example . . . 5-257
5.83.4 Status codes . . . 5-257
5.84 Logical-record clauses . . . 5-258
5.84.1 WHERE . . . 5-258
5.84.2 Syntax . . . 5-258
5.84.3 Parameters . . . 5-259
5.84.4 Parameters . . . 5-259
5.84.5 Examples . . . 5-262
5.84.6 ON clause . . . 5-262
5.84.7 Syntax . . . 5-263
5.84.8 Parameters . . . 5-264
5.84.9 Examples . . . 5-264
5.84.10 Status codes . . . 5-264

Appendix A. DML Precompile, COBOL Compile, and Link-Edit JCL . . . A-1
A.1 About this appendix . . . A-3
A.2 Compiling a COBOL program . . . A-4
A.3 MVS JCL . . . A-6
A.4 VSE JCL . . . A-9
A.4.1 Local mode . . . A-10

A.4.2 IDMSLBLS Procedure A-11
A.5 BS2000 JCL A-17
A.6 CMS commands A-19
A.7 Link-edit considerations A-22
A.8 Passing Parameters to the Precompiler A-23

Appendix B. Sample Batch Program B-1
B.1 About this appendix B-3
B.2 Sample batch program as input to the DML compiler B-4
B.3 Sample batch program as output from the DML compiler B-10
B.4 Sample batch program from the COBOL precompiler B-21

Appendix C. Sample Online Program C-1
C.1 About this appendix C-3
C.2 Application components C-4
C.3 Application runtime requirements C-6
C.3.1 Sample online COBOL program as input to the DML precompiler . . C-6
C.3.2 Sample online COBOL program as output from the DML precompiler C-11
C.3.3 Sample online COBOL program from the COBOL compiler C-25

Appendix D. CA-IDMS Call Formats D-1
D.1 About this appendix D-3
D.2 DB call formats D-4
D.3 DC call formats D-12
D.4 Sample COBOL program calling IDMSIN01 . . . . . D-15

Appendix E. CA-IDMS Keywords E-1
E.1 About this appendix E-3
E.2 List of keywords E-4

Appendix F. Notes to Teleprocessing Monitor Users F-1
F.1 About this appendix F-3
F.2 TP monitor coding guidelines F-4
F.3 TP monitor coding requirements F-5

Appendix G. EMPLOYEE Database Definition G-1
G.1 About this appendix G-3
G.2 IDMSRPTS utility report listings G-4
G.3 EMPLOYEE database structure diagram G-12

Appendix H. VS COBOL II Support H-1
H.1 About this appendix H-3
H.2 Features supported by CA-IDMS H-4
H.3 Features not supported by CA-IDMS H-7

Appendix I. Considerations for IBM Language Environment I-1
I.1 Overview I-3
I.2 Considerations about LE runtime I-4
I.3 Running LE-compliant compiler programs under CA-IDMS/DC I-6
I.4 Supported LE functions I-10

I.5 Unsupported LE functions I-11
I.6 Performance improvements with RHDCLEFE I-12

Appendix J. 18-Byte Communications Blocks J-1
J.1 About this appendix J-3
J.2 18-byte IDMS block J-4
J.3 18-byte IDMS-DC block J-5

Appendix K. FORTRAN DML Precompiler K-1
K.1 About this appendix K-3
K.2 Compiling a FORTRAN program with DML statements K-4
K.3 IDMS communications block K-5
K.3.1 Record and element synonyms K-5
K.4 Compiler options statements K-7
K.5 Compiler-directive statements K-8
K.5.1 INVOKE statement K-8
K.5.1.1 Syntax K-9
K.5.1.2 Parameters K-9
K.5.1.3 Usage K-10
K.5.2 INCLUDE IDMS K-10
K.5.2.1 Syntax K-10
K.5.2.2 Parameters K-11
K.5.2.3 Usage K-12
K.5.3 INCLUDE IDMS SUBSCHEMA-BINDS K-13
K.5.3.1 Syntax K-14
K.5.3.2 Parameters K-14
K.5.3.3 Usage K-14
K.5.4 INCLUDE IDMS MODULE K-14
K.5.4.1 Syntax K-15
K.5.4.2 Parameters K-15
K.6 Control statements K-16
K.7 Retrieval statements K-17
K.8 Modification statements K-18
K.9 Save statements K-19
K.10 ACCEPT IDMS STATISTICS K-20
K.10.1.1 Syntax K-20
K.10.1.2 Usage K-20
K.11 Logical Record Facility statements K-23
K.12 Program compilation K-24
K.13 Link-edit considerations K-29
K.14 Sample CA-IDMS FORTRAN program K-30

Appendix L. RPG II DML Precompiler L-1
L.1 About this appendix L-3
L.2 The DMLR precompiler (IDMSDMLR) L-4
L.2.1 Standard and non-standard data transfer L-5
L.3 Programming considerations L-7
L.3.1 IDMS communications block L-7
L.3.2 Synonym processing L-9
L.3.3 Copying non-IDMS record descriptions L-10
L.3.4 Logical Record Facility programming considerations L-10

L.3.4.1 LRC block L-11
L.3.4.2 Path status L-11
L.3.4.3 DML statement restrictions L-12
L.3.5 Precompiler options L-13
L.4 Database access under standard RPG II data transfer L-15
L.4.1 Compiler-directive statements L-16
L.4.1.1 Subschema/Schema identification L-17
L.4.1.2 Defining records/logical records L-20
L.4.1.3 Copying input records/logical records L-22
L.4.1.4 Copying output records/logical records L-25
L.4.1.5 Copying modules L-27
L.5 Database access under non-standard RPG II data transfer L-29
L.5.1 Compiler-directive statements L-30
L.5.1.1 Subschema/Schema identification L-30
L.5.1.2 Copying subschema records L-33
L.5.1.3 Copying individual record descriptions L-34
L.5.1.4 Copying modules L-35
L.5.1.5 Binding records L-36
L.5.2 Functional DML statements L-37
L.5.3 Control Statements L-38
L.5.4 Logical record DML statements L-39
L.6 Database access under combined RPG II data transfer L-40
L.7 Compiling and link-editing L-42
L.7.1 Compiling an RPG II Program L-42
L.7.2 JCL L-43
L.8 Link-edit considerations L-47
L.9 Standard RPG II data transfer return codes L-48

Appendix M. Optional Online COBOL Functionality M-1
M.1 About this appendix M-3
M.2 COBOL II and LE COBOL task management M-4
M.3 Suppression of certain program checks M-7
M.4 Loading VS-COBOL programs into XA storage M-10

Index X-1

How to Use This Manual

What this manual is about

This manual contains reference material for writing applications programs in the COBOL language to use CA-IDMS DB, DC, and UCF services.

Who should use this manual

This manual is intended to be used by COBOL programmers whose programs access CA-IDMS databases and who want to use the DC/UCF system facilities.

In addition, this manual contains two language supplements for FORTRAN and RPG II programmers (found in appendixes K and L, respectively).

Programmers using Assembly language or PL/I should refer to *CA-IDMS DML Reference — Assembler* or *CA-IDMS DML Reference — PL/I*.

How this manual is organized

The information in this manual is grouped as follows:

- **Chapters 1 through 4** present introductory material about preparing, compiling, and executing COBOL applications under CA-IDMS; about the Data Manipulation Language (DML) compiler options and statements; and about communications blocks and error detection in the CA-IDMS COBOL environment
- **Chapter 5** presents DML statements
- **Appendixes A through J** present supporting information about JCL, teleprocessing applications, CA-IDMS call formats and keywords, 18-byte communications blocks, VS COBOL II and LE370 COBOL support, sample programs, and a sample database
- **Appendixes K and L** present information about using DML with FORTRAN and RPG II, respectively

How product names are referenced

This manual uses the term CA-IDMS to refer to any one of the following CA-IDMS components:

- CA-IDMS/DB — The database management system
- CA-IDMS/DC — The data communications system and proprietary teleprocessing monitor
- CA-IDMS/UCF — The universal communications facility for accessing CA-IDMS database and data communications services through another teleprocessing monitor, such as CICS
- CA-IDMS/DDS — The distributed database system

This manual uses the terms DB, DC, UCF, DC/UCF, and DDS to identify the specific CA-IDMS component only when it is important to your understanding of the product.

Related documentation

For further information related to this manual, refer to the following documents:

- *CA-IDMS DML Reference — Assembler*
- *CA-IDMS Database Administration*
- *IDD DDDL Reference*
- *CA-IDMS Messages and Codes*
- CA-IDMS installation manual for your operating system
- *CA-IDMS Mapping Facility*
- *CA-IDMS DML Reference — PL/I*
- *CA-IDMS System Generation*
- *CA-IDMS System Operations*

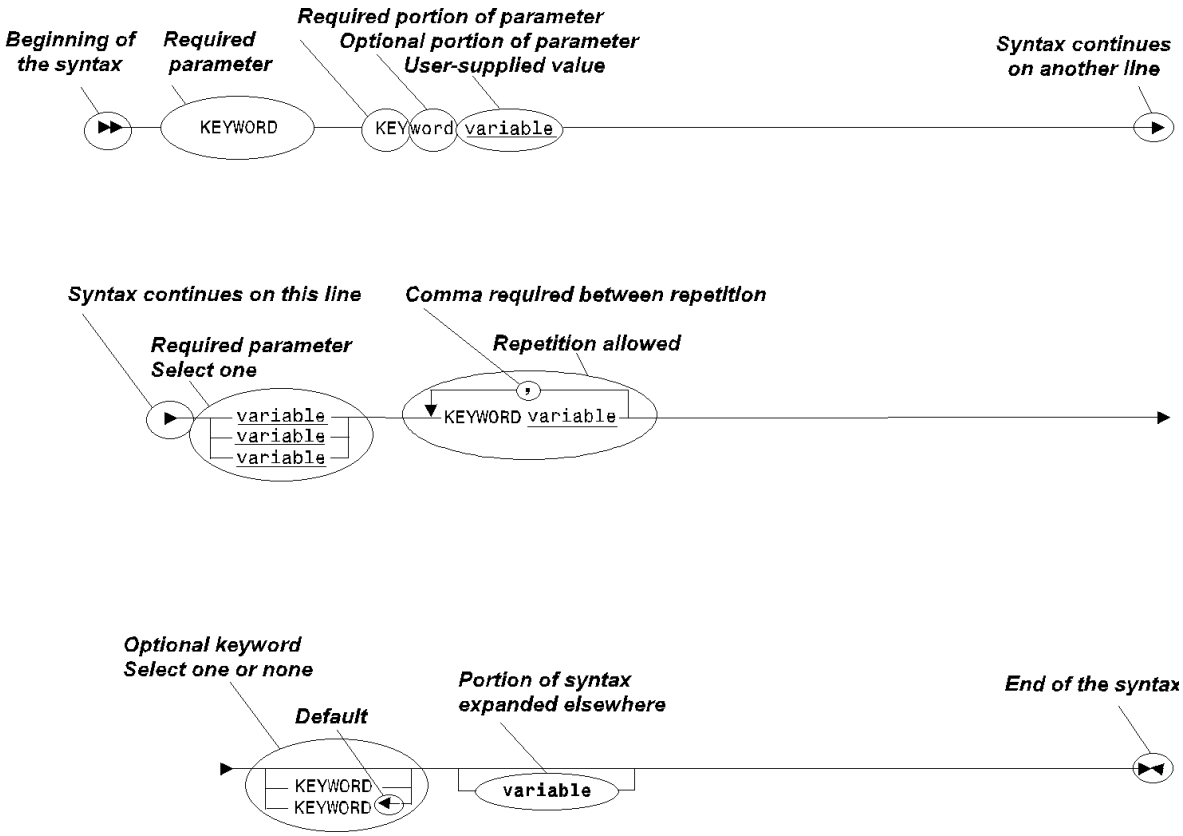
Understanding syntax diagrams

Look at the list of notation conventions below to see how syntax is presented in this manual. The example following the list shows how the conventions are used.

`UPPERCASE`
`OR`
`SPECIAL CHARACTERS`
	Represents a required keyword, partial keyword, character, or symbol that must be entered completely as shown.

`lowercase`	Represents an optional keyword or partial keyword that, if used, must be entered completely as shown.
`underlined lowercase`	Represents a value that you supply.
◄	Points to the default in a list of choices.
`lowercase bold`	Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.
►►────────────────	Shows the beginning of a complete piece of syntax.
────────────────►◄	Shows the end of a complete piece of syntax.
────────────────►	Shows that the syntax continues on the next line.
►────────────────	Shows that the syntax continues on this line.
────────────────►─	Shows that the parameter continues on the next line.
─►────────────────	Shows that a parameter continues on this line.
►── parameter ────►	Shows a required parameter.
►┬─ parameter ─┬──► └─ parameter ─┘	Shows a choice of required parameters. You must select one.
►┬──────────┬──► └─ parameter ─┘	Shows an optional parameter.
►┬──────────┬──► ├─ parameter ─┤ └─ parameter ─┘	Shows a choice of optional parameters. Select one or none.
►─┬─↓─ parameter ─┬──►	Shows that you can repeat the parameter or specify more than one parameter.
►─┬─↓─ parameter ─┬──► └──── , ────┘	Shows that you must enter a comma between repetitions of the parameter.

Sample syntax diagram



Chapter 1. Introduction to CA-IDMS Data Manipulation Language

1.1 About this chapter 1-3
1.2 Programming in the CA-IDMS environment 1-5
1.2.1 Accessing the database 1-5
1.2.1.1 Navigational DML statements 1-5
1.2.1.2 LRF DML statements 1-6
1.2.1.3 SQL DML statements 1-7
1.2.2 Programming in the online environment 1-7
1.3 Compiling and executing CA-IDMS programs 1-9
1.3.1 Compiling programs 1-9
1.3.2 Executing programs 1-10

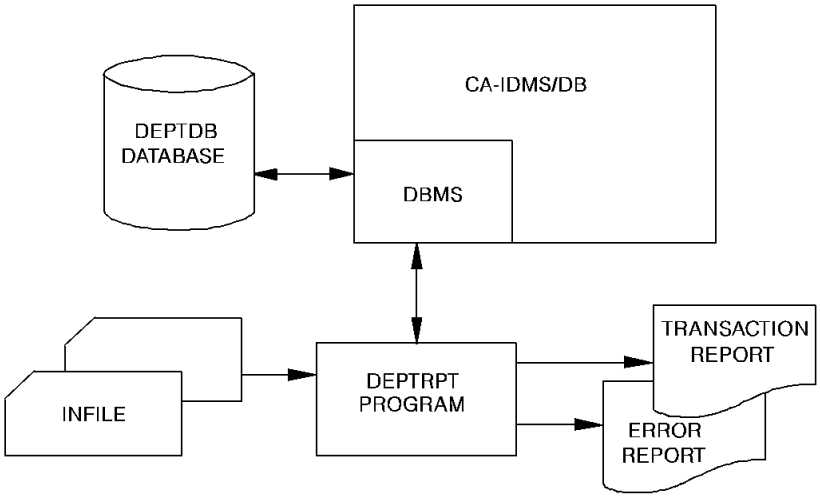
1.1 About this chapter

The CA-IDMS data manipulation language (DML) consists of statements that direct CA-IDMS database (DB) and data communications (DC) processing. DML statements are coded in the program source as if they were a part of the host language. The precompiler converts DML statements into standard COBOL statements and performs source-level error checking.

Depending on the operating environment, your program will use different sets of DML statements. For example, a batch program uses only database DML statements; an online program can use both database and data communications DML statements.

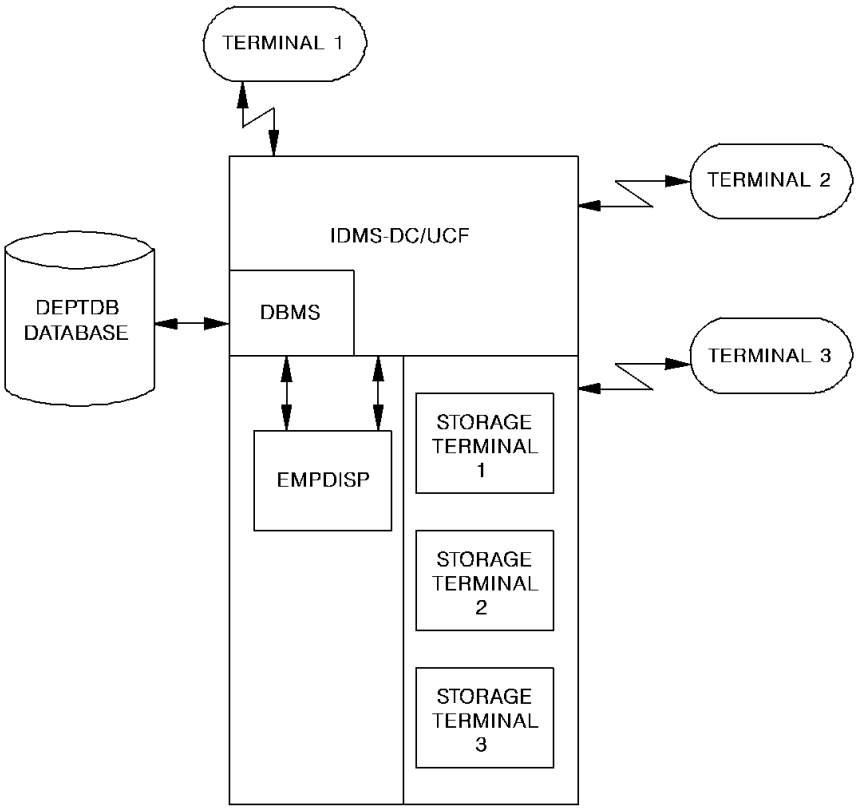
Batch processing typically involves large volumes of transactions, sequential processing, and output in the form of files and reports. Batch programs use database DML statements only.

The following figure illustrates the flow of a typical batch application. Input to DEPTRPT consists of department IDs. Output consists of a listing of departments and their employees. The error report lists the department IDs of missing and empty departments.



Online processing typically involves transaction requests entered from terminals connected directly to the computer, transaction results displayed at the terminal, multiple requests from multiple sources, and sharing one copy of a program among multiple users. Additionally, online processing is immediate; fast response time is essential in processing large volumes of transactions from multiple online users. Online programs use data communications DML statements and can include database DML statements.

The following figure illustrates the flow of a typical online application. EMPDISP retrieves information for an operator-specified employee ID. Output to the terminal consists of DEPARTMENT, EMPLOYEE, JOB, and OFFICE information.



The CA-IDMS programming environment is outlined below, followed by a discussion of compile and run-time considerations.

1.2 Programming in the CA-IDMS environment

CA-IDMS DML statements are divided into two categories:

- **Database** statements perform retrieval and update functions in either the batch or the online environment. There are three categories of database DML statements:
  - **Navigational** statements access database records and sets, one record at a time
  - **LRF** statements access groups of database records using the Logical Record Facility (LRF)
  - **SQL** statements access groups of database records using the Structured Query Language (SQL)
- **Data communications** (also called online) statements, request data communications services such as for online programs

You can include database DML statements in batch programs or combine them with data communications DML statements in online programs that require database access. A discussion of accessing the database by using DML statements is presented below, followed by a discussion of additional considerations for coding online programs.

1.2.1 Accessing the database

Database access under CA-IDMS can be accomplished by using navigational, LRF DML, or SQL DML statements. Navigational statements are used with a subschema usage mode of either DML or MIXED.

LRF DML statements, which use the Logical Record Facility (LRF), are used with a subschema usage mode of either LR or MIXED.

SQL DML statements, which use Structured Query Language, access records without reference to subschemas.

Some statements, such as BIND RUN-UNIT, READY, and FINISH, are used in all three environments. They are noted in the individual discussions of each DML statement in Chapter 5, “Data Manipulation Language Statements” on page 5-1.

Navigational, LRF, and SQL DML statements are discussed separately below.

1.2.1.1 Navigational DML statements

Navigational DML statements access database records and sets one record at a time, checking and maintaining currency in order to assure correct results. Navigational DML statements give you control over error checking and flexibility in choosing database access strategy. To use navigational DML statements, you must have a thorough knowledge of the database structure. For an example of a data structure diagram, refer to Appendix G, “EMPLOYEE Database Definition” on page G-1.

Navigational DML statements provide:

- **Control over error checking** — You can check the result of each navigational statement, enabling more thorough error detection
- **Flexibility in choosing database access strategy** — You can enter the database either sequentially (area sweep), by using a symbolic key value (CALC), or by using a database key value (DIRECT)

Navigational DML statements are grouped into four categories:

- **Control** statements initiate and terminate processing, effect recovery, prevent concurrent updates, and evaluate set conditions
- **Retrieval** statements locate data in the database and make it available to the application program
- **Modification** statements update the database
- **Accept** statements pass database keys, storage address information, and statistics to the program

1.2.1.2 LRF DML statements

LRF DML statements use the Logical Record Facility (LRF) to access database records. LRF allows you to access fields from multiple database records as if they were data fields in a single record. LRF DML statements allow you to specify selection criteria (by using the WHERE clause) that enable you to access only those logical records you need.

LRF DML statements provide:

- **Easy access to database records** — You need not be familiar with database structure; your programs need not include database navigation logic.
- **Data flexibility** — Modification and recompilation of LRF programs are not necessarily required when the physical or logical structure of the database is changed.
- **Run-time efficiency** — LRF minimizes communication between the program and the DBMS.

The LRF DML statements are listed below:

- **ERASE** deletes a logical record as specified in the path definition
- **MODIFY** modifies a logical record as specified in the path definition
- **OBTAIN** retrieves a logical record as specified in the path definition
- **STORE** stores a new logical record as specified in the path definition

1.2.1.3 SQL DML statements

You can use SQL DML to access the same databases you access using navigational DML. Additionally, you can use SQL DML to access databases that have been defined using SQL DDL.

Using SQL DML, you do not have to be familiar with database structure and your programs do not have to include database navigation logic.

You can perform the following functions using SQL DML statements:

- Select rows
- Update rows
- Delete rows
- Insert rows

►► For more information about SQL DML statements, refer to *CA-IDMS SQL Reference*.

1.2.2 Programming in the online environment

The CA-IDMS DC system is fully integrated with the CA-IDMS DBMS and the data dictionary. It enables you to request both data communications and database services through standard subroutine calls generated by the precompiler from DML statements. The following figure illustrates a typical stream of online DML statements in a COBOL program. This example maps in a user-specified employee ID, retrieves and displays the specified information, and performs a DC RETURN naming TSK02 as the next task to be performed.

```
PROCEDURE DIVISION.
    BIND MAP EMPMAPLR.
    BIND MAP EMPMAPLR RECORD EMPLOYEE.
    ACCEPT TASK CODE INTO TASK-CODE-IN.
    IF TASK-CODE-IN = 'TSK01'
                    GO TO INITIAL-MAPOUT.
    MAP IN USING EMPMAPLR.
         .
         .
 navigational, LRF, or SQL database DML statements
         .
         .
    MAP OUT USING EMPMAPLR
            OUTPUT DATA IS YES
            MESSAGE IS DISPLAY-MESSAGE LENGTH 80.
    DC RETURN NEXT TASK CODE 'TSK02'.
```

Online DML statements, which request CA-IDMS to perform data communications services, are grouped into nine categories:

1. **Program management** statements govern flow of control and abend processing
2. **Storage management** statements allocate and release variable storage

3. **Task management** statements provide run-time services that enhance control over task processing
4. **Time management** statements obtain the time and date, and define time-related events
5. **Scratch management** statements create, delete, or retrieve records from the scratch area
6. **Queue management** statements create, delete, or retrieve records in a queue area
7. **Terminal management** statements transfer data between the application program and a terminal
8. **Utility function** statements retrieve task-related information or statistics, send messages, and monitor access to database records
9. **Recovery** statements perform functions relating to database, scratch, and queue area recovery in the event of a system failure

1.3 Compiling and executing CA-IDMS programs

A CA-IDMS COBOL source program contains DML statements that are processed by the precompiler. The precompiler converts DML statements into COBOL CALL statements and copies information maintained in the data dictionary into the application program. After successful compilation and link editing, the application program can be executed. The compilation and run-time processes are described separately below.

1.3.1 Compiling programs

There are three components that prepare a COBOL DML program for execution: the precompiler, the COBOL compiler, and the linkage editor.

1. The precompiler converts DML statements in the source program to COBOL CALL statements and copies information maintained in the data dictionary into the application program. For example, database record descriptions, file definitions, map records, map definitions, and other predefined modules such as the IDMS communications block can be copied into the program.

   Output from the precompiler is a source file that serves as input to the COBOL precompiler and as an optional source listing. The output file differs from the source input to the precompiler in the following ways:

   - Source code (such as the IDMS communications block and the IDMS-STATUS routine) has been added to the program.
   - DML statements have been replaced by COBOL CALL statements and changed to comment entries (asterisk in column 7).

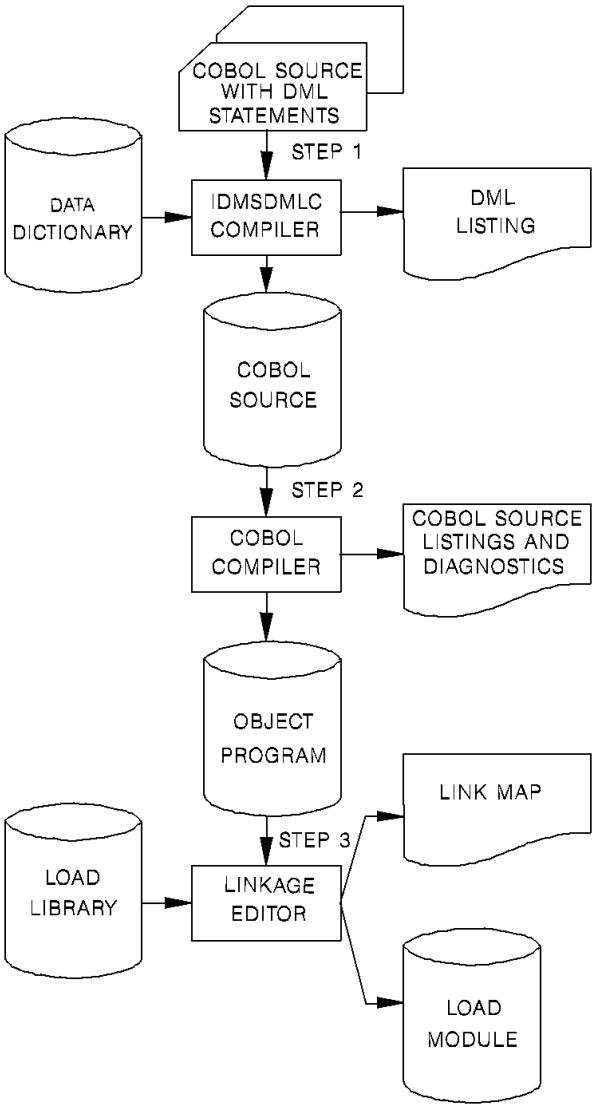
   Additionally, the precompiler produces a listing of the following errors:

   - Incorrect DML entries
   - Statements inconsistent with the program's declared subschema view
   - Any other error conditions detected during DMLC processing
   - Warning messages indicating source code conditions that could adversely affect run units using the program

2. The COBOL compiler compiles the source program after it has been successfully processed by the precompiler. Output from the COBOL compiler consists of an object program and a source listing that includes any generated diagnostics.
3. The linkage editor link edits the object program into a specified load library. Output from the linkage editor consists of a load module (or phase) and a link map.

The job control language required to execute each step is listed in Appendix A, “DML Precompile, COBOL Compile, and Link-Edit JCL” on page A-1.

The component steps needed to prepare a COBOL DML program for execution are illustrated in the following figure:



1.3.2 Executing programs

At run time, CA-IDMS requests are treated as application program subroutine calls. When the subroutine call is executed, control passes to the DBMS or the DC system, which processes the requested function.

A CA-IDMS program must be defined to the CA-IDMS system in which it will operate. The program can be defined either at system generation or at run time by using a DCMT VARY DYNAMIC PROGRAM command.

The following COBOL features are *not* used in programs running in an online environment under CA-IDMS:

- ENVIRONMENT and DATA DIVISION entries normally associated with file management (for example, INPUT-OUTPUT SECTION, FILE SECTION)
- The Report Writer and Segmentation features, as well as features invoked by the SORT, EXHIBIT, TRACE, DISPLAY, ACCEPT, STRING, UNSTRING, and INSPECT commands. The EXAMINE and TRANSFORM verbs, though valid under VS/COBOL, will not compile under VS/COBOL-II since the logic of these verbs has been incorporated into the inspect verb and are not valid verbs in COBOL-II. Additionally, the EXTERNAL clause of the record statement is invalid for all versions of COBOL.

  Note: Under the BS2000 operating system, INSPECT, STRING, and UNSTRING commands do not issue a supervisor call, so they may be used in programs running under the DC system.
- The I/O statements READ, WRITE, OPEN, and CLOSE
- The COBOL compiler DEBUG option; the COUNT, FLOW, STATE, ENDJOB, TEST, RESIDENT, DYNAM, and SYMDMP commands (OS only); or the COUNT, FLOW, STATE, STXIT, and SYMDMP commands (VSE only)
- Any feature that can lead to the issuance of a supervisor call (SVC)

These features lead to a supervisor call (SVC), which will inhibit system performance and can also crash the DC system.

Chapter 2. Precompiler Options

2.1 About this chapter 2-3
2.2 Dictionary ready override 2-4
2.2.1 Syntax 2-4
2.2.2 Parameters 2-4
2.3 Comment generation 2-5
2.3.1 Syntax 2-5
2.4 List generation 2-6
2.4.1 Syntax 2-6
2.4.2 Parameters 2-6
2.5 Log suppression 2-7
2.5.1 Syntax 2-7

2.1 About this chapter

This chapter contains the syntax for CA-IDMS COBOL precompiler options. These options, included as special format entries in the COBOL source code input to the precompiler, are used to:

- Override the DDLDML area default usage mode
- Enable the printing of data dictionary and subschema comments
- Control the generation of precompiler source listings
- Suppress the logging of program activity statistics

Each of these options is discussed separately below.

2.2 Dictionary ready override

When the DDLDML area of the data dictionary (that is, the main area of the dictionary accessed by the precompiler) is readied, a number of different options are available. The default mode used is shared update. Shared update mode readies the DDLDML area for both retrieval and update and allows other concurrently executing run units to ready the DDLDML area in shared update or shared retrieval usage mode. An application program can override the default usage mode by specifying either retrieval or protected update usage.

2.2.1 Syntax

```
Begin in column 7.

►►─┬─ *RETRIEVAL ──────────┬──────────────────────────────────────►◄
   └─ *PROTECTED-UPDATE ─┘
```

2.2.2 Parameters

***RETRIEVAL**

Readies the DDLDML area for retrieval only and allows other concurrently executing run units to open the area in shared retrieval, shared update, protected retrieval, or protected update usage modes.

Note: If the DDLDML area is readied for retrieval only, no program activity statistics can be logged.

***PROTECTED-UPDATE**

Readies the DDLDML area for both retrieval and update and allows other concurrently executing run units to ready the area in retrieval usage mode only. The protected update usage mode prevents concurrent update of the area by run units executing under the same central version.

If used, the dictionary ready override statement must precede all source input statements.

2.3 Comment generation

The *SCHEMA-COMMENTS option causes schema-defined data-item comments and IDD-defined record-element comments in the data dictionary to be printed on the precompiler source listing and inserted into the program.

2.3.1 Syntax

```
►►── *SCHEMA-COMMENTS ──────────────────────────────────────────►◄
```

Code the *SCHEMA-COMMENTS statement after the dictionary ready override statements (if any) and before any DML or COBOL statements: If the *SCHEMA-COMMENTS statement is not included with the input, comment lines are not generated.

2.4 List generation

The source statement listing output by the precompiler can be turned on or off by inserting a list generation option into the source program.

2.4.1 Syntax

```
►►─┬─ *DMLIST ──────────────────────────────────────────────►◄
   └─ *NODMLIST ◄ ─┘
```

2.4.2 Parameters

***DMLIST**
Specifies that the source listing is to be generated for the statements that follow.

***NODMLIST**
Specifies that no source listing is to be generated for the statements that follow.

This is the default.

Generation of the list can be turned on or off any number of times within one source program by inserting appropriate *DMLIST and/or *NODMLIST entries in the code.

Note: A listing of error messages is always produced. The *DMLIST option controls output of the processor source listing.

2.5 Log suppression

The *NO-ACTIVITY-LOG option suppresses the logging of program activity statistics. The precompiler generates and logs the following program activity statistics unless the *NO-ACTIVITY-LOG option is specified:

- Program name
- Language
- Date last compiled
- Number of lines
- Number of compilations
- Date created
- Subschema name (if any)
- File statistics
- Database access statistics (records and modules copied from the data dictionary; subprograms called; and records, sets, and areas accessed by DML verbs)

2.5.1 Syntax

```
►►── *NO-ACTIVITY-LOG ──────────────────────────────────────────►◄
```

The *NO-ACTIVITY-LOG statement follows the NODENAME/DBNAME and dictionary ready override statements.

Note: Program activity statistics cannot be logged if the DDLDML area is readied for retrieval only.

Chapter 3. Communications Blocks and Error Detection

3.1 About this chapter 3-3
3.2 Communications blocks 3-4
3.2.1 IDMS communications block 3-4
3.2.2 LRC block 3-10
3.2.3 IDMS-DC communications block 3-12
3.3 ERROR-STATUS field and codes 3-18
3.3.1 DB status codes 3-18
3.3.2 DC status codes 3-23
3.4 Error detection 3-29
3.4.1 IDMS-STATUS routine 3-29
3.5 IDMS-STATUS routine used under DC 3-30
3.5.1 AUTOSTATUS protocols 3-33
3.5.1.1 Syntax 3-33
3.5.1.2 Parameters 3-33
3.5.2 USER-DEFINED protocols 3-34

3.1 About this chapter

This chapter describes the 16-byte communications block available under CA-IDMS. These blocks return status information about requested database and data communications services to the application program. This chapter also describes the ERROR-STATUS field in the IDMS and IDMS-DC communications blocks, status codes, and error detection routines.

►► For information on 18-byte communications blocks, see Appendix J, “18-Byte Communications Blocks” on page J-1.

3.2 Communications blocks

Communications blocks return status information about requested database and data communications services to the application program. Depending on the usage mode (LR, DML, or MIXED) defined in the subschema, your program will use one or two of the following blocks:

- **IDMS communications block** — The IDMS communications block is used when the operating mode is either BATCH or BATCH-AUTOSTATUS.
- **Logical-record request control (LRC) block** — The LRC block is used when the subschema usage mode is either LR or MIXED.

  The LRC block is copied in with either the IDMS communications block (operating mode of BATCH or BATCH-AUTOSTATUS) or the IDMS-DC communications block (operating mode of IDMS-DC or DC-BATCH).
- **IDMS-DC communications block** — The IDMS-DC communications block is used when the operating mode is either IDMS-DC or DC-BATCH.

Each of these blocks is discussed in detail below. For more information on operating modes and protocols, see Chapter 4, “Precompiler-Directive Statements” on page 4-1.

3.2.1 IDMS communications block

The IDMS communications block is used when the operating mode is either BATCH or BATCH-AUTOSTATUS; it serves as an interface between the database management system (DBMS) and the application program. Whenever a run unit issues a call to the DBMS for a database operation, the DBMS returns information about the outcome of the requested service to the application program's IDMS communications block.

The data description (identified as SUBSCHEMA-CTRL) of the IDMS communications block is copied from the data dictionary into the WORKING-STORAGE SECTION or LINKAGE SECTION of the program. When you submit the program to the precompiler, the IDMS communications block is copied automatically unless an IDMS-RECORDS MANUAL statement is included in the ENVIRONMENT DIVISION. In that case, the program can explicitly call in the data description by using a COPY IDMS SUBSCHEMA-CTRL statement.

For more information on the IDMS-RECORDS MANUAL and the COPY IDMS statements, see Chapter 4, “Precompiler-Directive Statements” on page 4-1.

You should examine the ERROR-STATUS field of the IDMS communications block after every call to the DBMS. Depending on the value contained in this field, you should perform the IDMS-STATUS routine (see 3.4.1, “IDMS-STATUS routine” on page 3-29 later in this chapter). For example, if the ERROR-STATUS field contains the value 0307 (DB-END-OF-SET) while walking a set, you should perform end-of-set processing; otherwise, IDMS-STATUS should be performed.

The following figure show the layout of the 16-byte IDMS communications block. Each field is described separately following the figure.

16-CHARACTER IDMS COMMUNICATIONS BLOCK

		Field	Data Type	Length (bytes)	Suggested Initial Value
*	1 8	PROGRAM-NAME	Alphanumeric	8	Program Name
	9 12	ERROR-STATUS	Alphanumeric	4	'1400'
	13 16	DBKEY	Binary	4 (Fullword)	0000
	17 32	RECORD-NAME	Alphanumeric	16	Spaces
	33 48	AREA-NAME	Alphanumeric	16	Spaces
	49 64	ERROR-SET	Alphanumeric	16	Spaces
	65 80	ERROR-RECORD	Alphanumeric	16	Spaces
	81 96	ERROR-AREA	Alphanumeric	16	Spaces
**	97 100	PAGE-INFO	Binary	4 (Fullword)	0000
	97 ... 196	IDBMSCOM-AREA	Alphanumeric	100	Low Values
	197 200	DIRECT-DBKEY	Binary	4 (Fullword)	0000
NON-CICS	201 207	DATABASE-STATUS	Alphanumeric	7	Spaces
	208	FILLER	...	1	...
	209 212	RECORD-OCCUR	Binary	4 (Fullword)	0000
	213 216	DML-SEQUENCE	Binary	4 (Fullword)	0000
CICS	201 216	FILLER	...	16	Spaces
	217 223	DATABASE-STATUS	Alphanumeric	7	Spaces
	224	FILLER	...	1	...
	225 228	RECORD-OCCUR	Binary	4 (Fullword)	0000
	229 232	DML-SEQUENCE	Binary	4 (Fullword)	0000

* word aligned
** PAGE-INFO-GROUP overlays bytes 97 and 98 and PAGE-INFO-DBK-FORMAT overlays bytes 99 and 100. Both of these fields are binary datatype, each with a length of two bytes. Suggested initial values for both are 00. Together these two fields represent PAGE-INFO.

The IDMS-DB communications block contains the following fields that describe program status information:

Field name	Description
PROGRAM-NAME	The name of the program being executed, as defined in the program's IDENTIFICATION DIVISION. This field is initialized automatically at the beginning of program execution if the program contains a COPY IDMS SUBSCHEMA-BINDS statement in its PROCEDURE DIVISION. Otherwise, it must be initialized by the programmer.
ERROR-STATUS	An alphanumeric value indicating the outcome of the last DML statement executed. The ERROR-STATUS field must be initialized to 1400 by the program. The ERROR-STATUS field is updated by the DBMS after (attempted) performance of a requested database service and before control is returned to the program. A program that consists of more than one run unit must reinitialize the ERROR-STATUS field to 1400 after finishing one run unit and before binding the next.
DBKEY	The database key of the last record accessed by the run unit. For example, after successful execution of a FIND command, DBKEY is updated with the database key of the located record. DBKEY is not changed if the call to the DBMS results in a nonzero status condition.
RECORD-NAME	The name of the last record accessed successfully by the run unit. This field is left-justified and padded with spaces on the right.
AREA-NAME	The name of the last area accessed successfully by the run unit. This field is left-justified and padded with spaces on the right.
ERROR-SET	The name of the set involved in the last operation to produce a nonzero status code. This field is left-justified and padded with spaces on the right.
ERROR-RECORD	The name of the record involved in the last operation to produce a nonzero status code. This field is left-justified and padded with spaces on the right.

Field name	Description
ERROR-AREA	The name of the area involved in the last operation to produce a nonzero status code. This field is left-justified and padded with spaces on the right.

PAGE-INFO	Two binary halfwords that represent the page information associated with the last record accessed by the run unit. PAGE-INFO is not changed if the call to the DBMS results in a non-zero status. The first halfword (PAGE-INFO-GROUP) represents the page group number. The second halfword (PAGE-INFO-DBK-FORMAT) represents the db-key radix.
	
	The db-key radix portion of the page information can be used in interpreting a db-key for display purposes and in formatting a db-key from page and line numbers. The db-key radix represents the number of bits within a db-key value that are reserved for the line number of a record. By default, this value is 8, meaning that up to 255 records can be stored on a single page of the area. Given a db-key, you can separate its associated page number by dividing the db-key by 2 raised to the power of the db-key radix. For example, if the db-key radix is 4, you would divide the db-key value by 2**4. The resulting value is the page number of the db-key. To separate the line number, you would multiply the page number by 2 raised to the power of the db-key radix and subtract this value from the db-key value. The result would be the line number of the db-key. The following two formulas can be used to calculate the page and line numbers from a db-key value:
	
	`Page-number = db-key value /`
	`  (2 ** db-key radix)`
	`Line-number = db-key value -`
	`  (page-number *`
	`  ( 2 ** db-key radix))`

IDBMSCOM-AREA	Used internally by the DBMS for specification of run-time function information.
DIRECT-DBKEY	Either a user-specified db-key value or a null db-key value of -1. This field is used for storing a record with a location mode of DIRECT. It must be initialized by the user; it is not updated by the DBMS.

Native VSAM Users: The DIRECT-DBKEY field can be used only when storing a record in a native VSAM relative record data set (RRDS) or when storing

records with DIRECT location mode. You must initialize DIRECT-DBKEY to the relative record number of the record being stored.

The IDMS communications block fields in the following table are for non-CICS application programs:

Field name	Description
DATABASE-STATUS	Reserved for use by the DBMS.
FILLER	Used to ensure fullword alignment.
RECORD-OCCUR	A record occurrence sequence identifier used internally by the DBMS.
DML-SEQUENCE	The source level sequence number generated by the precompiler. This field is updated before each call to the DBMS if DEBUG is specified in the program's ENVIRONMENT DIVISION; it is not used by the run-time system.

The IDMS communications block fields in the following table are for CICS application programs:

Field name	Description
FILLER	A work area reserved for use by CICS applications.
DATABASE-STATUS	Reserved for use by the DBMS.
FILLER	Used to ensure fullword alignment.
RECORD-OCCUR	A record occurrence sequence identifier used internally by the DBMS.
DML-SEQUENCE	The source-level sequence number generated by the precompiler. This field is updated before each call to the DBMS if DEBUG is specified in the program's ENVIRONMENT DIVISION; it is not used by the run-time system.

After a call has been made to the DBMS, one or more of the fields described above may have been updated, depending on the DML statement issued and whether the statement was executed successfully. The following figure illustrates the IDMS communications block fields updated by successful and unsuccessful calls to the DBMS; only those fields accessed by the run-time system are shown. Fields used internally by the DBMS are not shown. Blank fields are not updated by DML statements.

	SUCCESSFUL										UNSUCCESSFUL									
	PROGRAM-NAME	ERROR-STATUS	DBKEY	RECORD-NAME	AREA-NAME	ERROR-SET	ERROR-RECORD	ERROR-AREA	PAGE-INFO	DIRECT-DBKEY	PROGRAM-NAME	ERROR-STATUS	DBKEY	RECORD-NAME	AREA-NAME	ERROR-SET	ERROR-RECORD	ERROR-AREA	PAGE-INFO	DIRECT-DBKEY
Control statements																				
BIND RUN-UNIT		O										14nn								
BIND RECORD		O										14nn				Y	Y	Y		
BIND PROCEDURE		O										14nn				Y	Y	Y		
READY		O										09nn				C	C	C		
FINISH		O	N	C		C	C	C				01nn				C	C	C		
COMMIT (ALL)		O	N	C		C	C	C				18nn				C	C	C		
ROLLBAK (CONTINUE)		O	N	C		C	C	C				19nn				C	C	C		
KEEP (EXCLUSIVE)		O	Y	Y	Y	C	C	C	Y			06nn				Y	Y	Y		
IF SET		*	Y	Y	Y	C	C	C	Y			16nn				Y	Y	Y		
IF NOT SET		*	Y	Y	Y	C	C	C	Y			16nn				Y	Y	Y		
Retrieval statements																				
FIND/OBTAIN RECORD		O	Y	Y	Y	C	C	C	Y			03nn				Y	Y	Y		
GET RECORD		O	Y	Y	Y	C	C	C	Y			05nn				Y	Y	Y		
RETURN RECORD		O	Y	Y	Y	C	C	C	Y			17nn				Y	Y	Y		
Modification statements																				
STORE RECORD		O	Y	Y	Y	C	C	C	Y			12nn				Y	Y	Y		
CONNECT RECORD		O	Y	Y	Y	C	C	C	Y			07nn				Y	Y	Y		
MODIFY RECORD		O	Y	Y	Y	C	C	C	Y			08nn				Y	Y	Y		
DISCONNECT RECORD		O	Y	Y	Y	C	C	C	Y			11nn				Y	Y	Y		
ERASE RECORD		O	N	Y	Y	C	C	C				02nn				Y	Y	Y		
Accept statements																				
ACCEPT DBKEY OF CURRENCY		O				C	C	C				15nn				Y	Y	Y		
ACCEPT DBKEY OF N/P/O		O				C	C	C				15nn				Y	Y	Y		
ACCEPT IDMS STATISTICS		O				C	C	C				15nn				Y	Y	Y		
ACCEPT BIND RECORD		O				C	C	C				15nn				Y	Y	Y		
ACCEPT PROCEDURE		O				C	C	C				82nn				Y	Y	Y		
ACCEPT page-info-location		O				C	C	C				15nn				Y	Y	Y		

Accept statements																			
ACCEPT FROM CURRENCY		O				C	C	C			15nn				Y	Y	Y	
ACCEPT FROM N/P/O CURRENCY		O				C	C	C			15nn				Y	Y	Y	
ACCEPT FROM IDMS-STATISTICS		O				C	C	C			15nn				Y	Y	Y	
ACCEPT FROM BIND		O				C	C	C			15nn				Y	Y	Y	
ACCEPT FROM PROCEDURE		O				C	C	C			15nn				Y	Y	Y	

```
*   If true, field is set to zone decimal zeroes (0000)
    If false, field is set to 1601

O   Field is set to zone decimal zeroes

Y   Field is updated

C   Field is cleared to spaces

N   Field is set to null db-key value (-1)

nn  Specific minor error code
```

3.2.2 LRC block

The logical-record request control (LRC) block is used when the subschema usage mode is LR or MIXED. The LRC block, which is used in conjunction with the IDMS or IDMS-DC communications block, provides an interface between LRF and the application program. It passes information about a logical-record request to LRF and returns path status information about the processing of the request to the program.

The data description (identified as SUBSCHEMA-LR-CTRL) of the LRC block is copied from the data dictionary into the WORKING-STORAGE SECTION or LINKAGE SECTION of the program. When the program is submitted to the precompiler, the LRC block and the IDMS or IDMS-DC communications block are copied automatically, unless the IDMS-RECORDS MANUAL statement is included in the ENVIRONMENT DIVISION. In that case, both descriptions can be called in explicitly by the program by using a COPY IDMS SUBSCHEMA-LR-CTRL statement

For more information on the IDMS-RECORDS MANUAL and the COPY IDMS statements, see Chapter 4, “Precompiler-Directive Statements” on page 4-1.

You should examine the LR-STATUS field of the LRC block for all possible statuses after every call to LRF. If the value returned is LR-ERROR, you should then examine the ERROR-STATUS field of the IDMS or IDMS-DC communications block.

The following figure shows the layout of the LRC block. Each field is described separately, following the figure.

Logical-Record Request Control Block:

LRC BLOCK

	Field	Data Type	Length (bytes)	Suggested Initial Value
1 2	LRC-LRPXELNG	Binary	2 (Halfword)	00
3 4	LRC-MAXVXP	Binary	2 (Halfword)	00
5 8	LRIDENT	Alphanumeric	4	'LRC'
9 16	LRVERB	Alphanumeric	8	Spaces
17 32	LRNAME	Alphanumeric	16	Spaces
33 48	LR-STATUS	Alphanumeric	16	Spaces
49 64	FILLER	...	16	...
65 ... (variable-length)	PXE	Mixed	...	...

* word aligned

The LRC block contains the following fields:

Field name	Position	Description
LRC-LRPXELNG	1-2	Specifies the length of the LRC block.
LRC-MAXVXP	3-4	Specifies the length of the work area required to evaluate the WHERE clause.
LRIDENT	5-8	The constant 'LRC' followed by a space.
LRVERB	9-16	The verb passed to LRF.
LRNAME	17-32	The name of the logical record being accessed.
LR-STATUS	33-48	The path status of a logical-record request. Path statuses are 1- to 16-character strings; they can be either standard or defined in the subschema by the DBA. LRF provides three standard path statuses: LR-FOUND, LR-NOT-FOUND, and LR-ERROR. For more information on path statuses, refer to 5.84, “Logical-record clauses” on page 5-258.
FILLER	49-64	Used internally by LRF.

Field name	Position	Description
PXE	65-end	The variable-length expansion of the WHERE clause. From 0 to 512 1-byte elements. The 512-byte limit can be raised or lowered by using the SIZE IS parameter of the COPY IDMS SUBSCHEMA-LR-CTRL statement. For more information on the SIZE IS parameter and the COPY IDMS statement, see Chapter 4, “Precompiler-Directive Statements” on page 4-1.

3.2.3 IDMS-DC communications block

The IDMS-DC communications block replaces the IDMS communications block when the operating mode is either IDMS-DC or DC-BATCH. At run time, CA-IDMS uses the IDMS-DC communications block to pass information about the outcome of requested data communications and database services to an application program.

The data description (identified as SUBSCHEMA-CTRL) of the IDMS-DC communications block is copied from the data dictionary into the WORKING-STORAGE SECTION or LINKAGE SECTION of the program. When the program is submitted to the precompiler, the IDMS-DC communications block is copied automatically unless the IDMS-RECORDS MANUAL statement is included in the ENVIRONMENT DIVISION. In that case, the program can explicitly call in the data description by using a COPY IDMS SUBSCHEMA-CTRL statement (for more information on the IDMS-RECORDS MANUAL and the COPY IDMS statements, see Chapter 4, “Precompiler-Directive Statements” on page 4-1).

The following figure shows the layout of the IDMS-DC communications block. Each field is described separately below.

		Field	Data Type	(bytes)	Initial Value
*	1 8	PROGRAM-NAME	Alphanumeric	8	Program Name
	9 12	ERROR-STATUS	Alphanumeric	4	'1400'
	13 16	DBKEY	Binary	4 (Fullword)	0000
	17 32	RECORD-NAME	Alphanumeric	16	Spaces
	33 48	AREA-NAME	Alphanumeric	16	Spaces
	49 64	ERROR-SET	Alphanumeric	16	Spaces
	65 80	ERROR-RECORD	Alphanumeric	16	Spaces
	81 96	ERROR-AREA	Alphanumeric	16	Spaces
**	97 100	PAGE-INFO	Binary	4 (Fullword)	0000
	97 ... 196	IDBMSCOM-AREA	Alphanumeric	100	Low Values
	197 200	DIRECT-DBKEY	Binary	4 (Fullword)	0000
	201 ... 300	DCBMSCOM-AREA	Alphanumeric	100	Low Values
	301 304	SSC-ERRSTAT-SAVE	Alphanumeric	4	Spaces
	305 308	SSC-DMLSEQ-SAVE	Binary	4 (Fullword)	0000
	309 312	DML-SEQUENCE	Binary	4 (Fullword)	0000
	313 316	RECORD-OCCUR	Binary	4 (Fullword)	0000
	317 320	SUBSCHEMA-CTRL-END	Alphanumeric	4	Spaces

* word aligned
** PAGE-INFO-GROUP overlays bytes 97 and 98 and PAGE-INFO-DBK-FORMAT overlays bytes 99 and 100. Both of these fields are binary datatype each having a length of two bytes. Suggested initial values for both are 00. Together these two fields represent PAGE-INFO.

Field descriptions: The IDMS-DC communications block contains the following fields that describe program status information:

Field name	Position	Description
PROGRAM-NAME	1-8	The name of the program being executed, as defined in the program's IDENTIFICATION DIVISION. This field is initialized automatically at the beginning of program execution if the program contains a COPY IDMS SUBSCHEMA-BINDS statement in its PROCEDURE DIVISION. Otherwise, it must be initialized by the programmer.

Field name	Position	Description
ERROR-STATUS	9-12	A value indicating the outcome of the last DML statement executed. The ERROR-STATUS field must be initialized to 1400 by the program. This field is updated by CA-IDMS after (attempted) performance of a requested database or data communications service and before control is returned to the program. The ERROR-STATUS field and its use are described in greater detail under 3.3, “ERROR-STATUS field and codes” on page 3-18 later in this chapter. A program that consists of more than one run unit must reinitialize the ERROR-STATUS field to 1400 after finishing one run unit and before binding the next.
DBKEY	13-16	The database key of the last record accessed by the run unit. For example, after successful execution of a FIND command, DBKEY is updated with the database key of the located record. DBKEY is not changed if the database call results in a nonzero status condition.
RECORD-NAME	17-32	The name of the last record accessed successfully by the run unit. This field is left-justified and padded with spaces on the right.
AREA-NAME	33-48	The name of the last area accessed successfully by the run unit. This field is left-justified and padded with spaces on the right.
ERROR-SET	49-64	The name of the set involved in the last operation to produce a nonzero status code. This field is left-justified and padded with spaces on the right.

Field name	Position	Description
ERROR-RECORD	65-80	The name of the record involved in the last operation to produce a nonzero status code. This field is left-justified and padded with spaces on the right.
ERROR-AREA	81-96	The name of the area involved in the last operation to produce a nonzero status code. This field is left-justified and padded with spaces on the right.

Field name	Position	Description

PAGE-INFO	97-100	Two binary halfwords that represent the page information associated with the last record accessed by the run unit. PAGE-INFO is not changed if the call to the DBMS results in a non-zero status. The first halfword (PAGE-INFO-GROUP) represents the page group number. The second halfword (PAGE-INFO-DBK-FORMAT) represents the db-key radix.
		
		The db-key radix portion of the page information can be used in interpreting a db-key for display purposes and in formatting a db-key from page and line numbers. The db-key radix represents the number of bits within a db-key value that are reserved for the line number of a record. By default, this value is 8, meaning that up to 255 records can be stored on a single page of the area. Given a db-key, you can separate its associated page number by dividing the db-key by 2 raised to the power of the db-key radix. For example, if the db-key radix is 4, you would divide the db-key value by 2**4. The resulting value is the page number of the db-key. To separate the line number, you would multiply the page number by 2 raised to the power of the db-key radix and subtract this value from the db-key value. The result would be the line number of the db-key. The following two formulas can be used to calculate the page and line numbers from a db-key value:
		`Page-number = db-key value /`
		`  (2 ** db-key radix)`
		`Line-number = db-key value -`
		`  (page-number *`
		`  ( 2 ** db-key radix))`

IDBMSCOM-AREA	97-196	Used internally by CA-IDMS for specification of DBMS run-time function information.

Field name	Position	Description
DIRECT-DBKEY	197-200	Either a user-specified db-key value or a null db-key value of -1. This field is used for storing a record with a location mode of DIRECT. It must be initialized by the user; it is not updated by CA-IDMS. **Native VSAM users:** The DIRECT-DBKEY field can be used when storing a record in a native VSAM relative record data set (RRDS). You must initialize DIRECT-DBKEY to the relative record number of the record being stored.
DCBMSCOM-AREA	201-300	Used internally by CA-IDMS for specification of run-time function information.
SSC-ERRSTAT-SAVE	301-304	Used by the IDMS-STATUS routine to save a nonzero ERROR-STATUS in the event of an abend.
SSC-DMLSEQ-SAVE	305-308	Used by the IDMS-STATUS routine to save the value of DML-SEQUENCE in the event of an abend.
DML-SEQUENCE	309-312	The source level sequence number generated by the precompiler. This field is updated before each call to CA-IDMS if DEBUG is specified in the program's ENVIRONMENT DIVISION; it is not used by the run-time system.
RECORD-OCCUR	313-316	A record occurrence sequence identifier used internally by CA-IDMS.
SUBSCHEMA-CTRL-END	317-320	Marks the end of the IDMS-DC communications block.

3.3 ERROR-STATUS field and codes

You can use the ERROR-STATUS field of the IDMS or IDMS-DC communications block to determine whether a DML request has been processed successfully. The DBMS or the DC system returns a value to the ERROR-STATUS field indicating the result of each DML request. For more information on using the ERROR-STATUS field, see 3.4, “Error detection” on page 3-29 later in this chapter.

LRF users: You should check the LR-STATUS field of the LRC block before checking the ERROR-STATUS field.

The ERROR-STATUS field is a four-byte zoned decimal (PIC X(4) DISPLAY) field. The first two bytes represent a major code; the second two bytes represent a minor code. Major codes identify the function performed; minor codes describe the status of that function.

A value of 0000 indicates successful completion of the requested function. A value other than 0000 indicates completion of the function in a manner that may or may not be in error, depending on your expectations. For example, 0326 (DB-REC-NOT-FOUND) should be anticipated after FIND CALC retrieval; this allows you to trap the condition and continue processing.

DB status codes have a major code in the range 01 to 20 and occur during database access in either batch processing or online processing. DC status codes occur in either online or DC-BATCH processing. Status codes with a major code of 00 apply to all DML functions. Database status codes and DC status codes are discussed separately below.

3.3.1 DB status codes

The following tables list database major and minor codes and their meanings.

For a complete description of DB run-time status codes, refer to *CA-IDMS Messages and Codes*.

Major DB status codes

Code	Database Function
00	Any DML statement
01	FINISH
02	ERASE
03	FIND/OBTAIN
05	GET
06	KEEP

Code	Database Function
07	CONNECT
08	MODIFY
09	READY
11	DISCONNECT
12	STORE
14	BIND
15	ACCEPT
16	IF
17	RETURN
18	COMMIT
19	ROLLBACK
20	LRF requests

Minor DB status codes

Code	Database Function Status
00	Combined with a major code of 00, this code indicates successful completion of the DML operation. Combined with a nonzero major code, this code indicates that the DML operation was not completed successfully due to central version causes, such as time-outs and program checks.
01	An area has not been readied. When this code is combined with a major code of 16, an IF operation has resulted in a valid false condition.
02	Either the db-key used with a FIND/OBTAIN DB-KEY statement or the direct db-key suggested for a STORE is not within the page range for the specified record name.
04	The occurrence count of a variably occurring element has been specified as either less than zero or greater than the maximum number of occurrences defined in the control element.
05	The specified DML function would have violated a duplicates-not-allowed option for a CALC, sorted, or index set.
06	No currency has been established for the named record, set, or area.
07	The end of a set, area, or index has been reached or the set is empty.
08	The specified record, set, procedure, or LR verb is not in the subschema or the specified record is not a member of the set.

Code	Database Function Status
09	The area has been readied with an incorrect usage mode.
10	An existing access restriction or subschema usage prohibits execution of the specified DML function. For LRF users, the subschema in use allows access to database records only. Combined with a major code of 00, this code means the program has attempted to access a database record, but the subschema in use allows access to logical records only.
11	The record cannot be stored in the specified area due to insufficient space.
12	There is no db-key for the record to be stored. This is a system internal error and should be reported.
13	A current record of run unit either has not been established or has been nullified by a previous ERASE statement.
14	The CONNECT statement cannot be executed because the requested record has been defined as a mandatory automatic member of the set.
15	The DISCONNECT statement cannot be executed because the requested record has been defined as a mandatory member of the set.
16	The record cannot be connected to a set of which it is already a member.
18	The record has not been bound.
20	The current record is not the same type as the specified record name.
21	Not all areas being used have been readied in the correct usage mode.
22	The record name specified is not currently a member of the set name specified.
23	The area name specified is either not in the subschema or not an extent area; or the record name specified has not been defined within the area name specified.
25	No currency has been established for the named set.
26	No duplicates exist for the named record or the record occurrences cannot be found.
28	The run unit has attempted to ready an area that has been readied previously.
29	The run unit has attempted to place a lock on a record that is locked already by another run unit. A deadlock results. Unless the run unit issued either a FIND/OBTAIN KEEP EXCLUSIVE or a KEEP EXCLUSIVE, the run unit is aborted.
30	An attempt has been made to erase the owner record of a nonempty set.
31	The retrieval statement format conflicts with the record's location mode.

Code	Database Function Status
32	An attempt to retrieve a CALC/DUPLICATE record was unsuccessful; the value of the CALC field in variable storage is not equal to the value of the CALC control element in the current record of run unit.
33	At least one set in which the record participates has not been included in the subschema.
40	The WHERE clause in an OBTAIN NEXT logical-record request is inconsistent with a previous OBTAIN FIRST or OBTAIN NEXT command for the same record. Previously specified criteria, such as reference to a key field, have been changed. A path status of LR-ERROR is returned to the LRC block.
41	The subschema contains no path that matches the WHERE clause in a logical-record request. A path status of LR-ERROR is returned to the LRC block.
42	An ON clause included in the path by the DBA specified return of the LR-ERROR path status to the LRC block; an error has occurred while processing the LRF request.
43	A program check has been recognized during evaluation of a WHERE clause; the program check indicates that either a WHERE clause has specified comparison of a packed decimal field to an unpacked nonnumeric data field, or data in variable storage or a database record does not conform to its description. A path status of LR-ERROR is returned to the LRC block unless the DBA has included an ON clause to override this action in the path.
44	The WHERE clause in a logical-record request does not supply a key element (sort key, CALC key, or db-key) expected by the path. A path status of LR-ERROR is returned to the LRC block.
45	During evaluation of a WHERE clause, a program check has been recognized because a subscript value is neither greater than 0 nor less than its maximum allowed value plus 1. A path status of LR-ERROR is returned to the LRC block unless the DBA has included an ON clause to override this action in the path.
46	A program check has revealed an arithmetic exception (for example: overflow, underflow, significance, divide) during evaluation of a WHERE clause. A path status of LR-ERROR is returned to the LRC block unless the DBA has included an ON clause to override this action in the path.
53	The subschema definition of an indexed set does not match the indexed set's physical structure in the database.
54	Either the prefix length of an SR51 record is less than zero or the data length is less than or equal to zero.
55	An invalid length has been defined for a variable-length record.

Code	Database Function Status
56	An insufficient amount of memory to accommodate the CA-IDMS compression/decompression routines is available.
57	A retrieval-only run unit has detected an inconsistency in an index that should cause an 1143 abend, but optional APAR bit 216 has been turned on.
60	A record occurrence type is inconsistent with the set named in the ERROR-SET field in the IDMS communications block. This code usually indicates a broken chain.
61	No record can be found for an internal db-key. This code usually indicates a broken chain.
62	A system-generated db-key points to a record occurrence, but no record with that db-key can be found. This code usually indicates a broken chain.
63	The DBMS cannot interpret the DML function to be performed. When combined with a major code of 00, this code means invalid function parameters have been passed on the call to the DBMS. For LRF users, a WHERE clause includes a keyword that is longer than the 32 characters allowed.
64	The record cannot be found; the CALC control element has not been defined properly in the subschema.
65	The database page read was not the page requested.
66	The area specified is not available in the requested usage mode.
67	The subschema invoked does not match the subschema object tables.
68	The CICS interface was not started.
69	A BIND RUN-UNIT may not have been issued; the CV may be inactive or not accepting new run units; or the connection with the CV may have been broken due to time out or other factors. When combined with a major code of 00, this code means the program has been disconnected from the DBMS.
70	The database will not ready properly; a JCL error is the probable cause.
71	The page range or page group for the area being readied or the page requested cannot be found in the DMCL.
72	There is insufficient memory to dynamically load a subschema or database procedure.
73	A central version run unit will exceed the MAXERUS value specified at system generation.

Code	Database Function Status
74	The dynamic load of a module has failed. If operating under the central version, a subschema or database procedure module either was not found in the data dictionary or the load (core image) library or, if loaded, will exceed the number of subschema and database procedures provided for at system generation.
75	A read error has occurred.
76	A write error has occurred.
77	The run unit has not been bound or has been bound twice. When combined with a major code of 00, this code means either the program is no longer signed on to the subschema or the variable subschema tables have been overwritten.
78	An area wait deadlock has occurred.
79	The run unit has requested more db-key locks than are available to the system.
80	The target node is either not active or has been disabled.
81	The converted subschema requires specified database name to be in the DBNAME table.
82	The subschema must be named in the DBNAME table.
83	An error has occurred in accessing native VSAM data sets.
87	The owner and member records for a set to be updated are not in the same page group or do not have the same db-key radix.
91	The subschema requires a DBNAME to do the bind run unit.
92	No subschema areas map to DMCL.
93	A subschema area symbolic was not found in DMCL.
94	The specified dbname is neither a dbname defined in the DBNAME table, nor a SEGMENT defined in the DMCL.
95	The specified subschema failed DBTABLE mapping using the specified dbname.

3.3.2 DC status codes

The following tables list the DC major and minor codes and their meanings.

For a complete description of CA-IDMS/DC run-time status codes, refer to *CA-IDMS Messages and Codes*.

Major DC status codes

Code	Function
00	Any DML statement
30	TRANSFER CONTROL
31	WAIT/POST
32	GET STORAGE/FREE STORAGE
33	SET ABEND EXIT/ABEND CODE
34	LOAD/DELETE TABLE
35	GET TIME/SET TIMER
36	WRITE LOG
37	ATTACH/CHANGE PRIORITY
38	BIND/ACCEPT/END TRANSACTION STATISTICS
39	ENQUEUE/DEQUEUE
40	SNAP
43	PUT/GET/DELETE SCRATCH
44	PUT/GET/DELETE QUEUE
45	BASIC MODE TERMINAL MANAGEMENT
46	MAPPING MODE TERMINAL MANAGEMENT
47	LINE MODE TERMINAL MANAGEMENT
48	ACCEPT/WRITE PRINTER
49	SEND MESSAGE
50	COMMIT TASK/ROLLBACK TASK/FINISH TASK/WRITE JOURNAL
51	KEEP LONGTERM

Minor DC status codes

Code	Function Status
00	Combined with a major code of 00, this code indicates either successful completion of the DML function or that all tested resources have been enqueued.
01	The requested operation cannot be performed immediately; waiting will cause a deadlock.
02	Either there is insufficient storage in the storage pool or the storage required for control blocks is unavailable.
03	The scratch area ID cannot be found.

Code	Function Status
04	Either the queue ID (header) cannot be found or a paging session was in progress when a second STARTPAGE command was received (that is, an implied ENDPAGE was processed before this STARTPAGE was executed successfully).
05	The specified scratch record ID or queue record cannot be found.
06	No resource control element (RCE) exists for the queue record; currency has not been established.
07	Either an I/O error has occurred or the queue upper limit has been reached.
08	The requested resource is not available.
09	The requested resource is available.
10	New storage has been assigned.
11	A maximum task condition exists.
12	The named task code is invalid.
13	The named resource cannot be found.
14	The requested module is defined as nonconcurrent and is currently in use.
15	The named module has been overlaid and cannot be reloaded immediately.
16	The specified interval control element (ICE) address cannot be found.
17	The record has been replaced.
18	No printer terminals have been defined for the current DC system.
19	The return area is too small; data has been truncated.
20	An I/O, program-not-found, or potential-deadlock status condition exists.
21	The message destination is undefined, the longterm ID cannot be found, or a KEEP LONGTERM request was issued by a nonterminal task.
22	A record already exists for the scratch area specified.
23	No storage or resource control element (RCE) could be allocated for the reply area.
24	The maximum number of outstanding replies has been exceeded.
25	An attention interrupt has been received.
26	There is a logical error in the output data stream.
27	A permanent I/O error has occurred.
28	The terminal dial-up line is disconnected.

Code	Function Status
29	An invalid parameter has been passed in the list set up by the DML processor.
30	The named function has not yet been implemented.
31	An invalid parameter has been passed; the TRB, LRB, or MRB contains an invalid field; or the request is invalid because of a possible logic error in the application program. In a DC-BATCH environment, a possible cause is that the record length specified by the command exceeds the maximum length based on the packet size.
32	The derived length of the specified variable storage is negative or zero.
33	Either the named table or the named map cannot be found in the data dictionary load area.
34	The named variable-storage area must be an 01-level entry in the LINKAGE SECTION.
35	A GET STORAGE request is invalid because the LINKAGE SECTION variable has already been allocated.
36	The program either was not defined during system generation or is marked out-of-service.
37	A GET STORAGE operand is invalid because the specified variable storage area is in the WORKING-STORAGE SECTION instead of the LINKAGE SECTION.
38	Either no GET STORAGE operand was specified or the specified LINKAGE SECTION variable has not been allocated.
39	The terminal device being used is out of service.
40	NOIO has been specified but the datastream cannot be found.
41	An IF operation resulted in a valid true condition.
42	The named map does not support the terminal device in use.
43	A line I/O session has been cancelled by the terminal operator.
44	The referenced field does not participate in the specified map; a possible cause is an invalid subscript.
45	An invalid terminal type is associated with the issuing task.
46	A terminal I/O error has occurred.
47	The named area has not been readied.
48	The run unit has not been bound.
49	NOWAIT has been specified but WAIT is required.
50	Statistics are not being kept.

Code	Function Status
51	A lock manager error occurred during the processing of a KEEP LONGTERM request
52	The specified table is missing or invalid.
53	An error occurred from a user-written edit routine.
54	Either there is invalid internal data or a data conversion error has occurred.
55	The user-written edit routine cannot be found.
56	No DFLDS have been defined for the map.
57	The ID cannot be found, is not a long-term permanent ID, or is being used by another run unit.
58	Either the LRID cannot be found or the maximum number of concurrent task threads was exceeded.
59	An error occurred in transferring the KEEP LONGTERM request to IDMSKEEP
60	The requested KEEP LONGTERM lock id was already in use with a different page group
63	Invalid function parameters have been passed on the call to the DBMS.
64	No detail exists currently for update; no action has been taken. Alternatively, the requested node for a header or detail is either not present or not updated.
68	There are no more updated details to MAP IN or the amount of storage defined for pageable maps at sysgen is insufficient. In the latter case, subsequent MAP OUT DETAIL statements are ignored.
72	No detail occurrence, footer, or header fields exist to be mapped out by a MAP OUT RESUME command, or the scratch record that contains the requested detail could not be accessed. The latter case is a mapping internal error and should be reported.
76	The first screen page has been transmitted to the terminal.
77	Either the program is no longer signed on to the subschema or the variable subschema tables have been overwritten.
80	The target node is either not active or has been disabled.
96	There are too many active run units for the internal table.
97	An invalid status has been received from DBIO/DBMS; check the DC system log file for details.
98	An unsupported COBOL compiler option (for example, DEBUG) may have been specified.

Code	Function Status
99	An unexpected internal return code has been received; the terminal device is out of service.

3.4 Error detection

The value returned to the ERROR-STATUS field must be checked after each DML request. When using the Logical Record Facility, you should check the LR-STATUS field of the LRC block before checking the ERROR-STATUS field.

CA-IDMS offers two aids for error detection: the IDMS-STATUS routine and the AUTOSTATUS protocols. Each of these aids is described below.

3.4.1 IDMS-STATUS routine

The data dictionary contains an error-checking routine called IDMS-STATUS that can be copied into the program by coding the following statement at the end of the PROCEDURE DIVISION:

```
COPY IDMS IDMS-STATUS.
```

For more information on the use of the COPY IDMS IDMS-STATUS statement, refer to Chapter 4, “Precompiler-Directive Statements” on page 4-1.

3.5 IDMS-STATUS routine used under DC

The following figure shows the code copied into batch programs by the COPY IDMS IDMS-STATUS statement.

```
*****************************************************************
 IDMS-STATUS                                            SECTION.
*****************************************************************
         IF DB-STATUS-OK GO TO ISABEX.
         PERFORM IDMS-ABORT.
         DISPLAY '***************************'
                 ' ABORTING - ' PROGRAM-NAME
                 ', '           ERROR-STATUS
                 ', '           ERROR-RECORD
                 ' **** RECOVER IDMS ****'
                UPON CONSOLE.
         DISPLAY 'PROGRAM NAME ------ ' PROGRAM-NAME.
         DISPLAY 'ERROR STATUS ------ ' ERROR-STATUS.
         DISPLAY 'ERROR RECORD ------ ' ERROR-RECORD.
         DISPLAY 'ERROR SET --------- ' ERROR-SET.
         DISPLAY 'ERROR AREA -------- ' ERROR-AREA.
         DISPLAY 'LAST GOOD RECORD -- ' RECORD-NAME.
         DISPLAY 'LAST GOOD AREA ---- ' AREA-NAME.
         DISPLAY 'DML SEQUENCE -------' DML-SEQUENCE.
         ROLLBACK.
         CALL 'ABORT'.
```

The following figure shows the code copied into DC programs by the COPY IDMS IDMS-STATUS statement.

```
*****************************************************************
 IDMS-STATUS                                            SECTION.
******************** IDMS-STATUS FOR IDMS/DC ********************
         IF DB-STATUS-OK GO TO ISABEX.
         PERFORM IDMS-ABORT.
         MOVE ERROR-STATUS TO SSC-ERRSTAT-SAVE
         MOVE DML-SEQUENCE TO SSC-DMLSEQ-SAVE
         SNAP FROM SUBSCHEMA-CTRL TO SUBSCHEMA-CTRL-END
                   ON ANY-STATUS
                                NEXT SENTENCE.
         ABEND CODE SSC-ERRSTAT-SAVE
                   ON ANY-STATUS
                                NEXT SENTENCE.
 ISABEX. EXIT.
 DMLC-DC-GEN-GOBACK SECTION.
     GOBACK.
```

The IDMS-STATUS routine abends the program if a nonzero value is present in the ERROR-STATUS field. Since certain returned values do not constitute a processing error, your program should check ERROR-STATUS for any expected nonzero values before this routine is performed. The most common codes to check for before execution of IDMS-STATUS are listed in the following table, along with their assigned level-88 condition names.

ERROR-STATUS condition names

Code	Condition name	Explanation
0000	DB-STATUS-OK	No error
0307	DB-END-OF-SET	End of set, area, or SPF index
0326	DB-REC-NOT-FOUND	No record found
0001 to 9999	ANY-ERROR-STATUS	Any nonzero status
0000 to 9999	ANY-STATUS	Any status
3101 3201 3401 3901	DC-DEADLOCK	Waiting will cause a deadlock
3202 3402	DC-NO-STORAGE	Insufficient space available
4303	DC-AREA-ID-UNK	ID cannot be found
4404	DC-QUEUE-ID-UNK	Queue header cannot be found
4305 4405	DC-REC-NOT-FOUND	Record cannot be found
3908	DC-RESOURCE-NOT-AVAIL	Resource not available
3909	DC-RESOURCE-AVAIL	Resource is available
3210	DC-NEW-STORAGE	New space allocated
3711	DC-MAX-TASKS	Maximum attached tasks
4317	DC-REC-REPLACED	Record has been replaced
4319 4419 4519 4719	DC-TRUNCATED-DATA	Return area too small
4525 4625	DC-ATTN-INT	Attention interrupt received
4743	DC-OPER-CANCEL	Session cancelled

The condition names that are automatically included when using pageable maps are listed in the following table.

Pageable Map ERROR-STATUS condition names

Code	Condition name	Explanation
4604	DC-SECOND-STARTPAGE	Second consecutive STARTPAGE
4664	DC-DETAIL-NOT-FOUND	No current detail
4668	DC-NO-MORE-UPD-DETAILS	All details mapped in
4668	DC-MAX-SPACE-REACHED	Pageable map space exceeded
4672	DC-NO-DETAILS	Nothing to map out

Code	Condition name	Explanation
4676	DC-FIRST-PAGE-SENT	First page transmitted

When the IDMS-STATUS procedure is performed, it exits immediately if the status check indicates successful completion of the function (that is, an ERROR-STATUS of 0000).

- When the operating mode is **BATCH**, a nonzero status causes IDMS-STATUS to:
  - Print status information on the unsuccessful function
  - Issue a rollback
  - Abend the program

  The status information retrieved from the IDMS communications block includes program name, error status, error record, error set, error area, record name (the last record successfully accessed), and area name (the last area successfully accessed).
- When the operating mode is **IDMS-DC**, a nonzero error status causes IDMS-STATUS to:
  - Snap the IDMS-DC communications block (SUBSCHEMA-CTRL)
  - Abend the program

  The status information retrieved from the IDMS-DC communications block includes program name, error status, error record, error set, error area, record name (the last record successfully accessed), and area name (the last area successfully accessed).

IDMS-STATUS includes a call to perform a routine named IDMS-ABORT, which you can use for additional error processing. CA-IDMS supplies only the PERFORM statement; if the IDMS-ABORT routine is to be used, you must supply the routine itself by coding the section name and exit as shown below:

```
IDMS-ABORT SECTION.
IDMS-ABORT-EXIT.
    EXIT.
```

For example, you can use the IDMS-ABORT SECTION to display information regarding the LRC block as shown below:

```
IDMS-ABORT SECTION.
        IF LR-STATUS = 'LR-ERROR'
           DISPLAY 'LOGICAL RECORD ERROR'
                   'LR NAME -- ' LR-NAME
                   'LR VERB -- ' LR-VERB.
IDMS-ABORT-EXIT.
        EXIT.
```

A routine can be coded directly into the program or copied in as a module, according to the requirements of the program. However, if no abort routine is to be performed, the reference to IDMS-ABORT must be deleted from IDMS-STATUS by the DBA.

3.5.1 AUTOSTATUS protocols

The precompiler automatically generates a PERFORM IDMS-STATUS statement after each DML command (except IF) if the protocol in use includes AUTOSTATUS. For each standard protocol (for example BATCH or CICS) provided at installation time, an AUTOSTATUS protocol (for example BATCH-AUTOSTATUS or CICS-AUTOSTATUS) is also provided. (The IDMS-DC and DC-BATCH protocols already include AUTOSTATUS.) The DBA determines which protocol should be used; you must specify this protocol in the ENVIRONMENT DIVISION by means of the MODE IS statement (for more information on protocols, see Chapter 4, "Precompiler-Directive Statements" on page 4-1).

When AUTOSTATUS is in use, the PERFORM IDMS-STATUS statement can still be preceded by a check for a nonzero return code by including an ON clause at the end of the DML command. If the DBMS returns the specified status code to the IDMS communications block, the imperative statement included in the ON clause is executed; if the status code tested for is not returned, IDMS-STATUS is performed.

Any DML command can include an ON clause; only one ON clause is allowed per command.

3.5.1.1 Syntax

```
►►── ON condition-name imperative-statement . ──────────────────────────►◄
```

3.5.1.2 Parameters

ON parameter

Tests for a nonzero status returned as a result of a DML command.

condition-name

A preassigned nonzero status condition name. Valid condition names include DB-STATUS-OK, DB-END-OF-SET, DB-REC-NOT-FOUND, ANY-ERROR-STATUS, and any condition names defined by the DBA.

imperative-statement

Specifies the program action to be taken if the nonzero status identified by *condition-name* results from the DML command.

The example below illustrates use of the ON clause. A DML source program might contain the following statements:

```
0800-OBTAIN-REC.
    OBTAIN CALC OFFICE ON DB-REC-NOT-FOUND GO TO 0900-NO-REC.
                  .
                  .
                  .
0900-NO-REC.
    STORE OFFICE.
```

The precompiler converts the DML statements to comments, translates the ON clause into an IF statement, and generates the following expanded COBOL source code:

```
 0800-OBTAIN-REC.
*    OBTAIN CALC OFFICE ON DB-REC-NOT-FOUND
                 MOVE 0001 TO DML-SEQUENCE
                 CALL 'IDMS' USING SUBSCHEMA-CTRL
                                   IDBMSCOM (32)
                                   SR450
                                   IDBMSCOM (43)
                 IF NOT DB-REC-NOT-FOUND PERFORM IDMS-STATUS;
                 ELSE
                                            GO TO 0900-NO-REC.
                   .
                   .
                   .

 0900-NO-REC.
*    STORE OFFICE.
                 MOVE 0002 TO DML-SEQUENCE
                 CALL 'IDMS' USING SUBSCHEMA-CTRL
                                   IDBMSCOM (42)
                                   SR450
                   PERFORM IDMS-STATUS.
```

For further details on the expansion of calls to CA-IDMS, see Appendix D, “CA-IDMS Call Formats” on page D-1.

3.5.2 USER-DEFINED protocols

To establish a user-defined protocol, follow these steps:

1. Establish a uniquely named user-defined MODE.
2. Identify an existing CA-supplied protocol that meets the program's requirements, and use this protocol, with modifications as needed, to create a new protocol with the same name as the user-defined MODE.
3. Modify the appropriate SUBSCHEMA-CTRL record definition to include the user-defined MODE.
4. Specify the user-defined MODE in the PROTOCOL parameter of the program.

For example, to create a version of the DC-BATCH protocol that does not include AUTOSTATUS, follow these steps:

1. Define the user-defined MODE:

   ```
   ADD ATTRIBUTE DC-BATCH-NOAUTO WITHIN CLASS MODE.
   ```

2. Define the user-defined protocol based on the CA-supplied DC-BATCH protocol, editing the DC-BATCH protocol to remove the @AUTOSTATUS references:

   ```
   ADD MODULE NAME DC-BATCH-NOAUTO  VERSION 1  LANGUAGE IS COBOL
           MODE IS DC-BATCH-NOAUTO
           MODULE SOURCE FOLLOWS
                     .
                     .
                     .
           MSEND.
   ```

3. Modify the SUBSCHEMA-CTRL record for MODE IS DC-BATCH to include the user-defined MODE:

```
MODIFY RECORD SUBSCHEMA-CTRL  VERSION 1  LANGUAGE COBOL
        MODE IS DC-BATCH-NOAUTO.
```

4. Specify the user-defined MODE in the program:

```
PROTOCOL MODE IS DC-BATCH-NOAUTO
                  .
                  .
                  .
```

Chapter 4. Precompiler-Directive Statements

4.1 About this chapter 4-3
4.2 IDENTIFICATION DIVISION 4-4
4.2.1 Syntax 4-4
4.2.2 Parameters 4-4
4.3 ENVIRONMENT DIVISION 4-5
4.3.1 Syntax 4-5
4.3.2 Parameters 4-5
4.4 DATA DIVISION 4-9
4.4.1 FILE SECTION 4-9
4.4.1.1 Syntax 4-9
4.4.1.2 Parameters 4-9
4.4.2 SCHEMA SECTION 4-10
4.4.2.1 Syntax 4-10
4.4.2.2 Parameters 4-10
4.4.3 MAP SECTION 4-10
4.4.3.1 Syntax 4-11
4.4.3.2 Parameters 4-11
4.4.4 WORKING-STORAGE and LINKAGE SECTIONs 4-12
4.4.4.1 Syntax 4-14
4.4.4.2 Parameters 4-15
4.5 PROCEDURE DIVISION 4-22
4.5.1 Syntax 4-22
4.5.2 Parameters 4-22

4.1 About this chapter

Compiler-directive statements instruct the precompiler to copy source code from the data dictionary into the COBOL application program. These statements do not produce any executable commands. Compiler-directive statements are coded beginning in columns 8-11 of the IDENTIFICATION and ENVIRONMENT DIVISIONs, and in columns 8-72 of the DATA and PROCEDURE DIVISIONs, as follows:

- **IDENTIFICATION DIVISION** — The PROGRAM-ID statement specifies a program name and version number.
- **ENVIRONMENT DIVISION** — The IDMS-CONTROL SECTION establishes the operating mode, debug sequencing, and variable storage allocation.
- **DATA DIVISION** — The following sections are included in the DATA DIVISION:
  - **FILE SECTION** — COPY IDMS FILE statements copy descriptions of non-IDMS files from the data dictionary.
  - **SCHEMA SECTION** — The DB statement identifies the subschema view to be used by the program.
  - **MAP SECTION** — These statements notify the precompiler that mapping mode terminal I/O is being used, define the program's maps, and specify the size of map field lists.
  - **WORKING-STORAGE and LINKAGE SECTIONs** — COPY IDMS statements copy source data descriptions or non-IDMS data description code for records from the data dictionary.
- **PROCEDURE DIVISION** — COPY IDMS statements copy source data for BIND statements or program source modules defined in the data dictionary.

All CA-IDMS compiler-directive statements are optional except the SCHEMA SECTION and DB statement. If a program accesses the database, it must include a SCHEMA SECTION that contains a DB statement identifying the subschema. All other compiler-directive statements can be omitted; the precompiler will generate the required source code components automatically.

If the program does not access the database (that is, does not invoke a subschema and does not issue any DML statements), the SCHEMA SECTION and DB statement can be omitted as well.

The COPY IDMS and other compiler-directive statements are explained separately for each of the following divisions. References to the IDMS communications block apply equally to the IDMS-DC communications block.

4.2 IDENTIFICATION DIVISION

The PROGRAM-ID statement in the IDENTIFICATION DIVISION identifies your program to the precompiler.

4.2.1 Syntax

```
►►── IDENTIFICATION DIVISION. ──────────────────────────────────►
 ►── PROGRAM-ID.program-name ──┬──────────────────────────────►◄
                               └─ VERSION version-number ─┘
```

4.2.2 Parameters

PROGRAM ID

Specifies the program.

program-name

The name of the program. If the program has been previously defined in the data dictionary through IDD facilities, *program-name* must match the name assigned to the program when it was defined in order for the precompiler to recognize it as the same program.

VERSION

Qualifies *program-name* with a version number (for example, for purposes of testing or development).

version-number

An integer in the range 1 through 9999. By default, if you do not specify a number, the default is either the highest number defined in the data dictionary for the named program or 1 if the program does not already exist in the data dictionary.

4.3 ENVIRONMENT DIVISION

An IDMS-CONTROL SECTION is required in the ENVIRONMENT DIVISION to establish the following:

- **Operating mode** — The environment in which the program will execute, and the form and content of calling sequences produced by the precompiler
- **Debug sequencing** — Whether each PROCEDURE DIVISION DML command will be numbered for identification during error reporting
- **Variable storage allocation** — How source data description code for the IDMS communications block and other DATA DIVISION components will be inserted in the program

4.3.1 Syntax

```
►►── ENVIRONMENT DIVISION. ──────────────────────────────────────────►◄

►──┬──────────────────────────────────────────────────────────────────►
   └─ IDMS-CONTROL SECTION. ──────────────────────────────────────────

►──┬──────────────────────────────────────────────────────────────────►
   ├── PROTOCOL. ──┬──────────────────────────────────────────────────┤
   │               └─ MODE is ─┬─ BATCH ◄ ──────────┬──────────────────┤
   │                           ├─ IDMS-DC ──────────┤  └─ DEBUG ─┘     │
   │                           ├─ DC-BATCH ─────────┤                  │
   │                           └─ user-mode-name ───┘                  │

►──┬──────────────────────────────────────────────────────────────────►
   └─ SUBSCHEMA-NAMES LENGTH IS ─┬─ 16 ◄─┬──┤
                                 └─ 18 ──┘  │

►──┬──────────────────────────────────────────────────────────────────►
   │
   └─ IDMS-RECORDS ───────────────────────────────────────────────────

►──┬──────────────────────────────────────────────────────────────────►◄
   ├─ WITHIN WORKING-STORAGE section ◄──┬──────────────────────────┬─ . ┘
   ├─ WITHIN LINKAGE section ───────────┘  └─ levels INCREMENTED by count ┤
   └─ MANUAL ──────────────────────────────────────────────────────────┘
```

4.3.2 Parameters

PROTOCOL

Specifies how CA-IDMS CALL statements are generated and whether the debugging sequence option is included.

MODE IS

Identifies the operating mode used by the precompiler to generate call statements for the program's PROCEDURE DIVISION DML statements.

BATCH

Indicates that the program will execute in batch mode.

This is the default.

The IDMS communications block is copied into variable storage; standard CALL statements (CALL 'IDMS') are generated in the PROCEDURE DIVISION.

IDMS-DC

Indicates that the program will execute in IDMS-DC mode.

The IDMS-DC communications block is copied into variable storage; CA-IDMS CALL statements (CALL 'IDMSCOBI') are generated in the PROCEDURE DIVISION for DC requests.

DC-BATCH

Indicates that the program will execute in DC-BATCH mode. The IDMS-DC communications block is copied into variable storage; DC-BATCH CALL statements (CALL 'IDMSDCCI') are generated in the PROCEDURE DIVISION for DC requests.

Specify MODE IS DC-BATCH to access DC queues and printers from batch applications running under the CA-IDMS central version.

user-mode-name

Indicates that the program will execute in a special environment (for example, under a teleprocessing monitor or in a user-defined operating mode) as determined by the DBA. The appropriate communications block is copied into variable storage; mode-specific CALL statements (for example, in CICS: CALL 'IDMSINC1' USING DFHCADS) are generated in the PROCEDURE DIVISION. The following list provides the standard operating modes (protocols) available for COBOL programs.

If *user-mode-name* specifies an AUTOSTATUS protocol (for example, CICS-AUTOSTATUS), the precompiler automatically generates an IDMS-STATUS statement after every DML command except IF. When using an AUTOSTATUS protocol, be sure to include the COPY IDMS IDMS-STATUS statement in the PROCEDURE DIVISION. For details on programming under an AUTOSTATUS protocol, see Chapter 3, “Communications Blocks and Error Detection” on page 3-1.

DEBUG

Specifies that a unique DML sequence number is placed in the IDMS communications block for each DML statement. These numbers appear in columns 81-88 of the COBOL compiler output listing in the form DMLC*nnnn*. The precompiler generates numbers to identify the sequence in which DML statements appear in the program. Depending on the error routine defined by the DBA, you can use the DML sequence number to help debug your program.

If DEBUG is not specified, the precompiler does not associate sequence numbers with source statements.

SUBSCHEMA-NAMES LENGTH IS

Specifies whether to use a 16-byte or 18-byte communications block.

For information about 16-byte communications blocks, see Chapter 3, “Communications Blocks and Error Detection” on page 3-1. For information

about 18-byte communications blocks, see Appendix J, “18-Byte Communications Blocks” on page J-1.

IDMS-RECORDS

Specifies whether source CA-IDMS data description code is inserted into the DATA DIVISION automatically.

WITHIN WORKING-STORAGE section

Instructs the processor to insert automatically the copied DATA DIVISION components as the last entries in the WORKING-STORAGE SECTION of the source program.

This is the default.

WITHIN LINKAGE section

Instructs the processor to automatically insert the copied DATA DIVISION components as the last entries in the LINKAGE SECTION of the source program. Any VALUE clauses present in source code will be dropped automatically.

levels INCREMENTED by

Varies the level numbers for inserted descriptions from those stored in the data dictionary. If you specify a level number, the first level of code will be inserted to the level specified by *count*; all other levels will be adjusted accordingly. If you do not specify a level, the descriptions inserted will begin at 01 and have the same level numbers as originally specified in the data dictionary.

count

An integer in the range 1 through 48.

Specifies the value by which the DATA DIVISION level numbers (including the 01 level number) of all stored elements are to be incremented.

Note: Using the LEVELS INCREMENTED BY clause may cause unpredictable results if record fields have been defined with a SYNCHRONIZED clause. Such fields may contain extra bytes (slack bytes) inserted to ensure correct alignment. Because CA-IDMS does not recognize slack bytes as functional, it may misinterpret data fields that contain them. Therefore, you should ensure that all fields and records are properly structured.

MANUAL

Indicates that CA-IDMS-related source data description code (for example, SUBSCHEMA-CTRL or SUBSCHEMA-NAMES) will be inserted explicitly into the source program by means of DATA DIVISION COPY IDMS statements. If MANUAL is not specified, the required DATA DIVISION code is inserted automatically by the precompiler.

Standard modes available for COBOL programs

```
BATCH                   DC-BATCH                  TASKMASTER
BATCH-AUTOSTATUS        IDMS-DC                   TASKMASTER-AUTO
CICS                    INTERCOMM                 UTM
CICS-AUTOSTATUS         INTERCOMM-AUTO            UTM-AUTOSTATUS
CICS-EXEC               INTERCOMM-REENT           WESTI
CICS-EXEC-AUTO          ICOMM-REENT-AUTO          WESTI-AUTOSTATUS
CICS-STANDARD           SHADOW                    WESTI-REENT
CICS-STD-AUTO           SHAD-AUTOSTATUS           WESTI-REENT-AUTO
```

The following example illustrates the statements used to code the IDMS-CONTROL SECTION of a program running under DC with DEBUG sequencing and automatic insertion of IDMS-RECORDS in WORKING-STORAGE SECTION:

```
ENVIRONMENT DIVISION.
  IDMS-CONTROL SECTION.
  PROTOCOL.
  MODE IS IDMS-DC
    DEBUG
    IDMS-RECORDS WITHIN WORKING-STORAGE SECTION.
```

4.4 DATA DIVISION

Compiler-directive statements can be in the following sections of the DATA DIVISION:

- **FILE SECTION** — COPY IDMS statements copy descriptions of non-IDMS files from the data dictionary
- **SCHEMA SECTION** — A DB statement identifies the subschema view to be used by the program
- **MAP SECTION** — These statements notify the precompiler that mapping mode terminal I/O is being used, define the program's maps, and specify the size of map field lists
- **WORKING-STORAGE SECTION** — COPY IDMS statements copy source data description or non-IDMS data description code for records from the data dictionary
- **LINKAGE SECTION** — COPY IDMS statements copy source data description or non-IDMS data description code for records from the data dictionary

4.4.1 FILE SECTION

The FILE SECTION can include one or more COPY IDMS statements to copy non-IDMS file descriptions from the data dictionary into the program. Each COPY IDMS statement generates the file definition that includes record size, block size, and recording mode from the data dictionary. Additionally, any records defined within the file through the IDD facilities are also copied.

4.4.1.1 Syntax

```
►►── FILE SECTION. ──────────────────────────────────────────►
►┬──────────────────────────────────────────────────────────┬►◄
 └┬─────────────────────────────────────────────────────┬┘
  ↓─ COPY IDMS FILE file-name ┬──────────────────────┬ . ┘
                              └─ VERSION version-number ┘
```

4.4.1.2 Parameters

COPY IDMS FILE

Copies the description of a non-IDMS file into the DATA DIVISION.

file-name

Either the primary name or a synonym for a file defined in the data dictionary.

VERSION

Qualifies *file-name* with a version number.

If you do not specify a version number, the default is the highest version number defined in the data dictionary for *file-name*.

version-number
An integer in the range 1 through 9999.

4.4.2 SCHEMA SECTION

For any program that accesses the database, a SCHEMA SECTION is included in the DATA DIVISION to identify a subschema view to the precompiler. The subschema named in the DB statement of the SCHEMA SECTION determines which CA-IDMS record descriptions can be copied into the program from the data dictionary. Every DML command issued by the program is checked against the record, set, and area access restrictions specified in this subschema.

4.4.2.1 Syntax

```
►►── SCHEMA SECTION. ──────────────────────────────────────────►
 ►── DB subschema-name WITHIN schema-name ┬──────────────────────┬ . ──►
                                          └ VERSION version-number ┘
```

4.4.2.2 Parameters

DB subschema-name
Specifies a subschema defined in the data dictionary. If the DBA has chosen to preregister valid program names for this subschema in the data dictionary, the program named in the IDENTIFICATION DIVISION must be associated with *subschema-name* in the data dictionary.

WITHIN schema-name
Specifies the schema under which *subschema-name* is compiled.

VERSION
Qualifies *schema-name* with a version number.

If you do not specify a version number, the default is the highest version number defined in the data dictionary for *file-name*.

version-number
An integer in the range 1 through 9999.

4.4.3 MAP SECTION

The MAP SECTION notifies the precompiler that mapping mode terminal I/O is being used, defines the program's maps, and specifies the size of map field lists.

4.4.3.1 Syntax

```
►►── MAP SECTION. ──────────────────────────────────────────►
►┬──────────────────────────────────────────────────────────►
 └─ MAP FIELD LIST is field-list-size ─┘
►┬──────────────────────────────────────────────────────────►
 │
 ↓─ MAP map-name ┬──────────────────────────────────────────►─
                 └─ VERSION version-number ─┘
──────────────────────────────────────────────────────────┐
►┬──────────────────────────────────────────────────────┬──►◄
 └─ TYPE is ┬─ STANDARD ◄ ┬──────────┬ . ┘
            └─ EXTENDED ──┘ └─ PAGING ─┘
```

4.4.3.2 Parameters

MAX FIELD LIST is

Specifies the size of field lists used in MODIFY MAP and INQUIRE MAP statements.

field-list-size

The field list size or the size is expressed as a numeric constant.

The specified size must be at least one greater than the size of the largest field list used by the program. For example, if the largest map field list contains 5 fields, the value of *field-list-size* must be at least 6.

The MAX FIELD LIST statement must be specified if the program uses a field list in a MODIFY MAP or INQUIRE MAP request.

MAP

Defines the map used by the program. This parameter can be repeated as necessary to define each map to be used.

map-name

The name of a map used by the program.

VERSION

Qualifies the named map with a version number.

version-number

An integer in the range 1 through 9999.

There is no default for *version-number*. If your site uses multiple versions, you must specify a version number.

TYPE IS

Specifies whether the map request block (MRB) built for the map is to be standard or extended.

STANDARD

Specifies that the map has standard 3270-type terminal attributes.

This is the default.

EXTENDED
Specifies that the map has extended 3279-type terminal attributes (for example, color, blinking fields, reverse video).

PAGING
Specifies that the named map is a pageable map. For more information on pageable maps, see "MAP OUT" and 5.54, “MAP IN” on page 5-155, or refer to the *CA-IDMS Mapping Facility*.

The following example shows the DATA DIVISION statements required to access the EMPSS09 subschema and the EMPMAPLR map; the largest map field list allowed is 4.

```
DATA DIVISION.
SCHEMA SECTION.
DB  EMPSS09 WITHIN EMPSCHM.
MAP SECTION.
MAX FIELD LIST IS 5.
MAP EMPMAPLR VERSION 1 TYPE IS STANDARD.
```

4.4.4 WORKING-STORAGE and LINKAGE SECTIONs

COPY IDMS statements can be coded in the WORKING-STORAGE and LINKAGE SECTIONs, allowing you to explicitly copy source code from the data dictionary into the program. *No COPY IDMS statements are required in the DATA DIVISION* unless the IDMS-RECORDS MANUAL clause has been specified in the IDMS-CONTROL SECTION of the ENVIRONMENT DIVISION.

If the source code to be copied into the LINKAGE SECTION includes VALUE clauses, these clauses are not copied.

WORKING-STORAGE SECTION and LINKAGE SECTION source code requirements differ according to the usage mode defined in the program's subschema: DML, LR, or MIXED. These usage modes determine whether the program can access database records only, logical records only, or both database records and logical records. The program should not copy components that conflict with its subschema's usage mode (for example, do not copy SUBSCHEMA-LR-CTRL if the subschema's usage mode is DML).

An explanation of each usage mode and the required source code components in the program is shown below:

- **DML** allows a program to access database records only and requires the following source code components:
  - **SUBSCHEMA-CTRL** — The IDMS communications block, through which the application program and the DBMS communicate. For further details, see Chapter 3, “Communications Blocks and Error Detection” on page 3-1.
  - **SUBSCHEMA-NAMES** — The name of the program's subschema and the names of all records, sets, and areas to which the program has access through this subschema. SUBSCHEMA-NAMES is used by the precompiler to

generate appropriate CA-IDMS CALL statements in the PROCEDURE DIVISION.

- **SUBSCHEMA-RECORDS** — The description of all records to which the subschema permits access.

- **LR** allows a program to access logical records only and requires the following source code components:
  - **SUBSCHEMA-CTRL** — The IDMS communications block, through which LRF and the DBMS communicate. For further details, see Chapter 3, “Communications Blocks and Error Detection” on page 3-1.
  - **SUBSCHEMA-LR-CTRL** — The logical-record request control (LRC) block, through which the application program and LRF communicate. For further details, see Chapter 3, “Communications Blocks and Error Detection” on page 3-1.
  - **SUBSCHEMA-LR-NAMES** — The name of the program's subschema and the names of all database areas that can be accessed through the subschema. Logical-record names are not copied into the program; rather, they are moved as literals into the LRC block when needed to process a logical-record request.
  - **SUBSCHEMA-LR-RECORDS** — The descriptions of all logical records contained in the subschema.

- **MIXED** allows a program to access both database records and logical records; this usage mode requires the following source code components:
  - SUBSCHEMA-CTRL
  - SUBSCHEMA-NAMES
  - SUBSCHEMA-RECORDS
  - SUBSCHEMA-LR-CTRL
  - SUBSCHEMA-LR-RECORDS

  The use of MIXED mode is not recommended for the following reasons:
  - Issuing both logical-record and database requests requires that the program take into account the database currencies maintained in the paths used to service logical-record requests.
  - Accessing both logical records and database records in the same program can diminish the program's independence from the database structure and possibly interfere with the execution of paths invoked to provide requested logical-record access.
  - Logical-record path processing can interfere with program access to database records. You may need to insert a DML statement after a logical-record request to reestablish the appropriate currency.

The precompiler inserts the required data descriptions into the program automatically unless IDMS RECORDS MANUAL is specified in the IDMS-CONTROL SECTION

of the ENVIRONMENT DIVISION. If IDMS RECORDS MANUAL is specified, you must explicitly copy the required components, as outlined above, by coding COPY IDMS statements in the DATA DIVISION.

UTM modes only: You must include SUBSCHEMA-CTRL and all subschema records in the LINKAGE SECTION. You must include SUBSCHEMA-NAMES in the WORKING-STORAGE SECTION.

4.4.4.1 Syntax

```
►►─┬─ WORKING STORAGE SECTION. ─┬──────────────────────────────►
   └─ LINKAGE SECTION. ─────────┘

►──┬──────────────────────────────────────────────────────────►◄
   │   ┌──────────────────────────────────────────────────────┐
   └─↓─┬──────────────────┬─ COPY-IDMS ─────────────────────────┘
       └─ level-number ───┘
```

```
──►──────────────────────────────────────────────────────────►◄
   │                                                          │
   │──────────────────────────────────────────────────────────┘
   ├── SUBSCHEMA-DML-LR-DESCRIPTION ──────────────────────────── . ──┘
   │
   ├── SUBSCHEMA-DESCRIPTION ──────────────────────────────────┤
   │
   ├── SUBSCHEMA-CONTROL ──────────────────────────────────────┤
   │
   ├── SUBSCHEMA-CTRL ─────────────────────────────────────────┤
   │
   ├── SUBSCHEMA-NAMES ────────────────────────────────────────┤
   │
   ├── SUBSCHEMA-SSNAME ───────────────────────────────────────┤
   │
   ├── SUBSCHEMA-RECNAMES ─────────────────────────────────────┤
   │
   ├── SUBSCHEMA-SETNAMES ─────────────────────────────────────┤
   │
   ├── SUBSCHEMA-AREANAMES ────────────────────────────────────┤
   │
   ├── SUBSCHEMA-RECORDS ──────────────────────────────────────┤
   │
   ├── RECORD rec-name ──┬──────────────────────────────────┬──┤
   │                     └── VERSION vers-num ──┘ └── REDEFINES rec-name ──┘
   │
   ├── SUBSCHEMA-LR-DESCRIPTION ───────────────────────────────┤
   │
   ├── SUBSCHEMA-LR-CONTROL ───────────────────────────────────┤
   │
   ├── SUBSCHEMA-LR-CTRL ──┬──────────────────────────────────┤
   │                       ├── SIZE IS lrc-block-size ──┤
   │                       └── 512 ◄ ──────────────────┘
   │
   ├── SUBSCHEMA-LR-NAMES ─────────────────────────────────────┤
   │
   ├── SUBSCHEMA-LR-RECORDS ───────────────────────────────────┤
   │
   ├── LR logical-record-name ──┬──────────────────────────────┤
   │                            └── REDEFINES record-name ──┘
   │
   ├── MAPS ───────────────────────────────────────────────────┤
   │
   ├── MAP map-name ───────────────────────────────────────────┤
   │
   ├── MAP CONTROLS ───────────────────────────────────────────┤
   │
   ├── MAP CONTROL map-name ───────────────────────────────────┤
   │
   └── MAP RECORDS ────────────────────────────────────────────┘
```

4.4.4.2 Parameters

level-number

An integer in the range 01 through 48.

Instructs the precompiler to copy the descriptions into the program at a level other than that originally specified for the description in the data dictionary. If you specify a level number, the first level of code will be copied to the specified level; all other levels will be adjusted accordingly. If you do not specify a level, the descriptions copied will begin at 01 and have the same level numbers as originally specified in the data dictionary.

Note: Using the *level-number* clause can cause unpredictable results if record fields have been defined with a SYNCHRONIZED clause. Such fields may contain slack bytes, inserted to ensure correct alignment. Because CA-IDMS does not regard slack bytes as functional, it may misrepresent fields that contain such bytes. Therefore, you should ensure that all fields and records are properly structured.

COPY IDMS

Requests that the specified source data description code be copied into the DATA DIVISION at the location of the COPY IDMS statement.

SUBSCHEMA-DML-LR-DESCRIPTION

Copies all components required to access both database and logical records (SUBSCHEMA-CTRL, SUBSCHEMA-NAMES, SUBSCHEMA-RECORDS, SUBSCHEMA-LR-CTRL, SUBSCHEMA-LR-RECORDS). SUBSCHEMA-DML-LR-DESCRIPTION should be specified only when the subschema's usage mode is MIXED; do not specify SUBSCHEMA-DML-LR-DESCRIPTION if the usage mode is DML or LR.

SUBSCHEMA-DESCRIPTION

Copies all components required to access database records (SUBSCHEMA-CTRL, SUBSCHEMA-NAMES, and SUBSCHEMA-RECORDS). Do not specify SUBSCHEMA-DESCRIPTION if the subschema's usage mode is LR.

SUBSCHEMA-CONTROL

Copies both the SUBSCHEMA-CTRL and SUBSCHEMA-NAMES components. Do not specify SUBSCHEMA-CONTROL if the subschema's usage mode is LR.

SUBSCHEMA-CTRL

Copies the IDMS communications block data description; if the operating mode is IDMS-DC or DC-BATCH, SUBSCHEMA-CTRL copies the IDMS-DC communications block.

SUBSCHEMA-NAMES

Copies the eight-character literal name of the subschema and the literal names of all database records, sets, and areas contained in the subschema. SUBSCHEMA-NAMES includes SUBSCHEMA-SSNAME, SUBSCHEMA-RECNAMES, SUBSCHEMA-SETNAMES, and SUBSCHEMA-AREANAMES. Do not specify SUBSCHEMA-NAMES if the subschema's usage mode is LR.

SUBSCHEMA-SSNAME

Copies the eight-character literal name of the program's subschema. Do not specify SUBSCHEMA-SSNAME if the subschema's usage mode is LR.

SUBSCHEMA-RECNAMES

Copies the literal names of all database records contained in the subschema. Do not specify SUBSCHEMA-RECNAMES if the subschema's usage mode is LR.

SUBSCHEMA-SETNAMES

Copies the literal names of all sets contained in the subschema. Do not specify SUBSCHEMA-SETNAMES if the subschema's usage mode is LR.

SUBSCHEMA-AREANAMES

Copies the literal names of all database areas that can be accessed through the subschema. Do not specify SUBSCHEMA-AREANAMES if the subschema's usage mode is LR.

SUBSCHEMA-RECORDS

Copies the descriptions of all records contained in the subschema. COBOL synonyms defined for the subschema records in the data dictionary may be copied into the program, according to the rules of synonym usage. Do not specify SUBSCHEMA-RECORDS if the subschema's usage mode is LR.

Note: The OCCURS DEPENDING ON clause will be commented out for all schema-owned records. Therefore, although the maximum length of variable storage will be reserved, only the correct amount of data will be transferred to variable storage at run time.

Since COBOL will doubleword align an 01 level record, the precompiler adds up to seven bytes, if necessary, to make the record length divisible by eight when copying in a schema-owned record to an 01 level.

RECORD

Copies the description of a record defined in the data dictionary. If the subschema's usage mode is LR, only copy in IDD work records.

rec-name

The name of the record to be copied. Either the primary name or a synonym for a record stored in the data dictionary.

Schema-owned records cannot be copied into non-IDMS programs (that is, programs that do not use a subschema and that do not access the database). However, a synonym defined for a schema-owned record *can* be copied into a non-IDMS program (use the VERSION clause to identify the synonym).

IDD records: If an operating mode is associated with *record-name* in the data dictionary, it must agree with the mode in effect for the program (see "ENVIRONMENT DIVISION" earlier in this chapter).

VERSION

Optionally qualifies IDD records (but not schema-owned records) with a version number.

If you do not specify a version number, the default is the highest version number defined in the data dictionary for the language and operating mode under which the program is being compiled.

When copying a record that is schema owned using a synonym name, a version clause is needed, even if the synonym is not schema owned. The only time the version clause may be left off when copying a record using a synonym name is when the record is IDD owned. Once a record becomes schema owned, version clauses are needed.

vers-num

An integer in the range 1 through 9999.

You cannot specify a version number for a *rec-name* specified in the

subschema named in the DB *subschema-name* statement. The precompiler will automatically copy the correct version into the program.

REDEFINES

Copies a record description to an area previously defined by another record description. Two record descriptions can thus provide alternative definitions of the same storage location.

rec-name

The name of the record to be redefined.

SUBSCHEMA-LR-DESCRIPTION

Copies all components required to access logical records (SUBSCHEMA-CTRL, SUBSCHEMA-LR-CTRL, SUBSCHEMA-LR-NAMES, and SUBSCHEMA-LR-RECORDS). Do not include SUBSCHEMA-LR-DESCRIPTION if the subschema's usage mode is DML.

SUBSCHEMA-LR-CONTROL

Copies the SUBSCHEMA-CTRL, SUBSCHEMA-LR-CTRL, and SUBSCHEMA-LR-NAMES components. Do not include SUBSCHEMA-LR-CONTROL if the subschema's usage mode is DML.

SUBSCHEMA-LR-CTRL

Copies the LRC block data description.

SIZE IS

Specifies the size of that portion of the LRC block that contains information about the logical-request request WHERE clause (PXE).

If included, this parameter should specify a size large enough to accommodate the most complex WHERE clause in the program. The default, 512, is large enough to include approximately 32 operators, operands, and literals.

Do not include SUBSCHEMA-LR-CTRL if the subschema's usage mode is DML.

lrc-block-size

A positive integer in the range 0 through 9999.

Calculate the size as follows:

1. Multiply the greatest number of operands and operators that will be included in a single WHERE clause by 16 bytes
2. Add the number of bytes, rounded up to the nearest multiple of 8, associated with the data field for each operand that is a keyword or a program variable or logical-record field named in the OF LR clause
3. Add the length, rounded up to the nearest multiple of eight, of each operand that is a character literal
4. Add 12 bytes for each operand that is a numeric literal

Do not specify a block size if none of the logical-record requests issued by the program will include WHERE clauses.

SUBSCHEMA-LR-NAMES

Copies the literal name of the program's subschema and the literal names of all database areas that can be accessed through the subschema. Logical-record names are not copied into the program. Do not include SUBSCHEMA-LR-NAMES if the subschema's usage mode is DML.

SUBSCHEMA-LR-RECORDS

Copies the descriptions of all logical records defined in the subschema. All participating database records become 02-level group fields, permitting the program to reference as a group field that portion of a logical record that corresponds to a database record. Do not include SUBSCHEMA-LR-RECORDS if the subschema's usage mode is DML.

Note: The OCCURS DEPENDING ON clause will be commented out for all schema-owned records. Therefore, although the maximum length of variable storage will be reserved, only the correct amount of data will be transferred to variable storage at runtime.

When copying a schema-owned record to a level other than 01, the precompiler adds up to seven bytes, if necessary, to make the record length divisible by eight for doubleword alignment.

LR

Copies the description of an individual logical record contained in the subschema.

logical-record-name

The name of the logical record to copy.

REDEFINES

Copies a redefinition of the data contained in another logical record, a database record, or a non-IDMS record, while maintaining the same location in variable storage.

Do not include this statement if the subschema's usage mode is DML.

record-name

The name of the record to be redefined.

MAPS

Copies the map request block (MRB) and map records associated with all maps defined in the MAP SECTION.

MAP map-name

Copies the MRB and map records associated with the named map. The map version number defaults to the version specified for the map in the MAP SECTION.

MAP-CONTROLS

Copies the MRBs associated with all maps specified in the MAP SECTION.

MAP-CONTROL map-name

Copies the MRB for the named map. The map version number defaults to the version specified for the map in the MAP SECTION.

MAP-RECORDS

Copies the map records associated with all maps specified in the MAP SECTION.

Results of COPY IDMS specifications: The following figure shows the code copied into the DATA DIVISION as a result of COPY IDMS specifications.

COPY IDMS statements coded in the DATA DIVISION	Source code components brought in from the data dictionary by the DMLC processor															
	SUBSCHEMA | CTRL	SUBSCHEMA | SSNAME	SUBSCHEMA | RECNAMES	SUBSCHEMA | SETNAMES	SUBSCHEMA | AREANAMES	SUBSCHEMA | RECORDS	record | name	SUBSCHEMA | LR | CTRL	SUBSCHEMA | LR | NAMES	SUBSCHEMA | LR | RECORDS	logical | record | name	All Map Request Blocks	Named Map Request Block	All Map Records	map | name Map Records
SUBSCHEMA-DML-LR-DESCRIPTION	X	X	X	X	X	X									
SUBSCHEMA-DESCRIPTION	X	X	X	X	X	X		X		X					
SUBSCHEMA-CONTROL	X	X	X	X	X										
SUBSCHEMA-CTRL	X														
SUBSCHEMA-NAMES		X	X	X	X										

SUBSCHEMA-SSNAME		X												
SUBSCHEMA-RECNAMES			X											
SUBSCHEMA-SETNAMES				X										
SUBSCHEMA-AREANAMES					X									
SUBSCHEMA-RECORDS						X								
RECORD record-name							X							
SUBSCHEMA-LR-DESCRIPTION	X							X	X	X				
SUBSCHEMA-LR-CONTROL	X							X	X					
SUBSCHEMA-LR-CTRL								X						
SUBSCHEMA-LR-NAMES									X					
SUBSCHEMA-LR-RECORDS										X				
LR logical-record-name											X			
MAPS												X		X	
MAP-CONTROLS												X			
MAP CONTROL map-name													X		
MAP map-name													X		X
MAP-RECORDS														X	

4.5 PROCEDURE DIVISION

The COPY IDMS statements in the PROCEDURE DIVISION allow inclusion into the source program of BIND statements for CA-IDMS records and for procedure source statements defined as modules in the data dictionary by the DBA.

4.5.1 Syntax

```
►►── PROCEDURE DIVISION. ──────────────────────────────────────────►◄
►──┬──────────────────────────┬──────────────────────────────────►
  └─ COPY IDMS SUBSCHEMA-BINDS. ─┘
►──┬──────────────────────────────┬──────────────────────────────►
  └─ COPY IDMS SUBSCHEMA-RECORD-BINDS. ─┘
►──┬──────────────────────────┬──────────────────────────────────►
  └─ COPY IDMS MAP-BINDS. ─┘
►──┬──┬──────────────────────────────────────────────────────┬──►◄
   │  │                                                      │
   │  └↓─ COPY IDMS module module-name ──┬──────────────────┬─ . ─┘
   │                                     └─ VERSION version-number ─┘
```

4.5.2 Parameters

COPY IDMS SUBSCHEMA-BINDS

Initializes the PROGRAM-NAME field in the IDMS communications block and copies a standard BIND RUN-UNIT statement and appropriate standard BIND *record-name* commands for each CA-IDMS record in the program's DATA DIVISION. COPY IDMS SUBSCHEMA-BINDS does not generate BIND RECORD statements for logical records, nor are any needed.

In cases where more than one copy of a given database record description (including synonyms) is present in the program, COPY IDMS SUBSCHEMA-BINDS will not automatically generate bind record statements. Individual bind record statements must be issued to bind the record to the correct location.

If IDMS-RECORDS MANUAL has been specified in the ENVIRONMENT DIVISION, the COPY IDMS SUBSCHEMA-BINDS statement generates BINDS only for subschema records explicitly copied into the DATA DIVISION by means of COPY IDMS statements; it does not automatically generate BINDS for all subschema records.

Do not use the COPY IDMS SUBSCHEMA-BINDS statement when binding several records to the same location. Instead, code DML BIND statements in the PROCEDURE DIVISION for each record (for further details, see 5.14, “BIND RECORD” on page 5-41).

Note: If AUTOSTATUS is in use, a PERFORM IDMS-STATUS occurs automatically after each BIND generated by a COPY IDMS SUBSCHEMA-BINDS statement. If AUTOSTATUS is not in use, you

should explicitly code the BIND RUN-UNIT and BIND RECORD statements so that a PERFORM IDMS-STATUS can be coded after each BIND. For more information on AUTOSTATUS, see Chapter 3, “Communications Blocks and Error Detection” on page 3-1.

COPY IDMS SUBSCHEMA-RECORD-BINDS

Copies appropriate standard BIND *record-name* commands for each CA-IDMS record in the program's DATA DIVISION.

In cases where more than one copy of a given database record description (including synonyms) is present in the program, COPY IDMS SUBSCHEMA-RECORD-BINDS will not automatically generate bind record statements. Individual bind record statements must be issued to bind the record to the correct location.

If IDMS-RECORDS MANUAL has been specified in the ENVIRONMENT DIVISION, the COPY IDMS SUBSCHEMA-RECORD-BINDS statement generates BINDS only for subschema records explicitly copied into the DATA DIVISION by means of COPY IDMS statements; it does not automatically generate BINDS for all subschema records.

Do not use the COPY IDMS SUBSCHEMA-RECORD-BINDS statement when binding several records to the same location. Instead, code DML BIND statements in the PROCEDURE DIVISION for each record (for further details, see 5.14, “BIND RECORD” on page 5-41).

Note: If AUTOSTATUS is in use, a PERFORM IDMS-STATUS occurs automatically after each BIND generated by a COPY IDMS SUBSCHEMA-BINDS statement. If AUTOSTATUS is not in use, you should explicitly code the BIND RUN-UNIT and BIND RECORD statements so that a PERFORM IDMS-STATUS can be coded after each BIND. For more information on AUTOSTATUS, see Chapter 3, “Communications Blocks and Error Detection” on page 3-1.

COPY IDMS MAP-BINDS

Copies map- and map-record-specific BIND MAP statements for all maps in the program's MAP SECTION. For more information, see 5.12, “BIND MAP” on page 5-37.

COPY IDMS module

Copies source statements from a module stored in the data dictionary into the source program.

The unmodified module is placed into the program by the precompiler at the location of the request. The module can, but need not, contain DML statements. Any DML statements will be examined and expanded within the context of the program's subschema view and compile mode as if they were coded directly.

COPY IDMS MODULE statements *can* be nested (that is, code invoked by a COPY IDMS MODULE entry can itself contain a COPY IDMS MODULE statement). However, you must ensure that a copied module does not, in turn, copy itself.

module-name

The name of a module previously defined by the DBA by means of the IDD DDDL compiler.

The following standard modules are available for COBOL programs:

- IDMS-STATUS

Note: The IDMS-STATUS module must be copied into the program if an AUTOSTATUS protocol is in effect, as specified in the IDMS-CONTROL SECTION of the ENVIRONMENT DIVISION.

- IDMS-STATUS (BATCH-AUTOSTATUS)
- IDMS-STATUS (DC)
- IDMS-WAIT (DC)
- IDMS-WAIT (CICS)
- IDMS-WAIT (CICS STANDARD)
- IDMS-WAIT (CICS AUTOSTATUS)
- IDMS-WAIT (CICS STANDARD AUTOSTATUS)

VERSION

Optionally qualifies *module-name* with a version number.

If you do not specify a version number, the default is the highest version number defined in the data dictionary for the language mode under which the program is being compiled (for example, BATCH or IDMS-DC).

If no mode-specific version exists for *module-name*, the non-mode-specific version (if present) is copied. If neither a mode-specific entry nor a non-mode-specific entry for *module-name* has been established, an error results. The same rules apply to the module's language (that is, *version-number* defaults to the highest value defined in the data dictionary for the language in which the program is written).

version-number

An integer in the range 1 through 9999.

By default, if you do not specify a version number, the highest value defined in the data dictionary will be used.

Chapter 5. Data Manipulation Language Statements

5.1 About data manipulation language (DML) 5-11
5.2 ABEND 5-19
5.2.1 Syntax 5-19
5.2.2 Parameters 5-19
5.2.3 Example 5-19
5.2.4 Status codes 5-19
5.3 ACCEPT 5-20
5.3.1 Syntax 5-20
5.3.2 Parameters 5-20
5.3.3 Example 5-21
5.3.4 Status codes 5-21
5.4 ACCEPT BIND ADDRESS 5-22
5.4.1 Syntax 5-22
5.4.2 Parameters 5-22
5.4.3 Example 5-22
5.4.4 Status codes 5-22
5.5 ACCEPT DATABASE STATISTICS 5-23
5.5.1 Syntax 5-23
5.5.2 Parameters 5-23
5.5.3 Example 5-24
5.5.4 Status codes 5-24
5.6 ACCEPT DB-KEY FROM CURRENCY 5-25
5.6.1 Syntax 5-25
5.6.2 Parameters 5-25
5.6.3 Example 5-25
5.6.4 Status codes 5-26
5.7 ACCEPT DB-KEY RELATIVE TO CURRENCY 5-27
5.7.1 Syntax 5-27
5.7.2 Parameters 5-27
5.7.3 Example 5-28
5.7.4 Status codes 5-28
5.8 ACCEPT page-info-location 5-29
5.8.1 Syntax 5-29
5.8.2 Parameters 5-29
5.8.3 Example 5-30
5.8.4 Status codes 5-30
5.9 ACCEPT PROCEDURE CONTROL LOCATION 5-31
5.9.1 Syntax 5-31
5.9.2 Parameters 5-31

5.9.3 Example 5-31
5.9.4 Status codes 5-31
5.10 ACCEPT TRANSACTION STATISTICS 5-32
5.10.1 Syntax 5-32
5.10.2 Parameters 5-32
5.10.3 Example 5-34
5.10.4 Status codes 5-34
5.11 ATTACH 5-35
5.11.1 Syntax 5-35
5.11.2 Parameters 5-35
5.11.3 Example 5-36
5.11.4 Status codes 5-36
5.12 BIND MAP 5-37
5.12.1 Syntax 5-37
5.12.2 Parameters 5-37
5.12.3 Example 5-38
5.12.4 Status codes 5-38
5.13 BIND PROCEDURE 5-39
5.13.1 Syntax 5-39
5.13.2 Parameters 5-39
5.13.3 Example 5-39
5.13.4 Status codes 5-39
5.14 BIND RECORD 5-41
5.14.1 Syntax 5-41
5.14.2 Parameters 5-41
5.14.3 Example 5-42
5.14.4 Status codes 5-42
5.15 BIND RUN-UNIT 5-43
5.15.1 Syntax 5-43
5.15.2 Parameters 5-43
5.15.3 Example 5-44
5.15.4 Status codes 5-45
5.16 BIND TASK 5-47
5.16.1 Syntax 5-47
5.16.2 Parameters 5-47
5.16.3 Example 5-47
5.16.4 Status codes 5-47
5.17 BIND TRANSACTION STATISTICS 5-48
5.17.1 Syntax 5-48
5.17.2 Example 5-48
5.17.3 Status codes 5-48
5.18 CHANGE PRIORITY 5-49
5.18.1 Syntax 5-49
5.18.2 Parameters 5-49
5.18.3 Example 5-49
5.18.4 Status codes 5-49
5.19 CHECK TERMINAL 5-50
5.19.1 Syntax 5-50
5.19.2 Parameters 5-50

5.19.3 Example . . . 5-51
5.19.4 Status codes . . . 5-51
5.20 COMMIT . . . 5-53
5.20.1 Syntax . . . 5-53
5.20.2 Parameters . . . 5-53
5.20.3 Example . . . 5-53
5.20.4 Status codes . . . 5-53
5.21 CONNECT . . . 5-55
5.21.1 Syntax . . . 5-56
5.21.2 Parameters . . . 5-56
5.21.3 Example . . . 5-56
5.21.4 Status codes . . . 5-57
5.22 DC RETURN . . . 5-58
5.22.1 Syntax . . . 5-60
5.22.2 Parameters . . . 5-60
5.22.3 Example . . . 5-62
5.22.4 Status codes . . . 5-62
5.23 DELETE QUEUE . . . 5-63
5.23.1 Syntax . . . 5-63
5.23.2 Parameters . . . 5-63
5.23.3 Example . . . 5-63
5.23.4 Status codes . . . 5-63
5.24 DELETE SCRATCH . . . 5-65
5.24.1 Syntax . . . 5-65
5.24.2 Parameters . . . 5-65
5.24.3 Example . . . 5-66
5.24.4 Status codes . . . 5-66
5.25 DELETE TABLE . . . 5-67
5.25.1 Syntax . . . 5-67
5.25.2 Parameters . . . 5-67
5.25.3 Example . . . 5-67
5.25.4 Status codes . . . 5-68
5.26 DEQUEUE . . . 5-69
5.26.1 Syntax . . . 5-69
5.26.2 Parameters . . . 5-69
5.26.3 Example . . . 5-69
5.26.4 Status codes . . . 5-69
5.27 DISCONNECT . . . 5-71
5.28 Disconnecting a record from a set . . . 5-72
5.28.1 Syntax . . . 5-73
5.28.2 Parameters . . . 5-73
5.28.3 Example . . . 5-73
5.28.4 Status codes . . . 5-73
5.29 END LINE TERMINAL SESSION . . . 5-74
5.29.1 Syntax . . . 5-74
5.29.2 Example . . . 5-74
5.29.3 Status codes . . . 5-74
5.30 END TRANSACTION STATISTICS . . . 5-75
5.30.1 Syntax . . . 5-75

5.30.2 Parameters 5-75
5.30.3 Example 5-75
5.30.4 Status codes 5-75
5.31 ENDPAGE 5-77
5.31.1 Syntax 5-77
5.31.2 Example 5-77
5.31.3 Status codes 5-77
5.32 ENQUEUE 5-78
5.32.1 Syntax 5-78
5.32.2 Parameters 5-78
5.32.3 Examples 5-79
5.32.4 Status codes 5-79
5.33 ERASE 5-81
5.33.1 Syntax 5-81
5.33.2 Parameters 5-82
5.33.3 Example 5-82
5.33.4 Status codes 5-85
5.34 ERASE (LRF) 5-87
5.34.1 Syntax 5-87
5.34.2 Parameters 5-87
5.34.3 Example 5-88
5.35 FIND/OBTAIN 5-89
5.36 FIND/OBTAIN CALC/DUPLICATE 5-90
5.36.1 Syntax 5-90
5.36.2 Parameters 5-90
5.36.3 Example 5-91
5.36.4 Status codes 5-91
5.37 FIND/OBTAIN CURRENT 5-92
5.37.1 Syntax 5-92
5.37.2 Parameters 5-92
5.37.3 Example 5-92
5.37.4 Status codes 5-94
5.38 FIND/OBTAIN DB-KEY 5-95
5.38.1 Syntax 5-95
5.38.2 Parameters 5-95
5.38.3 Example 5-96
5.38.4 Status codes 5-96
5.39 FIND/OBTAIN OWNER 5-98
5.39.1 Syntax 5-98
5.39.2 Parameters 5-98
5.39.3 Example 5-99
5.39.4 Status codes 5-99
5.40 FIND/OBTAIN WITHIN SET USING SORT KEY 5-101
5.40.1 Syntax 5-101
5.40.2 Parameters 5-101
5.40.3 Example 5-103
5.40.4 Status codes 5-103
5.41 FIND/OBTAIN WITHIN SET/AREA 5-104
5.41.1 Syntax 5-105

5.41.2 Parameters 5-105
5.41.3 Example 5-106
5.41.4 Status codes 5-108
5.42 FINISH 5-110
5.42.1 Syntax 5-110
5.42.2 Parameters 5-110
5.42.3 Example 5-110
5.42.4 Status codes 5-110
5.43 FREE STORAGE 5-112
5.43.1 Syntax 5-112
5.43.2 Parameters 5-112
5.43.3 Example 5-112
5.43.4 Status codes 5-112
5.44 GET 5-114
5.44.1 Syntax 5-114
5.44.2 Parameters 5-114
5.44.3 Example 5-114
5.44.4 Status codes 5-114
5.45 GET QUEUE 5-116
5.45.1 Syntax 5-116
5.45.2 Parameters 5-116
5.45.3 Example 5-118
5.45.4 Status codes 5-118
5.46 GET SCRATCH 5-120
5.46.1 Syntax 5-120
5.46.2 Parameters 5-120
5.46.3 Example 5-122
5.46.4 Status codes 5-122
5.47 GET STORAGE 5-123
5.47.1 Syntax 5-123
5.47.2 Parameters 5-123
5.47.3 Example 5-126
5.47.4 Status codes 5-126
5.48 GET TIME 5-128
5.48.1 Syntax 5-128
5.48.2 Parameters 5-128
5.48.3 Example 5-129
5.48.4 Status codes 5-129
5.49 IF 5-130
5.49.1 Syntax 5-130
5.49.2 Parameters 5-130
5.49.3 Examples 5-131
5.49.4 Status codes 5-131
5.50 INQUIRE MAP 5-133
5.50.1 Syntax 5-133
5.50.2 Parameters 5-133
5.50.3 Example 5-136
5.50.4 Syntax 5-137
5.50.5 Parameters 5-137

5.50.6 Example 5-137
5.50.7 Syntax 5-138
5.50.8 Parameters 5-138
5.50.9 Example 5-138
5.50.10 Syntax 5-139
5.50.11 Parameters 5-139
5.50.12 Examples 5-141
5.50.13 Status codes 5-142
5.51 KEEP CURRENT 5-143
5.51.1 Syntax 5-143
5.51.2 Parameters 5-143
5.51.3 Example 5-143
5.51.4 Status codes 5-143
5.52 KEEP LONGTERM 5-145
5.52.1 Syntax 5-147
5.52.2 Parameters 5-147
5.52.3 Lock-options 5-148
5.52.4 Example 5-149
5.52.5 Status codes 5-149
5.53 LOAD TABLE 5-151
5.53.1 Syntax 5-151
5.53.2 Parameters 5-151
5.53.3 Examples 5-153
5.53.4 Status codes 5-154
5.54 MAP IN 5-155
5.54.1 Syntax 5-155
5.54.2 Parameters 5-155
5.54.3 Examples 5-157
5.54.4 Status codes 5-158
5.55 MAP OUT 5-160
5.55.1 Syntax 5-160
5.55.2 Parameters 5-161
5.55.3 Examples 5-164
5.55.4 Status codes 5-165
5.56 MAP OUTIN 5-167
5.56.1 Syntax 5-167
5.56.2 Parameters 5-167
5.56.3 Example 5-169
5.56.4 Status codes 5-169
5.57 MODIFY 5-171
5.57.1 Syntax 5-172
5.57.2 Parameters 5-172
5.57.3 Example 5-172
5.57.4 Status codes 5-172
5.58 MODIFY (LRF) 5-174
5.58.1 Syntax 5-174
5.58.2 Parameters 5-174
5.58.3 Example 5-175
5.59 MODIFY MAP 5-176

5.59.1 Syntax 5-176
5.59.2 Parameters 5-177
5.59.3 Examples 5-184
5.59.4 Status codes 5-185
5.60 OBTAIN (LRF) 5-186
5.60.1 Syntax 5-186
5.60.2 Parameters 5-186
5.60.3 Example 5-187
5.61 POST 5-189
5.61.1 Syntax 5-189
5.61.2 Parameters 5-189
5.61.3 Example 5-189
5.61.4 Status codes 5-189
5.62 PUT QUEUE 5-190
5.62.1 Syntax 5-190
5.62.2 Parameters 5-190
5.62.3 Example 5-191
5.62.4 Status codes 5-192
5.63 PUT SCRATCH 5-193
5.63.1 Syntax 5-193
5.63.2 Parameters 5-193
5.63.3 Example 5-194
5.63.4 Status codes 5-194
5.64 READ LINE FROM TERMINAL 5-195
5.64.1 Syntax 5-195
5.64.2 Parameters 5-195
5.64.3 Examples 5-196
5.64.4 Status codes 5-196
5.65 READ TERMINAL 5-198
5.65.1 Syntax 5-198
5.65.2 Parameters 5-198
5.65.3 Example 5-199
5.65.4 Status codes 5-200
5.66 READY 5-201
5.66.1 Syntax 5-201
5.66.2 Parameters 5-201
5.66.3 Example 5-202
5.66.4 Status codes 5-203
5.67 RETURN 5-204
5.67.1 Syntax 5-204
5.67.2 Parameters 5-204
5.67.3 Example 5-205
5.67.4 Status codes 5-206
5.68 ROLLBACK 5-207
5.68.1 Syntax 5-207
5.68.2 Parameters 5-207
5.68.3 Example 5-208
5.68.4 Status codes 5-208
5.69 SEND MESSAGE 5-209

5.69.1 Syntax 5-209
5.69.2 Parameters 5-209
5.69.3 Examples 5-210
5.69.4 Status codes 5-211
5.70 SET ABEND EXIT 5-212
5.70.1 Syntax 5-212
5.70.2 Parameters 5-212
5.70.3 Examples 5-212
5.70.4 Status codes 5-213
5.71 SET TIMER 5-214
5.71.1 Syntax 5-214
5.71.2 Parameters 5-214
5.71.3 Examples 5-216
5.71.4 Status codes 5-217
5.72 SNAP 5-218
5.72.1 Syntax 5-218
5.72.2 Parameters 5-218
5.72.3 Example 5-219
5.72.4 Status codes 5-219
5.73 STARTPAGE 5-220
5.73.1 Syntax 5-220
5.73.2 Parameters 5-220
5.73.3 Examples 5-221
5.73.4 Status codes 5-221
5.74 STORE 5-223
5.74.1 Syntax 5-224
5.74.2 Parameters 5-225
5.74.3 Example 5-225
5.74.4 Status codes 5-226
5.75 STORE (LRF) 5-228
5.75.1 Syntax 5-228
5.75.2 Parameters 5-228
5.75.3 Example 5-229
5.76 TRANSFER CONTROL 5-230
5.76.1 Syntax 5-230
5.76.2 Parameters 5-230
5.76.3 Examples 5-231
5.76.4 Status codes 5-231
5.77 WAIT 5-232
5.77.1 Syntax 5-232
5.77.2 Parameters 5-232
5.77.3 Example 5-233
5.77.4 Status codes 5-233
5.78 WRITE JOURNAL 5-234
5.78.1 Syntax 5-234
5.78.2 Parameters 5-234
5.78.3 Example 5-235
5.78.4 Status codes 5-235
5.79 WRITE LINE TO TERMINAL 5-236

5.79.1 Syntax 5-236
5.79.2 Parameters 5-237
5.79.3 Examples 5-238
5.79.4 Status codes 5-238
5.80 WRITE LOG 5-240
5.80.1 Syntax 5-241
5.80.2 Parameters 5-241
5.80.3 Example 5-244
5.80.4 Status codes 5-244
5.81 WRITE PRINTER 5-246
5.81.1 Syntax 5-247
5.81.2 Parameters 5-247
5.81.3 Examples 5-249
5.81.4 Status codes 5-249
5.82 WRITE TERMINAL 5-251
5.82.1 Syntax 5-251
5.82.2 Parameters 5-251
5.82.3 Example 5-252
5.82.4 Status codes 5-252
5.83 WRITE THEN READ TERMINAL 5-254
5.83.1 Syntax 5-254
5.83.2 Parameters 5-254
5.83.3 Example 5-257
5.83.4 Status codes 5-257
5.84 Logical-record clauses 5-258
5.84.1 WHERE 5-258
5.84.2 Syntax 5-258
5.84.3 Parameters 5-259
5.84.4 Parameters 5-259
5.84.5 Examples 5-262
5.84.6 ON clause 5-262
5.84.7 Syntax 5-263
5.84.8 Parameters 5-264
5.84.9 Examples 5-264
5.84.10 Status codes 5-264

5.1 About data manipulation language (DML)

CA-IDMS data manipulation language (DML) consists of statements that enable you to access the database management system (DBMS) and to request Logical Record Facility (LRF) and DC system services. The DML statements can be grouped into categories by function:

- **Control** statements:
  - Initiate and terminate processing
  - Effect recovery
  - Prevent concurrent retrieval and update of database records
  - Evaluate set conditions
- **Retrieval** statements locate records in the database and make them available to the application program.
- **Modification** statements add new records to the database and modify and delete existing records.
- **Accept** statements move special information such as database keys, storage addresses, and statistics from the DBMS to program variable storage.
- **Logical-record** statements retrieve, modify, store, and erase logical records.
- **Program management** statements:
  - Pass and return control from one program to another
  - Load and delete programs and tables
  - Define exit routines to be performed before an abnormal program termination (abend)
  - Force an abend condition
- **Storage management** statements allocate and release variable storage.
- **Task management** statements:
  - Initiate a new task
  - Change the dispatching priority of the issuing task
  - Enqueue and dequeue system resources
  - Signal that a task is to wait pending completion of an event
  - Post an event control block (ECB) indicating completion of an event
- **Time management** statements obtain the time and date, and define time-related events. These events include:
  - Placing the issuing task in a wait state for a specified duration of time
  - Posting a user-specified ECB after a specified interval

  - Initiating a new task after a specified interval

- **Scratch management** statements create, delete, or retrieve records from the scratch area.
- **Queue management** statements create, delete, or retrieve records from the queue area.
- **Terminal management** statements transfer data between the application program and the terminal.
- **Utility function** statements:
  - Request retrieval of task-related information
  - Request a memory dump of selected parts of storage
  - Retrieve and send a predefined message stored in the data dictionary
  - Send a specified message to one or more users or logical terminals
  - Collect, retrieve, and write CA-IDMS statistics on a transaction basis
  - Establish longterm database locks and monitor access to database records used across tasks during a pseudo-conversational transaction
- **Recovery** statements perform functions relating to database, scratch, and queue area recovery in the event of a system failure. These functions:
  - Establish checkpoints in the journal file for database, scratch, and queue records used by the issuing task
  - Roll back user database, scratch, and queue areas to the last checkpoint established
  - Establish an end-of-task checkpoint and relinquish control of all database, scratch, and queue areas associated with the issuing task
  - Write user-defined records to the journal file

This section describes each DML statement that requests a CA-IDMS database access or an online service. The DML statements are presented in two ways to help you understand their function in the CA-IDMS environment. The following table presents the DML statements by function (for example, retrieval statements and program management statements). Statements that apply to the online environment only are marked with (o). Statements that apply to DC-BATCH only are marked with (dcb). Statements that apply to DC-BATCH or the online environment only are marked with (o,dcb). Following the table, each DML statement is presented in alphabetical order; function, syntax, syntax rules, examples, and associated error-status codes are described in detail. Run-time currency affected by DML statements that navigate the database is described where appropriate.

The WHERE and ON clauses, which are used with LRF DML statements, are described in detail at the end of this section.

Note: All DML operands are positional.

DML statements grouped by function

Function	DML Statement
Control Statements	■ BIND RUN-UNIT — Signs on the application program to the DBMS ■ BIND TASK — Establishes a connection with the CA-IDMS/DC/UCF system from a batch program and allows certain online functions, such as writing to queues or printing to a printer controlled by the CA-IDMS/DC/UCF system (dcb) ■ BIND RECORD — Establishes addressability in variable storage for one or more records included in the program's subschema ■ BIND PROCEDURE — Establishes communication between the application program and a DBA-defined database procedure ■ READY — Prepares database areas for processing ■ FINISH — Releases database areas from program control ■ IF — Evaluates the presence of records in a set or a record's membership status and specifies action based on the outcome ■ COMMIT — Writes a checkpoint to the journal file and releases record locks ■ ROLLBACK — Requests recovery of database, scratch, and queue areas ■ KEEP — Places locks on record occurrences

Function	DML Statement
Retrieval Statements	■ FIND/OBTAIN DB-KEY — Accesses a record using a db-key previously saved by the program ■ FIND/OBTAIN CURRENT — Accesses a record using previously established currencies ■ FIND/OBTAIN WITHIN SET/AREA — Accesses a record based on its logical location within a set or its physical location within an area ■ FIND/OBTAIN OWNER — Accesses the owner record of a set occurrence ■ FIND/OBTAIN CALC/DUPLICATE — Accesses a record using its CALC-key value ■ FIND/OBTAIN USING SORT KEY — Accesses a record in a sorted set using its sort-key value ■ GET — Moves all data associated with a previously located record into program variable storage ■ RETURN — Retrieves the database key and symbolic key of an indexed record entry
Modification Statements	■ STORE — Adds a new record to the database ■ MODIFY — Changes the contents of an existing record ■ CONNECT — Links a record to a set ■ DISCONNECT — Removes a member record from a set ■ ERASE — Deletes a record from the database
Recovery Functions	■ COMMIT — Establishes a checkpoint in the journal file for database, scratch, and queue record activity ■ FINISH — Relinquishes control of database, scratch, and queue areas ■ ROLLBACK — Rolls back database, scratch, and queue areas to the last checkpoint

Function	DML Statement
Accept Statements	■ ACCEPT DB-KEY FROM CURRENCY — Saves the db-key of the current record of run unit, record type, set, or area ■ ACCEPT DB-KEY RELATIVE TO CURRENCY — Saves the db-key of the next, prior, or owner record relative to the current record of a set ■ ACCEPT IDMS STATISTICS — Returns system run-time statistics to the program ■ ACCEPT BIND RECORD — Returns a record's bind address to the program ■ ACCEPT PROCEDURE — Returns information from the application program information block associated with a database procedure to the program
Logical Record Facility	■ ERASE — Deletes a logical record ■ MODIFY — Modifies a logical record ■ OBTAIN — Accesses a logical record ■ STORE — Stores a logical record
Program Management	■ TRANSFER CONTROL (LINK) — Passes control to another program with the expectation of receiving it back (o) ■ TRANSFER CONTROL (XCTL) — Passes control to another program with no expectation of receiving it back (o) ■ DC RETURN — Returns control to the next higher level calling program (o) ■ LOAD TABLE — Loads a program or table into the CA-IDMS system program pool (o) ■ DELETE TABLE — Signals that a program has finished using a program or a table in the program pool (o) ■ SET ABEND EXIT (STAE) — Establishes linkage to a program or routine that will receive control in the event of an abend (o) ■ ABEND — Abnormally terminates the issuing task (o)
Storage Management	■ GET STORAGE — Allocates variable storage from a CA-IDMS system storage pool (o) ■ FREE STORAGE — Frees all or part of a block of variable storage (o)

Function	DML Statement
Task Management	■ ATTACH — Attaches a new task within the CA-IDMS system (o) ■ CHANGE PRIORITY — Changes the dispatching priority of the issuing task (o) ■ ENQUEUE — Acquires a resource or a list of resources (o) ■ DEQUEUE — Releases a resource (o) ■ WAIT — Relinquishes control to the CA-IDMS system while awaiting completion of an event (o) ■ POST — Posts an event control block (ECB) (o)
Time Management	■ GET TIME — Obtains the time and date from the system ■ SET TIMER — Defines a time-delayed event (o)
Scratch Management	■ PUT SCRATCH — Stores a scratch record (o) ■ GET SCRATCH — Retrieves a scratch record (o) ■ DELETE SCRATCH — Deletes a scratch record (o)
Queue Management	■ PUT QUEUE — Stores a queue record (o,dcb) ■ GET QUEUE — Retrieves a queue record (o,dcb) ■ DELETE QUEUE — Deletes a queue record (o,dcb)
Terminal Management (Basic Mode)	■ READ TERMINAL — Requests a synchronous or asynchronous data transfer from the terminal to program variable storage (o) ■ WRITE TERMINAL — Requests a synchronous or asynchronous data transfer from program variable storage to the terminal buffer (o) ■ WRITE THEN READ TERMINAL — Requests a synchronous or asynchronous data transfer from program variable storage to the terminal buffer and, upon an operator signal, back to variable storage (o) ■ CHECK TERMINAL — Ensures that a previously issued asynchronous I/O operation is complete (o)

Function	DML Statement
Terminal Management (Line Mode)	■ READ LINE FROM TERMINAL — Requests a synchronous data transfer from the terminal to the issuing program (o) ■ WRITE LINE TO TERMINAL — Requests a synchronous or asynchronous data transfer from the issuing program to the terminal (o) ■ END LINE TERMINAL SESSION — Terminates the current line I/O session (o) ■ WRITE PRINTER — Requests transmission of data from a task to a printer (o,dcb)
Terminal Management (Mapping Mode)	■ BIND MAP — Identifies the location of the map request block (MRB) and initializes MRB fields (o) ■ MAP IN — Requests a transfer of data from the terminal to program variable storage (o) ■ MAP OUT — Requests a transfer of data from program variable storage to the terminal (o) ■ MAP OUTIN — Requests a transfer of data from program variable storage to the terminal and, upon an operator signal, back to variable storage (o) ■ INQUIRE MAP — Obtains information or tests conditions concerning the previous map operation (o) ■ MODIFY MAP — Requests modifications of mapping options for a map (o) ■ STARTPAGE — Begins a map paging session and specifies options for that session (o) ■ ENDPAGE — Terminates a map paging session (o)

Function	DML Statement
Utility Functions	■ ACCEPT — Retrieves task-related information (o) ■ SNAP — Requests a memory dump of selected parts of storage (o) ■ SEND MESSAGE — Sends a message to a user, logical terminal, or list of users or logical terminals (o) ■ BIND TRANSACTION STATISTICS — Defines the beginning of a transaction for the purpose of collecting transaction statistics (o) ■ ACCEPT TRANSACTION STATISTICS — Returns the contents of the transaction statistics block (TSB) to program variable storage (o) ■ END TRANSACTION STATISTICS — Defines the end of a transaction (o) ■ KEEP LONGTERM — Either modifies a prior KEEP LONGTERM request, or enables database longterm locks or database monitoring for records, sets, or areas ■ WRITE JOURNAL — Writes user-defined records to the journal file ■ WRITE LOG — Retrieves a message from the data dictionary and sends it to a predefined destination (o)

5.2 ABEND

The ABEND statement terminates the issuing task abnormally. ABEND also requests CA-IDMS to invoke or ignore previously established abend exits and to write a task dump to the log file. After completion of the ABEND function, processing control is returned to the CA-IDMS program-control module.

5.2.1 Syntax

```
►►── ABEND CODE abend-code ─┬─ DUMP   ─┬─┬─ EXITS ─┬─ INVOKED ◄ ─┬─ . ──►◄
                            └─ NODUMP ◄ ─┘         └─ IGNORED   ─┘
```

5.2.2 Parameters

CODE
Specifies a four-character user-defined abend code.

abend-code
Either the symbolic name of a variable-storage field that contains the abend code or the code itself enclosed in quotation marks.

Note: Because the specified abend code appears in the system log and is displayed at the task's terminal, you should not use CA-IDMS system abend codes.

DUMP/NODUMP
Specifies whether CA-IDMS will write a formatted task dump to the log file. The default is NODUMP.

EXITS INVOKED/IGNORED
Specifies whether CA-IDMS will invoke or ignore abend routines established by SET ABEND EXIT (STAE) requests. The default is INVOKED.

5.2.3 Example

The following example specifies an abend request that terminates the issuing task abnormally and requests CA-IDMS to write a task dump to the log file and to ignore any abend exits:

```
ABEND CODE 'U876'
   DUMP
   EXITS IGNORED.
```

5.2.4 Status codes

Because control is passed to the CA-IDMS program-control module, there is no need for the program to check the ERROR-STATUS field.

5.3 ACCEPT

The ACCEPT statement retrieves the following task-related information:

- Current task code
- Task identifier
- Logical terminal identifier
- Physical terminal identifier
- CA-IDMS system version
- User identifier (the ID of the user signed on to the task's logical terminal)
- Physical terminal screen dimensions

5.3.1 Syntax

```
►►── ACCEPT ─┬─ TASK CODE ──┬─ INTO return-location . ──────────────────►◄
            ├─ TASK ID ────┤
            ├─ LTERM ID ───┤
            ├─ PTERM ID ───┤
            ├─ SYSVERSION ─┤
            ├─ USER ID ────┤
            └─ SCREENSIZE ─┘
```

5.3.2 Parameters

ACCEPT

Retrieves the specified information.

TASK CODE

Retrieves the 1 through 8 character code used to invoke the current task.

TASK ID

Retrieves the task identifier assigned by CA-IDMS. The task identifier is a unique sequence number stored in a PIC S9(8) COMP SYNC (fullword) field. At system startup, the identifier is set to zero; each time a task is executed, the ID is incremented by one.

LTERM ID

Retrieves the 1 through 8 character identifier of the logical terminal associated with the current task. If the current task is not associated with a terminal, CA-IDMS returns a null value of all spaces.

PTERM ID

Retrieves the 1 through 8 character identifier of the physical terminal associated with the current task. If the current task is not associated with a terminal, CA-IDMS returns a null value of all spaces.

SYSVERSION

Retrieves the version of the current CA-IDMS system. The version number is an integer in the range 0 through 9999 stored in a PIC S9(4) COMP (halfword) field.

USER ID

Retrieves the 32-character identifier of the user signed on to the logical terminal associated with the current task. If no user is signed on, CA-IDMS returns a null value of all spaces.

SCREENSIZE

Retrieves the screen dimensions of the physical terminal associated with the current task. The screen size is returned to a field that is divided into two PIC S9(4) COMP (halfword) fields. The first halfword contains the row, the second halfword contains the column. For example, a 24-line by 80-character screen is represented by a value of 24 in the first halfword and 80 in the second halfword. If the current task is not associated with a terminal, CA-IDMS returns a null value of zero.

INTO

Specifies the location to which CA-IDMS returns the requested task-related information.

return-location

The symbolic name of a user-defined field; the picture and usage of this field must be compatible with the picture and usage of the requested data.

5.3.3 Example

The following example illustrates ACCEPT statements that retrieve the ID of the current task and the ID of the user signed on to the logical terminal associated with that task:

```
ACCEPT TASK ID INTO TASK-ID.
ACCEPT USER ID INTO USER-ID.
```

5.3.4 Status codes

After completion of the ACCEPT function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4829	An invalid parameter has been passed from the program

5.4 ACCEPT BIND ADDRESS

The ACCEPT BIND ADDRESS statement moves the bind address of a record to a specified location in program variable storage. This statement is typically requested by a subprogram that requires the address of a record in order to access it.

Currency: The ACCEPT BIND ADDRESS statement updates no currencies and requires no currencies to be set relative to the specified record.

5.4.1 Syntax

```
►►── ACCEPT bind-address FROM record-name BIND . ──────────────────────►◄
```

5.4.2 Parameters

bind-address
A PIC S9(8) COMP SYNC (fullword) field, containing the location into which the bind address of the specified record will be copied.

FROM ... BIND
Specifies the record whose bind address will be copied into the specified location in variable storage.

record-name
The name of a record previously bound by the run unit.

5.4.3 Example

The following statement moves the bind address for an EMPLOYEE record to a location identified as REG1 in the requesting subprogram:

```
ACCEPT REG1 FROM EMPLOYEE BIND.
```

5.4.4 Status codes

After completion of the ACCEPT BIND ADDRESS function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
1508	The named record is not in the specified subschema

5.5 ACCEPT DATABASE STATISTICS

The ACCEPT DATABASE STATISTICS statement copies system run-time statistics located in the program's IDMS statistics block to program variable storage. This statement can be issued any number of times during the execution of a run unit. For example, you might request database statistics after storing a variable-length record to determine whether the entire record was stored in one place or if fragments were placed in an overflow area.

The ACCEPT DATABASE STATISTICS statement does not reset any of the statistics fields to zero; resetting of IDMS statistics block fields occurs only upon issuing a FINISH command.

The ACCEPT DATABASE STATISTICS statement is used in both the navigational and the non-navigational environments.

5.5.1 Syntax

```
►►── ACCEPT db-statistics FROM IDMS-STATISTICS . ──────────────────────►◄
```

5.5.2 Parameters

db-statistics

The name of a fullword-aligned 100-byte field in program variable storage.

The data copied from IDMS-STATISTICS to *db-statistics* is formatted as follows:

```
01  DB-STATISTICS
  03  DATE-TODAY              PIC X(8).
  03  TIME-TODAY              PIC X(8).
  03  PAGES-READ              PIC S9(8)    COMP.
  03  PAGES-WRITTEN           PIC S9(8)    COMP.
  03  PAGES-REQUESTED         PIC S9(8)    COMP.
  03  CALC-TARGET             PIC S9(8)    COMP.
  03  CALC-OVERFLOW           PIC S9(8)    COMP.
  03  VIA-TARGET              PIC S9(8)    COMP.
  03  VIA-OVERFLOW            PIC S9(8)    COMP.
  03  LINES-REQUESTED         PIC S9(8)    COMP.
  03  RECS-CURRENT            PIC S9(8)    COMP.
  03  CALLS-TO-IDMS           PIC S9(8)    COMP.
  03  FRAGMENTS-STORED        PIC S9(8)    COMP.
  03  RECS-RELOCATED          PIC S9(8)    COMP.
 *03  LOCKS-REQUESTED         PIC S9(8)    COMP.
 *03  SEL-LOCKS-HELD          PIC S9(8)    COMP.
 *03  UPD-LOCKS-HELD          PIC S9(8)    COMP.
 *03  RUN-UNIT-ID             PIC S9(8)    COMP.
 *03  TASK-ID                 PIC S9(8)    COMP.
 *03  LOCAL-ID                PIC X(8).
  03  FILLER                  PIC X(8).

 *Applies to the CA-IDMS central version only
```

The LOCAL-ID field consists of the four-byte identifier of the interface in which the run unit originated (for example, BATC, DBDC, or CICS) and a unique identifier (fullword) assigned to the run unit by that interface. For batch and CMS run units, this identifier specifies the internal machine time. For CICS run units, this identifier specifies the CICS transaction number assigned to the run unit.

To display the originating interface identifier and the run-unit identifier for a program, the LOCAL-ID field can be moved to a work field:

```
01   WORK-LOCAL-ID.
  02  WORK-LOCAL-ORIGIN      PIC X(4).
  02  WORK-LOCAL-NUMBER      PIC S9(8)   COMP.
```

Alternatively, the DB-STATISTICS record from the data dictionary can be modified by your DBA to define two subordinate fields for the LOCAL-ID field. The DB-STATISTICS record describes the IDMS statistics block. To use this record, code the following statement in program variable storage:

```
01 COPY IDMS DB-STATISTICS.
```

5.5.3 Example

The following statements establish currency for the sets in which a new EXPERTISE record will participate as a member, store the EXPERTISE record, and move statistics regarding the stored EXPERTISE record to the DB-STATISTICS location in main storage:

```
MOVE EMP-ID-IN TO EMP-ID-0415.
FIND CALC EMPLOYEE.
MOVE SKILL-ID-IN TO SKILL-ID-0455.
FIND CALC SKILL.
STORE EXPERTISE.
ACCEPT DB-STATISTICS FROM IDMS-STATISTICS.
```

5.5.4 Status codes

After completion of the ACCEPT DATABASE STATISTICS function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
1518	The database statistics location has not been bound properly

5.6 ACCEPT DB-KEY FROM CURRENCY

The ACCEPT DB-KEY FROM CURRENCY statement moves the db-key of the current record of run unit, record type, set, or area to a specified location in program variable storage. Records whose db-keys are saved in this manner are available for subsequent direct access by using a FIND/OBTAIN DB-KEY statement.

Note: You must establish currency before using this statement. If no currency has been established, the DBMS returns 0000 to the ERROR-STATUS field and -1 to the *db-key* field.

Currency: ACCEPT DB-KEY FROM CURRENCY does not update any currencies.

5.6.1 Syntax

```
►►── ACCEPT db-key-location FROM ┬──────────────┬─ CURRENCY . ──────────►◄
                                ├─ record-name ─┤
                                ├─ set-name ────┤
                                └─ area-name ───┘
```

5.6.2 Parameters

db-key-location
A PIC S9(8) COMP SYNC (fullword) field. Identifies the location in variable storage that will contain the db-key of the specified record.

FROM CURRENCY
Specifies the record whose db-key will be placed in the specified location. By default, if you omit a record, set, or area qualifier, the db-key of the record that is current of the run unit is saved.

record-name
Saves the db-key of the record that is current of the specified record type.

set-name
Saves the db-key of the record that is current of the specified set.

area-name
Saves the db-key of the record that is current of the specified area.

5.6.3 Example

The following statements establish a DEPARTMENT record as current of run unit and save its db-key in location SAVE-DB-KEY:

```
MOVE '8683' TO DEPT-ID-0410.
FIND CALC DEPARTMENT.
ACCEPT SAVE-DB-KEY FROM CURRENCY.
```

Note: The same results can be accomplished using the following COBOL MOVE statement:

```
MOVE DB-KEY TO SAVE-DB-KEY.
```

5.6.4 Status codes

After completion of the ACCEPT DB-KEY FROM CURRENCY function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
1508	The named record or set is not in the subschema. The program has probably invoked the wrong subschema.
1523	The named area is not in the subschema.

5.7 ACCEPT DB-KEY RELATIVE TO CURRENCY

The ACCEPT DB-KEY RELATIVE TO CURRENCY statement moves a selected db-key to a specified location in program variable storage. The db-key moved to variable storage can be the db-key of the next, prior, or owner record relative to the current record of set.

This version of the ACCEPT statement allows you to save the db-key of a record within a set without actually having to access the record. Records whose db-keys are saved in this manner are available for subsequent direct access by using a FIND/OBTAIN DB-KEY statement.

Note: You must establish currency before using this statement. If no set currency has been established, the DBMS returns 0000 to the ERROR-STATUS field and -1 to the *db-key-location* field.

Currency: ACCEPT DB-KEY RELATIVE TO CURRENCY does not update any currencies.

5.7.1 Syntax

```
►►── ACCEPT db-key-location FROM set-name ─┬─ NEXT ─┬─ CURRENCY . ──────►◄
                                          ├─ PRIOR ─┤
                                          └─ OWNER ─┘
```

5.7.2 Parameters

db-key-location
A PIC S9(8) COMP SYNC (fullword) field. Identifies the location in variable storage that will contain the db-key of the requested record.

FROM .. CURRENCY
Identifies the record whose db-key will be moved into the specified location.

set-name
The name of a set included in the subschema.

Native VSAM users: NEXT/PRIOR/OWNER CURRENCY cannot be requested for sets defined for native VSAM records.

NEXT
Saves the db-key of the next record relative to the record that is current of the specified set. NEXT CURRENCY cannot be requested unless the specified set has prior pointers; prior pointers ensure that the next pointer in the prefix of the current record does not point to a logically deleted record.

PRIOR
Saves the db-key of the prior record relative to the record that is current of the specified set. PRIOR CURRENCY cannot be requested unless the specified set has prior pointers.

Note: No indication of an end-of-set condition is possible for an ACCEPT NEXT or PRIOR. A retrieval command must be issued to determine whether the next or prior record in the set occurrence is the owner record.

OWNER

Saves the db-key of the owner of the record that is current of the specified set. A request for OWNER CURRENCY cannot be executed unless the specified set has owner pointers. However, if the current record of the named set is the owner record occurrence, requests for OWNER CURRENCY return the db-key of the record itself, even if this set does not have owner pointers.

Note: When a record declared as an optional or manual member of a set is accessed, it is *not* established as current of set if it is not currently connected to an occurrence of the specified set. A subsequent attempt to access the owner record will locate instead the owner of the current record of set. In such cases, determine whether the retrieved record is actually a set member before executing the ACCEPT DB-KEY FROM OWNER CURRENCY statement. The IF statement, explained later in this chapter, can be used for this purpose.

5.7.3 Example

The following statements access the EMP-EXPERTISE set and save the db-key of the owner record of the SKILL-EXPERTISE set:

```
MOVE '0119' TO EMP-ID-0415.
FIND CALC EMPLOYEE.
FIND FIRST WITHIN EMP-EXPERTISE.
ACCEPT SAVE-DB-KEY FROM SKILL-EXPERTISE OWNER CURRENCY.
```

5.7.4 Status codes

After completion of the ACCEPT DB-KEY RELATIVE TO CURRENCY function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
1508	The named set is not in the subschema. The program has probably invoked the wrong subschema.

5.8 ACCEPT page-info-location

The ACCEPT page-info-location statement moves the page information for a given record to a specified location in program variable storage. Page information that is saved in this manner is available for subsequent direct access by using a FIND/OBTAIN DB-KEY statement.

The dbkey radix portion of the page information can be used in interpreting a dbkey for display purposes and in formatting a dbkey from page and line numbers. The dbkey radix represents the number of bits within a dbkey value that are reserved for the line number of a record. By default, this value is 8, meaning that up to 255 records can be stored on a single page of the area. Given a dbkey, you can separate its associated page number by dividing the dbkey by 2 raised to the power of the dbkey radix. For example, if the dbkey radix is 4, you would divide the dbkey value by 2**4. The resulting value is the page number of the dbkey. To separate the line number, you would multiply the page number by 2 raised to the power of the dbkey radix and subtract this value from the dbkey value. The result would be the line number of the dbkey. The following two formulas can be used to calculate the page and line numbers from a dbkey value:

- Page-number = dbkey value / (2 ** dbkey radix)
- Line-number = dbkey value - (page-number * (2 ** dbkey radix))

5.8.1 Syntax

```
►►── ACCEPT page-info-location FOR record-name . ──────────────────►◄
```

5.8.2 Parameters

ACCEPT
Retrieves the specified information.

page-info-location
Specifies a four-byte field that may be defined either as a group field or as a fullword field (PIC S9(8) COMP). Identifies the location in variable storage that contains page information for the specified record. Upon successful completion of this statement, the first two bytes of the field contain the page group number and the last two bytes contain a value that may be used for interpreting dbkeys.

FOR
Specifies the record whose page information will be placed in the specified location.

record-name
Specifies the record whose page information will be placed in the specified location.

5.8.3 Example

The following example retrieves the page information for the DEPARTMENT record and uses the dbkey format information to transform a page number into a dbkey.

```
01 W-PG-INFO.
   02 W-GRP-NUM       PIC S9(4) COMP.
   02 W-DBK-FORMAT    PIC 9(4) COMP.

   ACCEPT W_PG_INFO FOR DEPARTMENT.
   MOVE W-PAGE TO W-DBKEY.
   PERFORM ADJUST-PAGE W-DBK-FORMAT TIMES.

ADJUST-PAGE SECTION.
   MULTIPLY W-DBKEY BY 2.}
```

5.8.4 Status codes

After completion of the ACCEPT page-info-location function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
1508	The named record is not in the subschema. The program probably invoked the wrong subschema.

5.9 ACCEPT PROCEDURE CONTROL LOCATION

The ACCEPT PROCEDURE CONTROL LOCATION statement copies the 256-byte application program information block associated with a previously defined database procedure to a specified location in program variable storage. A BIND PROCEDURE statement (explained later in this chapter) previously placed information into this block; this information may have been subsequently updated by the procedure.

The ACCEPT PROCEDURE CONTROL LOCATION statement should be used only by programs running under, but in a different partition from, the central version.

5.9.1 Syntax

```
►►── ACCEPT procedure-control-location FROM procedure-name PROCEDURE . ──►◄
```

5.9.2 Parameters

procedure-control-location
The fullword-aligned 256-byte location in variable storage to which the application program information block is to be copied.

FROM procedure-name PROCEDURE
The name of the database procedure whose application program information block is to be copied into variable storage. *Procedure-name* must refer to an eight-character field in variable storage.

5.9.3 Example

The following statement copies the application program information block used by the CHECKALL procedure to the location identified as CHECK-IT in main storage:

```
ACCEPT CHECK-IT FROM CHECKALL PROCEDURE.
```

5.9.4 Status codes

After completion of the ACCEPT PROCEDURE CONTROL LOCATION function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
1508	The named procedure is not in the specified subschema
1518	The procedure control location has not been bound properly

5.10 ACCEPT TRANSACTION STATISTICS

The ACCEPT TRANSACTION STATISTICS statement copies the contents of the transaction statistics block (TSB) to a location in program variable storage and optionally writes the TSB to the DC system log file.

5.10.1 Syntax

```
►►── ACCEPT TRANSACTION STATISTICS ──────────────────────────────►
►─┬─ WRITE ◄ ─┬─┬─ INTO return-stat-data-location ─┬─ . ──────────►◄
  └─ NOWRITE ─┘ │                                  │
                └──────────────────────────────────┘
```

5.10.2 Parameters

WRITE
Specifies that the TSB will be written to the DC system log file.

This is the default.

NOWRITE
Specifies that the TSB will not be written to the DC system log file.

INTO
Specifies the location to which CA-IDMS copies the TSB.

return-stat-data-location
A fullword-aligned 388-byte field. The data copied from the TSB is formatted as follows:

```
01  STATISTICS-BLOCK.
    03  SYS-INTERN1    PIC X(8)          SYSTEM INTERNAL USE ONLY
    03  PROG-CALL      PIC S9(8) COMP    # OF PROGRAMS CALLED
    03  PROG-LOAD      PIC S9(8) COMP    # OF PROGRAMS LOADED
    03  TERM-READ      PIC S9(8) COMP    # OF TERMINAL READS
    03  TERM-WRITE     PIC S9(8) COMP    # OF TERMINAL WRITES
    03  TERM-ERROR     PIC S9(8) COMP    # OF TERMINAL ERRORS
    03  STORAGE-GET    PIC S9(8) COMP    # OF STORAGE GETS
    03  SCRATCH-GET    PIC S9(8) COMP    # OF SCRATCH GETS
    03  SCRATCH-PUT    PIC S9(8) COMP    # OF SCRATCH PUTS
    03  SCRATCH-DEL    PIC S9(8) COMP    # OF SCRATCH DELETES
    03  QUEUE-GET      PIC S9(8) COMP    # OF QUEUE GETS
    03  QUEUE-PUT      PIC S9(8) COMP    # OF QUEUE PUTS
    03  QUEUE-DEL      PIC S9(8) COMP    # OF QUEUE DELETES
    03  GET-TIME       PIC S9(8) COMP    # OF GET TIMES
    03  SET-TIME       PIC S9(8) COMP    # OF SET TIMES
    03  DB-SRVREQ      PIC S9(8) COMP    # OF DB SERVICE REQUESTS
    03  MAX-STACK      PIC S9(8) COMP    MAX WORDS USED IN STACK
    03  USER-TIME      PIC S9(8) COMP    USER MODE TIME (10**-4 SEC)
    03  SYS-TIME       PIC S9(8) COMP    SYS MODE TIME (10**-4 SEC)
    03  WAIT-TIME      PIC S9(8) COMP    WAIT TIME (10** -4 SEC)
    03  MAX-RCE-USED   PIC S9(8) COMP    MAXIMUM NUMBER OF RCES USED
    03  MAX-RLE-USED   PIC S9(8) COMP    MAXIMUM NUMBER OF RLES USED
    03  MAX-DPE-USED   PIC S9(8) COMP    MAXIMUM NUMBER OF DPES USED
    03  STG-HI-MARK    PIC S9(8) COMP    STORAGE HIGH WATER MARK
    03  FREESTG-REQ    PIC S9(8) COMP    # OF FREE STORAGE REQUESTS
    03  SYS-SERV       PIC S9(8) COMP    # OF SYSTEM SERVICE REQUEST
    03  SYS-INTERN2    PIC X(8)          SYSTEM INTERNAL USE ONLY
    03  PAGES-READ     PIC S9(8) COMP    # OF PAGES READ
    03  PAGES-WRIT     PIC S9(8) COMP    # OF PAGES WRITTEN
    03  PAGES-REQ      PIC S9(8) COMP    # OF PAGES REQUESTED
    03  CALC-NO        PIC S9(8) COMP    # OF CALC RECS NO OFLOW
    03  CALC-OF        PIC S9(8) COMP    # OF CALC RECS OFLOW
    03  VIA-NO         PIC S9(8) COMP    # OF VIA RECS NO OFLOW
    03  VIA-OF         PIC S9(8) COMP    # OF VIA RECS OFLOW
    03  RECS-REQ       PIC S9(8) COMP    # OF RECS REQUESTED
    03  RECS-CURR      PIC S9(8) COMP    # OF RECS CURR OF RU
    03  DB-CALLS       PIC S9(8) COMP    # OF DBMS CALLS
    03  FRAG-STORED    PIC S9(8) COMP    # OF FRAGMENTS STORED
    03  RECS-RELO      PIC S9(8) COMP    # OF RECS RELOCATED
    03  TOT-LOCKS      PIC S9(8) COMP    TOTAL # OF LOCKS ACQUIRED
    03  SHR-LOCKS      PIC S9(8) COMP    # OF SHARE LOCKS HELD
    03  NSH-LOCKS      PIC S9(8) COMP    # OF NON-SHARE LOCKS HELD
    03  LOCKS-FREED    PIC S9(8) COMP    # OF LOCKS FREED
    03  SR8-SPLITS     PIC S9(8) COMP    # OF SR8 SPLITS
    03  SR8-SPAWN      PIC S9(8) COMP    # OF SR8 SPAWNS
    03  SR8-STORE      PIC S9(8) COMP    # OF SR8S STORED
    03  SR8-ERASE      PIC S9(8) COMP    # OF SR8S ERASED
    03  SR7-STORE      PIC S9(8) COMP    # OF SR7S STORED
    03  SR7-ERASE      PIC S9(8) COMP    # OF SR7S ERASED
    03  BTREE-SRCH     PIC S9(8) COMP    # OF BTREE SEARCHES
    03  BTREE-LEVEL    PIC S9(8) COMP    # OF BTREE LEVELS SEARCHED
    03  ORPHANS        PIC S9(8) COMP    # OF ORPHANS ADAPTED
    03  BTREE-LEV-B    PIC S9(4) COMP    # OF LVLS SRCH'D (BEST CASE)
    03  BTREE-LEV-W    PIC S9(4) COMP    # OF LVLS SRCH'D (WORST CASE)
    03  RECS-UPD       PIC S9(8) COMP    # OF RECS UPDATED
```

```
03  PAGE-INCACHE   PIC S9(8) COMP   # OF PAGES FOUND IN CACHE
03  PAGE-INPREFET  PIC S9(8) COMP   # OF PAGES FOUND IN PREFETCH
03  RESERVED       PIC X(8)         RESERVED FOR FUTURE USE
03  SYS-INTERN3    PIC X(8)         SYSTEM INTERNAL USE ONLY
03  USER-ID        PIC X(32)        DC USER ID
03  LTERM-ID       PIC X(8)         LOGICAL TERMINAL ID
03  USER-SUPP-ID   PIC X(8)         USER-SUPPLIED ID
03  BIND-DATE      PIC S9(7) COMP-3 DATE BIND COMMAND ISSUED
03  BIND-TIME      PIC S9(8) COMP   TIME BIND COMMAND ISSUED
03  TRANSTAT-FLGS  PIC S9(8) COMP   FOUR 1-BYTE FLAGS
03  SYS-INTERN4    PIC X(8)         SYSTEM INTERNAL USE ONLY
03  SQL-COMMANDS   PIC S9(8) COMP   # OF SQL COMMANDS EXECUTED
03  SQL-FETCH      PIC S9(8) COMP   # OF ROWS FETCHED
03  SQL-INSERT     PIC S9(8) COMP   # OF ROWS INSERTED
03  SQL-UPDATE     PIC S9(8) COMP   # OF ROWS UPDATED
03  SQL-DELETE     PIC S9(8) COMP   # OF ROWS DELETED
03  SQL-SORTS      PIC S9(8) COMP   # OF SORTS PERFORMED
03  SQL-ROWSORT    PIC S9(8) COMP   # OF ROWS SORTED
03  SQL-MINRSORT   PIC S9(8) COMP   MINIMUM ROWS SORTED
03  SQL-MAXRSORT   PIC S9(8) COMP   MAXIMUM ROWS SORTED
03  SQL-AMCMPL     PIC S9(8) COMP   # OF AM RECOMPILES
03  SQL-RESERVED   PIC X(32)        RESERVED FOR FUTURE USE
```

5.10.3 Example

The following statement returns the contents of the TSB to STATISTICS-BLOCK and writes transaction statistics to the log file:

```
ACCEPT TRANSACTION STATISTICS
   WRITE
   INTO STATISTICS-BLOCK.
```

5.10.4 Status codes

After completion of the ACCEPT TRANSACTION STATISTICS function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3801	Storage for the transaction statistics block is not available; to wait would cause a deadlock
3813	No transaction statistics block exists; a BIND TRANSACTION STATISTICS request has not been issued
3831	Either the parameter list is invalid or no logical terminal element (LTE) is associated with the issuing task
3850	The collection of transaction statistics or task statistics has not been enabled during system generation

5.11 ATTACH

The ATTACH statement instructs CA-IDMS to initiate a new task by acquiring the necessary control blocks and storage and by adding the task to its dispatching list. CA-IDMS initializes the attached task and queues it up for execution; the issuing program receives control in accordance with normal dispatching priority.

5.11.1 Syntax

```
►►── ATTACH TASK CODE 'task-code' ┬──────────────────────┬─┬──────────┬─ . ──►◄
                                 └─ PRIORITY priority ─┘ ├─ WAIT ◄ ─┤
                                                         └─ NOWAIT ─┘
```

5.11.2 Parameters

task-code

Either the symbolic name of a user-defined field that contains the task code or the code itself enclosed in quotation marks.

The referenced task code must have been defined during system generation or dynamically by using the DCMT VARY DYNAMIC TASK command.

PRIORITY

Specifies the dispatching priority of the attached task.

priority

Either the symbolic name of a user-defined field that contains the dispatching priority or the priority itself expressed as a numeric constant in the range 000 through 240. By default, if you do not specify a priority or its location, CA-IDMS will use the priority established during system generation for the specified task code, terminal, and user.

WAIT

Specifies that the issuing task will wait until the maximum task condition no longer exists and CA-IDMS can attach the specified task.

This is the default.

NOWAIT

Specifies that the issuing task will not wait for CA-IDMS to attach the task. When NOWAIT is specified, the program should check the ERROR-STATUS field in the CA-IDMS communications block to determine if the ATTACH request has been completed. If the ERROR-STATUS value is 3711, indicating that a maximum task condition exists, the request has not been serviced and the program should perform alternative processing before reissuing the ATTACH request.

5.11.3 Example

The following example requests CA-IDMS to initiate task TASKATCH and assigns it a dispatching priority of 199:

```
ATTACH TASK CODE 'TASKATCH'
   PRIORITY 199
   NOWAIT.
```

5.11.4 Status codes

After completion of the ATTACH function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3711	The task cannot be attached because the maximum number of tasks has already been attached
3712	The specified task code is not known to the CA-IDMS system
3758	The task cannot be attached because the maximum number of concurrent tasks threads was exceeded

5.12 BIND MAP

The BIND MAP statement identifies the location of a map request block (MRB) and initializes MRB fields. For each MRB used by a program, code a BIND MAP statement; for each record defined to a map, code a BIND MAP RECORD statement.

BIND MAP statements can be global or record-specific, as follows:

- **Global** — The BIND MAP statement applies to the map as a whole. It initializes the entire MRB and fills in fields that apply to the map in general.
- **Record-specific** — The BIND MAP statement applies only to the named map record. It initializes the variable storage address of the named record in the MRB.

Typically, a program issues a global BIND MAP statement for each map, followed by BIND MAP statements for each map record used by the program.

You can request the precompiler to include global and record-specific BIND MAP statements automatically by using a COPY IDMS MAP-BINDS statement (see Chapter 4, “Precompiler-Directive Statements” on page 4-1). COPY IDMS MAP-BINDS includes the necessary BINDS for all maps and map records defined for the program.

The program can alter the storage address for a map record at any time by issuing another BIND MAP statement for that record. After the initial global bind (BIND MAP), all map records are considered unbound; map operations that use those records will have no effect on storage. After binding a map record to a storage address (BIND MAP RECORD), subsequent map operations will use that address to access the record. To unbind a map record, issue a record-specific BIND MAP statement that specifies the TO NULL option.

5.12.1 Syntax

```
►►─┬─────────────────────────────────────────────────────────────────┬─►◄
   ↓─ BIND MAP map-name ─┬──────────────────────────────────────────┬─ . ─┘
                         └─ RECORD rec-name ─┬──────────────────────┬┘
                                             └─ TO ─┬─ NULL ────────┬┘
                                                    └─ rec-location ─┘
```

5.12.2 Parameters

map-name

The name of an existing map. The map version defaults to the version specified for the map in the program's MAP SECTION.

RECORD

Initializes the variable storage address of the named record in the MRB.

record-name

The name of a record used by the map.

TO

Specifies whether the record is to be unbound or bound to a specified address.

NULL

Leaves the record unbound.

rec-location

The symbolic name of a user-defined field that contains the address to which the record is to be bound. *Record-location* defaults to *record-name*. Subsequent I/O operations will use this area of storage for any operation associated with the record.

5.12.3 Example

The following statements bind the map EMPMAPLR and its five associated map records:

```
BIND MAP EMPMAPLR.
BIND MAP EMPMAPLR RECORD EMPLOYEE.
BIND MAP EMPMAPLR RECORD DEPARTMENT.
BIND MAP EMPMAPLR RECORD JOB.
BIND MAP EMPMAPLR RECORD OFFICE.
BIND MAP EMPMAPLR RECORD EMP-DATE-WORK-REC.
```

5.12.4 Status codes

After completion of the BIND MAP function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully

5.13 BIND PROCEDURE

The BIND PROCEDURE statement establishes communication between a program and a DBA-written database procedure (for example, a security routine). You should use this statement only when the application program is required to pass to the procedure more information than is provided by the DBMS itself. Such instances are unusual; in most cases, you will not be aware of which procedures gain control before or after various DML functions.

The BIND PROCEDURE statement is used in both the navigational and the non-navigational environments.

5.13.1 Syntax

```
►►── BIND PROCEDURE FOR procedure-name TO procedure-control-location . ──►◄
```

5.13.2 Parameters

procedure-name
Specifies the database procedure in program variable storage to be made available to the program.

TO procedure-control-location
Specifies the 256-byte (fixed-length) location to which the named procedure will be bound.

A program that runs in a different partition from the central version may need to pass certain information to the database procedure. When the DBMS invokes the database procedure, this information is copied from the program storage area identified by *procedure-control-location* into the IDMS application program information block. The information passed is the information in *procedure-control-location* when the BIND PROCEDURE was performed; it is not the information in the program's storage at the time of the procedure call.

5.13.3 Example

The following statement binds the procedure with the variable name PROGCHEK to the 256-byte area PROC-CTL:

```
BIND PROCEDURE FOR PROGCHEK TO PROC-CTL.
```

5.13.4 Status codes

After completion of the BIND PROCEDURE function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.

Status code	Meaning
1400	The BIND PROCEDURE statement cannot be recognized. This code usually indicates that the IDMS communications block (SUBSCHEMA-CTRL) is not aligned on a fullword boundary.
1408	The named procedure is not in the specified subschema.
1418	The procedure has been bound improperly to location 0.
1472	The available memory is insufficient to dynamically load the database procedure.
1474	An attempt to load a module from the load/core-image library or DDLDCLOD has failed.

5.14 BIND RECORD

The BIND RECORD statement establishes addressability for a record in program variable storage. In most cases, you need not issue individual BIND RECORD statements since the necessary statements are generated as a group by the COPY IDMS SUBSCHEMA-BINDS statement (see Chapter 4, “Precompiler-Directive Statements” on page 4-1). However, you can issue BIND RECORD commands separately as necessary (for example, to bind several records to the same storage location). In any case, addressability must be established for each subschema record to be used by the program.

The program should perform the IDMS-STATUS routine after each BIND RECORD statement to ensure that the statement was executed successfully. When AUTOSTATUS is in use (see 3.5.1, “AUTOSTATUS protocols” on page 3-33), a PERFORM IDMS-STATUS operation occurs automatically after each BIND RECORD statement, even if the BIND RECORD statements are generated as a group by a COPY IDMS SUBSCHEMA-BINDS statement. You should use COPY IDMS SUBSCHEMA-BINDS only when AUTOSTATUS is in use.

5.14.1 Syntax

```
►►── BIND ─┬─ record-name ─┬──────────────────────┬─┬─ . ──────────────►◄
          │               └─ TO record-location ─┘ │
          │                                        │
          └─ record-location WITH record-name ─────┘
```

5.14.2 Parameters

record-name

Specifies the record to be bound to a location in variable storage.

The specified record must be included in the subschema.

TO record-location

The location to which the record is to be bound. The location corresponds to the record description as copied into the program manually or automatically through DATA DIVISION statements.

Record-location must be the same length as *record-name*.

Note: Exercise caution when using the TO *record-location* option because source-object mismapping can result from improper use. In cases where more than one copy of a given database record description is present in the program, you must ensure that the proper record description is bound at the proper time.

record-location WITH record-name

Binds a record name literal, specified by *record-name*, with a variable storage record description, specified by *record-location*. *Record-name* must specify a record included in the subschema.

5.14.3 Example

The following statement binds the EMPLOYEE record:

```
BIND EMPLOYEE.
```

5.14.4 Status codes

After completion of the BIND RECORD function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
1400	The BIND RECORD statement cannot be recognized. This code usually indicates that the IDMS communications block (SUBSCHEMA-CTRL) is not aligned on a fullword boundary.
1408	The named record is not in the subschema. The program has probably invoked the wrong subschema.
1418	The record has been bound improperly to location 0.
1472	The available memory is insufficient to dynamically load a database procedure.
1474	An attempt to load a module from the load/core-image library or DDLDCLOD has failed.

5.15 BIND RUN-UNIT

The BIND RUN-UNIT statement establishes a run unit for accessing the database, identifies the location of the IDMS communications block being used, and names the subschema to be loaded for the run unit. BIND RUN-UNIT can also name the node under which the run unit will execute and identify the database to be accessed. BIND RUN-UNIT must be the first functional DML call passed to the DBMS at execution time; it must logically precede all other DML statements (for example, BIND RECORD, READY, FIND) in the program's PROCEDURE DIVISION.

UTM modes only: You must move LOW VALUES to SUBSCHEMA-CTRL before issuing the BIND RUN-UNIT statement.

When AUTOSTATUS is in use, COPY IDMS SUBSCHEMA BINDS can be used to automatically invoke the BIND RUN-UNIT statement and the appropriate BIND RECORD statements (see Chapter 4, “Precompiler-Directive Statements” on page 4-1).

If program registration is in effect (that is, all programs must be registered in the data dictionary before compilation), the program must initialize the PROGRAM-NAME field of the IDMS communications block either automatically or manually:

- **Automatically** — A COBOL MOVE statement automatically generated by COPY IDMS SUBSCHEMA-BINDS moves the program name (stated in the IDENTIFICATION DIVISION) to the PROGRAM-NAME field.
- **Manually** — A COBOL MOVE statement is coded by the programmer before the BIND RUN-UNIT statement is executed. For example:

  ```
  MOVE 'EMPDISP' TO PROGRAM-NAME.
  ```

The BIND RUN-UNIT statement is used in both the navigational and the non-navigational environments.

5.15.1 Syntax

```
►►── BIND RUN-UNIT ──┬──────────────────────────────────────────►
                     └─ FOR subschema-name ─┘

►──┬──────────────────┬──┬──────────────────────────────────────►
   └─ DBNODE nodename ─┘  └─ DBNAME database-name ─┘

►──┬──────────────────┬──┬──────────────────────────────────┬─ . ──►◄
   └─ DICTNODE nodename ─┘  └─ DICTNAME dictionary-name ─┘
```

5.15.2 Parameters

FOR subschema-name

Identifies a subschema view other than that specified in the DB clause of the SCHEMA SECTION. It must be the symbolic name of a user-defined eight-character field in variable storage.

By default, if you do not specify a subschema, the run unit uses the subschema named in the DB clause of the SCHEMA SECTION.

Note: Exercise care when using the FOR *subschema-name* option; improper use can lead to mismappings between the named subschema and record descriptions in variable storage.

DBNODE
Specifies the node where the database resides.

nodename
Either the symbolic name of a user-defined eight-character field in variable storage or the database name itself enclosed in quotation marks.

DBNAME
Specifies the database to be accessed by the run unit.

database-name
Either the symbolic name of a user-defined eight-character field in variable storage or the database name itself enclosed in quotation marks.

DICTNODE
Specifies the node that controls the dictionary where the subschema resides.

nodename
Either the symbolic name of a user-defined eight-character field in variable storage or the node name itself enclosed in quotation marks.

DICTNAME
Specifies the dictionary where the subschema resides.

dictionary-name
Either the symbolic name of a user-defined eight-character field in variable storage or the dictionary name itself enclosed in quotation marks.

Note: The DBNODE, DBNAME, DICTNODE, and DICTNAME parameters can be overridden at runtime by the DCUF SET DBNODE/DBNAME and DCUF SET DICTNODE/DICTNAME commands.

Batch users: The DBNODE AND DBNAME parameters can be overridden at runtime if the IDMSOPTI module or the SYSCTL clause in the system generation SYSTEM statement specifies a nodename or a dbname with the ALWAYS option. For more information on the use of DBNODE, DBNAME, DICTNODE, and DICTNAME, see *CA-IDMS System Generation*.

5.15.3 Example

The following statement binds the run unit to the DBMS:

```
BIND RUN-UNIT.
```

5.15.4 Status codes

After completion of the BIND RUN-UNIT function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
1400	The BIND RUN-UNIT statement cannot be recognized. This code usually indicates that the IDMS communications block (SUBSCHEMA-CTRL) is not aligned on a fullword boundary.
1410	Security violation; an existing access restriction or subschema usage prohibits execution of the specified DML function. For LRF users, the subschema in use allows access to database records only. Combined with a major code of 00, this code means the program has attempted to access a database record, but the subschema in use allows access to logical records only.
1467	Invalid subschema load module; the subschema invoked does not match the subschema object tables. subschema object tables.
1469	The run unit is not bound to the DBMS. This code indicates that the central version is not active, that the central version is not accepting new run units, or that the run unit's connection to the central version is broken due to timeout or other factors, as noted on the CV log.
1470	A journal file will not open (local mode only); under OS, the most probable cause is that a DD statement for the journal file is missing in the JCL.
1472	There is insufficient memory to dynamically load a subschema or database procedure.
1473	The central version is not accepting new run units.
1474	The subschema was not found in the dictionary load area or in the load library.
1477	The run unit has been bound previously.
1480	The node specified is not active or has been disabled.
1481	The converted subschema requires specified database name to be in the DBNAME table.
1482	The subschema must be named in the DBNAME table.
1483	The available memory is insufficient to allocate native VSAM work areas.
1491	The subschema requires a DBNAME to do the bind run unit.
1492	No subschema areas map to DMCL.

Status code	Meaning
1493	A subschema area symbolic was not found in DMCL.
1494	The specified dbname is neither a dbname defined in the DBNAME table, nor a SEGMENT defined in the DMCL.
1495	The specified subschema failed DBTABLE mapping using the specified dbname.

5.16 BIND TASK

The BIND TASK statement initiates a CA-IDMS task when the operating mode is DC-BATCH. This statement establishes communication with the DC system and, if accessing CA-IDMS queues, allocates a packet-data movement buffer to contain the queue data. Once a task is started, the program can issue any number of consecutive BIND-READY-FINISH sequences.

Note: Do not issue this command in local mode.

5.16.1 Syntax

```
►►── BIND TASK ──────────────────────────────────────────►
►┬──────────────────────────┬─ . ──────────────────────►◄
 └─ NODENAME (nodename) ────┘
```

5.16.2 Parameters

NODENAME

Specifies the node to which the task will be bound.

nodename

Either the symbolic name of a user-defined field that contains the nodename or the nodename itself enclosed in quotation marks. The specified node name must match the node named in the DDS statement at system generation.

5.16.3 Example

The following statement establishes communication with a DC system.

```
BIND TASK.
```

5.16.4 Status codes

After completion of the BIND TASK function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.

5.17 BIND TRANSACTION STATISTICS

The BIND TRANSACTION STATISTICS statement defines the beginning of a transaction for the purposes of collecting transaction statistics. CA-IDMS allocates a block of storage in which to accumulate these statistics. Because this block is owned by the logical terminal associated with the current task, the BIND TRANSACTION STATISTICS statement cannot be used with nonterminal tasks.

Note: If a transaction statistics block (TSB) is already allocated for the logical terminal associated with the current task, the BIND request clears the block and writes any previously accumulated transaction statistics to the log file.

When a BIND TRANSACTION STATISTICS request is issued, CA-IDMS assigns the transaction a 40-character identifier; the first 32 characters are the identifier of the signed-on user (if any) and the last eight characters are the identifier of the logical terminal associated with the current task.

5.17.1 Syntax

```
►►── BIND TRANSACTION STATISTICS . ──────────────────────────────►◄
```

5.17.2 Example

The following example illustrates the BIND TRANSACTION STATISTICS statement:

```
BIND TRANSACTION STATISTICS.
```

5.17.3 Status codes

After completion of the BIND TRANSACTION STATISTICS function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully; any existing transaction statistics block was written to the log file before being cleared
3801	Storage for the transaction statistics block is not available; to wait would cause a deadlock
3810	A new transaction statistics block has been allocated
3831	Either the parameter list is invalid or no logical terminal element (LTE) is associated with the issuing task
3850	The collection of transaction statistics or task statistics has not been enabled during system generation

5.18 CHANGE PRIORITY

The CHANGE PRIORITY statement changes the dispatching priority of the issuing task. The new dispatching priority applies only to the current execution of the task. CHANGE PRIORITY does not relinquish control to another task and cannot be used to alter the priority of other tasks.

5.18.1 Syntax

```
►►── CHANGE PRIORITY to priority . ──────────────────────────────►◄
```

5.18.2 Parameters

priority

The new dispatching priority for the issuing task.

Either the symbolic name of a user-defined field that contains the priority value or the value itself expressed as a numeric constant in the range 0 through 240.

5.18.3 Example

The following example changes the dispatching priority of the issuing task to the value contained in the PRIORITY-210 field:

```
CHANGE PRIORITY TO PRIORITY-210.
```

5.18.4 Status codes

After completion of the CHANGE PRIORITY function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully

5.19 CHECK TERMINAL

The CHECK TERMINAL statement delays task processing until a previously issued I/O request has completed.

If a READ TERMINAL, WRITE TERMINAL, or WRITE THEN READ TERMINAL request specifies the NOWAIT option, the program must issue a CHECK TERMINAL request before specifying any other I/O operation. If the I/O operation is not complete, CA-IDMS suspends task execution. When the I/O operation is complete, the task resumes execution according to its established dispatching priority.

5.19.1 Syntax

```
►►── CHECK TERMINAL ┬──────────────┬──────────────────────────────────►
                   └─ GET STORAGE ─┘

►── INTO input-data-location ─┬─ TO end-input-data-location ──────────►
                              └─ MAX LENGTH input-data-max-length ────┘

►┬──────────────────────────────────────────────┬─ . ──────────────►◄
 └─ RETURN LENGTH into input-data-actual-length ─┘
```

5.19.2 Parameters

GET STORAGE

Asynchronous requests only. Acquires an input buffer for the data being read into the program; CA-IDMS allocates the required storage when the read operation is complete.

INTO

Specifies the 01-level WORKING-STORAGE SECTION or LINKAGE SECTION data area reserved for the input data stream.

input-data-location

The symbolic name of a user-defined field.

If GET STORAGE is specified, the data area reserved for the input data stream must be an unallocated 01-level LINKAGE SECTION entry. If GET STORAGE is not specified, the data area must be a previously allocated WORKING-STORAGE SECTION or LINKAGE SECTION entry.

TO

Indicates the end of the data area reserved for the input.

end-input-data-location

Either the symbolic name of a user-defined dummy byte field or a field that contains a data item not associated with the data area reserved for the input data stream.

MAX LENGTH

Defines the length, in bytes, of the data area reserved for the input data stream.

input-data-max-length

Either the symbolic name of a user-defined field that contains the length of the data area or the length itself expressed as a numeric constant.

If the input data stream is larger than the data area reserved in the WORKING-STORAGE SECTION or LINKAGE SECTION, CA-IDMS truncates the data stream as needed to fit the available space.

RETURN LENGTH INTO

Indicates the location to which CA-IDMS will return the actual length of the input data stream.

input-data-actual-length

The symbolic name of a user-defined field. If the data stream has been truncated, *input-data-actual-length* will contain the original length before truncation.

5.19.3 Example

The following statement determines whether an I/O operation is complete, acquires an input buffer, and reads 72 bytes of data into TERM-LINE:

```
CHECK TERMINAL
   GET STORAGE
   INTO TERM-LINE MAX LENGTH 72.
```

5.19.4 Status codes

After completion of the CHECK TERMINAL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4519	The input area specified for the return of data is too small; the returned data has been truncated to fit the available space.
4525	The output operation has been interrupted; the terminal operator has pressed ATTENTION or BREAK.
4526	A logical error (for example, an invalid control character) has been encountered in the output data stream.
4527	A permanent I/O error has occurred during processing.
4528	The dial-up line for the terminal being used has been disconnected.
4531	The terminal request block (TRB) contains an invalid field, indicating a possible error in the program's parameters.
4535	Storage for the input buffer cannot be acquired because the specified 01-level LINKAGE SECTION entry has been allocated.

Status code	Meaning
4537	Storage for the input buffer cannot be acquired because the specified data area is defined in the WORKING-STORAGE SECTION rather than in the LINKAGE SECTION.
4538	The specified 01-level LINKAGE SECTION entry has not been allocated and the GET STORAGE option has not been specified. No I/O has been performed.
4539	The terminal device associated with the issuing task is out of service.

5.20 COMMIT

The COMMIT statement requests CA-IDMS to write a checkpoint to the journal file to designate the start or end of specific database, scratch, or queue record access activities associated with the issuing run unit or task. COMMIT simulates a FINISH-BIND-READY sequence without relinquishing control of database resources.

The COMMIT statement is used in both the navigational and logical record facility environments.

Currency: Use of the ALL option, as in COMMIT ALL, sets all currencies to null.

5.20.1 Syntax

```
►►── COMMIT ─┬──────┬─┬─────┬─ . ──────────────────────────────►◄
          └ TASK ┘ └ ALL ┘
```

5.20.2 Parameters

TASK
Establishes checkpoints for all data areas associated with all run units, scratch activities, queue activities, and print activities initiated by the issuing task.

ALL
Releases all locks held on records in database, scratch, and queue areas associated with the issuing task (COMMIT TASK ALL) or run unit (COMMIT ALL) and sets all currencies to null.

5.20.3 Example

The following statement writes a commit checkpoint to the journal file:

```
COMMIT.
```

5.20.4 Status codes

After completion of the COMMIT function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
5031	The specified request is invalid; the program may contain a logic error
5096	Too many run units are currently active; check the DC system log for details

Status code	Meaning
5097	An invalid status has been received from DBIO/DBMS; check the DC system log for details

5.21 CONNECT

The CONNECT statement establishes a record occurrence as a member of a set occurrence. The specified record must be defined as an optional automatic, optional manual, or mandatory manual member of the set.

Native VSAM users: The CONNECT statement is not valid since all sets in native VSAM data sets must be defined as mandatory automatic.

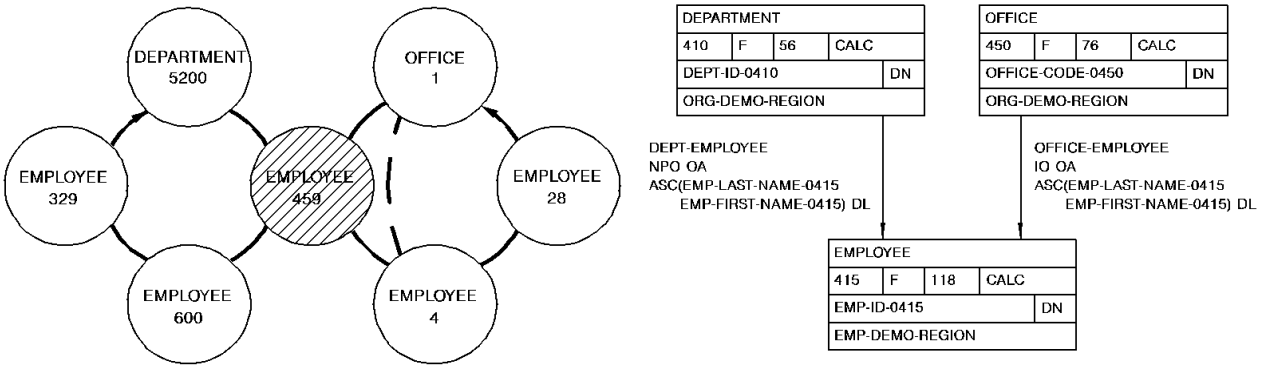
Before execution of the CONNECT statement, the following conditions must be satisfied:

- All areas affected either explicitly or implicitly by the CONNECT statement must be readied in one of the update usage modes (see 5.66, “READY” on page 5-201 later in this chapter).
- The specified record must be established as current of its record type.
- The occurrence of the set into which the specified record will be connected must be established. The current record of set determines the set occurrence and, if set order is NEXT or PRIOR, the position at which the specified record will be connected within the set.

Currency: Following successful execution of a CONNECT statement, the specified record is current of run unit, its record type, its area, and all sets in which it currently participates.

Connecting a record to a set: The following figure illustrates the steps required to connect an EMPLOYEE record to an occurrence of the OFFICE-EMPLOYEE set.

To connect EMPLOYEE 459 to OFFICE 1 in the OFFICE-EMPLOYEE set, establish EMPLOYEE 459 as current of record type, locate the proper occurrence of the OFFICE record, and issue the CONNECT command.



	CURRENCIES RUN UNIT, RECORD, SET, AREA							
	RUN UNIT	DEPARTMENT	EMPLOYEE	OFFICE	DEPT-EMPLOYEE	OFFICE-EMPLOYEE	ORG-DEMO-REGION	EMP-DEMO-REGION
MOVE 5200 TO DEPT-ID. FIND CALC DEPARTMENT.	5200	5200			5200		5200	
OBTAIN FIRST EMPLOYEE WITHIN DEPT-EMPLOYEE.	459	5200	459		459		5200	459
MOVE 1 TO OFFICE-CODE. FIND CALC OFFICE.	1	5200	459	1	459	1	1	459
CONNECT EMPLOYEE TO OFFICE-EMPLOYEE.	459	5200	459	1	459	459	1	459

5.21.1 Syntax

```
►►── CONNECT record-name TO set-name . ──────────────────────────────►◄
```

5.21.2 Parameters

record-name

Specifies the record whose current occurrence is to be connected to the current occurrence of the specified set. *Record-name* must be a record included in the subschema and must be defined as an optional automatic, optional manual, or mandatory manual member of the set to which it is being connected.

TO

Specifies the set to which the member record is to be connected.

set-name

The name of a set included in the subschema. The record is connected to the set in accordance with the ordering rules defined for that set in the schema.

5.21.3 Example

The following statement connects the current EMPLOYEE record to the current occurrence of the OFFICE-EMPLOYEE set:

```
CONNECT EMPLOYEE TO OFFICE-EMPLOYEE.
```

5.21.4 Status codes

After completion of the CONNECT function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0705	The CONNECT would violate a duplicates-not-allowed option.
0706	Currency has not been established for the named record or set.
0708	The named record is not in the subschema. The program has probably invoked the wrong subschema.
0709	The named record's area has not been readied in one of the update usage modes.
0710	The subschema specifies an access restriction that prohibits connecting the named record in the named set.
0714	The CONNECT statement cannot be executed because the named record has been defined as a mandatory automatic member of the set.
0716	The record cannot be connected to a set in which it is already a member.
0721	An area other than the area of the named record has been readied with an incorrect usage mode.
0725	Currency has not been established for the named set type.

5.22 DC RETURN

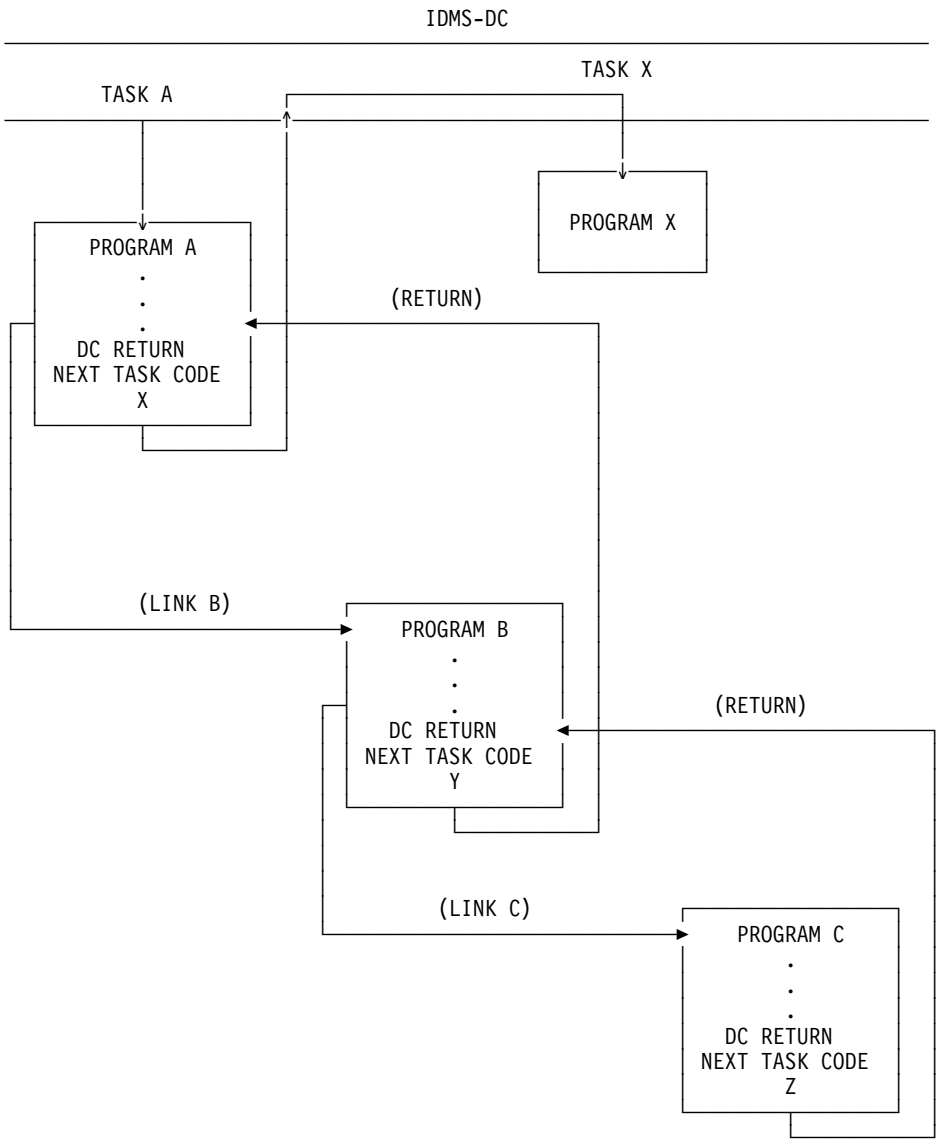
The DC RETURN statement returns control to a program at the next higher level within a task. Additionally, you can use the DC RETURN statement to specify:

- The next task to be initiated on the same terminal
- Recovery procedures for abend routines established by SET ABEND EXIT functions
- The action to be taken by CA-IDMS if the user fails to initiate the next task

Following a DC RETURN request, control returns to the program at the next higher level within the task. If the issuing program is the highest level program, control returns to CA-IDMS. Any DC RETURN statement can include a NEXT TASK CODE option to specify the next task to be initiated by CA-IDMS. However, the position of the issuing program within the task governs whether the specified task will, in fact, receive control.

DC RETURN processing: The following figure illustrates how CA-IDMS executes a task when DC RETURN statements within three programs specify the NEXT TASK CODE option.

Task A invokes program A. Program A links to program B, which in turn links to program C. Program C issues a DC RETURN NEXT TASK CODE 'Z' request; control returns to program B. Program B contains a DC RETURN NEXT TASK CODE 'Y' request, which takes precedence over program C's DC RETURN specification. Control returns to program A, which issues a DC RETURN NEXT TASK CODE 'X' request. Because program A is at the highest level in the task, task X will be invoked.



When CA-IDMS receives control from the highest level program that issued a DC RETURN NEXT TASK CODE request, the specified task is executed immediately if the specified task code has been assigned the NOINPUT attribute during system generation; if the task code was assigned the INPUT attribute, the task executes only when the user presses an attention identifier (AID) key. Typical AID keys include all PA and PF keys, ENTER, and CLEAR.

5.22.1 Syntax

```
►►── DC RETURN ──────────────────────────────────────────────────────►
►──┬──────────────────────────────────────────────────────────────────►
  └─ NEXT TASK CODE next-task-code ─┘
►──┬──────────────────────────────────────────────────────────────────►
  ├─ NORMAL ◄ ──┤
  ├─ ABORT ─────┤
  ├─ CONTINUE ──┤
  └─ IMMEDIATE ─┘
►──┬──────────────────────────────────────────────────────────────────►
  │                ┌──────────────────────────────┐
  └─ TIMEOUT ─────↓──┬─ INTERVAL timeout-interval ─┬──┘
                     └─ PROGRAM timeout-program ───┘
►──┬──────────────────────────────────────────────────────────────────►─
  └─ NEXT TASK INTERVAL start-interval EVENT TYPE ──┬─ INTERNAL ──────
                                                    └─ EXTERNAL ─┘
►──┬──────────────────────────────────────┬──────────────────────────►◄
   │                                      │
   ├─ EVENT ecb ──────────────────────────┤
   └─ EVENT NAME ecb-id ──────────────────┘
```

5.22.2 Parameters

NEXT TASK CODE

Specifies the next task to be initiated on the same terminal.

next-task-code

Either the symbolic name of a user-defined field that contains the task code or the task code itself enclosed in quotation marks. The task code must be defined to the DC system under which it is running, either during system generation or at runtime.

NORMAL/ABORT/CONTINUE

Defines the recovery action to be taken within the program logic (CA-IDMS recovery occurs automatically) and specifies whether CA-IDMS is to execute abend routines for higher-level programs. These options apply to DC RETURNs issued from abend routines established by SET ABEND EXIT (STAE) functions only.

NORMAL

Specifies that will not attempt recovery and will execute all abend routines established for programs at higher task levels.

This is the default.

ABORT

Specifies that CA-IDMS will not attempt recovery and will abort the task immediately without executing any abend routines established for programs at higher task levels.

CONTINUE

Specifies that CA-IDMS will return control to the program that failed at an address established in the abend control element (ACE) for the program.

IMMEDIATE

Is ignored when issued from ABEND routine; it is only applied when NOT issued from an ABEND routine.

TIMEOUT

Specifies the action CA-IDMS is to take if the user fails to enter data required to initiate a task. This parameter overrides resource timeout interval and program specifications established during system generation.

INTERVAL

Specifies the time, in seconds, that can elapse before CA-IDMS releases the resources held by the terminal on which the task is executing.

timeout-interval

Either the symbolic name of a user-defined PIC S9(4) COMP SYNC (halfword) field that contains the timeout interval or the interval itself expressed as a numeric constant.

PROGRAM

Specifies the program to be invoked to handle and release resources held by the terminal on which the task is executing when the specified timeout interval has been reached.

timeout-program

Either the symbolic name of a user-defined field that contains the program name or the name itself enclosed in quotation marks.

The specified program must be defined to the DC system either during system generation or at runtime.

NEXT TASK INTERVAL start-interval

Either the symbolic name of a user-defined PIC S9(4) COMP SYNC (halfword) field that contains the start interval or the interval itself expressed as a numeric constant.

Note: When specified alone, NEXT TASK INTERVAL will cause task to be initiated after *start-interval*. When specified along with EVENT/EVENT NAME, task will be initiated either after *start-interval* or posting of the EVENT(S)/EVENT NAME(S), whichever occurs first.

EVENT TYPE INTERNAL/EXTERNAL

Specifies events that happen either internal or external to the system.

INTERNAL

An event that occurs within IDMS-DC, such as waiting for space in a storage pool, or waiting for a completed task.

EXTERNAL

An event that occurs outside the system's control, such as waiting for a file to be read, or waiting for an I/O to complete.

EVENT

Defines one or more ECBs upon which the task will wait.

ecb

The symbolic name of a user-defined area that contains three PIC S9(8) COMP SYNC (fullword) fields. Multiple EVENT parameters must be separated by at least one blank.

EVENT NAME

Specifies the ECB upon which the task will wait.

Note: When specified alone, NEXT TASK INTERVAL will cause task to be initiated after *start-interval*. When specified along with EVENT/EVENT NAME, task will be initiated either after *start-interval* or posting of the EVENT(S)/EVENT NAME(S), whichever occurs first.

ecb-id

Either the symbolic name of a user-defined field that contains the ECB ID or the ID itself enclosed in quotation marks.

5.22.3 Example

The following statement illustrates the use of DC RETURN. The task code associated with MENU-TASK-CODE, if defined with the INPUT parameter, will be invoked when the user next presses an AID key; if MENU-TASK-CODE is defined with the NOINPUT parameter, it will be invoked immediately.

```
DC RETURN
   NEXT TASK CODE MENU-TASK-CODE.
```

5.22.4 Status codes

Because control is returned to the next-higher level, there is no need to check the ERROR-STATUS field.

5.23 DELETE QUEUE

The DELETE QUEUE statement deletes all or part of a queue. If only one queue record is deleted, CA-IDMS maintains currency within the queue by saving the next and prior currencies of the deleted record.

5.23.1 Syntax

```
►►── DELETE QUEUE ─┬──────────────────┬─┬──────────────────┬─ . ──────────►◄
                   └─ ID queue-id ────┘ │┌─ CURRENT ◄ ────┐│
                                        │└─ ALL ──────────┘│
```

5.23.2 Parameters

ID

Specifies the queue that contains the record to be deleted.

queue-id

Either the symbolic name of a user-defined field that contains the ID or the ID itself enclosed in quotation marks. If you do not specify an ID, a blank ID is assumed.

CURRENT

Deletes the current record of the queue associated with the requesting task.

This is the default.

ALL

Deletes all records in the queue and the queue header ID.

5.23.3 Example

The following example requests CA-IDMS to delete the current record in the RES-Q queue:

```
DELETE QUEUE
   ID 'RES-Q'
   CURRENT.
```

5.23.4 Status codes

After completion of the DELETE QUEUE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4404	The requested queue header record cannot be found.
4405	The requested queue record cannot be found.

Status code	Meaning
4406	No resource control element (RCE) exists for the queue record, indicating that currency has not been established.
4407	An I/O error has occurred during processing.
4431	The parameter list is invalid.

5.24 DELETE SCRATCH

The DELETE SCRATCH statement deletes one scratch record or all records in the scratch area.

5.24.1 Syntax

```
►►── DELETE SCRATCH ──────────────────────────────────────────►
►┬──────────────────────────────┬──────────────────────────────►
 └─ AREA ID scratch-area-id ────┘
►┬──────────────────────────────┬──────────────────────────────►
 ├─ CURRENT ◄ ──────────────────┤
 ├─ FIRST ──────────────────────┤
 ├─ LAST ───────────────────────┤
 ├─ NEXT ───────────────────────┤
 ├─ PRIOR ──────────────────────┤
 ├─ ALL ────────────────────────┤
 └─ RECORD ID scratch-record-id ┘
►┬──────────────────────────────────────────┬─ . ──────────►◄
 └─ RETURN RECORD ID into return-scratch-record ┘
```

5.24.2 Parameters

AREA ID

Specifies the scratch area associated with the scratch records being deleted.

scratch-area-id

Either the symbolic name of a user-defined field that contains the scratch area ID or the ID itself enclosed in quotation marks. If you do not specify an AREA ID, CA-IDMS assumes an area ID of eight blanks.

CURRENT

Deletes the current record in the scratch area (that is, that record most recently referenced by another scratch function).

This is the default.

FIRST

Deletes the first record in the specified scratch area.

LAST

Deletes the last record in the specified scratch area.

NEXT

Deletes the next record in the specified scratch area.

PRIOR

Deletes the prior record in the specified scratch area.

ALL

Deletes all records in the specified scratch area.

RECORD ID

Deletes the identified record.

scratch-record-id

The symbolic name of a user-defined field that contains the ID.

RETURN RECORD ID into

Specifies the location in the program to which CA-IDMS will return the ID of the last record deleted by means of the DELETE SCRATCH function.

return-scratch-record-id

The symbolic name of a user-defined four-byte field.

5.24.3 Example

The following example requests CA-IDMS to delete the scratch record that is prior to the current scratch record and return the ID of the deleted record to the SCR-REC-ID field:

```
DELETE SCRATCH
   PRIOR
   RETURN RECORD ID INTO SCR-REC-ID.
```

5.24.4 Status codes

After completion of the DELETE SCRATCH function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4303	The requested scratch area ID cannot be found
4305	The requested scratch record ID cannot be found
4307	An I/O error has occurred during processing
4331	The parameter list is invalid

5.25 DELETE TABLE

The DELETE TABLE statement notifies CA-IDMS that the issuing task has finished using a table that has been loaded into the program pool by using the LOAD TABLE function. DELETE TABLE does not physically delete reusable tables from the program pool; rather, it decrements the in-use count maintained by CA-IDMS. An in-use count of 0 signals CA-IDMS that the space occupied by the table can be reused.

5.25.1 Syntax

```
►►── DELETE TABLE from 01-level-program-location . ──────────────────────►
►─┬─────────────────────┬─┬─────────────────────────┬──────────────────────►
  └─ DICTNODE nodename ─┘ └─ DICTNAME dictionary-name ─┘
►─┬─────────────────────┬─ . ──────────────────────────────────────────►◄
  └─ LOADLIB library-name ─┘
```

5.25.2 Parameters

01-level-program-location
The LINKAGE SECTION entry of the 01-level record area specified in the associated LOAD TABLE request.

DICTNODE
Specifies the node that controls the dictionary where the subschema containing the table resides.

nodename
The symbolic name of a user-defined eight-character field in variable storage.

DICTNAME
Specifies the dictionary where the subschema containing the table resides.

dictionary-name
The symbolic name of a user-defined eight-character field in variable storage.

LOADLIB
Specifies the load library containing the table.

library-name
The symbolic name of a user-defined eight-character field in variable storage.

5.25.3 Example

The following example releases a previously loaded table from the location in variable storage identified by RATE-TABLE:

```
DELETE TABLE FROM RATE-TABLE.
```

5.25.4 Status codes

After completion of the DELETE TABLE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3433	The specified table was not loaded by the task

5.26 DEQUEUE

The DEQUEUE statement releases resources acquired by the issuing task with an ENQUEUE request. Acquired resources not released explicitly with a DEQUEUE request are released automatically at task termination.

5.26.1 Syntax

```
►►── DEQUEUE ─┬─ ALL ──────────────────────────────────────┬─ . ──────►◄
               └─┬─────────────────────────────────────────┬─┘
                 └─↓─ NAME resource-id LENGTH resource-id-length ─┘
```

5.26.2 Parameters

ALL
Releases all resources acquired by the issuing task by means of ENQUEUE requests.

NAME
Specifies a resource to be dequeued.

Multiple resource specifications must be separated by at least one blank.

resource-id
The symbolic name of a user-defined field that contains the resource ID.

LENGTH
Specifies the length of the resource.

resource-id-length
Either the symbolic name of a PIC S9(8) COMP SYNC (fullword) field that contains the length of the resource ID or the length itself expressed as a numeric constant.

5.26.3 Example

The following statement requests CA-IDMS to release all the resources enqueued by the issuing task:

```
DEQUEUE PAYROLL-LOCK
        LENGTH 16.
```

5.26.4 Status codes

After completion of the DEQUEUE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully

Status code	Meaning
3913	At least one resource ID cannot be found; all resources that were located have been dequeued

5.27 DISCONNECT

The DISCONNECT statement cancels the current membership of a record occurrence in a set occurrence. The named record must be defined as an *optional* member of the named set.

Native VSAM users: The DISCONNECT statement is not valid since all sets in native VSAM data sets must be defined as mandatory automatic.

Before execution of the DISCONNECT statement, the following conditions must be satisfied:

- All areas affected either explicitly or implicitly by the DISCONNECT statement must be readied with one of the three update usage modes (see 5.66, "READY" on page 5-201 later in this chapter).
- The named record must be established as current of its record type.
- The named record must currently participate as a member in an occurrence of the named set.

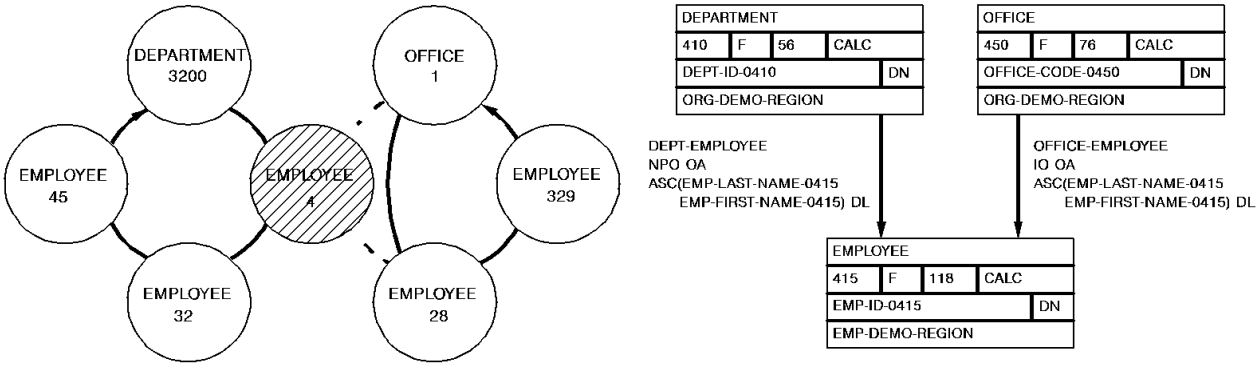
Following successful execution of the DISCONNECT statement, the named record can no longer be accessed through the set for which membership was cancelled. The disconnected record can still be accessed either by means of a complete scan of the area in which it participates or directly through its db-key, if known. A disconnected record can also be accessed either through any other sets in which it participates as a member or if it has a location mode of CALC.

Currency: A successfully executed DISCONNECT statement nullifies currency in the specified set. However, next, prior, and owner of set are maintained, enabling continued access within the set. The disconnected record is current of run unit, its record type, its area, and any other sets in which it participates.

5.28 Disconnecting a record from a set

The following figure illustrates the steps required to disconnect an EMPLOYEE record from an occurrence of the OFFICE-EMPLOYEE set.

To disconnect EMPLOYEE 4 from the OFFICE 1 of the OFFICE-EMPLOYEE set, enter the database on OFFICE, establish EMPLOYEE 4 as current of the EMPLOYEE record type, and disconnect it from the OFFICE-EMPLOYEE set.



	CURRENCIES RUN UNIT, RECORD, SET, AREA							
	RUN UNIT	DEPARTMENT	EMPLOYEE	OFFICE	DEPT-EMPLOYEE	OFFICE-EMPLOYEE	ORG-DEMO-REGION	EMP-DEMO-REGION
MOVE 1 TO OFFICE-CODE. FIND CALC OFFICE.	1			1		1	1	
FIND FIRST EMPLOYEE WITHIN OFFICE-EMPLOYEE.	4		4	1	4	4	1	4
DISCONNECT EMPLOYEE. FROM OFFICE-EMPLOYEE.	4		4	1	4	NPO	1	4

5.28.1 Syntax

```
►►── DISCONNECT record-name FROM set-name . ──────────────────────────►◄
```

5.28.2 Parameters

record-name
Specifies the record to disconnect from the named set. *Record-name* must be a record included in the subschema and must be defined as an optional member of the specified set.

FROM
Specifies the set from which the named record will be disconnected.

set-name
The name of a set included in the subschema.

5.28.3 Example

The following statement disconnects the current EMPLOYEE record from the OFFICE-EMPLOYEE set:

```
DISCONNECT EMPLOYEE FROM OFFICE-EMPLOYEE.
```

5.28.4 Status codes

After completion of the DISCONNECT function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
1106	Currency has not been established for the named record
1108	The named record is not in the subschema. The program has probably invoked the wrong subschema
1109	The named record's area has not been readied in one of the update usage modes
1110	The subschema specifies an access restriction that prohibits use of the DISCONNECT statement
1115	The DISCONNECT statement cannot be executed because the named record has been defined as a mandatory member of the set
1121	An area other than the area that contains the named record has been readied with an incorrect usage mode
1122	The named record is not currently a member of the specified set

5.29 END LINE TERMINAL SESSION

The END LINE TERMINAL SESSION statement terminates the current line I/O session. All output data lines that remain in the current buffer and all pages queued for asynchronous I/O operations are deleted.

5.29.1 Syntax

```
►►── END LINE TERMINAL session . ──────────────────────────────────────►◄
```

5.29.2 Example

The following statement terminates a line mode I/O session:

```
END LINE TERMINAL SESSION.
```

5.29.3 Status codes

There are no status codes associated with the END LINE TERMINAL SESSION command.

5.30 END TRANSACTION STATISTICS

The END TRANSACTION STATISTICS statement defines the end of a transaction. The transaction typically ends when the issuing task terminates. Optionally, END TRANSACTION STATISTICS can be used to write the transaction statistics block (TSB) to the DC system log file and to return the TSB to a preallocated location in variable storage. CA-IDMS returns a copy of the TSB to program variable storage.

5.30.1 Syntax

```
►►── END TRANSACTION STATISTICS ──────────────────────────────────►
►─┬─ WRITE ◄ ─┬─┬──────────────────────────────────────┬─ . ──────►◄
  │           │ │└─ INTO return-stat-data-location ─┘│
  └─ NOWRITE ─┘ │                                      │
                │                                      │
```

5.30.2 Parameters

WRITE

Specifies that the TSB will be written to the DC system log file when the task terminates.

This is the default.

NOWRITE

Specifies that the TSB will not be written to the DC system log file when the task terminates.

INTO

Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION data area into which CA-IDMS is to return the TSB.

return-stat-data-location

The symbolic name of a user-defined field.

5.30.3 Example

The following statement requests CA-IDMS to end a transaction, write statistics to the log file, and return a copy of the TSB to the STATISTICS-BLOCK field:

```
END TRANSACTION STATISTICS
   WRITE
   INTO STATISTICS-BLOCK.
```

5.30.4 Status codes

After completion of the END TRANSACTION STATISTICS function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
3801	Storage for the transaction statistics block is not available; to wait would cause a deadlock.
3813	No transaction statistics block exists; a BIND TRANSACTION STATISTICS request has not been issued.
3831	Either the parameter list is invalid or no logical terminal element (LTE) is associated with the issuing task.
3850	The collection of transaction statistics or task statistics has not been enabled during system generation.

5.31 ENDPAGE

The ENDPAGE statement terminates a map paging session, clears the scratch record for the session, and clears the map paging options for the completed session. A STARTPAGE/ENDPAGE pair encloses commands that handle a pageable map at runtime. The STARTPAGE command is discussed later in this chapter.

5.31.1 Syntax

```
►►── ENDPAGE session . ──────────────────────────────────────────►◄
```

5.31.2 Example

The following statement ends a map paging session:

```
ENDPAGE SESSION.
```

5.31.3 Status codes

After completion of the ENDPAGE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully

5.32 ENQUEUE

The ENQUEUE statement acquires or tests the availability of a resource or list of resources. Resources are defined during installation and system generation and typically include storage areas, common routines, queues, and processor time.

An enqueued resource can be exclusive or shared:

- **Exclusive** — The resource is owned exclusively by the issuing task and is not available to any other tasks. CA-IDMS prohibits other tasks from obtaining resources that have been ENQUEUED exclusively.

  Note: An exclusive ENQUEUE request prohibits another task from enqueueing a resource by name; however, it does not prohibit the use of the resource by another task. Therefore, to effect true resource protection, you must enqueue and dequeue resources consistently.

- **Shared** — The resource is available to all tasks. CA-IDMS allows other tasks to issue nonexclusive ENQUEUE requests for the resources, permitting the resources to be shared.

5.32.1 Syntax

```
►►── ENQUEUE ──┬──────────┬──────────────────────────────────────────►
              ├─ WAIT ◄─┤
              ├─ NOWAIT ─┤
              └─ TEST ───┘

►─┬─↓─ NAME resource-id LENGTH resource-length ──┬──────────────┬──┬── . ──────►◄
  │                                              ├─ EXCLUSIVE ◄─┤  │
  │                                              └─ SHARED ─────┘  │
  └──────────────────────────────────────────────────────────────┘
```

5.32.2 Parameters

WAIT

Specifies that CA-IDMS is to wait for all resources to be freed if it cannot service the request immediately.

This is the default.

NOWAIT

Specifies that CA-IDMS is not to wait to acquire resources that are not currently available. If NOWAIT is specified, the program should check the ERROR-STATUS field in the IDMS-DC communications block to determine if the function has been completed. If the ERROR-STATUS value is 3901, indicating that a resource could not be obtained immediately, the request has not been serviced and the program should perform alternative processing before reissuing the NOWAIT request.

TEST

Requests CA-IDMS to test the availability of the specified resources. If TEST is specified, the program should check the ERROR-STATUS field in the IDMS-DC communications block to determine the outcome of the test.

NAME

Specifies the ID associated with a resource.

Multiple resource specifications must be separated by at least one blank.

resource-id

The symbolic name of a user-defined field that contains the resource ID. The resource ID must be the name of a resource defined to the DC system. Any resource name can be specified, provided that all programs that access the resource use the same name.

LENGTH

Specifies the length of the resource.

resource-id-length

Either the symbolic name of a user-defined field that contains the length of the resource ID or the length itself expressed as a numeric constant.

EXCLUSIVE

Assigns the exclusive attribute to the named resource.

This is the default.

SHARED

Assigns the shared attribute to the named resource.

5.32.3 Examples

The statements below illustrate the use of the ENQUEUE statement:

Example 1: The following statement requests CA-IDMS to enqueue the CODE-VALUE and PAYROLL-LOCK resources. CODE-VALUE is reserved for the issuing task's exclusive use; PAYROLL-LOCK can be shared.

```
ENQUEUE
   WAIT
   NAME CODE-VALUE LENGTH 10
   NAME PAYROLL-LOCK LENGTH 16 SHARED.
```

Example 2: The following statement requests CA-IDMS to test the availability of the resource whose identifier is contained in the RESOURCE-NAME field:

```
ENQUEUE
   TEST
   NAME RESOURCE-NAME LENGTH RESOURCE-LENGTH.
```

5.32.4 Status codes

After completion of an ENQUEUE function to *acquire* resources, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
3901	At least one of the requested resources cannot be enqueued immediately; to wait would cause a deadlock. No new resources have been acquired.
3908	At least one of the requested exclusive resources is currently owned by another task. No new resources have been acquired.

After completion of an ENQUEUE function to *test* resources, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	All requested resources are available.
3908	At least one of the tested resources is already owned by another task.
3909	At least one of the tested resources is not yet owned by another task and is available to the issuing task.

5.33 ERASE

The ERASE statement performs the following functions:

- Disconnects the specified record from all set occurrences in which it participates as a member and logically or physically deletes the record from the database
- Optionally erases all records that are mandatory members of set occurrences owned by the specified record
- Optionally disconnects or erases all records that are optional members of set occurrences owned by the specified record

ERASE is a two-step procedure that first cancels the existing membership of the named record in specific set occurrences and then releases for reuse the space occupied by the named record and its db-key. Erased records are unavailable for further processing by any DML statement.

Before execution of the ERASE statement, the following conditions must be satisfied:

- All areas affected either implicitly or explicitly must be readied in one of the update usage modes (see 5.66, “READY” on page 5-201 later in this chapter)
- All sets in which the specified record participates as owner either directly or indirectly (for example, as owner of a set with a member that is owner of another set) and all member record types in those sets must be included in the subschema in use
- The specified record must be established as current of run unit

Currency: Following successful execution of an ERASE statement, currency is nullified for all record types involved in the erase both explicitly and implicitly. Run unit and area currency remain unchanged. Next, prior, and owner currencies are preserved for sets from which the last record occurrence was erased. These currencies enable you to retrieve the next or prior records within the area or the next, prior, or owner records within the set in which the erased record participated. An attempt to retrieve erased records results in a non-zero status condition.

5.33.1 Syntax

```
►►── ERASE record-name ┬──────────────────────┬── . ──────────────────►◄
                       ├─ PERMANENT MEMBERS ──┤
                       ├─ SELECTIVE MEMBERS ──┤
                       └─ ALL MEMBERS ────────┘
```

5.33.2 Parameters

record-name

The name of the record to be erased. It must be a record included in the subschema. The current of *record-name* must be current of run unit.

Unless PERMANENT, SELECTIVE, or ALL qualification follows, a non-zero status condition results if the named record is the owner of any nonempty set occurrences.

Native VSAM users: ERASE *record-name* is the only form of the ERASE statement valid for records in a native VSAM KSDS or RRDS; the ERASE statement is not valid for a native VSAM ESDS.

PERMANENT MEMBERS

Erases the specified record and all mandatory member record occurrences owned by the specified record. Optional member records are disconnected. If any of the erased mandatory members are themselves the owner of any set occurrences, the ERASE statement is executed on such records as if they were directly the object record of an ERASE PERMANENT statement (that is, all mandatory members of such sets are also erased). This process continues until all direct and indirect members have been processed.

SELECTIVE MEMBERS

Erases the specified record and all mandatory member record occurrences owned by the specified record. Optional member records are erased if they do not *currently participate* as members in other set occurrences. All erased member records that are themselves the owners of any set occurrences are treated as if they were the object of an ERASE SELECTIVE statement.

ALL MEMBERS

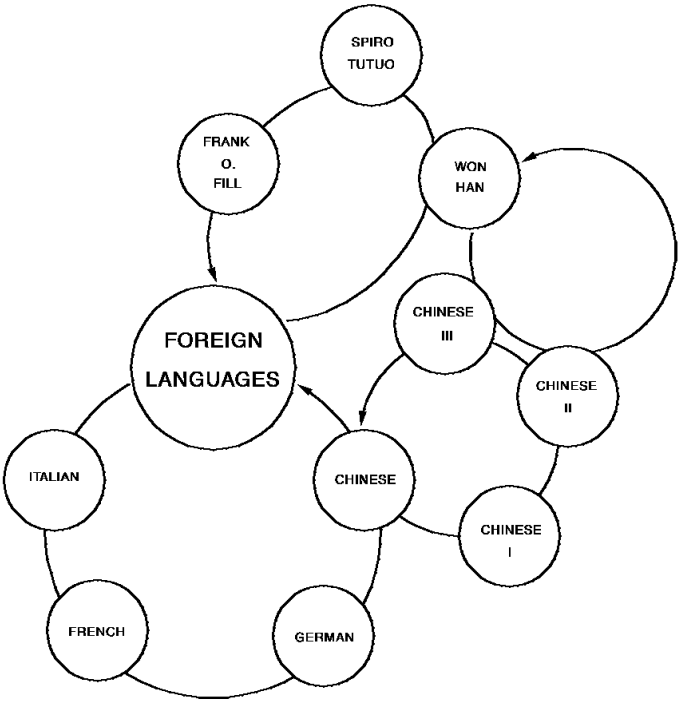
Erases the specified record and all mandatory and optional member record occurrences owned by the specified record. All erased member records that are themselves the owners of any set occurrences are treated as if they were the object record of an ERASE ALL statement.

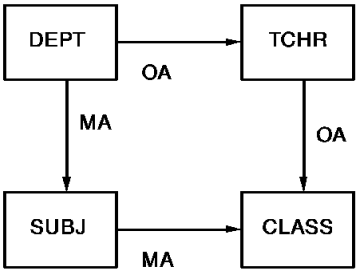
5.33.3 Example

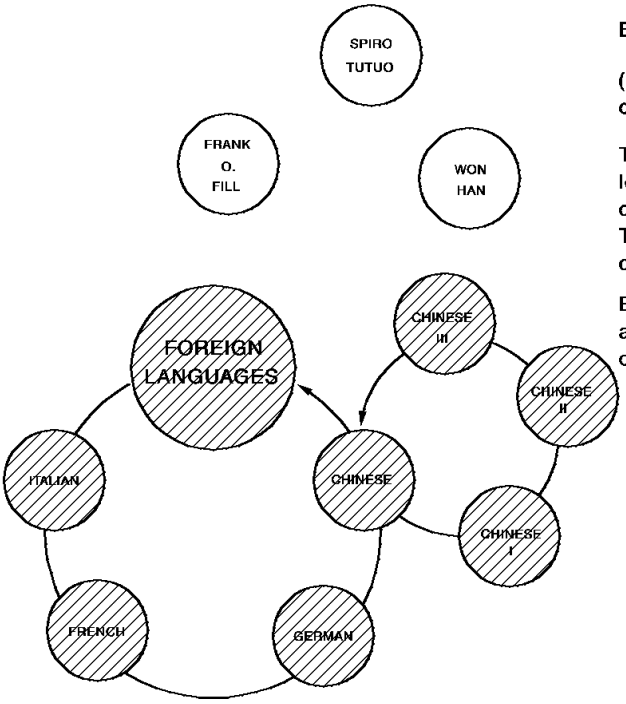
Use of the ERASE statement: The following figure illustrates use of the three parameters of the ERASE statement.

The outcome of the ERASE statement varies based on the qualifier specified (PERMANENT, SELECTIVE, or ALL). Although all three qualifiers cause all mandatory members owned by the specified record to be erased, they differ in their effect on optional members.

Because the sample employee database provides no appropriate examples of these parameters, this figure and the one after use a sample high school database instead.





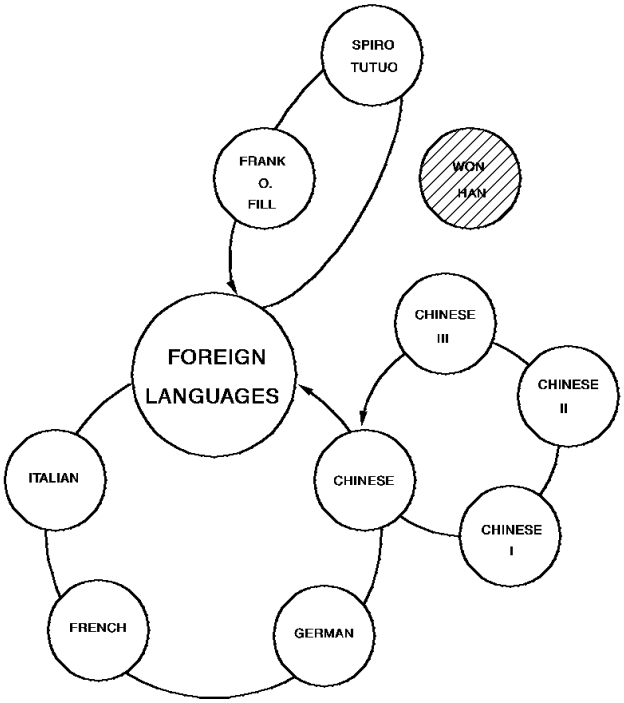


ERASE DEPT PERMANENT

(assuming that FOREIGN LANGUAGES is current of run unit)

The Foreign Languages Department can no longer be funded, so it is deleted from the database along with its subjects and classes. The teachers will be reassigned to other departments.

Erases the foreign language record and all mandatory members; disconnects optional members.

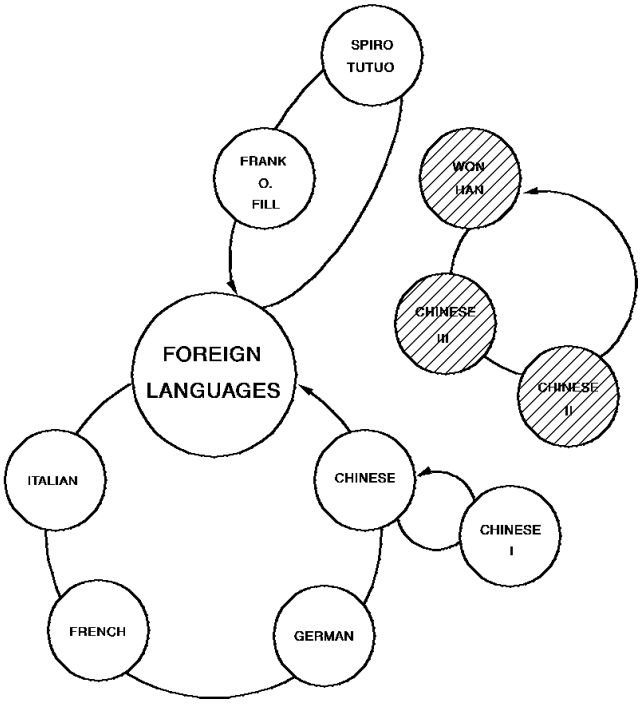


ERASE TCHR SELECTIVE

(assuming that WON HAN is current of the run unit)

Won Han has quit in the middle of the semester. His classes will be finished by another teacher, so only Won Han is erased. (Remember that an unqualified ERASE command cannot be used to erase the owner of a non-empty set.)

Erases the TCHR record occurrence, mandatory members (none, TCHR-CLASS is OA), and optional orphans (none, CHI is in the SUBJ-CLASS set).



ERASE TCHR ALL

(assuming that Won Han is current of run unit)

No one is available to teach Won Han's classes, so both he and his classes are deleted from the database.

Erases the TCHR record occurrence and all mandatory and optional members.

ERASE Currency: The following figure shows the effect each of the parameters has on currency.

	CURRENCIES: RUN UNIT, RECORD, SET, AREA									
	RUN UNIT	DEPT	TCHR	SUBJ	CLASS	DEPT-TCHR	DEPT-SUBJ	TCHR-CLASS	SUBJ-CLASS	SCHOOL-REGION
ESTABLISHED CURRENCIES	FOREIGN LANG.	FOREIGN LANG.		FRENCH	CHI I.	FOREIGN LANG.	FOREIGN LANG.	CHI I.	FRENCH	FOREIGN LANG.
ERASE DEPT PERMANENT	FOREIGN LANG.	NULL		NULL	NULL	NP	NULL	NP	NULL	FOREIGN LANG.
ESTABLISHED CURRENCIES	WON HAN	FOREIGN LANG.	WON HAN		CHI I.	WON HAN	FOREIGN LANG.	WON HAN	CHI I.	WON HAN
ERASE TCHR SELECTIVE	WON HAN	FOREIGN LANG.			CHI I.	NP	FOREIGN LANG.	NP	CHI I.	WON HAN
ESTABLISHED CURRENCIES	WON HAN		WON HAN	FRENCH		WON HAN	FRENCH	WON HAN	FRENCH	WON HAN
ERASE TCHR ALL	WON HAN			FRENCH	NULL	NP	FRENCH	NP	NP	WON HAN

5.33.4 Status codes

After completion of the ERASE function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0208	The object record is not in the specified subschema.
0209	The named record's area has not been readied in one of the three update usage modes.
0210	The subschema specifies an access restriction that prohibits use of the ERASE statement. For SPF users, this code can also indicate use of an invalid form of the ERASE statement.
0213	A current record of run unit has either not been established or has been nullified by a previous ERASE statement.
0217	A db-key has been encountered that contains a long-term permanent lock.
0220	The current record of run unit is not the same record type as the named record.
0221	An area other than the area of the specified record has been readied with an incorrect usage mode.

Status code	Meaning
0225	Currency has not been established.
0226	A broken chain has been encountered in the process of executing an ERASE ALL, PERMANENT, or SELECTIVE.
0230	An attempt has been made to erase the owner record of a nonempty set.
0233	Either erasure of the record occurrence is not allowed in this subschema or all sets in which the record participates have not been included in the subschema.
0260	A record occurrence has been encountered whose type is inconsistent with the set named in the ERROR-SET field of the IDMS communications block; probable causes include: a broken chain and improper database description.
0261	The record cannot be erased because of broken chains in the database.

5.34 ERASE (LRF)

The ERASE statement deletes a logical-record occurrence. The ERASE statement does not necessarily result in the deletion of all or any of the database records used to create the specified logical record; the path selected to service an ERASE logical-record request performs whatever database access operations the DBA has specified to service the request. For example, if a DEPARTMENT loses an employee, the EMP-JOB-LR logical record that contains information about that employee would be erased. However, only the information about the former employee would be erased from the database, not all the information about the department; that is, EMPLOYEE information would be erased, but not DEPARTMENT, JOB, or OFFICE information.

LRF uses field values present in the variable-storage location reserved for the logical record to update the database. You can specify an alternative storage location from which LRF is to take field values to make the appropriate updates to the database.

5.34.1 Syntax

```
►►── ERASE logical-record-name ──────────────────────────────────►
►─┬──────────────────────────────┬──────────────────────────────►
  └─ FROM alt-logical-record-location ─┘
►─┬──────────────────────────┬──────────────────────────────────►
  └─ WHERE boolean-expression ─┘
►─┬──────────────────────────────────┬─── . ──────────────────►◄
  └─ ON path-status imperative-statement ─┘
```

5.34.2 Parameters

logical-record-name

The name of the logical record to erase. The logical record must be defined in the subschema. Unless you specify FROM, LRF uses field values present in the variable-storage location reserved for the logical record to make any necessary updates to the database.

FROM alt-logical-record-location

Names an alternative variable-storage location from which LRF is to obtain field values to perform the appropriate database updates in response to this request. When erasing a logical record that has previously been retrieved into an alternative storage location, use the FROM clause to name the same location specified in the OBTAIN request. The alternate record location must be defined in the WORKING-STORAGE/LINKAGE SECTION.

WHERE boolean-expression

Specifies the selection criteria to be applied to the specified logical record. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

ON <u>path-status imperative-statement</u>

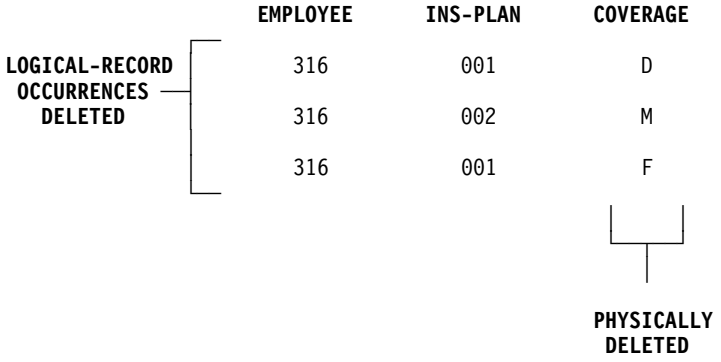
Specifies the action to be taken if *path-status* is returned to the LR-STATUS field in the LRC block. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

5.34.3 Example

The following example illustrates a request to erase all occurrences of a former employee's EMP-INSURANCE-LR logical record; the DBA-designated path status ALL-ERASED indicates that all occurrences of the EMP-INSURANCE-LR logical record have been erased.

```
ERASE EMP-INSURANCE-LR WHERE EMP-ID-0415 EQ '0316'
  ON ALL-ERASED PERFORM EMP-INS-DELETION-RPT.
```

ERASE EMP- INSURANCE-LR: As defined by the DBA, the ERASE EMP-INSURANCE-LR path group logically deletes all of the specified EMP-INSURANCE-LR occurrences but physically deletes only the COVERAGE records, as illustrated by the following figure.



5.35 FIND/OBTAIN

The FIND statement locates a record occurrence in the database; the OBTAIN statement locates a record and moves the data associated with the record to the record buffers. Because the FIND and OBTAIN command statements have identical formats, they are discussed together. The six formats of the FIND/OBTAIN statement are as follows:

- **FIND/OBTAIN CALC** accesses a record occurrence by using its CALC key value.
- **FIND/OBTAIN CURRENT** accesses a record occurrence by using established currencies.
- **FIND/OBTAIN DB-KEY** accesses a record occurrence by using its database key.
- **FIND/OBTAIN OWNER** accesses the owner record of a set occurrence.
- **FIND/OBTAIN WITHIN SET USING SORT KEY** accesses a record occurrence in a sorted set by using its sort key value.
- **FIND/OBTAIN WITHIN SET/AREA** accesses a record occurrence based on its logical location within a set or on its physical location within an area.

You can place locks on located record occurrences by using the KEEP clause of a FIND/OBTAIN statement. The KEEP clause sets a shared or exclusive lock:

- **KEEP** places a shared lock on the located record occurrence. Other concurrently executing run units can access but not update the locked record.
- **KEEP EXCLUSIVE** places an exclusive lock on the located record occurrence. Other concurrently executing run units can neither access nor update the locked record.

For more information on record locks, see 5.51, “KEEP CURRENT” on page 5-143 later in this chapter.

Each format of the FIND/OBTAIN statement is discussed separately on the following pages.

5.36 FIND/OBTAIN CALC/DUPLICATE

The FIND/OBTAIN CALC/DUPLICATE statement locates a record based on the value of an element defined as a CALC key in the record. The specified record must be stored in the database with a location mode of CALC. Before issuing the FIND/OBTAIN CALC/DUPLICATE statement, you must initialize a field in program variable storage with the CALC-key value.

You can use the DUPLICATE option to access duplicate records with the same CALC-key value as the record that is current of record type, provided that a FIND/OBTAIN CALC statement has previously accessed an occurrence of the same record type.

Currency: Following successful execution of a FIND/OBTAIN CALC/DUPLICATE statement, the accessed record becomes the current record of run unit, its record type, its area, and all sets in which it currently participates as member or owner.

5.36.1 Syntax

```
►►─┬─ FIND ──┬──┬──────────────────────┬──┬─ CALC ──┬──── record-name . ──►◄
   └─ OBTAIN ─┘  └─ KEEP ─┬──────────┬─┘  └─ ANY ──┘
                          └─ EXCLUSIVE ─┘    └─ DUPLICATE ─┘
```

5.36.2 Parameters

KEEP
Places a shared lock on the accessed record.

EXCLUSIVE
Places an exclusive lock on the accessed record.

CALC (ANY)
Locates the first or only occurrence of the specified record type whose CALC key matches the value of the CALC data item in program variable storage.

CALC and ANY are synonyms and can be used interchangeably.

DUPLICATE
Locates the next record with the same CALC key value as the current of the specified record type. Use of the DUPLICATE option requires prior selection of an occurrence of the same record type with the CALC option. If the value of the CALC key in variable storage is not equal to the CALC-key field of the current of record type, a status code of 0332 is returned.

record-name
The name of the record type to be located.

5.36.3 Example

To retrieve an occurrence of the EMPLOYEE record by using the FIND/OBTAIN CALC/DUPLICATE statement, you must first initialize the variable-storage field that contains the CALC control element. The following statements initialize the CALC field EMP-ID-0415 and retrieve an occurrence of the EMPLOYEE record:

```
MOVE EMP-ID-IN TO EMP-ID-0415.
OBTAIN CALC EMPLOYEE.
```

5.36.4 Status codes

After completion of the FIND/OBTAIN CALC/DUPLICATE function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0301	The area in which the named record participates has not been readied.
0306	A successful FIND/OBTAIN CALC has not yet been executed (applies to the DUPLICATE option only).
0308	The named record is not in the subschema. The program probably invoked the wrong subschema.
0310	The subschema specifies an access restriction that prohibits retrieval of the named record.
0318	The record has not been bound.
0326	Either the record or cannot be found or no more duplicates exist for the named record.
0331	The retrieval statement format conflicts with the record's location mode.
0332	The value of the CALC data item in program variable storage does not equal the value of the CALC data item in the current record (applies to the DUPLICATE option only).
0364	The CALC control element has not been described correctly either in the program or in the subschema.
0370	A database file will not open properly.

If the FIND/OBTAIN statement includes an explicit KEEP: 03 is the major code returned if an error occurs during FIND processing, 06 if the error occurs during KEEP processing.

5.37 FIND/OBTAIN CURRENT

The FIND/OBTAIN CURRENT statement locates the record that is current of its record type, set, or area. This form of the FIND/OBTAIN statement is an efficient means of establishing the appropriate record as current of run unit before executing a DML statement that uses run-unit currency (for example, ACCEPT, IF, GET, MODIFY, ERASE).

Currency: Following successful execution of a FIND/OBTAIN CURRENT statement, the accessed record is current of run unit, its record type, its area, and all sets in which it currently participates as member or owner.

5.37.1 Syntax

```
►►─┬─ FIND ──┬──┬──────────────────┬── CURRENT ──┬──────────────────────┬── . ─►◄
   └─ OBTAIN ─┘  └─ KEEP ─┬──────┬─┘             ├─ record-name ────────┤
                          └─ EXCLUSIVE ─┘          ├─ WITHIN set-name ───┤
                                                   └─ WITHIN area-name ──┘
```

5.37.2 Parameters

KEEP
Places a shared lock on the accessed record.

EXCLUSIVE
Places an exclusive lock on the accessed record.

CURRENT
Locates the current record occurrence of a specified record type, set, or area.

Record-name
Accesses the current record of the specified record type.

WITHIN set-name
Accesses the current record of the specified set.

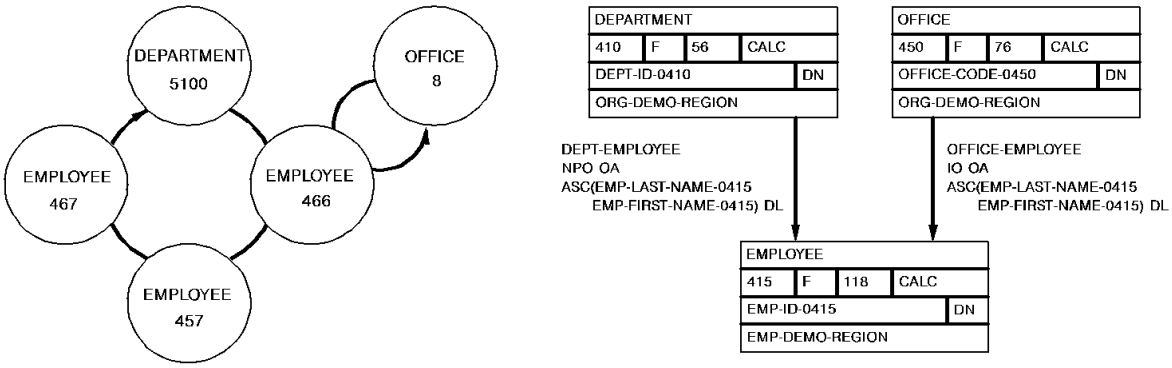
WITHIN area-name
Accesses the current record of the specified area.

5.37.3 Example

Using the FIND/OBTAIN CURRENT statement: The following figure illustrates use of the FIND/OBTAIN CURRENT statement to establish the proper record as current of run unit before the record is modified.

Enter the database on DEPARTMENT 5100 by using CALC retrieval. Then examine EMPLOYEE 466 by using within set retrieval and obtain further information from its owner OFFICE record (OFFICE 8). OFFICE 8 becomes current of run unit. Before modifying EMPLOYEE 466, you must issue the FIND CURRENT statement to reestablish EMPLOYEE 466 as current of run unit.

For a complete description of the MODIFY statement and its use, see 5.57, “MODIFY” on page 5-171 later in this chapter.



	CURRENCIES RUN UNIT, RECORD, SET, AREA							
	RUN UNIT	DEPARTMENT	EMPLOYEE	OFFICE	DEPT-EMPLOYEE	OFFICE-EMPLOYEE	ORG-DEMO-REGION	EMP-DEMO-REGION
MOVE 5100 TO DEPT-ID. FIND CALC DEPARTMENT.	5100	5100			5100		5100	
OBTAIN FIRST WITHIN DEPT-EMPLOYEE.	466	5100	466		466	466	5100	466
OBTAIN OWNER WITHIN OFFICE-EMPLOYEE.	8	5100	466	8	466	8	8	466
FIND CURRENT EMPLOYEE.	466	5100	466	8	466	466	8	466
MODIFY EMPLOYEE.	466	5100	466	8	466	466	8	466

5.37.4 Status codes

After completion of the FIND/OBTAIN CURRENT function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0301	The area in which the named record participates has not been readied.
0306	Currency has not been established for the named record, set, or area.
0308	The named record or set is not in the subschema. The program has probably invoked the wrong subschema.
0310	The subschema specifies an access restriction that prohibits retrieval of the named record.
0313	A current record of run unit either has not been established or has been nullified by a previous ERASE statement.
0323	The specified area name has not been included in the subschema invoked.

If the FIND/OBTAIN statement includes an explicit KEEP: 03 is the major code returned if an error occurs during FIND processing, 06 if the error occurs during KEEP processing.

5.38 FIND/OBTAIN DB-KEY

The FIND/OBTAIN DB-KEY statement locates a record occurrence directly using a database key that has been stored previously by the program. The DML ACCEPT statement, explained earlier in this chapter, or the COBOL MOVE statement can be used to save a db-key. Any record in the program's subschema can be accessed directly in this manner, regardless of its location mode.

Native VSAM users: This statement is not valid for accessing data records in a native VSAM key-sequenced data set (KSDS).

Currency: After successful execution of a FIND/OBTAIN DB-KEY statement, the accessed record becomes the current record of run unit, its record type, its area, and all sets in which it currently participates as member or owner. In addition, the RECORD-NAME field of the IDMS communications block is updated with the name of the accessed record.

Note that currency is not used to determine the specified record of the FIND/OBTAIN DB-KEY statement; the record is identified by its db-key and, optionally, by its record type.

5.38.1 Syntax

```
►►─┬─ FIND   ─┬─┬─ KEEP ─┬─────────────┬──────────────────────────────►
   └─ OBTAIN ─┘ │        └─ EXCLUSIVE ─┘ │
                └──────────────────────────┘

►─┬─ DB-KEY is db-key ─┬─────────────────────────────────────────────►◄
  │                    └─ PAGE-INFO page-info ─┘
  │
  └─────────────── DB-KEY is db-key ──────────────────────────────────┘
    └─ rec-name ─┘
```

5.38.2 Parameters

KEEP
Places a shared lock on the accessed record.

EXCLUSIVE
Places an exclusive lock on the accessed record.

DB-KEY IS
Locates a record directly by using a db-key value contained in program variable storage.

db-key
A field that identifies the location within program variable storage that contains a db-key previously saved by the program.

If a record name has been specified, *db-key* must contain the db-key of an occurrence of the named record type. If a record name has not been specified, *db-key* can contain the db-key of an occurrence of any record type in the subschema.

PAGE-INFO

Specifies page information that is used to determine the area with which the dbkey is associated. If not specified, the page information associated with the record that is current of rununit is used.

Note: Page information is only used if the subschema includes areas that have mixed page groups; otherwise, it is ignored.

page-info

A four-byte field that may be defined either as a group field or as a fullword field (PIC S9(8) COMP). Identifies the location in variable storage that contains the page information previously saved by the program.

Page information is returned in the PAGE-INFO field in the subschema control area if the subschema includes areas in mixed page groups. Page information may also be returned using an ACCEPT PAGE-INFO statement.

rec-name

The record type of the requested record. *Rec-name* must name a record that is included in the subschema.

5.38.3 Example

The following statement locates the occurrence of the HOSPITAL-CLAIM record whose db-key matches the value of a field in program variable storage called SAVED-KEY:

```
FIND HOSPITAL-CLAIM DB-KEY IS SAVED-KEY.
```

The located record becomes current of run unit, current of the HOSPITAL-CLAIM record type, current of the INS-DEMO-REGION area, and current of the COVERAGE-CLAIMS set.

5.38.4 Status codes

After completion of the FIND/OBTAIN DB-KEY function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0301	The area in which the named record participates has not been readied.
0302	The db-key is inconsistent with the area in which the record is stored. Either the db-key has not been initialized properly or the record name is incorrect.
0308	The named record is not in the subschema. The program has probably invoked the wrong subschema.

Status code	Meaning
0310	The subschema specifies an access restriction that prohibits retrieval of the named record.
0326	The record cannot be found; record occurrence not correct type.
0370	A database file will not open properly.
0371	The requested page cannot be found in the DMCL.

If the FIND/OBTAIN statement includes an explicit KEEP: 03 is the major code returned if an error occurs during FIND processing, 06 if the error occurs during KEEP processing.

5.39 FIND/OBTAIN OWNER

The FIND/OBTAIN OWNER statement locates the owner record of the current occurrence of a set. This statement can be used to retrieve the owner record of any set whether or not that set has been assigned owner pointers.

Native VSAM users: The FIND/OBTAIN OWNER statement is not valid since owner records are not defined in native VSAM data sets.

Currency: In order to execute a FIND/OBTAIN OWNER statement, currency must be established for the specified set.

Note: When a record declared as an optional or manual member of a set is retrieved, it is *not* established as current of set if it is not currently connected to the specified set. A subsequent attempt to retrieve the owner record will locate instead the owner of the current record of set. In such cases, you should determine whether the retrieved record is actually a member in the specified set before executing the FIND/OBTAIN OWNER statement. The IF MEMBER statement, explained later in this chapter, can be used for this purpose.

Following successful execution of a FIND/OBTAIN OWNER statement, the accessed record becomes the current record of run unit, its record type, its area, and all sets in which it currently participates as member or owner. If the current record of set is the owner record when the statement is executed, currency within the specified set remains unchanged.

5.39.1 Syntax

```
►►─┬─ FIND ──┬──┬──────────────────────┬── OWNER WITHIN set-name . ──────►◄
   └─ OBTAIN ─┘  └─ KEEP ─┬──────────┬─┘
                          └─ EXCLUSIVE ─┘
```

5.39.2 Parameters

KEEP
Places a shared lock on the accessed record.

EXCLUSIVE
Places an exclusive lock on the accessed record.

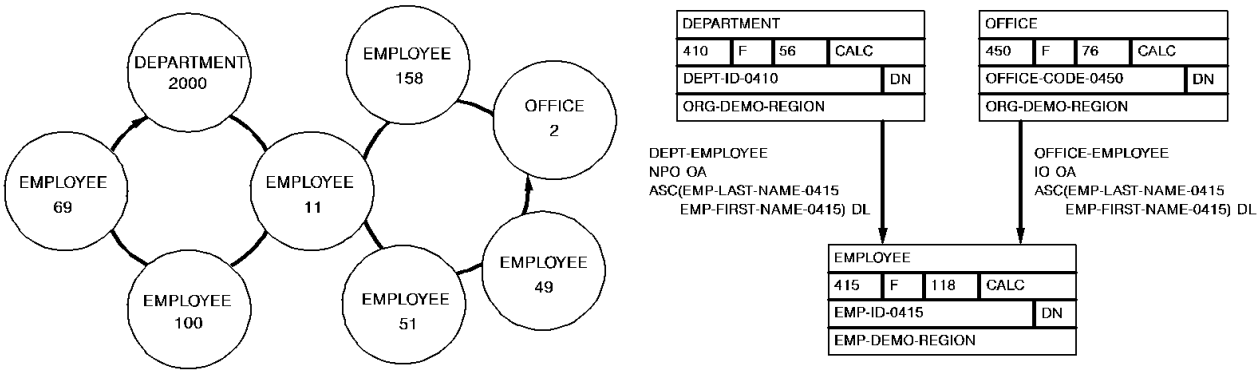
OWNER
Locates the owner record of the specified set.

WITHIN
Specifies the set whose owner record is to be retrieved.

set-name
The name of a set included in the subschema.

5.39.3 Example

Using the FIND/OBTAIN OWNER statement to move through the database: The following figure illustrates use of the FIND/OBTAIN OWNER statement.



	CURRENCIES RUN UNIT, RECORD, SET, AREA							
	RUN UNIT	*DEPARTMENT*	*EMPLOYEE*	*OFFICE*	*DEPT-EMPLOYEE*	*OFFICE-EMPLOYEE*	*ORG-DEMO-REGION*	*EMP-DEMO-REGION*
MOVE 2000 TO DEPT-ID. OBTAIN CALC DEPARTMENT.	2000	2000			2000		2000	
OBTAIN FIRST WITHIN DEPT-EMPLOYEE.	11	2000	11		11	11	2000	11
OBTAIN OWNER WITHIN OFFICE-EMPLOYEE.	2	2000	11	2	11	2	2	11

5.39.4 Status codes

After completion of the FIND/OBTAIN OWNER function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0301	The area in which the object record participates has not been readied.
0306	Currency has not been established for the record, set, or area.
0308	The named set is not in the subschema. The program has probably invoked the wrong subschema.
0310	The subschema specifies an access restriction that prohibits retrieval of the object record.
0360	A record occurrence has been encountered whose record type is not a member or owner of the set as it is defined in the subschema.
0370	A database file will not open properly.

If the FIND/OBTAIN statement includes an explicit KEEP: 03 is the major code returned if an error occurs during FIND processing, 06 if the error occurs during KEEP processing.

5.40 FIND/OBTAIN WITHIN SET USING SORT KEY

The FIND/OBTAIN WITHIN SET USING SORT KEY statement locates a member record in a sorted set. Sorted sets are ordered in ascending or descending sequence based on the value of a sort-control element in each member record. The search begins with either the current of set or the owner of the current of set and always proceeds through the set in the next direction.

Before issuing this statement, you must initialize the sort control element in program variable storage. The record occurrence selected will have a key value equal to the value of the sort control element. If more than one record occurrence contains a sort key equal to the key value in variable storage, the first such record will be selected.

In a batch environment, sorted sets can be processed more efficiently by sorting the input transactions.

Currency: Following successful execution of a FIND/OBTAIN WITHIN SET USING SORT KEY statement, the accessed record becomes current of run unit, its record type, its area, and all sets in which it currently participates as member or owner. If a member record with the requested sort-key value is not found, the current of set is nullified but the next of set and prior of set are maintained. The next of set is the member record with the next higher sort-key value (or next lower for descending sets) than the requested value; the prior of set is the member record with the next lower value (or higher for descending sets) than requested. Because these currencies are maintained, the program can walk the set to do a generic search on the sort-key value.

5.40.1 Syntax

```
►►─┬─ FIND ───┬─┬──────────────────────────┬─ record-name WITHIN set-name ──►
   └─ OBTAIN ─┘ └─ KEEP ─┬──────────────┬─┘
                         └─ EXCLUSIVE ─┘

►─┬──────────┬─ USING sort-key . ──────────────────────────────────────►◄
  └─ CURRENT ─┘
```

5.40.2 Parameters

KEEP
Places a shared lock on the accessed record.

EXCLUSIVE
Places an exclusive lock on the accessed record.

record-name
Specifies the record type to locate.

WITHIN

Specifies the set to be searched.

Unless you specify CURRENT, the search begins with the *owner* of the specified set.

set-name

The name of a sorted set included in the subschema.

CURRENT

Indicates that the search begins with the currencies already established for the specified set.

If the key value for the record that is current of set is higher than the key value of the requested record (assuming ascending set order), a non-zero status condition results. In a descending set order, if the key value for the record that is current of set is lower than the key value of the requested record, a non-zero status condition results.

USING

Specifies the sort control element to be used in searching the sorted set.

sort-key

The symbolic name of a field defined in working storage that contains the value of the sort control element.

Note: Due to the architecture of the client interface for CA-IDMS, 256 bytes will be moved regardless of the actual length of the working storage sort key. This additional storage should be accounted for in order to avoid potential program exceptions that can occur. While these exceptions are rare, they are more probable if the sort-key is defined in a FILE or LINKAGE SECTION definition. To avoid this problem, it is recommended that the sort-key be defined in the program's WORKING STORAGE SECTION, padded to a full 256 bytes; and moved in and out of the FILE or LINKAGE SECTION fields.

Here is an example of OBTAIN RECA WITHIN RECA-SET USING RECA-KEY. The record's sort key would be defined as follows in the WORKING-STORAGE SECTION:

```
01  RECA-KEY.
    02  RECA-FIELD1              PIC X(10).
    02  RECA-FIELD2              PIC X(10).
```

It should be changed to:

```
01  RECA-KEY.
    02  RECA-FIELD1              PIC X(10).
    02  RECA-FIELD2              PIC X(10).
    02  FILLER                   PIC X(236).
```

5.40.3 Example

The following example illustrates the use of a FIND/OBTAIN WITHIN SET USING SORT KEY statement. Assume that the SKILL-NAME-NDX set is ordered in ascending sequence based on the value stored in SKILL-NAME-0455 in each SKILL record occurrence. Retrieval of a SKILL record with a skill name equal to PL/I is accomplished by the following statements:

```
MOVE 'PL/I' TO SKILL-NAME-0455.
FIND SKILL WITHIN SKILL-NAME-NDX
                  USING SKILL-NAME-0455.
```

5.40.4 Status codes

After completion of the FIND/OBTAIN WITHIN SET USING SORT KEY function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0301	The area in which the named record participates has not been readied.
0306	Currency has not been established for the named set.
0308	Either the named record or set is not in the subschema or the named record is not a member of the named set. The program has probably invoked the wrong subschema.
0310	The subschema specifies an access restriction that prohibits retrieval of the named record.
0326	The record cannot be found.
0331	The retrieval statement format conflicts with the record's location mode.
0360	A record occurrence has been encountered whose record type is not a member or owner of the set as it is defined in the subschema.
0370	A database file will not open properly.

If the FIND/OBTAIN statement includes an explicit KEEP: 03 is the major code returned if an error occurs during FIND processing, 06 if the error occurs during KEEP processing.

5.41 FIND/OBTAIN WITHIN SET/AREA

The FIND/OBTAIN WITHIN SET/AREA statement locates records either logically, based on set relationships, or physically, based on database location. The formats of this statement allow you either to access serially each record in a set or area or to select specific occurrences of a given record type within the set or area.

The following rules apply to the selection of member records within a **set**:

- The set occurrence used as the basis for the operation is determined by the current record of the specified set. Set currency must be established before attempting to access records within a set.
- The next or prior record within a set is the subsequent or previous record relative to the *current record of the named set* in the logical order of the set. The prior record in a set can be retrieved only if the set has been assigned prior pointers.
- The first or last record within a set is the first or last member occurrence in terms of the logical order of the set. The selected record is the same as would be selected if the current of set were the owner record and the next or prior record had been requested. The last record in a set can be retrieved only if the set has prior pointers.
- The *n*th occurrence of a record within a set can be retrieved by specifying a sequence number that identifies the position of the record in the set. The DBMS begins its search with the *owner of the current of set* for the specified set and continues until it locates the *n*th record or encounters an end-of-set condition. If the specified sequence number is negative, the search proceeds in the prior direction within the set. A negative sequence number can be used only if the set has prior pointers; a sequence number of 0 produces a status code of 0304.
- When an end-of-set condition occurs, the owner record occurrence of the set becomes the current record of run unit, current of its record type, current of its area, and current record of *only* the set involved in this operation. Currency of other sets in which the specified record participates as owner or member remains unaffected.

  Note: If OBTAIN has been specified, the contents of the owner record are not moved to program variable storage (that is, OBTAIN under these circumstances is treated as a FIND).

  Native VSAM users: When an end-of-set condition occurs, all currencies remain unchanged.

The following rules apply to the selection of records within an **area**:

- The first record occurrence within an area is the one with the lowest database key; the last record is the one with the highest database key
- The next record within an area is the one with the next higher database key relative to the *current record of the named area*; the prior record is the one with the next lower database key relative to the current of area

- The first or last or *n*th record in an area must be retrieved to establish the correct starting position before next or prior records are requested

Currency: Following successful execution of a FIND/OBTAIN WITHIN SET/AREA statement, the accessed record becomes the current record of run unit, its record type, its area, and all sets in which it currently participates as member or owner.

When an end-of-set condition occurs selecting records within a set, the owner record occurrence of the set becomes the current record of run unit, its record type, its area, and *only* the set involved in this operation. Currency of other sets in which the specified record participates as owner or member remains unaffected.

5.41.1 Syntax

```
►►─┬─ FIND ──┬──┬──────────────────────┬──┬─ NEXT ──────────────────────►
   └─ OBTAIN ─┘  └─ KEEP ─┬──────────┬─┘  ├─ PRIOR ─┤
                          └─ EXCLUSIVE ─┘  ├─ FIRST ─┤
                                           ├─ LAST ──┤
                                           └─ number ─┘

►──────────────── WITHIN ─┬─ set-name ─┬─ . ──────────────────────►◄
   └─ record-name ─┘       └─ area-name ─┘
```

5.41.2 Parameters

KEEP
Places a shared lock on the accessed record.

EXCLUSIVE
Places an exclusive lock on the accessed record.

NEXT
Accesses the next record in the specified set or area relative to the current record.

PRIOR
Accesses the prior record in the specified set or area relative to the current record. The specified set must have prior pointers.

FIRST
Accesses the first record in the specified set or area.

LAST
Accesses the last record in the specified set or area. The specified set must have prior pointers.

number
Accesses the indicated record number in the specified set or area. *Number* must either be a non-zero number or the symbolic name of a numeric field that contains a non-zero value. If the number is negative, the specified set must have prior pointers.

record-name

Specifies that within a set or area, only occurrences of the named record type will be accessed. *Record-name* must be defined as a member of the specified set or contained within the specified area.

WITHIN

Locates a record based on its location within a set or area.

set-name

Specifies the set to be searched. The set must be included in the subschema.

area-name

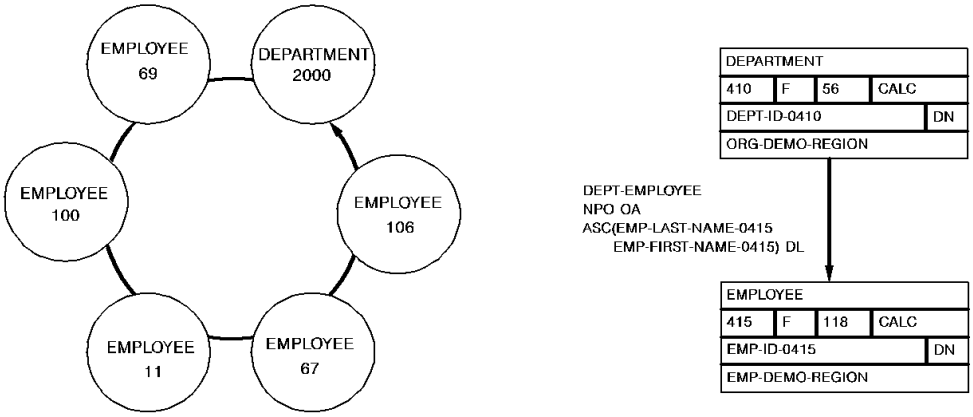
Specifies the area to be searched. The area must be included in the subschema.

Native VSAM users: FIRST, LAST, and *sequence* options are not valid for a native VSAM KSDS with spanned records.

5.41.3 Example

Retrieval of records in an occurrence of the DEPT-EMPLOYEE set: The following figure illustrates the retrieval of records within an occurrence of the DEPT-EMPLOYEE set.

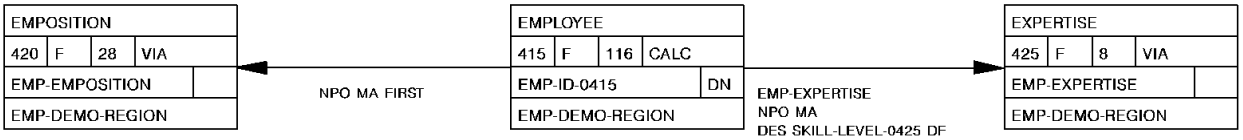
The FIND CALC statement establishes currency in the DEPT-EMPLOYEE set. Member EMPLOYEE records are then retrieved by a series of OBTAIN WITHIN SET statements. EMPLOYEE 106 is the last record in the set and the next OBTAIN statement returns an end-of-set condition, positioning run unit currency at the owner of the set, DEPARTMENT 2000.

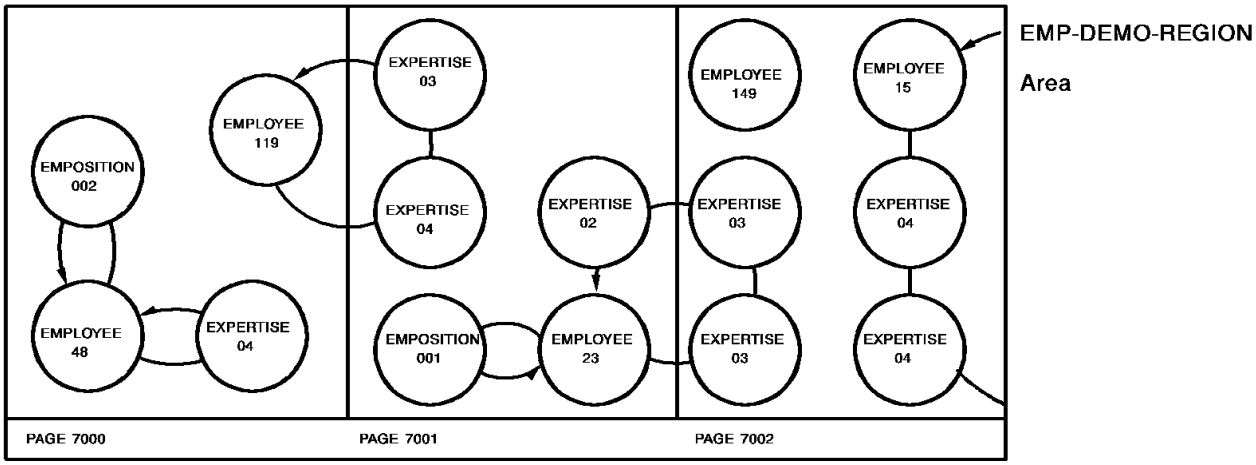


	CURRENCIES RUN UNIT, RECORD, SET, AREA							
	RUN UNIT	DEPARTMENT	EMPLOYEE	DEPT-EMPLOYEE	OFFICE-EMPLOYEE	ORG-DEMO-REGION	EMP-DEMO-REGION	
MOVE 2000 TO DEPT-ID. FIND CALC DEPARTMENT.	2000	2000		2000		2000		
OBTAIN FIRST WITHIN DEPT-EMPLOYEE.	69	2000	69	69	69	2000	69	
OBTAIN NEXT WITHIN DEPT-EMPLOYEE.	100	2000	100	100	100	2000	100	
OBTAIN 5 WITHIN DEPT-EMPLOYEE.	106	2000	106	106	106	2000	106	
OBTAIN NEXT WITHIN DEPT-EMPLOYEE.	2000	2000	106	2000	106	2000	106	ERROR-STATUS OF '0307'

Retrieving records in area containing multiple record types: The following figure illustrates special considerations relating to the retrieval of records within an area that contains multiple record types.

In this example, a sweep of the EMP-DEMO-REGION is performed, retrieving sequentially each EMPLOYEE record and all records in the associated EMPLOYEE-EXPERTISE set. The first command retrieves EMPLOYEE 119. Subsequent OBTAIN WITHIN SET statements retrieve the associated EXPERTISE records and establish currency on EXPERTISE 03. The FIND CURRENT statement is used to reestablish the proper position before retrieving EMPLOYEE 48. If FIND CURRENT EMPLOYEE is not specified, an attempt to retrieve the next EMPLOYEE record in the area would return EMPLOYEE 23.





	RUN UNIT	EMPLOYEE	EXPERTISE	EMP-EXPERTISE	EMP-DEMO-REGION
OBTAIN FIRST EMPLOYEE WITHIN EMP-DEMO-REGION.	119	119		119	119
OBTAIN FIRST EXPERTISE WITHIN EMP-EXPERTISE	04	119	04	04	04
OBTAIN NEXT EXPERTISE WITHIN EMP-EXPERTISE	03	119	03	03	03
FIND CURRENT EMPLOYEE	119	119	03	119	119
OBTAIN NEXT EMPLOYEE WITHIN EMP-DEMO-REGION	48	48	03	48	48

5.41.4 Status codes

After completion of the FIND/OBTAIN WITHIN SET/AREA function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0301	The area in which the named record participates has not been readied.
0306	A successful FIND/OBTAIN CALC has not yet been executed (applies to the DUPLICATE option only).
0307	The end of the set or area has been reached, or the set is empty.

Status code	Meaning
0308	The named record is not in the subschema. The program probably invoked the wrong subschema.
0310	The subschema specifies an access restriction that prohibits retrieval of the named record.
0318	The record has not been bound.
0326	Either the record or SPF index entry cannot be found or no more duplicates exist for the named record.
0331	The retrieval statement format conflicts with the record's location mode.
0332	The value of the CALC data item in program variable storage does not equal the value of the CALC data item in current record (applies to the DUPLICATE option only).
0364	The CALC control element has not been described correctly either in the program or in the subschema.
0370	A database file will not open properly.

If the FIND/OBTAIN statement includes an explicit KEEP: 03 is the major code returned if an error occurs during FIND processing, 06 if the error occurs during KEEP processing.

5.42 FINISH

The FINISH statement relinquishes control over all database areas associated with a program or task and optionally establishes an end-of-task checkpoint for scratch and queue areas associated with a task. FINISH writes statistical information for the database operations performed during run unit execution to the CA-IDMS journal file; it also defines and logs the end checkpoint for a recovery unit.

You can use the FINISH statement to change area usage modes defined by previously issued READY statements.

The FINISH statement is used in both the navigational and the non-navigational environments.

Currency: Following the successful execution of a FINISH request, all currencies are set to null; the issuing program or task cannot perform database access without executing another BIND/READY sequence.

5.42.1 Syntax

```
►►── FINISH ──┬──────┬── . ──────────────────────────────────────►◄
             └─ TASK ─┘
```

5.42.2 Parameters

TASK
Releases all data areas held by all run units executing under the issuing task.

5.42.3 Example

The following statement illustrates the use of the FINISH statement:

```
FINISH.
```

5.42.4 Status codes

After completion of the FINISH function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
5031	The specified request is invalid; the program may contain a logic error
5096	There are too many run units currently active; check the DC system log for details

Status code	Meaning
5097	An invalid status has been received from DBIO/DBMS; check the DC system log for details

5.43 FREE STORAGE

The FREE STORAGE statement instructs CA-IDMS to release all or a part of a variable storage area. The storage to be released must have been acquired by means of a GET STORAGE request in the issuing task or by another task running on the same terminal as the issuing task. A partial release is valid only for user storage; shared storage must be freed in its entirety.

5.43.1 Syntax

```
►►── FREE STORAGE ──────────────────────────────────────────────►
►─┬─ STGID storage-id ──────────────────────────────────────┬─ . ─►◄
  └─ FOR 01-level-storage-data-loc ─┬──────────────────────┬┘
                                    └─ FROM start-free-storage-loc ─┘
```

5.43.2 Parameters

STGID
Specifies variable storage area to be released.

storage-id
Either the symbolic name of a user-defined field that contains the ID or the ID itself enclosed in quotation marks.

FOR 01-level-storage-data-loc
Specifies the LINKAGE SECTION entry of the storage area to be released.

FROM
Releases storage from the specified location to the end of the storage area.

start-free-storage-loc
The symbolic name of a user-defined field that contains the starting point of the storage area to be released.

5.43.3 Example

The following example requests CA-IDMS to release the storage area identified by 09PA:

```
FREE STORAGE STGID '09PA'.
```

5.43.4 Status codes

After completion of the FREE STORAGE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3213	The requested storage ID cannot be found

Status code	Meaning
3232	The derived length of the variable storage area is zero or negative
3234	The request cannot be serviced because the variable storage area is not an 01-level entry in the LINKAGE SECTION

5.44 GET

The GET statement transfers the contents of a specified record occurrence from the record buffer into program variable storage. Elements in the specified record are moved to their respective locations in variable storage according to the subschema view of the record. The transferred elements will appear in storage at the location to which the record has been bound (for further details, see 5.14, “BIND RECORD” on page 5-41 earlier in this chapter).

Currency: The GET statement operates only on the record that is current of run unit. Following successful execution of a GET statement, the accessed record is current of run unit, its record type, its area, and all sets in which it participates as member or owner.

5.44.1 Syntax

```
►►── GET ┬──────────────┬ . ──────────────────────────────────►◄
        └─ record-name ─┘
```

5.44.2 Parameters

record-name

Specifies that the current of run unit must be an occurrence of the named record type.

5.44.3 Example

The following statement moves the record that is current of run unit (in this case, the OFFICE record) from the record buffer into program variable storage:

```
GET OFFICE.
```

5.44.4 Status codes

After completion of the GET function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
0506	Currency has not been established.
0508	The named record is not in the subschema. The program has probably invoked the wrong subschema.
0510	The subschema specifies an access restriction that prohibits retrieval of the named record.

Status code	Meaning
0513	A current record of run unit either has not been established or has been nullified by a previous ERASE statement.
0518	The record has not been bound.
0520	The current record is not the same type as the named record.
0526	The requested record has been erased.
0555	An invalid length has been returned for a variable-length record.

5.45 GET QUEUE

The GET QUEUE statement retrieves a queue record and places it in a storage area associated with the issuing program. If the queue record is larger than the designated storage area, the record is truncated. CA-IDMS automatically deletes the retrieved record from the queue unless the GET QUEUE statement explicitly requests to keep the record in the queue.

5.45.1 Syntax

```
►►── GET QUEUE ─┬──────────────────┬─┬──────────────────────────────────────────┬──►
                └─ ID queue-id ────┘ │┌─ NEXT ◄ ──────────────────────────┐│
                                     ├│─ FIRST ───────────────────────────│
                                     ├│─ LAST ────────────────────────────│
                                     ├│─ PRIOR ───────────────────────────│
                                     ├│─ SEQUENCE number ─────────────────│
                                     │└─ RECORD ID queue-record-id ───────┘│

►──┬─ DELETE ◄ ─┬─┬─ LOCK ◄ ─┬─┬─ WAIT ─────┬──────────────────────────────────►
   └─ KEEP ─────┘ └─ NOLOCK ─┘ └─ NOWAIT ◄ ─┘

►── INTO return-queue-data-location ─┬─ TO end-queue-data-location ──────────┬──►
                                     └─ MAX LENGTH queue-data-max-length ────┘

►──┬──────────────────────────────────────────────────────────────────── . ──►◄
   └─ RETURN LENGTH INTO queue-data-actual-length ──────────────────────────┘
```

5.45.2 Parameters

ID

Specifies the queue associated with the record to be retrieved.

queue-id

Either the symbolic name of a user-defined field that contains the ID or the ID itself enclosed in quotation marks. If the queue ID is not specified, a null ID of 16 blanks is assumed.

NEXT

Retrieves the next record in the queue.

This is the default.

If currency has not been established, NEXT is equivalent to FIRST.

FIRST

Retrieves the first record in the queue.

LAST

Retrieves the last record in the queue.

PRIOR

Retrieves the prior record in the queue. If currency has not been established, PRIOR is equivalent to LAST.

SEQUENCE

Retrieves the specified queue record.

number

Either the symbolic name of a user-defined field that contains the sequence number of the record or the sequence number itself expressed as a numeric constant.

RECORD ID

Retrieves the specified record.

queue-record-id

The symbolic name of the PIC S9(8) COMP (fullword) field that contains the queue record ID returned by the PUT QUEUE function.

DELETE

Deletes the record from the queue.

This is the default.

If DELETE is specified and the record has been truncated, the truncated data is lost.

KEEP

Keeps the record in the queue.

LOCK

Retains the lock on the current queue record until either a COMMIT TASK command is issued or the issuing task terminates.

This is the default.

While a queue record is locked, no other task can access that record (regardless of its position in the queue) until the lock has been released.

NOLOCK

Releases the lock on the current queue record following the execution of a subsequent queue I/O request.

WAIT

Suspends task execution until the requested queue exists.

NOWAIT

Continues task execution in the event of a nonexistent queue.

This is the default.

An ERROR-STATUS value of 4405 (DC-REC-NOT-FOUND) indicates that the requested queue record cannot be found.

INTO

Indicates the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the data area reserved for the requested queue record.

return-queue-data-location

The symbolic name of a user-defined field.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry reserved for the requested queue record.

end-queue-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the requested queue record.

MAX LENGTH

Explicitly defines the length of the data area reserved for the requested queue record.

queue-data-max-length

Either the symbolic name of the user-defined field that contains the length of the queue records data or the length itself expressed as a numeric constant.

RETURN LENGTH INTO

Specifies the location to which CA-IDMS is to return the actual length of the retrieved queue record.

queue-data-actual-length

The symbolic name of a user-defined four-byte field. If the record has been truncated, the value returned to this field is the actual length of the queue record before truncation.

5.45.3 Example

The following example requests CA-IDMS to retrieve the first record in the RES-Q queue, return it to the PEND-RES field, and keep the record in the queue:

```
GET QUEUE
   ID 'RES-Q'
   FIRST
   KEEP
   INTO PEND-RES MAX LENGTH 125.
```

5.45.4 Status codes

After completion of the GET QUEUE function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4404	The requested queue header record cannot be found.
4405	The requested queue record cannot be found.
4407	An I/O error has occurred during processing.

Status code	Meaning
4419	The program storage area specified for return of the queue record is too small; the returned record has been truncated as appropriate to fit the available space. In a DC-BATCH environment, a possible cause is that the size of the queue record exceeds the value specified in the MAX LENGTH parameter of the BIND TASK statement. In a DC-BATCH environment, a possible cause is that the size of the record read by a GET QUEUE statement exceeds the value specified in the max length parameter of the BIND TASK statement. This can also happen if the record size specified in the GET QUEUE statement is large enough for the queue record, but the maximum specified in the BIND TASK statement is too small. The record size is always truncated to the maximum length determined in the BIND TASK statement.
4431	The parameter list is invalid. In DC-BATCH, this code signifies that the specified record length has exceeded the maximum length based on the packet size.
4432	The derived length of the queue record data area is negative.

5.46 GET SCRATCH

The GET SCRATCH statement obtains a scratch record and places it in a storage area associated with the issuing program. The storage area must already be allocated to the requesting task; no implicit GET STORAGE function is performed during the GET SCRATCH operation. If the scratch record is larger than the designated storage area, data is truncated.

5.46.1 Syntax

```
►►── GET SCRATCH ┬──────────────────────────────┬──────────────────────────►
                └─ AREA ID scratch-area-id ─┘

►─┬──────────────────────────┬─┬──────────┬──────────────────────────────►
  ├─ NEXT ◄ ─────────────────┤ ├─ DELETE ◄ ┤
  ├─ FIRST ──────────────────┤ └─ KEEP ────┘
  ├─ LAST ───────────────────┤
  ├─ PRIOR ──────────────────┤
  ├─ CURRENT ────────────────┤
  └─ RECORD ID scratch-record-id ┘

►── INTO return-scratch-data-loc ┬─ TO end-scratch-data-loc ──────────┬──►
                                 └─ MAX LENGTH scratch-data-max-length ┘

►─┬──────────────────────────────────────────────┬─ . ──────────────────►◄
  └─ RETURN LENGTH into scratch-data-actual-length ┘
```

5.46.2 Parameters

AREA ID

Identifies the scratch area associated with the record being retrieved. If you do not specify an area ID, CA-IDMS assumes an area ID of eight blanks.

scratch-area-id

Either the symbolic name of a user-defined field that contains the scratch area ID or the ID itself enclosed in quotation marks.

NEXT

Retrieves the next record in the scratch area.

This is the default.

FIRST

Retrieves the first record in the scratch area.

LAST

Retrieves the last record in the scratch area.

PRIOR

Retrieves the prior record in the scratch area.

CURRENT

Retrieves the current record in the scratch area; the current record is the record most recently referenced by another scratch function.

RECORD ID

Retrieves the specified scratch record.

scratch-record-id

The symbolic name of a user-defined PIC S9(8) COMP SYNC (fullword) field that contains the four-byte scratch record ID.

DELETE

Deletes the record from the scratch area.

This is the default.

If DELETE is specified and the record has been truncated, the truncated data is lost. To maintain currency following a DELETE request, CA-IDMS saves the next and prior currencies of the scratch area.

KEEP

Keeps the record in the scratch area.

INTO

Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the data area to which CA-IDMS is to return the scratch record.

return-scratch-data-loc

The symbolic name of a user-defined field.

TO

Indicates the end of the data area to which CA-IDMS will return the scratch record.

end-scratch-data-loc

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the scratch record.

MAX LENGTH

Specifies the length in bytes of the data area associated with the requested scratch record.

scratch-data-max-length

Either the symbolic name of a WORKING-STORAGE SECTION or LINKAGE SECTION field that contains the length or the length itself expressed as a numeric constant.

RETURN LENGTH into

Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry to which CA-IDMS will return the actual length of the requested scratch record.

scratch-data-actual-length

The symbolic name of the entry. If the record has been truncated, *scratch-data-actual-length* will contain the length of the full, untruncated scratch record.

5.46.3 Example

The following statement requests CA-IDMS to return the contents of the current record in the scratch area to the variable-storage area defined by WORK-PROC-AREA and END-WORK-PROC-AREA:

```
GET SCRATCH
   CURRENT
   INTO WORK-PROC-AREA TO END-WORK-PROC-AREA.
```

5.46.4 Status codes

After completion of the GET SCRATCH function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4303	The requested scratch area ID cannot be found
4305	The requested scratch record ID cannot be found
4307	An I/O error has occurred during processing
4319	The program storage area specified for return of the scratch record is too small; the returned record has been truncated to fit the available space
4331	The parameter list is invalid
4332	The derived length of the scratch record is negative

5.47 GET STORAGE

The GET STORAGE statement is used either to acquire variable storage from a DC system storage pool or to obtain the address of a previously acquired storage area. Once acquired, the storage is available for use:

- By the issuing task only (user storage)
- By subsequent tasks running on the same terminal (user kept storage)
- By all tasks in the system (shared or shared kept storage)

Storage availability is governed by GET STORAGE parameter specifications.

5.47.1 Syntax

```
►►── GET STORAGE FOR 01-level-storage-data-location ──────────────────►
►┬──────────────────────────────────────────────────────────────────►
 └─ TO end-storage-data-location ─┘
►┬──────────────────────────────────────────────────────────────────►
 └─ LENGTH storage-data-length ─┘
►┬──────────────────────────────────────────────────────────────────►
 └─ POINTER storage-data-location-pointer ─┘
►┬──────────────────────────────────────────────────────────────────►
 ├─ WAIT ──┤ └─ KEEP ─┘ ├─ LONG ◄ ─┤ ├─ USER ◄ ─┤
 └─ NOWAIT ─┘           └─ SHORT ─┘ └─ SHARED ─┘
►┬──────────────────────────────────────────────────────────────────►
 └─ STGID storage-id ─┘ └─ VALUE IS ┬─ LOW-VALUE ──────┤
                                    ├─ HIGH-VALUE ─────┤
                                    └─ initial-value ──┘
►┬──────────────────────────────────────────────────────────────────►◄
 └─ LOCATION is ──┬──────────┬─┘ . ──
                  ├─ ANY ◄ ─┤
                  └─ BELOW ─┘
```

5.47.2 Parameters

FOR

Specifies the LINKAGE SECTION entry of the storage area to be acquired.

01-level-storage-data-location

The name of the 01-level entry used to acquire the storage area.

TO

Specifies the end of the storage area. This parameter is required when the precompiler execution option COBOL=1 is specified. It is accepted but not required if COBOL=2 is specified. See A.8, “Passing Parameters to the Precompiler” on page A-23 for more information on the COBOL= option.

end-storage-data-location

The symbolic name of a user-defined dummy byte field. It is specified as a subordinate item within the 01-level record area following the last real data field.

Note: CA-IDMS does not support the use of an OCCURS DEPENDING ON clause within 01-level-storage-data-location.

LENGTH

Specifies the length of the storage location (COBOL 85 only).

storage-data-length

The symbolic name of a user-defined field that contains the length of the storage location.

POINTER

Specifies a pointer that is to receive the address of the storage location (COBOL 85 only).

storage-data-location-pointer

The symbolic name of a user-defined field that contains a pointer to the address of the storage location.

WAIT

Specifies that the issuing task will wait until sufficient storage is available in a storage pool.

This is the default.

NOWAIT

Specifies that the issuing task will not wait for storage to become available if an insufficient storage condition exists. If NOWAIT is specified, the program should check the ERROR-STATUS field in the IDMS-DC communications block to determine if the GET STORAGE request has been completed. If the ERROR-STATUS value is 3202 (DC-NO-STORAGE), the program should perform alternative processing before reissuing the GET STORAGE request.

KEEP

Optionally specifies whether the storage area will be used by subsequent tasks executing on the same logical terminal. When KEEP is specified, the storage area can be accessed by subsequent tasks; otherwise the storage area cannot be accessed by subsequent tasks. For a more detailed discussion of the KEEP parameter, refer to *CA-IDMS Navigational DML Programming*.

LONG

Allocates storage from the bottom of the storage pool.

This is the default.

You should specify LONG when allocating kept storage to be held across pseudo-converses.

An incorrect LONG/SHORT specification will not affect normal program execution; however, it may affect the overall performance of the DC system.

SHORT

Allocates storage from the top of the storage pool. You should specify SHORT when allocating small pieces of storage for a short duration.

An incorrect LONG/SHORT specification will not affect normal program execution; however, it may affect the overall performance of the DC system.

USER

Specifies that *only* the issuing task can access the storage area or, if KEEP is specified, only subsequent tasks executing on the same terminal.

This is the default.

Note: During system execution, a program defined at sysgen with the NOPROTECT option can access any storage area within the system, including an area associated exclusively with another task. Thus, the USER attribute may not protect the storage area being acquired. However, storage areas can be protected on a system-wide or program-by-program basis during system generation and by the modes specified when storage is allocated.

SHARED

Specifies that any task in the system can access and modify the acquired storage. Each task must establish addressability to the storage area by explicitly issuing a GET STORAGE request.

STGID

Specifies storage area. The STGID parameter must be specified with GET STORAGE requests for either previously allocated storage areas or areas to be reallocated.

storage-id

Either the symbolic name of a user-defined field that contains the storage ID or the ID itself enclosed in quotation marks.

The specified storage ID must be unique; although multiple variable storage areas (that is, one shared and the others user) can have the same ID, only one such area can be owned by a given task at a time. To access the CA-IDMS common work area, specify STGID 'CWA'.

Note: If the STGID parameter specifies the address of an existing storage area, the USER/SHARED parameter must specify the same option as that specified in the GET STORAGE statement that originally allocated the storage area.

VALUE IS

Specifies how the storage area is to be initialized.

LOW-VALUE

Initializes the storage area to all zeros.

HIGH-VALUE

Initializes the storage area to the highest value in the computer collating sequence.

initial-value

Either the symbolic name of a user-defined field that contains the initial value or the value itself enclosed in quotation marks. All bytes of the acquired storage area are initialized to the same value.

LOCATION is

Specifies whether the storage is to be restricted to below the 16-megabyte line or if space above the 16-megabyte line is also eligible.

ANY

Specifies that space above the 16-megabyte line is eligible for allocation.

This is the default.

BELOW

Specifies that storage must be allocated from below the 16-megabyte line.

5.47.3 Example

The following statement requests CA-IDMS to allocate the shared kept storage area, 09PA, and initialize it to all zeros:

```
GET STORAGE FOR EMPLMENU-KEPT-STORAGE TO
   EMPLMENU-KEPT-STORAGE-END
   NOWAIT
   KEEP
   SHORT
   SHARED
   STGID '09PA'
   VALUE IS LOW-VALUE.
```

5.47.4 Status codes

After completion of the GET STORAGE function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3201	The requested storage cannot be allocated immediately; to wait would cause a deadlock
3202	The requested storage cannot be allocated because insufficient space exists in the storage pool
3210	The request specified a storage ID that did not previously exist; the required space has been allocated
3231	The request specifies an invalid parameter list
3232	The requested length is zero or negative
3234	The request cannot be serviced because the variable storage area is not an 01-level LINKAGE SECTION variable

Status code	Meaning
3235	The request cannot be serviced because the specified 01-level LINKAGE SECTION entry has either been previously allocated or contains an OCCURS DEPENDING ON clause

5.48 GET TIME

The GET TIME statement obtains the time of day and date from the operating system. The system time is returned to the issuing task in either fixed binary, packed decimal, or edited format. The date is returned to the program in packed decimal format.

5.48.1 Syntax

```
►►── GET TIME ──┬──────────────────────────────────────────────────────►
                └─ INTO return-time ─┬─ COMP ◄ ─┐
                                     ├─ COMP-3 ─┤
                                     └─ EDIT ───┘

►──┬──────────────────────────┐ . ──────────────────────────────────►◄
   └─ DATE INTO return-date ─┘
```

5.48.2 Parameters

INTO

Specifies the field to which CA-IDMS is to return the time.

return-time

The symbolic name of a user-defined field to which the current time will be returned. The following format options apply:

COMP

Returns the time as a fixed binary value representing the elapsed time since midnight in ten-thousandths of a second.

This is the default.

If COMP is specified, the field associated with *return-time* should be a PIC S9(8) COMP SYNC (fullword) field. The COMP option returns the most precise time.

COMP-3

Returns the time as a six-byte packed decimal value in the format *0hhmmsstttc* (padded zero, hours, minutes, seconds, ten-thousandths of a second, sign). If COMP-3 is specified, the field associated with *return-time* should be defined as PIC S9(11) COMP-3.

EDIT

Returns the time as an edited character string in the format *hh:mm:ss:hh* (hours, minutes, seconds, hundredths of a second). If EDIT is specified, *return-time* should be defined as PIC X(11) DISPLAY.

DATE INTO

Specifies the field to which CA-IDMS is to return the data obtained from the operating system.

return-date

The symbolic name of a user-defined COMP-3 PIC S9(7) field. The date is returned in the Julian format *0yyydddc* (padded zero, current year relative to

1900, date, sign). For example, 0099365C would represent December 31, 1999. 0100001C would represent January 1, 2000.

5.48.3 Example

The following statement requests CA-IDMS to return the current time and date to the CURRENT-TIME and CURRENT-DATE fields, respectively:

```
GET TIME
   INTO CURRENT-TIME EDIT
   DATE INTO CURRENT-DATE.
```

5.48.4 Status codes

After completion of the GET TIME function, the only possible value in the ERROR-STATUS field of the IDMS-DC communications block is 0000.

5.49 IF

The IF statement allows the program to test for the presence of member record occurrences in a set and to determine the membership status of a record occurrence in a specified set; once the set has been evaluated, the IF statement specifies further action based on the outcome of the evaluation. For example, an IF statement might be used to determine whether a set occurrence is empty and, if it is empty, to erase the owner record.

Note: DML IF statements cannot be nested within COBOL IF statements.

Native VSAM users: The IF statement is not valid for sets defined with member records that are stored in native VSAM data sets.

Depending on its format, the IF statement uses set or run-unit currency. The object set occurrence of an IF statement is determined by the owner of the current record of the named set; the object record occurrence is determined by the current of run unit.

Each IF statement contains a conditional phrase and an imperative statement. When an IF is issued, the precompiler first generates a call to the DBMS to execute the conditional phrase; the results of the test determine whether or not the imperative statement is executed.

Note: If AUTOSTATUS is in use (see 3.5.1, “AUTOSTATUS protocols” on page 3-33), IDMS-STATUS is *not* performed automatically when an IF statement is issued.

5.49.1 Syntax

```
►►─┬─ IF set-name is ──┬──────┬── EMPTY ──┬─ imperative-statement . ──────────►◄
   │                   └ NOT ┘           │
   └─ IF ──┬──────┬── set-name MEMBER ────┘
           └ NOT ┘
```

5.49.2 Parameters

set-name

Specifies the set whose owner should be examined for the presence of member record occurrences.

The specified set must be included in the subschema.

EMPTY

Specifies that the imperative statement be executed if the current occurrence of the named set is empty.

NOT EMPTY

Specifies that the imperative statement be executed if the current occurrence of the named set is **not** empty.

MEMBER
Specifies that the imperative statement be executed if the current record of the run unit is a member of any occurrence of the specified set.

NOT set-name MEMBER
Specifies that the imperative statement be executed if the current record of the run unit is **not** a member of any occurrence of the specified set.

imperative-statement
Identifies the action to execute if the specified condition is true.

5.49.3 Examples

The examples below illustrate the use of the IF statement.

Example 1: The following statement tests the COVERAGE-CLAIMS set for existing CLAIMS members and, if no occurrences of the CLAIMS record are found (ERROR-STATUS is 0000), moves a message to that effect to location CLAIMS-WS.

```
IF COVERAGE-CLAIMS IS EMPTY MOVE 'NONE' TO CLAIMS-WS.
```

If the current occurrence of the COVERAGE-CLAIMS set contains one or more occurrences of the CLAIMS record (ERROR-STATUS is 1601), the MOVE statement is ignored and the next statement in the program is executed.

Example 2: The following statement verifies that the EMPLOYEE record that is current of run unit is not a member of the current occurrence of the OFFICE-EMPLOYEE set before code is executed to connect the EMPLOYEE record to that set.

```
IF NOT OFFICE-EMPLOYEE MEMBER PERFORM LINK-SET.
```

If the EMPLOYEE record is not a member of the OFFICE-EMPLOYEE set (ERROR-STATUS is 1601), the program performs the LINK-SET paragraph. If the EMPLOYEE record is already a member of the OFFICE-EMPLOYEE set (ERROR-STATUS is 0000), the PERFORM statement is ignored and the next statement in the program is executed.

5.49.4 Status codes

After completion of the IF function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	Either the set is empty or the record that is current of run unit is a member of the set
1601	Either the set is not empty or the record that is current of run unit is not a member of the set

Status code	Meaning
1606	Currency has not been established for the named set
1608	Either an invalid set name has been specified or the current record of run unit is not a member of the named set
1613	A current record of run unit either has not been established or has been nullified by a preceding ERASE statement

5.50 INQUIRE MAP

The INQUIRE MAP statement is used after a map input request to accomplish one of the following actions related to the input operation:

- Move map-related information into variable storage
- Test for conditions relating to global map input operations
- Test specific map fields for the presence of the cursor
- Test for conditions relating to specific map fields

Each of these actions is discussed on the following pages.

The following rules apply to INQUIRE MAP statements:

- If any of the test conditions are requested, INQUIRE MAP must specify a statement that will be executed if the condition is found to be true.
- An INQUIRE MAP statement can specify only one field-oriented inquiry. This inquiry can be specified alone or with a map-specific inquiry.

Moving map-related data: This version of the INQUIRE MAP statement moves one of the following map-related data items into variable storage:

- The attention ID (AID) key used
- The current cursor position (row and column)
- The entered length of a specific map input field

5.50.1 Syntax

```
►►── INQUIRE MAP map-name ──────────────────────────────────────►
 ►── MOVE ─┬─ AID TO aid-indicator ──────────────────────┬─ . ──────►◄
        ├─ CURSOR TO cursor-row cursor-column ────────┤
        └─ IN LENGTH FOR field-name TO field-length ──┘
```

5.50.2 Parameters

map-name
Specifies the map for which the inquiry is to be made. The specified map must be included in the program's MAP SECTION.

MOVE
Move screen-related information to program variable storage.

AID TO
Return the attention ID to the specified location in variable storage.

aid-indicator

The symbolic name of a one-byte user-defined field that will be set to the 3270 AID character received in the last map input request. The table below lists the AID characters associated with each 3270-type control key.

Note: The data dictionary includes a record that defines the AID character values as level-88 items to test for particular keyed input by including a COPY IDMS DC-AID-CONDITION-NAMES statement in the WORKING-STORAGE SECTION.

CURSOR TO

Returns the cursor address from the last map input function to the specified location in program variable storage.

cursor-row cursor-column

The symbolic names of user-defined PIC S9(4) COMP fields to which the row and column cursor address will be returned.

IN LENGTH FOR

Requests CA-IDMS to return the length in bytes of the data in the specified map field.

field-name

The name of the map field for which the length is being requested.

TO

Tells CA-IDMS where to return the length of the field.

field-length

The symbolic name of a user-defined PIC S9(4) COMP field.

Attention ID (AID) key values

Key	AID Character	Key	AID Character
ENTER	"' (single quote)	PF14	'B'
CLEAR	'_' (underscore)	PF15	'C'
PF01	'1'	PF16	'D'
PF02	'2'	PF17	'E'
PF03	'3'	PF18	'F'
PF04	'4'	PF19	'G'
PF05	'5'	PF20	'H'
PF06	'6'	PF21	'I'
PF07	'7'	PF22	'¢'
PF08	'8'	PF23	'.'
PF09	'9'	PF24	'<'
PF10	':'	PA01	'%'
PF11	'#'	PA02	'>'
PF12	'@'	PA03	','
PF13	'A'		

The following figure shows the code copied into the program as a result of the COPY IDMS DC-AID-CONDITION-NAMES specification.

```
*01  COPY IDMS DC-AID-CONDITION-NAMES.
 01  DC-AID-CONDITION-NAMES.
      03  DC-AID-IND-V           PIC X.
                             88  ENTER-HIT VALUE QUOTE.
                             88  CLEAR-HIT VALUE '_'.
                             88  PF01-HIT VALUE '1'.
                             88  PF02-HIT VALUE '2'.
                             88  PF03-HIT VALUE '3'.
                             88  PF04-HIT VALUE '4'.
                             88  PF05-HIT VALUE '5'.
                             88  PF06-HIT VALUE '6'.
                             88  PF07-HIT VALUE '7'.
                             88  PF08-HIT VALUE '8'.
                             88  PF09-HIT VALUE '9'.
                             88  PF10-HIT VALUE ':'.
                             88  PF11-HIT VALUE '#'.
                             88  PF12-HIT VALUE '@'.
                             88  PF13-HIT VALUE 'A'.
                             88  PF14-HIT VALUE 'B'.
                             88  PF15-HIT VALUE 'C'.
                             88  PF16-HIT VALUE 'D'.
                             88  PF17-HIT VALUE 'E'.
                             88  PF18-HIT VALUE 'F'.
                             88  PF19-HIT VALUE 'G'.
                             88  PF20-HIT VALUE 'H'.
                             88  PF21-HIT VALUE 'I'.
                             88  PF22-HIT VALUE '¢'.
                             88  PF23-HIT VALUE '.'.
                             88  PF24-HIT VALUE '<'.
                             88  PA01-HIT VALUE '%'.
                             88  PA02-HIT VALUE '>'.
                             88  PA03-HIT VALUE ','.
                             88  PEN-ATTN-SPACE-NULL VALUE '='.
                             88  PEN-ATTN VALUE QUOTE.
```

5.50.3 Example

The following example illustrates the use of an INQUIRE MAP statement to move the 3270 AID character received in the last map input request to DC-AID-IND-V. If the AID character indicates that PF1 was pressed, the program performs a DC RETURN.

```
INQUIRE MAP EMPMAPLR
   MOVE AID TO DC-AID-IND-V.
IF CLEAR-HIT
   DC RETURN.
```

Testing for global map input conditions: This version of the INQUIRE MAP statement tests for one of the following global map input conditions:

- If the screen was not formatted before the input operation was performed
- If one or more input fields were truncated when transferred to variable storage data fields
- If one or more input fields were modified on the screen before being transferred
- If one or more fields that were modified on the screen are undefined in the map being used

5.50.4 Syntax

```
►►── INQUIRE MAP map-name ──────────────────────────────────────────►

►── IF INPUT ─┬─ UNFORMATTED ─┬─ THEN ─┬─↓─ imperative-statement ─┬─ . ──────►◄
             ├─ TRUNCATED ───┤        │                          │
             ├─ CHANGED ─────┤        │                          │
             └─ EXTRANEOUS ──┘        └──────────────────────────┘
```

5.50.5 Parameters

map-name

The name of the map for which the inquiry is being made. The map must be included in the program's MAP SECTION.

IF INPUT

Tests the outcome of the last map input request for conditions relating to the data input to the program.

UNFORMATTED

Tests whether the screen had been formatted before the input operation was performed.

TRUNCATED

Tests whether any of the map fields were truncated when transferred to variable-storage data fields.

CHANGED

Tests whether any of the map fields actually had been mapped to variable-storage data fields when the map input operation was performed.

EXTRANEOUS

Tests whether the input data stream contained any data from a field not defined to the map. If this condition is true, the undefined data field is ignored by CA-IDMS.

THEN

Specifies the action to be taken when the test condition is true.

imperative-statement

A COBOL statement, a DML statement, or a nested block of COBOL and/or DML statements.

5.50.6 Example

The following example illustrates an INQUIRE MAP statement that tests to determine if any fields in the EMPMAPLR map have been truncated and, if so, requests CA-IDMS to perform the DATA-TRUNC routine:

```
INQUIRE MAP EMPMAPLR
   IF INPUT TRUNCATED
     THEN PERFORM DATA-TRUNC.
```

Testing for cursor position: This version of the INQUIRE MAP statement tests a specified map field for the presence of the cursor.

5.50.7 Syntax

```
►►── INQUIRE MAP map-name ──────────────────────────────────────►
                                   ┌──────────────────────────┐
►── IF CURSOR at DFLD field-name THEN ─┬─ imperative-statement ─┴─ . ──►◄
                                       ↓
```

5.50.8 Parameters

map-name
The name of the map for which the inquiry is being made. The map must be included in the program's MAP SECTION.

IF CURSOR at DFLD
Determines whether the cursor was in the named map field during the last map input operation.

field-name
Identifies the field within the named map to be tested.

THEN
Specifies the action to be taken if the test condition is true.

imperative-statement
A COBOL statement, a DML statement, or a nested block of COBOL and/or DML statements.

5.50.9 Example

The following example illustrates an INQUIRE MAP statement that tests for the presence of the cursor in the PASSED-DATA-01 data field and, if so, performs the CHECK-2 routine:

```
INQUIRE MAP EMPMAPLR
     IF CURSOR AT DFLD EMP-LAST-NAME-0415
          THEN PERFORM CHECK-2.
```

Testing for input non-zero status conditions: This version of the INQUIRE MAP statement tests for the following input conditions relating to specific map fields:

- If map fields have been modified
- If map fields have been erased by operator action
- If map fields have been truncated
- If map fields are identical to map data currently in program variable storage
- If map fields are different from map data currently in program variable storage
- If the specified map fields are either in error (the error flag has been set on for those fields) or are correct (the error flag has been set off); this option applies only to those maps and map fields for which automatic editing is enabled

5.50.10 Syntax

```
►►── INQUIRE MAP map-name ──────────────────────────────────────────►
►── IF ─┬─┬─ CURRENT ──────────┬──────────────────────────────────────►
      │ │ ├─ ALL ──────────────┤
      │ │ ├─ NONE ─────────────┤
      │ │ ├─ ANY ──────────────┤
      │ │ ├─ SOME ─────────────┤
      │ │ └─ ALL BUT CURRENT ──┘
      │
      ├─┬─ ALL ◄ ──┬── DFLD field-name ─┤
      │ ├─ NONE ───┤
      │ ├─ ANY ────┤
      │ ├─ SOME ───┤
      │ └─ ALL BUT ─┘
      │
      └─ DFLD field-name ──────────────┘
►─┬─ DATA is ─┬─ YES ──────────┬─┬──────────────────────────────────►
  │           ├─ NO ───────────┤ │
  │           ├─ ERASE ────────┤ │
  │           ├─ TRUNCATED ────┤ │
  │           ├─ IDENTICAL ────┤ │
  │           └─ DIFFERENT ────┘ │
  │                              │
  └─ EDIT is ─┬─ ERROR ──────────┤
              └─ CORRECT ────────┘
►── THEN ─┬─↓─ imperative-statement ─┬─ . ──────────────────────────►◄
         └──────────────────────────┘
```

5.50.11 Parameters

map-name

Specifies the map for which the inquiry is being made. The map must be included in the program's MAP SECTION.

IF

Specifies the map fields to which the test applies.

CURRENT

Applies the test only to the current field; that is, the map field that was referenced in the last MODIFY MAP or INQUIRE MAP statement issued by the program. If the last MODIFY MAP or INQUIRE MAP statement specified a field list, no currency exists.

ALL

Specifies that the test is true if all map fields meet the specified condition.

NONE

Specifies that the test is true if none of the map fields meet the specified condition.

ANY

Specifies that the test is true if one or more of the map fields meet the specified condition.

SOME
Specifies that the test is true if one or more **but not all** of the map fields meet the specified condition.

ALL BUT CURRENT
Specifies that the test is true if all map fields except the current field meet the specified condition.

IF
Specifies the extent to which the condition applies to the map field.

ALL
Specifies that the test is true if all of the named map fields meet the specified condition.

NONE
Specifies that the test is true if none of the named map fields meet the specified condition.

ANY
Specifies that the test is true if one or more of the named map fields meet the specified condition.

SOME
Specifies that the test is true if one or more but not all of the named map fields meet the specified condition.

ALL BUT
Specifies that the test is true if all map fields except for the named field meet the specified condition.

DFLD
Specifies the individual map fields to which the test conditions apply.

Multiple DFLD specifications must be separated by at least one blank.

field-name
The name of a field within the named map.

DFLD field-name
Specifies the individual map field(s) to which the test condition applies. The specified field(s) must exist within the named map.

Multiple DFLD specifications must be separated by at least one blank.

DATA IS
Tests the input data in the specified map field(s).

YES
Determines if the terminal operator entered data in the specified map field(s).

NO
Determines if the terminal operator did not enter data in the specified map field(s).

ERASE
Determines if data has been erased from the specified map field(s).

TRUNCATED
Determines if data has been truncated in the specified map field(s).

IDENTICAL
Tests whether input data is identical to map data currently in program variable storage.

IDENTICAL is true in either of the following cases:

- The field's modified data tag (MDT) is off. On mapin, the MDT is usually off if the user did not type any characters in the field.
- The field's MDT is on, but each character that the user typed in is identical (including capitalization) to the data in variable storage.

DIFFERENT
Tests whether input data is different from map data currently in program variable storage.

DIFFERENT is true if the field's MDT is both:

- on

  and

- at least one input character differs from the data in variable storage.

EDIT IS
Tests for errors in the named map field(s).

If the EDIT parameter is specified, automatic editing must be enabled for the map and for each of the named map fields.

ERROR
Determines if the named map field(s) were found to be in error during automatic editing.

CORRECT
Determines if the named map field(s) were found to be correct during automatic editing.

THEN
Specifies the action to be taken when the test condition is true.

imperative-statement
A single COBOL statement, a DML statement, or a nested block of COBOL and/or DML statements.

5.50.12 Examples

The examples below illustrate the use of the INQUIRE MAP statement.

Example 1 — Testing for erroneous data: The following example determines if automatic editing has detected erroneous data in any field in the EMPMAPLR map; if so, the program modifies the map temporarily to display the erroneous fields with the bright and blinking attributes:

```
INQUIRE MAP EMPMAPLR
   IF ANY EDIT IS ERROR
      THEN MODIFY MAP EMPMAPLR TEMPORARY
           FOR ALL ERROR FIELDS
           ATTRIBUTES BRIGHT BLINK.
```

Example 2 — Testing for identical data: Use an INQUIRE MAP statement to test whether the user has entered an employee ID number:

- If the IDENTICAL condition is *true* (the user doesn't specify a different ID number), the program displays the menu screen.
- If the IDENTICAL condition is *false* (the user specifies a different ID number), the program obtains the corresponding employee record from the database.

The sample INQUIRE MAP statement is:

```
INQUIRE MAP MAP01
   IF DFLD EMP-ID-0415 DATA IS IDENTICAL THEN
      PERFORM EMP-PROMPT-20
   ELSE
      PERFORM EMP-OBTAIN-20.
```

Example 3 — Testing for changed data: Use an INQUIRE MAP statement to test whether the user has entered a new department ID or department name. If the user has changed either value (DIFFERENT is *true*), the program branches to DEPTUP-30:

```
INQUIRE MAP MAP02
   IF ANY DFLD DEPT-ID-0410
          DFLD DEPT-NAME-0410 DATA IS DIFFERENT
   THEN PERFORM DEPTUP-30.
```

5.50.13 Status codes

After completion of the INQUIRE MAP function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4629	An invalid parameter has been passed from the program.
4641	The test condition has been found to be true. (This condition is tested for automatically by COBOL DML expansion statements.)
4644	The referenced map field is not in the specified map; a possible cause is a reference to an invalid map field subscript.
4656	The referenced map contains no data fields.

5.51 KEEP CURRENT

The KEEP CURRENT statement places an explicit shared or exclusive lock on a record that is current of run unit, record, set, or area. Locks placed on records through the KEEP function are maintained either for the duration of the recovery unit or until explicitly released by means of the COMMIT or FINISH statements.

5.51.1 Syntax

```
►►── KEEP ─┬──────────────┬─ CURRENT ─┬──────────────────────┬─ . ──────────►◄
         └─ EXCLUSIVE ─┘           ├─ record-name ─────────┤
                                   ├─ WITHIN set-name ──────┤
                                   └─ WITHIN area-name ─────┘
```

5.51.2 Parameters

EXCLUSIVE
Specifies that the current record of run unit receive an exclusive lock. By default, if you do not specify EXCLUSIVE, the current record of run unit will receive a shared look.

record-name
Specifies that the lock will be placed on the current record of the specified record type.

WITHIN set-name
Specifies that the lock will be placed on the current record of the specified set.

WITHIN area-name
Specifies that the lock will be placed on the current record of the specified area.

5.51.3 Example

The following example places a shared lock on the current EMPLOYEE record occurrence:

```
KEEP CURRENT EMPLOYEE.
```

5.51.4 Status codes

After completion of the KEEP function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
0606	Currency has not been established for the named record, set, or area

Status code	Meaning
0608	Either the named record or set is not in the subschema or the current record of run unit is not a member of the named set
0610	The program's subschema specifies an access restriction that prohibits execution of the KEEP function
0623	The named area is not in the subschema
0626	The record to be kept has been erased
0629	A deadlock has occurred while attempting to set the lock.

5.52 KEEP LONGTERM

The KEEP LONGTERM statement establishes longterm record locks and/or monitors access to records between tasks. Longterm database locks are used in pseudo-conversational transactions and can be shared or exclusive:

- **Longterm shared locks** allow other run units to access the locked record but prevent run units from updating the record as long as the lock is maintained.
- **Longterm exclusive locks** prevent other run units from accessing the locked record. However, run units executing on the logical terminal associated with the issuing task are not restricted from accessing the locked record. Therefore, subsequent tasks in a transaction can access the locked record and complete the database processing required by the transaction.

If a record has been locked with a KEEP LONGTERM or KEEP request, restrictions exist on the type of lock that can be placed on that record by other run units. These restrictions are based on existing locks and whether the requesting run unit is executing on the same logical terminal as the run unit that originally placed the lock on the record. The following table illustrates these restrictions.

Keep longterm record lock options

Locks in effect	Locks allowed for other run units	Locks disallowed for other run units
Shared	▪ shared ▪ longterm shared	▪ exclusive ▪ longterm exclusive
Exclusive	None	▪ shared ▪ exclusive ▪ longterm shared ▪ longterm exclusive
Longterm shared	▪ For all run units: – shared – longterm shared ▪ For run units on the same terminal: – exclusive – longterm exclusive	For run units on other terminals: ▪ exclusive ▪ longterm exclusive

Locks in effect	Locks allowed for other run units	Locks disallowed for other run units
Longterm exclusive	For run units on the same terminal: ■ shared ■ exclusive ■ longterm shared ■ longterm exclusive	For run units on other terminals: ■ shared ■ exclusive ■ longterm shared ■ longterm exclusive

Tasks can monitor database activity associated with a specified record during a pseudo-converse and, if desired, can place a longterm lock on the record being monitored. A subsequent task can then make inquiries about that database activity for the record and take the appropriate action.

CA-IDMS maintains information on database activity by using five bit flags, each of which is either turned on (binary 1) or turned off (binary 0). This information is returned to the program as a numeric value. The bit assignments, the corresponding numeric value returned to the program, and a description of the associated database activity follow:

Numeric Value	Bit Assignment	Description
16	X'00000010'	The record was physically deleted
8	X'00000008'	The record was logically deleted
4	X'00000004'	The record's prefix was modified; that is, a set operation (for example, CONNECT or DISCONNECT) occurred involving the record
2	X'00000002'	The record's data was modified
1	X'00000001'	The record was obtained

To determine the action or combination of actions that has occurred, you can compare the numeric value returned to the program with an appropriate constant. For example:

- If the returned value is zero, no database activity occurred for the specified record.
- If the returned value is two, the record's data was modified.
- If the returned value is two or greater, the record was altered in some way.
- If the returned value is eight or greater, the record was deleted.

The maximum possible value is 31, indicating that all the above actions occurred for the specified record.

You may prefer to monitor database activity across a pseudo-converse rather than to set longterm locks. Monitoring does not restrict access to database records, sets, or areas by other run units; however, it does enable a program to test a record for alterations made by other run units. The presence of longterm locks can prevent other run units from accessing locked records for an undesirable amount of time if, during a pseudo-converse, the terminal operator fails to enter a response. If longterm locks are used, you may want to release them at specified intervals.

For more information regarding the use of timeout intervals, refer to *CA-IDMS System Generation*.

5.52.1 Syntax

```
►►── KEEP LONGTERM ─┬─ ALL ──────────────┬──────────────────────────────►
                    └─ longterm-id ──────┘

►─┬─ NOTIFY CURRENT ─┬─ record-name ──┬──────────────┬─ . ──────────────►◄
  │                  ├─ set-name ─────┤              │
  │                  └─ area-name ────┘              │
  │                                                  │
  ├─ lock-options ───────────────────────────────────┤
  │                                                  │
  ├─ TEST ─┬──────────────────────────────────────── │
  │        └─ RETURN NOTIFICATION into location ────┤
  │                                                  │
  └─ RELEASE ────────────────────────────────────────┘
```

Expansion of lock-options

```
►─┬─┬─ SHARE ──────┬─ CURRENT ─┬─ record-name ──┬──────────────────────►
  │ │               │           ├─ set-name ─────┤                      │
  │ └─ EXCLUSIVE ───┘           └─ area-name ────┘                      │
  │                                                                     │
  └─ UPGRADE ─┬─ SHARE ──────┬──┬──────────────────────────────────────┤
              └─ EXCLUSIVE ──┘  └─ RETURN-NOTIFICATION into location ──┘

►─┬──────────────────────────────────────────────────────────────────►
  ├─ WAIT ◄ ──────┤
  ├─ NOWAIT ──────┤
  └─ NODEADLOCK ──┘
```

5.52.2 Parameters

ALL

Used only with the RELEASE parameter, to release all longterm locks kept for the logical terminal associated with the current task.

longterm-id

Either the symbolic name of a user-defined field that contains the longterm ID or the ID itself enclosed in quotation marks. This ID will be used in any subsequent references to the lock, when it is changed or released.

NOTIFY CURRENT

Requests CA-IDMS to monitor database activity associated with the current record. When NOTIFY CURRENT is specified, CA-IDMS initializes a

preallocated location in the program to contain information on database activity for the specified record.

record-name
Monitors database activity associated with the current occurrence of *record-name*.

set-name
Monitors database activity associated with the record current of *set-name*.

area-name
Monitors database activity associated with the record current of *area-name*.

TEST RETURN NOTIFICATION into
Requests that CA-IDMS return information on database activity associated with the record identified by *longterm-id* to a previously allocated location in the program's storage.

The TEST request must specify a longterm lock ID that matches the longterm lock ID specified in a previous KEEP LONGTERM NOTIFY CURRENT request.

location
The symbolic name of a user-defined PIC S9(8) COMP (fullword) field that contains the WORKING-STORAGE or LINKAGE SECTION entry of the data area to which CA-IDMS will return the information.

RELEASE
Releases the longterm lock for the record identified by *longterm-id* or all record locks (ALL) owned by the logical terminal associated with the current task. RELEASE also releases the information associated with a previous KEEP LONGTERM NOTIFY request.

5.52.3 Lock-options

SHARE
Applies a longterm shared lock to the specified record.

EXCLUSIVE
Applies a longterm exclusive lock to the specified record.

CURRENT record-name
Applies the lock to the current occurrence of *record-name*.

CURRENT set-name
Applies the lock to the record current of *set-name*.

CURRENT area-name
Applies the lock to the record current of *area-name*.

UPGRADE
Upgrades a previous KEEP LONGTERM NOTIFY CURRENT request.

SHARE
Places a shared longterm lock on the record.

EXCLUSIVE
Places an exclusive longterm lock on the record.

RETURN NOTIFICATION into
Returns information on database activity for the specified record.

return-location
The symbolic name of a user-defined PIC S9(8) COMP (fullword) field that contains the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the data area to which CA-IDMS will return the information.

WAIT
Requests the issuing task to wait for an existing lock to be released.

This is the default.

If the wait would cause a deadlock, CA-IDMS terminates the task abnormally.

NOWAIT
Requests the issuing task not to wait for an existing lock to be released.

NODEADLOCK
Requests the issuing task to wait for an existing lock to be released, unless to do so would cause a deadlock. If the wait would cause a deadlock, CA-IDMS returns control to the task.

5.52.4 Example

The steps below illustrate the use of the KEEP LONGTERM statement.

1. Begin monitoring database activities for the current occurrence of the EMPLOYEE record by coding:

```
KEEP LONGTERM KEEP-ID NOTIFY CURRENT EMPLOYEE.
```

2. Return statistics of database activities for the record identified by KEEP-ID into STAT-VALUE by coding:

```
KEEP LONGTERM KEEP-ID TEST RETURN NOTIFICATION
              INTO STAT-VALUE.
```

3. Depending on the value returned to STAT-VALUE, you may want to put a longterm shared lock on the EMPLOYEE record identified by KEEP-ID by coding:

```
KEEP LONGTERM KEEP-ID UPGRADE SHARE.
```

4. After processing, release all longterm locks by coding:

```
KEEP LONGTERM ALL RELEASE.
```

5.52.5 Status codes

After completion of the KEEP LONGTERM function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
5101	The NODEADLOCK option has been specified; however, to wait would cause a deadlock. Control has returned to the issuing task.
5102	Unable to obtain storage for the required KEEP LONGTERM control blocks.
5105	Either the requested record type cannot be found or currency has not been established.
5113	The required area control block was not found in the DMCL.
5121	Either the requested longterm ID cannot be found or the KEEP LONGTERM request was issued by a nonterminal task.
5123	The specified area cannot be found.
5131	The parameter list is invalid.
5147	The KEEP LONGTERM area has not been readied.
5148	The run unit associated with the KEEP LONGTERM request has not been bound.
5149	The NOWAIT option has been specified; however, a wait is required.
5151	A lock manager error occurred during the processing of the KEEP LONGTERM request.
5159	An error occurred in transferring the KEEP LONGTERM request to IDMSKEEP.
5160	The requested KEEP LONGTERM lock ID was already in use with a different page group.
5161	The requested KEEP LONGTERM lock ID was already in use with a different BDKey format.

5.53 LOAD TABLE

The LOAD TABLE statement instructs CA-IDMS/DC to load a table (module or program) into the program pool and provide access to it through a COBOL LINKAGE SECTION entry.

5.53.1 Syntax

```
►►── LOAD TABLE program ──────────────────────────────────────────►

 ►── INTO 01-level-program-location ─┬─ TO end-program-location ──────┬──►
                                     └─ POINTER table-location-pointer ┘

 ►──┬──────────────────────┬──┬──────────────────────┬──────────────►
    └─ DICTNODE nodename ──┘  └─ DICTNAME dictionary-name ┘

 ►──┬──────────────────────┬──────────────────────────────────────►
    └─ LOADLIB library-name ┘

 ►──┬─ WAIT ◄ ─┬── . ──────────────────────────────────────────►◄
    └─ NOWAIT ─┘
```

5.53.2 Parameters

program

Either the symbolic name of a user-defined field that contains the table or the name itself enclosed in quotation marks.

INTO

Specifies the LINKAGE SECTION entry of the 01-level record area that references the loaded table.

01-level-program-location

The symbolic name of a user-defined field that contains the name of the 01-level LINKAGE SECTION entry used to load the table.

Note: CA-IDMS/DC does not support the use of an OCCURS DEPENDING ON clause within *01-level-program-location*.

TO

Specifies the end of the LINKAGE SECTION entry of the 01-level record area that references the loaded table.

This parameter is optional under COBOL 85.

end-program-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the module being loaded. *End-program-location* is an entry subordinate to the 01-level record.

POINTER

Specifies a pointer to the address of the table (COBOL 85 only).

table-location-pointer
The symbolic name of a user-defined field that is to contain the pointer to the address of the table.

DICTNODE
Specifies the node that controls the dictionary where the table resides.

nodename
Either the symbolic name of a user-defined eight-character field in variable storage or the node name itself enclosed in quotation marks.

DICTNAME
Specifies the dictionary where the table resides.

dictionary-name
Either the symbolic name of a user-defined eight-character field in variable storage or the dictionary name itself enclosed in quotation marks.

LOADLIB
Specifies the load library containing the table.

library-name
Either the symbolic name of a user-defined eight-character field in variable storage or the library name itself enclosed in quotation marks.

WAIT
Requests the issuing task to wait for sufficient storage in the event that program pool storage is not immediately available to meet the requirements of the LOAD TABLE request.

This is the default.

If you specify WAIT and CA-IDMS/DC encounters an insufficient storage condition, the issuing task is placed in an inactive state; when the LOAD TABLE function is completed, control returns to the issuing task according to its previously established dispatching priority.

NOWAIT
Requests the issuing task not to wait for storage to become available. If you specify NOWAIT, CA-IDMS/DC returns a value of 3402 (DC-NO-STORAGE) to the ERROR-STATUS field when an insufficient storage condition exists.

Example: The example below defines the 01-level LINKAGE SECTION entry for use with the LOAD TABLE request for a table built from an Assembler program.

Note: IDD edit and code tables contain special characters and variable-length fields. In general, such fields are not used in a COBOL program.

The following source code defines the 01-level LINKAGE SECTION entry for use with the LOAD TABLE request:

```
LINKAGE SECTION.

01  STATE-TABLE.
    02  STATES                  OCCURS 50 TIMES.
        03  STATE-ABB           PIC X(2).
        03  STATE-FULL          PIC X(15).
    02  END-STATE-TABLE         PIC X.
```

5.53.3 Examples

The examples below illustrate the use of the LOAD TABLE statement:

Example 1: The following statement loads the STATECON table into the 01-level LINKAGE SECTION entry STATE-TABLE:

```
LOAD TABLE 'STATECON'
     INTO STATE-TABLE TO END-STATE-TABLE.
```

Example 2: The example below defines the 01-level LINKAGE SECTION entry for use with the LOAD TABLE request for an IDD CODE TABLE, defined as follows:

```
ADD TABLE NAME IS DECODMTH
   TABLE DESCRIPTION IS 'MONTH CODE CONVERT'
    TYPE IS CODE
    SEARCH IS LINEAR
    ENCLODE DATA IS ALPHANUMERICPIC 9(4) COMP.
    TABLE IS UNSORTED
    DUPLICATES ARE NOT ALLOWED
    VALUES ARE (  01 JAN 02 FEB 03 MAR 04 APR
                  05 MAY 06 JUN 07 JUL 08 AUG
                  09 SEP 10 OCT 11 NOV 12 DEC ).
```

The following source code defines the 01-level LINKAGE SECTION entry for use with the LOAD TABLE request:

```
LINKAGE SECTION.

01  MONTH-TABLE.
    02  TABLE-HEADER.
        03  HDR-NUM-ENTRIES     PIC 9(4) COMP.
    02  TABLE-DATA.             OCCURS 12 TIMES.
        03  DTA-FILLER1         PIC X(2).
        03  DTA-MONTH-NUM       PIC 9(2).
        03  DTA-FILLER2         PIC X.
        03  DTA-MONTH-TXT       PIC X(3).
    02  END-MONTH-TABLE         PIC X.
```

The following statement loads the DECODMTH tables into the 01-level LINKAGE SECTION entry MONTH-TABLE:

```
PROCEDURE DIVISION USING MONTH-TABLE.

LOAD TABLE 'DECODMTH'
     INTO MONTH-TABLE TO END-MONTH-TABLE.
```

Note: For BS2000, starting from the COBOL85 compiler V2.2C and higher, each 01-level entry in the LINKAGE SECTION has to be defined in the USING-clause of the PROCEDURE DIVISION.

Note: For BS2000, TABLE definition must be the very last definition in the LINKAGE SECTION.

5.53.4 Status codes

After completion of the LOAD TABLE function, the ERROR-STATUS field in the CA-IDMS/DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
3401	The requested module cannot be loaded immediately due to insufficient storage; to wait would cause a deadlock.
3402	The requested module cannot be loaded because insufficient storage exists in the program pool.
3407	The requested module cannot be loaded because an I/O error has occurred during processing.
3414	The requested module cannot be loaded because it has been defined as nonconcurrent and is currently in use.
3415	The requested module has been overlaid temporarily in the program pool and cannot be reloaded immediately.
3435	The request cannot be serviced because the specified 01-level LINKAGE SECTION entry has either been previously allocated or contains an OCCURS DEPENDING ON clause.
3436	Either the requested program is not defined in the program definition table (PDT) or is marked out of service, or null PDEs are not specified or valid in this CA-IDMS/DC system.

5.54 MAP IN

The MAP IN statement requests a synchronous transfer of data from map fields on the screen to the corresponding variable-storage data fields. The MAP IN statement can also be used to transfer data from an area in variable storage that contains a 3270-like data stream to map-related variable-storage data fields; this is referred to as a native mode data transfer.

5.54.1 Syntax

```
►►── MAP IN USING map-name ──────────────────────────────────────►
►┬─ IO ◄ ──────────────────────────────────────────────────────┬►
 │                                                              │
 └─ NOIO DATASTREAM FROM mapped-data-location ─┬─ TO end-data-location ─┬
                                               └─ LENGTH data-length ───┘
►┬─ INPUT DATA is ─┬─ YES ─┬──────────────────────────────────────►
 │                 └─ NO ──┘
►┬─ DETAIL detail-options ─┬─┬─ PAGE is page-number ─┬─┬─ MODIFIED ─┬─ . ──►◄
 │                         │ │                       │ │            │
 └─ HEADER ────────────────┘ │                       │ │            │
```

Expansion of detail-options

```
►┬─ NEXT ◄ ──────────────────────────┬─┬─ RETURNKEY is data-field-name ─┬►
 │                                   │ │                                │
 │─ FIRST ───────────────────────────┤ │                                │
 │─ SEQUENCE NUMBER is seq-field-name┘ │                                │
 └─ KEY is key ────────────────────────┴────────────────────────────────┘
```

5.54.2 Parameters

map-name

The name of the map to be used for the MAP IN request. It must be a map included in the program's MAP SECTION.

IO

Requests CA-IDMS to transfer data from map fields to variable-storage data fields that are associated with the specified map.

This is the default type of data transfer.

NOIO DATASTREAM FROM

Requests CA-IDMS to transfer data from an area in program variable storage to the variable-storage data fields that correspond to the specified map. No terminal I/O is associated with the request.

mapped-data-location

The symbolic name of a user-defined field that contains the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the data stream to be read by CA-IDMS. The length of the data stream is determined by one of the following specifications:

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the data stream.

end-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the input data stream.

LENGTH

Explicitly defines the length in bytes of the input data stream.

data-length

Either the symbolic name of a user-defined field that contains the length of the data stream or the length itself expressed as a numeric constant.

INPUT DATA IS

I/O requests only. Specifies whether the contents of all fields of the specified map will be moved to variable-storage data fields, or left unchanged.

This specification applies to all variable-storage data fields unless overridden by an INPUT DATA IS YES/NO clause in a previously issued MODIFY MAP request.

YES

Moves the contents of all fields of the specified map to variable-storage data fields.

NO

Leaves the contents of all variable-storage data fields unchanged.

DETAIL

Pageable maps only. Specifies that the MAP IN operation is to retrieve data from a modified detail occurrence (MDT set on). The contents of all map fields in the detail occurrence are retrieved unless MODIFIED is specified for the MAP IN DETAIL statement; MODIFIED causes only modified fields to be retrieved. For more information on pageable maps, see the *CA-IDMS Mapping Facility*.

NEXT

Retrieves the next sequential modified detail occurrence.

This is the default.

An end-of-data condition (DC-NO-MORE-UPD-DETAILS) is returned in either of the following cases:

- No detail occurrences have been modified.
- All modified detail occurrences have been mapped in already.

FIRST

Retrieves the first available modified detail occurrence.

SEQUENCE-NUMBER is

Retrieves a detail occurrence by sequence number. Detail occurrences are built at run time by the application program and stored in the sequence in which they are created.

seq-field-name
A PIC S9(8) COMP (fullword) field.

A detail-not-found condition is returned in either of the following cases:

- The specified occurrence is not a modified detail occurrence.
- No detail occurrence with the specified value is found.

RETURNKEY IS
Specifies the variable field in which CA-IDMS stores the four-byte value (if any) associated with the retrieved detail occurrence. If no value is associated with the detail occurrence, CA-IDMS sets *data-field-name* to zero.

data-field-name
The symbolic name of either a PIC X(4) or PIC S9(8) COMP (fullword) field that contains the key value. *Data-field-name* does not have to be fullword aligned.

KEY IS
Retrieves a modified detail occurrence based on the value associated with the detail occurrence.

key
The name of a PIC S9(8) COMP (fullword) field.

Note: A value is associated with a detail occurrence by using the KEY IS parameter in the MAP OUT DETAIL command for that occurrence.

HEADER
Pageable maps only. Specifies that the MAP IN operation is to retrieve the contents of data fields in the header and footer areas. The contents of all data fields in the header and footer areas are retrieved unless MODIFIED is specified for the MAP IN HEADER statement; MODIFIED causes only modified fields to be retrieved.

PAGE IS
Pageable maps only. Specifies the name of a variable field to store the current value of the $PAGE field on mapin.

page-number
A PIC S9(8) COMP (fullword) field.

MODIFIED
Pageable maps only. Specifies that, within a modified detail occurrence, only modified fields (MDT set on) are to be retrieved in the MAP IN operation.

5.54.3 Examples

The examples below illustrate the use of the MAP IN statement.

Example 1: The following statement requests CA-IDMS to read the EMPMAPLR map. Data values are transferred from map fields on the EMPMAPLR map to the corresponding variable-storage data fields. Subsequent commands can evaluate the input values and perform appropriate processing.

```
MAP IN USING EMPMAPLR
   INPUT DATA IS YES.
```

Example 2: The following statement requests CA-IDMS to map in the next modified detail occurrence of the EMPMAPPG MAP:

```
MAP IN USING EMPMAPPG
   DETAIL
   NEXT MODIFIED.
```

5.54.4 Status codes

After completion of the MAP IN function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4627	A permanent I/O error has occurred during processing
4628	The dial-up line for the terminal has been disconnected
4631	The map request block (MRB) contains an invalid field, indicating a possible error in the program's parameters
4632	The derived length of the specified map input data area is zero or negative
4633	The map load module named in the MRB cannot be found
4634	The LINKAGE SECTION entry specified is not at COBOL 01-Level.
4638	The specified 01-level WORKING-STORAGE SECTION or LINKAGE SECTION entry has not been allocated
4639	The terminal being used is out of service
4640	The NOIO option has been specified but the requested data stream cannot be found
4642	The requested map does not support the terminal device being used
4652	The specified edit or code table either cannot be found or is invalid for use with the named map
4654	A data conversion error has occurred; internal map data does not match the map's data description
4655	The user-written edit routine specified for the named map cannot be found
4664	The requested node for a header or detail was either not present or not updated
4668	No more modified detail occurrences require mapin

Status code	Meaning
4672	The scratch record that contains the requested detail could not be accessed (internal error)

5.55 MAP OUT

The MAP OUT statement creates or modifies detail occurrences for a pageable map or requests a transfer of data from variable-storage data fields to map fields on the terminal screen. MAP OUT can also be used to transfer data to another area in program variable storage; this is referred to as a native mode data transfer.

5.55.1 Syntax

```
►►── MAP OUT USING map-name ──────────────────────────────────────►

►─┬─ WAIT ◄ ─┬──────────────────────────────────────────────────►
  ├─ NOWAIT ─┘

►─┬─ IO ◄ ──────────────────────────────────────────────────────►─
  │
  └─ NOIO DATASTREAM INTO data-location ─┬─ TO end-data-location ─┬──────────
                                         └─ max LENGTH data-length ─┘

►──────────────────────────────────────────────────────────────────►
  │
  │
  └─ RETURN LENGTH into data-actual-length ─┘

►─┬─ OUTPUT ─┬──────────────────────────────────────────────────┬──►
  │          └─ DATA is ─┬─ YES ──────┬─┬─ NEWPAGE ─┬─┬─ LITERALS ─┬
  │                      ├─ NO ───────┤ └─ ERASE ───┘ └────────────┘
  │                      ├─ ERASE ────┤
  │                      └─ ATTRIBUTE ─┘

►─┬─ MESSAGE IS message-text ─┬─ TO end-message-data-location ─┬──►
  │                           └─ LENGTH data-length ───────────┘

►─┬─ DETAIL ─┬──────────────────────────┬──────────────────────── . ──►◄
  │          ├─ NEW ◄ ─┬─┬─ KEY is key ─┬
  │          └─ CURRENT ─┘ │              │
  │                        │              │
  └─ RESUME ─┬──────────────────────────┬
             └─ PAGE is ─┬─ CURRENT ◄ ──┬
                         ├─ NEXT ────────┤
                         ├─ PRIOR ───────┤
                         ├─ FIRST ───────┤
                         ├─ LAST ────────┤
                         └─ page-number ─┘
```

5.55.2 Parameters

map-name

The map to be used for the MAP OUT request. The map must be included in the program's MAP SECTION.

WAIT

Specifies that the data transfer will be synchronous. CA-IDMS places the issuing task in an inactive state. When the MAP OUT operation is complete, the task resumes processing according to its established dispatching priority.

This is the default.

NOWAIT

Specifies that the data transfer will be asynchronous; the task will continue executing. If NOWAIT is specified, the program must issue a CHECK TERMINAL before performing any other I/O operation.

IO

Requests CA-IDMS to transfer data from variable-storage data fields associated with the named map to the terminal device associated with the issuing task.

This is the default.

NOIO DATASTREAM INTO

Requests CA-IDMS to transfer data from variable-storage data fields associated with the named map to another area of program variable storage; no terminal I/O is associated with the request.

data-location

The symbolic name of a user-defined field that contains the WORKING-STORAGE SECTION or LINKAGE SECTION entry to which the data is to be transferred.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry for the output data stream and is specified following the last data-item entry in *data-location*.

end-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the output data stream.

max LENGTH

Defines the maximum length of the output data stream.

data-length

Either the symbolic name of the user-defined field that contains the length of the data stream or the length itself expressed as a numeric constant.

RETURN LENGTH INTO

Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry to which CA-IDMS will return the length in bytes of the output data stream.

data-actual-length

If the data stream has been truncated, contains the length before truncation.

OUTPUT

IO requests only. Specifies screen display options for the data being output.

DATA IS

Specifies whether the variable-storage data fields are to be transmitted to the terminal. This specification applies to all variable-storage data fields unless overridden by an OUTPUT DATA IS YES/NO clause in a previously issued MODIFY MAP request.

YES

Transmits the contents of variable-storage data fields to the corresponding map fields.

NO

Does not transmit the contents of variable-storage data fields to the corresponding map fields. However, if the automatic error handling facility detects an error in any field, CA-IDMS will transmit the applicable attribute bytes.

ERASE

Does not transmit the contents of variable-storage data fields and fills the corresponding map fields with null values.

ATTRIBUTE

Transmits only the attribute bytes for variable-storage data fields. Data in the record buffer is not sent to the terminal.

NEWPAGE (ERASE)

The keywords NEWPAGE and ERASE are synonymous.

Activates the erase-write function; CA-IDMS clears the screen and transmits both literal and variable fields to the map. If NEWPAGE is not specified, CA-IDMS will write over any existing screen display without first erasing it.

To erase individual map fields, use the OUTPUT DATA IS ERASE option of the MODIFY MAP statement. To request CA-IDMS to erase all screen fields and to activate the erase-write function, the MAP OUT statement must specify OUTPUT DATA IS ERASE NEWPAGE.

LITERALS

Transmits literal fields as well as variable-storage data fields to the terminal. If LITERALS is not specified, CA-IDMS will write literal fields to the map only when a MAP OUT request specifies the ERASE option.

MESSAGE IS

IO requests only. Specifies the message to be displayed in the map's message area.

message-text

The symbolic name of a WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the message text.

TO

Specifies the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the message text and is specified following the last data item in *message-text*.

end-message-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the output data stream.

LENGTH

Defines the length in bytes of the message text.

data-length

Either the symbolic name of a user-defined field that contains the length or the length itself expressed as a numeric constant.

Note: The MESSAGE parameter can only be used with MAP OUT DETAIL if the $MESSAGE field is associated with the detail occurrence at map generation.

Note: To reference a message stored in the data dictionary, use the ACCEPT TEXT INTO parameter of the WRITE LOG statement to copy the message into *message-text*.

DETAIL

Pageable maps only. Specifies that the MAP OUT command is to create or modify a detail occurrence, and optionally associates a numeric key value with the occurrence. For more information on pageable maps, see the *CA-IDMS Mapping Facility*.

NEW

Creates a detail occurrence of a pageable map.

This is the default.

Occurrences are displayed in the order in which they are created by the application program.

CURRENT

Modifies the detail occurrence that was referenced by the most recent MAP IN DETAIL or MAP OUT DETAIL statement.

KEY IS

Specifies a value to be associated with the created or modified detail occurrence. The four-byte numeric value is not displayed on the terminal screen.

When the KEY IS parameter is used with the MAP OUT DETAIL CURRENT command, the specified value replaces the value (if any) previously associated with the detail occurrence.

key

The name of a PIC S9(8) COMP (fullword) field that contains the key of a database record associated with the detail occurrence.

RESUME

Pageable maps only. Specifies the page of detail occurrences to be mapped out to the terminal.

PAGE is CURRENT

Redisplays the current page.

This is the default.

If no page has been displayed, the first page of the pageable map is displayed.

PAGE is NEXT

Displays the page that follows the current page. If no page follows the current page, the current page is redisplayed.

PAGE is PRIOR

Displays the page that precedes the current page. If no page precedes the current page, the current page is redisplayed.

PAGE is FIRST

Displays the first available page of detail occurrences.

PAGE is LAST

Displays the page of detail occurrences with the highest available page number.

PAGE is page-number

A user field that contains the number of the page to be displayed. A page number is stored in the variable field by a preceding MAP IN PAGE IS *page-number* statement that names the same numeric variable field.

5.55.3 Examples

The examples below illustrate the use of the MAP OUT statement:

Example 1: The following statement requests CA-IDMS to write all literal and data fields associated with the EMPMAPLR map to the terminal:

```
MAP OUT USING EMPMAPLR
   OUTPUT DATA IS YES
   NEWPAGE
   MESSAGE IS INITIAL-MESSAGE LENGTH 80.
```

Example 2: The following statement maps out the current detail; no terminal I/O is associated with this request if the first page of the pageable map is not yet filled.

```
MAP OUT USING EMPMAPPG
   DETAIL
   KEY IS DETAIL-KEY.
```

5.55.4 Status codes

After completion of the MAP OUT function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4625	The output operation has been interrupted; the operator has pressed ATTENTION or BREAK.
4626	A logical error (for example, an invalid control character) has been encountered in the output data stream.
4627	A permanent I/O error has occurred during processing.
4628	The dial-up line for the terminal has been disconnected.
4631	The map request block (MRB) contains an invalid field, indicating a possible error in the program's parameters.
4632	The derived length of the specified map output data area is zero or negative.
4633	The map load module named in the MRB cannot be found.
4634	The LINKAGE SECTION entry specified is not at COBOL 01-Level.
4638	The WORKING-STORAGE SECTION or LINKAGE SECTION entry specified for return of the output data stream has not been allocated.
4639	The terminal being used is out of service.
4640	The NOIO option has been specified but the requested data stream cannot be found.
4642	The requested map does not support the terminal device being used.
4652	The specified edit or code table either cannot be found or is invalid for use with the named map.
4653	An error has occurred in a user-written edit routine.
4654	A data conversion error has occurred; internal map data does not match the map's data description.
4655	The user-written edit routine specified for the named map cannot be found.
4664	There is no current detail occurrence to be updated (MAP OUT DETAIL CURRENT only). No action is taken.

Status code	Meaning
4668	The amount of storage defined for pageable maps at system generation time is insufficient. No action is taken. This and subsequent MAP OUT DETAIL statements are ignored.
4672	No detail occurrence, footer, or header fields exist to be mapped out by a MAPOUT RESUME command.
4676	The first screen page has been transmitted to the terminal.
4680	A pageable map page has been built but the page has not been displayed. This can happen after you specify STARTPAGE NOAUTODISPLAY. Test for it after each MAP OUT DETAIL statement.

5.56 MAP OUTIN

The MAP OUTIN statement requests an output data transfer (MAP OUT) followed by an input data transfer (MAP IN). MAP OUTIN combines the functions of the MAP OUT and MAP IN requests; however, it cannot be used to perform pageable map functions or native mode data transfers. By definition, the MAP OUTIN request is synchronous; it forces the program to be conversational.

5.56.1 Syntax

```
►►── MAP OUTIN USING map-name ──────────────────────────────────────►
►──┬──────────────────────────────────────────────────────────────┬──►
  └─ OUTPUT ──┬──────────────────────────────────────────────────┬─┘
              └─ DATA is ─┬─ YES ──────┬─┬─ NEWPAGE ─┬─┬─ LITERALS ─┬
                          ├─ NO ───────┤ └─ ERASE ───┘ └────────────┘
                          ├─ ERASE ────┤
                          └─ ATTRIBUTE ┘

►──┬──────────────────────────────────────────────────────────────►
  └─ INPUT DATA is ─┬─ YES ─┬
                    └─ NO ──┘

►──┬──────────────────────────────────────────────────────────────┬──►◄
  └─ MESSAGE IS message-text ─┬─ TO end-message-data-location ─┬─ . ─
                              └─ LENGTH data-length ───────────┘
```

5.56.2 Parameters

map-name

Specifies the map to be used for the MAP OUTIN request. Must be the name of a map included in the program's MAP SECTION.

OUTPUT

Specifies screen display options for the data being output.

DATA is

Specifies whether variable-storage data fields are to be transmitted to the terminal. This specification applies to all variable-storage data fields unless overridden by an OUTPUT DATA IS YES/NO clause in a previously issued MODIFY MAP request.

YES

Transmits the contents of variable-storage data fields to the corresponding map fields.

NO

Does not transmit the contents of variable-storage data fields to the corresponding map fields. However, if the automatic error handling facility detects an error in any field, CA-IDMS will transmit the applicable attribute bytes.

ERASE
Does not transmit the contents of variable-storage data fields and fills the corresponding map fields with null values.

ATTRIBUTE
Transmits only the attribute bytes for variable-storage data fields. Data in the record buffer is not sent to the terminal.

NEWPAGE (ERASE)
The keywords NEWPAGE and ERASE are synonymous.

Activates the erase-write function; CA-IDMS clears the screen and transmits both literal and variable fields to the map. If NEWPAGE is not specified, CA-IDMS will write over any existing screen display without first erasing it.

To erase individual map fields, use the OUTPUT DATA IS ERASE option of the MODIFY MAP statement (described later in this chapter). To request CA-IDMS to erase all screen fields and to activate the erase-write function, the MAP OUT statement must specify OUTPUT DATA IS ERASE NEWPAGE.

LITERALS
Requests CA-IDMS to transmit literal fields as well as variable-storage data fields to the terminal. If LITERALS is not specified, CA-IDMS will write literal fields to the map only when a MAP OUT request specifies the ERASE option.

INPUT DATA is
Specifies whether the contents of map fields will be moved to variable-storage data fields (YES), or left unchanged (NO).

This specification applies to all variable-storage data fields unless overridden by an INPUT DATA IS YES/NO clause in a previously issued MODIFY MAP request.

YES
Moves the contents of map fields to variable-storage data fields.

NO
Leaves the contents of map fields unchanged.

MESSAGE IS
Specifies the message to be displayed in the map's message area.

message-text
The symbolic name of a WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the message text.

TO
Specifies the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the message text and is specified following the last data item in *message-text*.

end-message-data-location
The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the output data stream.

LENGTH

Defines the length in bytes of the message text.

data-length

Either the symbolic name of a user-defined field that contains the length or the length itself expressed as a numeric constant.

Note: To reference a message stored in the data dictionary, use the ACCEPT TEXT INTO parameter of the WRITE LOG statement (described later in this chapter) to copy the message into *message-text*.

5.56.3 Example

The following statement requests CA-IDMS to erase the screen, transmit literal and variable map fields (null values), and perform a MAP IN when the operator presses an AID key:

```
MAP OUTIN USING EMPMAPLR
   OUTPUT DATA IS ERASE NEWPAGE
   INPUT DATA IS YES.
```

5.56.4 Status codes

After completion of the MAP OUTIN function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4625	The I/O operation has been interrupted; the terminal operator has pressed ATTENTION or BREAK
4626	A logical error (for example, an invalid control character) has been encountered in the output data stream
4627	A permanent I/O error has occurred during processing
4628	The dial-up line for the terminal is disconnected
4631	The map request block (MRB) contains an invalid field, indicating a possible error in the program's parameters
4633	The map load module named in the MRB cannot be found
4639	The terminal being used is out of service
4642	The requested map does not support the terminal device being used
4652	The specified edit or code table either cannot be found or is invalid for use with the named map
4653	An error has occurred in a user-written edit routine
4654	A data conversion error has occurred; internal map data does not match the map's data description

Status code	Meaning
4655	The user-written edit routine specified for the named map cannot be found

5.57 MODIFY

The MODIFY statement replaces element values of the specified record occurrence in the database with new element values defined in program variable storage.

Before execution of the MODIFY statement, the following conditions must be satisfied:

- All areas affected either implicitly or explicitly must be readied in one of the update usage modes (see 5.66, “READY” on page 5-201 later in this chapter).
- The specified record must be established as current of run unit. If the record that is current of run unit is not an occurrence of the specified record, a non-zero status condition results.
- The values of all elements defined for the specified record in the program's subschema view must be in variable storage. If the MODIFY statement is not preceded by an OBTAIN statement, you must initialize the appropriate values. The best practice, however, is to precede MODIFY with an OBTAIN statement to ensure that all the elements in the modified record are present in variable storage.

The following special considerations apply to the modification of CALC- and sort-control elements:

- If modification of a CALC- or sort-control element will violate a duplicates-not-allowed option, the record is not modified and a non-zero status condition results.
- If a CALC-control element is modified, successful execution of the MODIFY statement enables the record to be accessed on the basis of its new CALC-key value. The db-key of the specified record is not changed.
- If a sort-control element is to be modified, the sorted set in which the specified record participates must be included in the subschema invoked by the program. A record occurrence that is a member of a set not defined in the subschema can be modified *if the undefined set is not sorted.*
- If any of the modified elements in the specified record are defined as sort-control elements for any set occurrence in which that record is currently a member, the set occurrence is examined. If necessary, the specified record is disconnected and reconnected in the set occurrence to maintain the set order specified in the schema.

The following special considerations apply to the modification of records in native VSAM data sets:

- The length of a record in an entry-sequenced data set (ESDS) cannot be changed even if the records are variable length.
- The prime key for a key-sequenced data set (KSDS) cannot be modified.

Currency: The specified record must be established as current of run unit.

Following successful execution of the MODIFY statement, the modified record becomes the current record of run unit, its record type, its area, and all sets in which it participates as member or owner.

5.57.1 Syntax

```
►►── MODIFY record-name . ──────────────────────────────────────►◄
```

5.57.2 Parameters

record-name

The record type to update. The record must be a type included in the subschema. The occurrence of *record-name* residing in program variable storage will be updated.

5.57.3 Example

The following example illustrates the steps involved in modifying an occurrence of the EMPLOYEE record. Assume that the employee address is to be changed.

1. Retrieve the desired EMPLOYEE record, moving its contents to variable storage:

   ```
   MOVE EMP-ID-IN TO EMP-ID-0415.
   OBTAIN CALC EMPLOYEE.
   ```

2. Update the value of the EMP-ADDRESS-0415 field by moving the new address into the proper location in the EMPLOYEE record:

   ```
   MOVE NEW-ADDRESS TO EMP-ADDRESS-0415.
   ```

3. Issue a MODIFY statement to return all data items in the EMPLOYEE record to the database:

   ```
   MODIFY EMPLOYEE.
   ```

5.57.4 Status codes

After completion of the MODIFY function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
0804	The OCCURS DEPENDING ON item is less than zero or greater than the maximum number of occurrences of the control element
0805	Modification of the record would violate a duplicates-not-allowed option for a CALC record, a sorted set, or an index set
0806	Currency has not been established for the named record

Status code	Meaning
0808	The specified record cannot be found; the record name has probably been misspelled
0809	The named record's area has not been readied in one of the update usage modes
0810	The subschema specifies an access restriction that prohibits modification of the named record
0811	There is insufficient space to hold the modified variable-length record occurrence
0813	A current record of run unit has not been established or has been nullified by a previous ERASE statement
0818	The record has not been bound
0820	The current record of run unit is not the same type as the named record
0821	An area other than the area of the named record has been readied with an incorrect usage mode
0825	No current record of set type has been established
0833	At least one sorted set in which the named record participates has not been included in the subschema
0855	An invalid length has been defined for a variable length record
0860	A record occurrence has been encountered whose type is inconsistent with the set named in the ERROR-SET field of the IDMS communications block; probable causes include: a broken chain and improper database description
0883	Either the length of a record in a native VSAM ESDS has been changed or a prime key in a native VSAM KSDS has been modified

5.58 MODIFY (LRF)

The MODIFY statement changes field values in an existing logical-record occurrence. LRF uses the field values present in the variable storage location reserved for the logical record to update the appropriate database records in the database. You can optionally specify an alternative variable storage location from which the changed field values are to be taken.

5.58.1 Syntax

```
►►── MODIFY logical-record-name ──────────────────────────────────►
►─┬──────────────────────────────┬──────────────────────────────►
  └─ FROM alt-logical-record-location ─┘
►─┬──────────────────────────────┬──────────────────────────────►
  └─ WHERE boolean-expression ─┘
►─┬──────────────────────────────────────┬─ . ──────────────────►◄
  └─ ON path-status imperative-statement ─┘
```

5.58.2 Parameters

logical-record-name

Updates data field values in the named logical record. Unless the FROM clause is specified (see below), the field values used to update the database are taken from the area in program variable storage reserved for the named logical record. The logical record must be defined in the subschema.

FROM

Specifies an alternative variable storage location from which the field values used to perform the requested modification are to be obtained. When modifying a logical record that was retrieved into an alternative location in variable storage, the FROM clause should name the same location specified in the OBTAIN request.

alt-logical-record-location

A record location defined in the WORKING-STORAGE SECTION or LINKAGE SECTION.

WHERE

Specifies the selection criteria to be applied to the named logical record. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

boolean-expression

The selection criteria to apply.

ON parameter

Specifies the action to be taken depending on the value returned to the LR-STATUS field in the LRC block. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

path-status
The value of the LR-STATUS field in the LRC block which triggers the specified action.

imperative-statement
The action to take.

5.58.3 Example

The following example illustrates the steps taken to modify an occurrence of the EMP-SKILL-LR logical record. Assume that the skill level for employee 120 is to be upgraded from 02 (COMPETENT-0425) to 03 (PROFICIENT-0425).

1. Retrieve the desired logical-record occurrence:

```
OBTAIN FIRST EMP-SKILL-LR WHERE EMP-ID-0415 EQ '0120'
      AND SKILL-ID-0455 EQ '3610'
      AND COMPETENT-0425.
```

2. Update the SKILL-LEVEL-0425 field:

```
MOVE '03' TO SKILL-LEVEL-0425.
```

3. Issue the MODIFY statement for the updated EMP-SKILL-LR logical record:

```
MODIFY EMP-SKILL-LR.
```

MODIFY EMP-SKILL-LR: The following figure illustrates the above example by showing three occurrences of the EMP-SKILL-LR logical record.

LRF retrieves the EMP-SKILL-LR logical record where

- EMP-ID-0415 = '0120'
- SKILL-ID-0455 = '0120'
- SKILL-LEVEL-0425 = '02' (COMPETENT-0425)

The bottom EXPERTISE occurrence represents the only data physically modified in the database.

EMPLOYEE	EXPERTISE	SKILL
120	04	7620
120	03	3710
120	(02) 03	3610

5.59 MODIFY MAP

The MODIFY MAP statement requests CA-IDMS to modify options in the map request block (MRB) for a map; modifications can be designated as permanent or temporary. Requested revisions can be field-specific, map-specific, or both; field-specific revisions apply to the map's variable data fields.

Note: The MODIFY MAP statement parameters used to revise predefined map and/or map data field attributes have no defaults. If a MODIFY MAP parameter is not specified, the applicable option remains set to the value specified at map generation or to the value specified in a previously issued MODIFY MAP PERMANENT statement.

5.59.1 Syntax

```
►►── MODIFY MAP map-name ──────────────────────────────────────────►◄

►┬─────────────────────────────────────────────────────────────────►
 ├─ PERMANENT ◄─┤
 └─ TEMPORARY ──┘

►┬─────────────────────────────────────────────────────────────────►
 └─ CURSOR at ─┬─ cursor-row cursor-column ─┤
               └─ DFLD field-name ──────────┘

►┬─────────────────────────────────────────────────────────────────►
 │                 ┌──────────────────────┐
 └─ WCC ─┬─┬─┬─┬─┬─┴─┬─ RESETMDT ─┬─┬─┬─┬─┴─┬─┬─┬─┬─┬
```

```
►─┬─ ALL ─┬─ BUT ────┬─ CURRENT ──────────────────────────────────►
  │       └─ EXCEPT ─┘                                            │
  │                                                               │
  ├─ ALL ─┬─ CORRECT ─┬─ FIELDS ──────────────────────────────────┤
  │       └─ ERROR ───┘                                           │
  │                                                               │
  │                                   ┌──────────────────────┐   │
  │                                   ↓─ DFLD field-name ────┘   │
  │   ┌─ ALL ◄ ──────────────────────┐                           │
  │   └─ all ─┬─ BUT ────┬──────────┘                            │
  │           └─ EXCEPT ─┘                                        │
```

Expansion of modification-options

```
►─┬─ BACKSCAN ───┬──────────────────────────────────────────── . ──────────►
  │ └─ NOBACKSCAN ─┘                                                        │
  ├─ OUTPUT DATA is ─┬─ YES ──────┬──────────────────────────────────────┤
  │                  ├─ NO ───────┤                                      │
  │                  ├─ ERASE ────┤                                      │
  │                  └─ ATTRIBUTE ─┘                                     │
  ├─ INPUT DATA is ─┬─ YES ─┬──────────────────────────────────────────┤
  │                 └─ NO ──┘                                           │
  ├─ RIGHT ─┬─ JUSTIFY ──────────────────────────────────────────────────┤
  │ └─ LEFT ─┘                                                          │
  ├─ PAD ─┬─ pad-character ──┬──────────────────────────────────────────┤
  │       ├─ LOW-VALUE ──────┤                                          │
  │       └─ HIGH-VALUE ─────┘                                          │
  ├─ EDIT is ─┬─ ERROR ───┬──────────────────────────────────────────────┤
  │           └─ CORRECT ─┘                                             │
  ├─ REQUIRED ─┬──────────────────────────────────────────────────────────┤
  │ └─ OPTIONAL ─┘                                                      │
  ├─ error message is ─┬─ ACTIVE ───┬──────────────────────────────────┤
  │                    └─ SUPPRESS ─┘                                   │
  │                                                                     │
  └─ ATTRIBUTES ─┬─┬─ SKIP ──────────────────────────────────────────────┤
                 │ │                                                   │
                 │ ├─ ALPHAMERIC ─┬──────────────────────────────────┤
                 │ │ └─ NUMERIC ──┘                                   │
                 │ ├─ PROTECTED ─┬──────────────────────────────────┤
                 │ │ └─ UNPROTECTED ─┘                                │
                 │ ├─ DISPLAY ──┬──────────────────────────────────┤
                 │ │ ├─ DARK ───┤                                    │
                 │ │ └─ BRIGHT ─┘                                    │
                 │ ├─ DETECT ──────────────────────────────────────┤
                 │ ├─ MDT ──────┬──────────────────────────────────┤
                 │ │ └─ NOMDT ──┘                                   │
                 │ ├─ BLINK ────┬──────────────────────────────────┤
                 │ │ └─ NOBLINK ─┘                                  │
                 │ ├─ REVERSE-VIDEO ─┬──────────────────────────────┤
                 │ │ └─ NORMAL-VIDEO ─┘                             │
                 │ ├─ UNDERSCORE ───┬──────────────────────────────┤
                 │ │ └─ NOUNDERSCORE ─┘                             │
                 │ └─ NOCOLOR ──┬──────────────────────────────────┤
                 │   ├─ BLUE ───┤                                   │
                 │   ├─ RED ────┤                                   │
                 │   ├─ PINK ───┤                                   │
                 │   ├─ GREEN ──┤                                   │
                 │   ├─ TURQUOISE ─┤                                │
                 │   ├─ YELLOW ─┤                                   │
                 │   └─ WHITE ──┘                                   │
```

5.59.2 Parameters

map-name

The name of the map to be modified. It must be a map included in the program's MAP SECTION.

PERMANENT

Specifies that modifications will apply to all mapping mode I/O requests issued until the program terminates or until a subsequent MODIFY MAP request overrides the requested revisions.

This is the default.

TEMPORARY

Specifies that modifications will apply only to the next mapping mode I/O request (that is, MAP IN, MAP OUT, or MAP OUTIN).

CURSOR AT

Identifies the screen location at which the cursor will be positioned during output operations.

cursor-row

The row on the terminal screen to which the cursor will be moved. Either the symbolic name of the user-defined field that contains the row value or the value itself expressed as a numeric constant. Typically, fields that contains cursor row and column coordinates are level-77 data items defined as PIC S9(4) USAGE COMP (halfword).

cursor-column

The column on the terminal screen to which the cursor will be moved. Either the symbolic name of a user-defined field that contains the column value or the value itself expressed as a numeric constant. Typically, fields that contains cursor row and column coordinates are level-77 data items defined as PIC S9(4) USAGE COMP (halfword).

DFLD

Specifies that the cursor will be moved to the first position in the specified field.

field-name

The name of a map field.

WCC

Specifies the write-control character (WCC) options requested for the output operation.

If a MODIFY MAP request alters any WCC option, CA-IDMS resets unspecified options to the following values:

- NOMDT
- NOKBD
- NOALARM

RESETMDT

Specifies that the modified data tags (MDTs) for the map fields will be reset (turned off) automatically when the map is displayed.

NOMDT

Specifies that the modified data tags (MDTs) for the map fields will be not reset (turned off) automatically when the map is displayed. In this case, the associated data is retransmitted to variable-storage data fields during the next MAP IN request.

RESETKBD
Specifies that the keyboard will be unlocked automatically when the map is displayed.

NOKBD
Specifies that the keyboard will not be unlocked automatically when the map is displayed.

ALARM
Specifies that the terminal audible alarm (if installed) will sound automatically when the map is displayed.

NOALARM
Specifies that the terminal audible alarm will not sound automatically when the map is displayed.

STARTPRT
3280-type printers only. Specifies that the contents of the terminal buffer will be printed automatically when the data has been transmitted to the terminal.

NOPRT
3280-type printers only. Specifies that the contents of the terminal buffer will not be printed automatically when the data has been transmitted to the terminal.

NLCR
Specifies that no line formatting will be performed on the printer output. Printing will begin on a new line only if the printer encounters new line (NL) and carriage control (CR) characters.

40CR
Specifies that the contents of the 3280-type printer buffer will be printed at 40 characters per line.

64CR
Specifies that the contents of the 3280-type printer buffer will be printed at 64 characters per line.

80CR
Specifies that the contents of the 3280-type printer buffer will be printed at 80 characters per line.

FOR
Specifies the map fields to be modified or excluded from modification.

Field specifications

ALL BUT (EXCEPT) CURRENT
Modifies all fields except the current field. The current field is the map field that was referenced in the last MODIFY MAP or INQUIRE MAP request issued by the program. However, if that request referenced a list of fields rather than a single map field, no currency exists and all map fields are modified.

BUT and EXCEPT are synonyms and can be used interchangeably.

ALL CORRECT FIELDS

Modifies all fields found to be correct, during automatic editing or by a user-written edit module.

To specify, ALL CORRECT FIELDS, automatic editing must be enabled for the map.

ALL ERROR FIELDS

Modifies all fields found to be in error, during automatic editing or by a user-written edit module.

To specify, ALL ERROR FIELDS, automatic editing must be enabled for the map.

ALL

Specifies that all named map fields will receive the requested modifications.

This is the default.

all BUT (EXCEPT)

Specifies that all map fields except those named will receive the requested modifications.

BUT and EXCEPT are synonyms and can be used interchangeably.

DFLD

Specifies the map field(s) to modify or exclude from modification. Multiple DFLD specifications must be separated by at least one blank and must come from the same map record.

field-name

The name of the field(s) to modify or exclude from modification.

Field names that are not unique within the program must be qualified with the name of the associated record. Likewise, multiply-occurring fields must be qualified with the appropriate subscripts.

Use the following syntax:

```
map-data-field-name ──┬──────────────┬──┬──┬─ IN ─┬─ record-name ─┬──
                      └─ subscript ─┘  │  └─ OF ─┘               │
                                       └─────────────────────────┘
```

Modification options

BACKSCAN

Tells CA-IDMS to backscan the specified fields to remove trailing blanks before performing a mapout operation. Only characters up to the last nonblank will be sent to the terminal; fields remaining on the screen will contain whatever characters were present before the MAP OUT or MAP OUTIN request was issued. If the MAP OUT or MAP OUTIN request specifies the ERASE option, CA-IDMS erases the contents of all terminal data fields.

NOBACKSCAN

Tells CA-IDMS not to backscan the specified fields to remove trailing blanks before performing a mapout operation.

OUTPUT DATA IS

Tells CA-IDMS how to treat the output map fields.

YES

Sets the fields to the value of the corresponding variable-storage data fields.

NO

Leaves the fields unchanged.

ERASE

Erases the fields.

ATTRIBUTE

Transmits only the attribute byte of the fields.

INPUT DATA is YES

Moves map fields automatically to the corresponding variable-storage data fields during an input operation.

INPUT DATA is NO

Does not move map fields to the corresponding variable-storage data fields during an input operation.

RIGHT JUSTIFY

Right justifies the variable-storage fields on input.

LEFT JUSTIFY

Left justifies the variable-storage fields on input.

PAD

Indicates whether variable-storage data fields will be padded on input and, if so, defines the value or character with which the fields are to be padded.

If RIGHT JUSTIFY is specified, fields will be padded on the left; if LEFT JUSTIFY is specified, fields will be padded on the right.

pad-character

Either the symbolic name of a user-defined PIC X DISPLAY field that contains the pad character or the character itself enclosed in quotation marks.

The fields will be padded with the specified character.

LOW-VALUE

Pads the fields with zeros.

HIGH-VALUE

Pads the fields with the highest value in the computer collating sequence.

EDIT IS ERROR

Explicitly sets the error flag **on** for the specified map fields.

Automatic editing must be enabled for the map.

The ability to set the error flag enables programs to perform their own editing and validation in addition to that provided by the automatic editing feature.

On a MAPOUT operation, if any field is flagged to be in error, then for all fields (both CORRECT and INCORRECT), only attribute bytes are transmitted; no data is moved from program variable storage to the screen.

EDIT IS CORRECT

Explicitly sets the error flag **off** for the specified map fields.

Automatic editing must be enabled for the map.

The ability to set the error flag enables programs to perform their own editing and validation in addition to that provided by the automatic editing feature.

On a MAPOUT operation, if any field is flagged to be in error, then for all fields (both CORRECT and INCORRECT), only attribute bytes are transmitted; no data is moved from program variable storage to the screen.

REQUIRED

Requires the user to enter data in the specified map fields. An error results on mapin if you specify REQUIRED and the user fails to enter data in a required field.

Automatic editing must be enabled for the map and for the specified map fields.

OPTIONAL

Does not require the user to enter data in the specified map fields.

error message is

Suppresses or enables display of an error message associated with the field.

ACTIVE

Enables display of the error message associated with the field.

This is the default.

You typically enable display of a message only after specifying ERROR MESSAGE SUPPRESS for the map in a previous MODIFY MAP PERMANENT statement.

SUPPRESS

Disables display of the error message associated with the field. When the map is redisplayed because of errors, the message defined for the map field will not be displayed even if the field contains edit errors.

ATTRIBUTES

Indicates the 3270- and 3279-type terminal display attributes for the specified map fields.

Multiple attributes must be separated by blanks.

Only the named attributes will be modified in the map's MRB.

SKIP

Repositions the cursor automatically past the map fields to the next unprotected field. When you specify SKIP, the affected map fields are assigned the NUMERIC and PROTECTED attributes (described below) automatically.

ALPHAMERIC
Allows the data input to the map fields by the user to be any character on the 3270 keyboard.

NUMERIC
Allows the data input to the map fields by the user to be numeric only. If the terminal does not have the numeric lock option, a specification of NUMERIC is ignored.

PROTECTED
Protects the specified map fields from data entry or modification by the user.

UNPROTECTED
Makes the specified map fields available for data entry or modification by the user.

You cannot specify both UNPROTECTED and SKIP.

DISPLAY
Displays the specified map fields in normal intensity.

DARK
Does not display the specified map fields.

You cannot specify both DARK and DETECT.

BRIGHT
Displays the specified map fields in bright intensity.

Fields assigned the BRIGHT attribute are automatically detectable by a light pen.

DETECT
Makes the specified map fields detectable by a light pen.

Fields assigned the BRIGHT attribute are automatically detectable by a light pen.

MDT
Sets the modified data tag automatically for the map fields when they are displayed.

NOMDT
Does not set the modified data tag automatically for the map fields when they are displayed.

BLINK
3279s only. Displays the specified map fields with blinking characters.

If you specify BLINK, you cannot specify REVERSE-VIDEO or UNDERSCORE.

NOBLINK
3279s only. Does not display the specified map fields with blinking characters.

REVERSE-VIDEO
3279s only. Displays the specified map fields in reverse video (background and character colors reversed).

If you specify REVERSE-VIDEO, you cannot specify BLINK or UNDERSCORE.

NORMAL-VIDEO
3279s only. Displays the specified map fields in normal video.

UNDERSCORE
3279s only. Displays the specified map fields with underlined characters. If you specify UNDERSCORE, you cannot specify BLINK or REVERSE-VIDEO.

NOUNDERSCORE
3279s only. Displays the specified map fields without underlined characters.

NOCOLOR/BLUE/RED/PINK/GREEN/TURQUOISE/YELLOW/WHITE
3279s only. Specifies the color which the specified map fields will be displayed.

5.59.3 Examples

The following examples illustrate the use of the MODIFY MAP statement.

Example 1: The following statement positions the cursor at EMP-ID-0415 and prohibits the user from entering data in any field except EMP-ID-0415 and DEPT-ID-0410:

```
MODIFY MAP EMPMAPLR TEMPORARY
   CURSOR AT DFLD EMP-ID-0415
   FOR ALL BUT DFLD EMP-ID-0415
               DFLD DEPT-ID-0410
   ATTRIBUTES PROTECTED.
```

Example 2: The following statement sets the edit flag on for the TASK-CODE-01 field, thereby overriding automatic editing and error handling for the next MAP IN request:

```
MODIFY MAP EMPMAPLR TEMPORARY
   FOR DFLD TASK-CODE-01
   EDIT IS ERROR.
```

Example 3: Use MODIFY MAP to suppress display of default error messages for fields EMP-ID and DEPT-ID on the current map:

```
MODIFY MAP EMPMAPLR TEMPORARY
   FOR DFLD EMP-ID  DFLD DEPT-ID
      ERROR MESSAGE IS SUPPRESS.
```

Because this MODIFY MAP statement specifies TEMPORARY, error messages for these fields are suppressed for the next mapout only. If PERMANENT (default) were used, the error messages would be suppressed until the program terminated or until the error message specifications were overridden by a subsequent MODIFY MAP statement.

5.59.4 Status codes

After completion of the MODIFY MAP function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4629	An invalid parameter has been passed from the program
4644	The map field is not in the specified map; a possible cause is a reference to an invalid map field subscript
4656	The referenced map contains no data fields

5.60 OBTAIN (LRF)

The OBTAIN statement retrieves the named logical record and places it in the variable-storage location reserved for that logical record. The OBTAIN statement can be issued to retrieve a single logical record, or it can be issued in iterative logic to retrieve all logical records that meet criteria specified in the WHERE clause. Additionally, the OBTAIN statement can specify that the retrieved logical record is to be placed into an alternative variable storage location.

5.60.1 Syntax

```
►►── OBTAIN ─┬──────────┬─ logical-record-name ──────────────────────►
           ├─ FIRST ──┤
           └─ NEXT ◄ ─┘

►─┬──────────────────────────────────┬──────────────────────────────►
  └─ INTO alt-logical-record-location ─┘

►─┬──────────────────────────────┬──────────────────────────────────►
  └─ WHERE boolean-expression ───┘

►─┬──────────────────────────────────────────┬─ . ──────────────────►◄
  └─ ON path-status imperative-statement ────┘
```

5.60.2 Parameters

FIRST

Retrieves the first occurrence of the logical record. OBTAIN FIRST is typically used to retrieve the first in a series of logical-record occurrences following the iterative retrieval of a different series of logical-record occurrences.

NEXT

Retrieves a (subsequent) occurrence of the named logical record, in the order specified by the DBA in the path.

This is the default.

OBTAIN NEXT is typically issued in iterative logic to retrieve a series of logical-record occurrences (possibly including the first).

When LRF receives repeated OBTAIN NEXT commands, it replaces field values in program variable storage with new values obtained through repeated access to the appropriate database records, thereby supplying the program with new occurrences of the desired logical record.

If an OBTAIN FIRST statement is followed by an OBTAIN NEXT statement to retrieve a series of occurrences of the same logical record, the OBTAIN statements must direct LRF to the same path. For this reason, you must ensure that the selection criteria specified in the WHERE clause that accompanies the OBTAIN FIRST and OBTAIN NEXT statements describe the same attributes of the desired logical record.

If the program issues an OBTAIN NEXT statement without issuing an OBTAIN FIRST, or if the last path status returned for the path was LR-NOT-FOUND, LRF

interprets the OBTAIN NEXT as OBTAIN FIRST. After LR-ERROR or a DBA-defined path status, LRF does *not* interpret OBTAIN NEXT as OBTAIN FIRST.

logical-record-name
The name of a logical record defined in the subschema.

INTO
Specifies an alternative location in variable storage into which LRF is to place the retrieved logical record. Any subsequent MODIFY, STORE, or ERASE statements for a logical record placed in *alt-logical-record-location* should name that area as the one from which LRF is to obtain the data to be used to update the logical record.

alt-logical-record-location
A record location defined in the WORKING-STORAGE SECTION or LINKAGE SECTION.

WHERE
Specifies the selection criteria to be applied to the named logical record. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

boolean-expression
The selection criteria to apply.

ON parameter
Specifies the action to be taken depending on the value returned to the LR-STATUS field in the LRC block. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

path-status
The value of the LR-STATUS field in the LRC block which triggers the specified action.

imperative-statement
The action to take.

5.60.3 Example

The following example illustrates the use of the OBTAIN NEXT statement to retrieve a series of logical-record occurrences. The program issues the OBTAIN NEXT statement iteratively to retrieve the first and all subsequent occurrences of the EMP-JOB-LR logical record for all employees in the specified department.

```
GET-AN-ORDER.
  MOVE DEPT-ID-IN TO DEPT-ID-0410.
  OBTAIN NEXT EMP-JOB-LR WHERE DEPT-ID-410 EQ DEPT-ID-0410 OF LR.
   IF LR-STATUS = LR-ERROR
     PERFORM ERROR-PROCESSING.
   IF LR-STATUS = LR-NOT-FOUND
     PERFORM END-PROCESSING.
   .
   .
   .
  GO TO GET-AN-ORDER.
```

OBTAIN NEXT EMP-JOB-LR: The following figure illustrates the information retrieved by each OBTAIN NEXT statement. The EMP-JOB-LR logical record consists of DEPARTMENT, OFFICE, EMPLOYEE, and JOB information.

	DEPARTMENT	EMPLOYEE	OFFICE	JOB
ONE OCCURRENCE OF EMP-JOB-LR	5100	466	8	SNOWBLOWER
	5100	467	8	WINDKEEPER
	5100	334	5	RAINDANCE
	5100	457	8	STURM UND DRANG

5.61 POST

The POST statement alters an event control block (ECB), either by posting it to indicate completion of an event upon which another task is waiting or by clearing it to an unposted status.

5.61.1 Syntax

```
►►── POST ─┬─ EVENT ecb ──────────────────┬─ . ──────────────────────►◄
          └─ EVENT NAME ecb-id ─┬──────┬─┘
                                └─ CLEAR ─┘
```

5.61.2 Parameters

EVENT
Identifies the ECB to be posted.

ecb
The symbolic name of a user-defined area that contains three PIC S9(8) COMP SYNC (fullword) fields. Program-allocated ECBs are cleared by moving zeros to *ecb*.

EVENT NAME
Specifies the ECB to be posted or cleared.

ecb-id
Either the symbolic name of a user-defined field that contains the ECB ID or the ID itself enclosed in quotation marks.

CLEAR
Clears the specified ECB to an unposted status.

Programs posting and waiting on ECBs are responsible for clearing ECBs before issuing subsequent WAIT requests.

5.61.3 Example

The following example requests CA-IDMS to post the event whose ECB identifier is in the FOUND-ECB field and to clear the ECB to an unposted status:

```
POST
   EVENT NAME FOUND-ECB
   CLEAR.
```

5.61.4 Status codes

After completion of the POST function, the only possible value in the ERROR-STATUS field of the IDMS-DC communications block is 0000.

5.62 PUT QUEUE

The PUT QUEUE statement stores a queue record in either the DDLDCRUN or the DDLDCQUE area of the data dictionary. CA-IDMS assigns an ID to the queue record and places it at the beginning or end of its associated queue.

5.62.1 Syntax

```
►►── PUT QUEUE ─┬──────────────────┬──┬──────────┬──────────────────────────────►
                └─ ID queue-id ────┘  ├─ FIRST ──┤
                                      └─ LAST ◄ ─┘

►── FROM queue-data-location ─┬─ TO end-queue-data-location ─┬──────────────────►
                               └─ LENGTH queue-data-length ───┘

►─┬──────────────────────────────────────────────────────────┬──────────────────►
  └─ RETURN RECORD ID INTO return-queue-record-id ────────────┘

►─┬──────────────────────────────┬─ . ──────────────────────────────────────►◄
  └─ RETENTION queue-retention-period ─┘
```

5.62.2 Parameters

ID

Directs the queue record to a previously defined queue.

queue-id

Either the symbolic name of a user-defined field that contains the ID or the ID itself enclosed in quotation marks.

If you do not specify an ID, a null ID of 16 blanks is assumed.

FIRST

Places the queue record at the beginning of the queue.

LAST

Places the queue record at the end of the queue.

This is the default.

FROM

Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry associated with the data to be stored in the queue record.

queue-data-location

The symbolic name of a user-defined field.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the data to be stored in the queue.

end-queue-data-location

The symbolic name of a user-defined dummy byte field or a field that contains a data item not associated with the queue record.

LENGTH

Explicitly defines the length, in bytes, of the area that contains the data to be stored in the queue record.

queue-data-length

Either the symbolic name of a user-defined field that contains the length or the length itself expressed as a numeric constant.

RETURN RECORD ID INTO

Specifies the location in the program to which CA-IDMS will return the system assigned ID of the queue record.

The returned ID is used to reference the queue record in subsequent GET QUEUE and DELETE QUEUE statements.

return-queue-record-id

The symbolic name of a user-defined PIC S9(8) COMP (fullword) field.

RETENTION

Specifies the time in days that CA-IDMS will retain the queue in the data dictionary. At system startup, queues having expired retention periods are deleted automatically by CA-IDMS. The retention period begins when the first record is stored in the queue.

The specified retention period takes precedence over retention periods associated with previously defined queues. The RETENTION parameter is ignored if the record being allocated is not the first record in the queue.

queue-retention-period

Either the symbolic name of a user-defined fixed binary field that contains the retention period or the retention period itself expressed as a numeric constant in the range 0 through 255.

A retention period of 255 indicates that the queue is never to be deleted automatically by CA-IDMS.

5.62.3 Example

The following example requests CA-IDMS to allocate a queue record in the beginning of the RES-Q queue, return the ID of the record to the Q-REC-ID field, and retain the queue for 45 days:

```
PUT QUEUE
   ID 'RES-Q'
   FIRST
   FROM NEW-RES TO END-NEW-RES
   RETURN RECORD ID INTO Q-REC-ID
   RETENTION 45.
```

5.62.4 Status codes

After completion of the PUT QUEUE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
0019	In a DC-BATCH environment, the record size exceeds the value specified in the MAX LENGTH parameter of the BIND TASK statement
4407	Either an I/O error has occurred during processing or the queue upper limit has been reached
4431	The parameter list is invalid; under DC-BATCH, this status indicates that the specified record length exceeds the maximum length based on the packet size
4432	The derived length of the specified queue record is either zero or negative

5.63 PUT SCRATCH

The PUT SCRATCH statement stores or replaces a scratch record in the DDLDCSCR area of the data dictionary. For new records, PUT SCRATCH generates an index entry in a scratch area associated with the issuing task. If the scratch area does not already exist, CA-IDMS allocates it dynamically in the storage pool.

5.63.1 Syntax

```
►►── PUT SCRATCH ─┬──────────────────────────────────────────►
                  └─ AREA ID scratch-area-id ─┘

►── FROM scratch-data-location ─┬─ TO end-scratch-data-location ─┬──►
                                └─ LENGTH scratch-data-length ───┘

►──┬──────────────────────────────────────────────────────────►
   └─ RECORD ID scratch-record-id ─┬──────────────┬
                                   └─ REPLACE ────┘

►──┬──────────────────────────────────────────────┬── . ──────►◄
   └─ RETURN RECORD ID into return-scratch-record-id ─┘
```

5.63.2 Parameters

AREA ID

Specifies the scratch area associated with the record being allocated.

If you do not specify an AREA ID, CA-IDMS assumes an area ID of eight blanks.

scratch-area-id

Either the symbolic name of a user-defined field that contains the ID or the ID itself enclosed in quotation marks.

FROM

Specifies the data to be stored in the scratch record.

scratch-data-location

The symbolic name of a user-defined WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the data.

TO

Indicates the end of the data area to be stored in the scratch record.

end-scratch-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the scratch data being stored.

LENGTH

Defines the length in bytes of the data area.

scratch-data-length

The symbolic name of a user-defined field that contains the length or the length itself expressed as a numeric constant.

RECORD ID

Specifies the ID of the scratch record being stored.

scratch-record-id

Either the symbolic name of a user-defined PIC S9(8) COMP (fullword) field that contains the ID or the ID itself expressed as a numeric constant.

REPLACE

Specifies that the specified scratch record replaces an existing scratch record. If you specify REPLACE, and the specified scratch record does not exist, the record is stored and the status code is set to 0000.

RETURN RECORD ID into

Returns the automatically assigned ID of a scratch record to the program.

return-scratch-record-id

The symbolic name of a user-defined field into which CA-IDMS will place the four-byte scratch record ID.

5.63.3 Example

The following statement requests CA-IDMS to replace the scratch record identified by SCR-REC-ID with data in the WORK-PROC-AREA field:

```
PUT SCRATCH
   FROM WORK-PROC-AREA LENGTH 125
   RECORD ID SCR-REC-ID REPLACE.
```

5.63.4 Status codes

After completion of the PUT SCRATCH function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request to add a scratch record has been serviced successfully
4305	The requested scratch record ID cannot be found
4307	An I/O error has occurred during processing
4317	The request to replace a scratch record has been serviced successfully
4322	The request to add a scratch record cannot be serviced because the specified scratch record already exists in the scratch area and REPLACE has not been specified
4331	The parameter list is invalid.
4332	The derived length of the specified scratch record is either zero or negative

5.64 READ LINE FROM TERMINAL

The READ LINE FROM TERMINAL statement requests a synchronous, line-by-line transfer of data from the terminal to the issuing program.

5.64.1 Syntax

```
►►── READ LINE from TERMINAL ──┬──────────┬──┬──────────────────┬──►
                              └─ ECHO ───┘  └─ NOBACKPAGE ─────┘

►── INTO input-data-location ──┬─ TO end-input-data-location ──────────►
                              └─ MAX LENGTH input-data-max-length ────┘

►──┬──────────────────────────────────────────────────────────┬── . ──►◄
   └─ RETURN LENGTH into input-data-actual-length ────────────┘
```

5.64.2 Parameters

ECHO

3270-type devices only. Saves the line of data being input in the current page (as displayed on the screen). If you do not specify ECHO, data entered will not be retained and will not be available for review by the user.

NOBACKPAGE

3270-type devices only. Tells CA-IDMS not to save previously input pages in a scratch area. If you specify NOBACKPAGE, the user can view only the current page of data. NOBACKPAGE is valid only with the first input request in a line mode session.

INTO

Indicates the WORKING-STORAGE SECTION or LINKAGE SECTION entry reserved for the input data.

input-data-location

The symbolic name of a user-defined field.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION reserved for the input data stream.

end-input-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the data area reserved for the input data stream.

MAX LENGTH

Defines the length in bytes of the input data stream.

If the input data stream is larger than the data area reserved in the WORKING-STORAGE SECTION or LINKAGE SECTION, CA-IDMS truncates the data to fit the available space.

input-data-max-length
Either the symbolic name of a user-defined field that contains the length of the data area or the length itself expressed as a numeric constant.

RETURN LENGTH into
Indicates the location to which CA-IDMS will return the actual length of the input data stream.

input-data-actual-length
The symbolic name of a user-defined field. If the data stream has been truncated, the field will contain the original length before truncation.

5.64.3 Examples

The following examples illustrate the use of the READ LINE FROM TERMINAL statement.

Example 1: The following statement requests CA-IDMS to read the specified data from a 3270-type device into the specified location in the program and to echo the input data on the screen:

```
READ LINE FROM TERMINAL
   ECHO
   INTO EMPL-DATA TO END-EMPL-DATA.
```

Example 2: The following statement requests CA-IDMS to read the specified data into the program but not to save pages associated with the line I/O session:

```
READ LINE FROM TERMINAL
   NOBACKPAGE
   INTO EMPL-DATA MAX LENGTH 8
   RETURN LENGTH INTO REC-DATA-LENGTH.
```

5.64.4 Status codes

After completion of the READ LINE FROM TERMINAL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4707	A logical or permanent I/O error has been encountered in the input data stream
4719	The input area specified for the return of data is too small; the returned data has been truncated to fit the available space
4731	The line request block (LRB) contains an invalid field, indicating a possible error in the program's parameters
4732	The derived length of the specified line input area is zero or negative

Status code	Meaning
4738	The specified 01-level LINKAGE SECTION entry has not been allocated as required A prior GET STORAGE request must be issued
4743	The line I/O session has been canceled; the user has pressed CLEAR (3270s), ATTENTION (2741s), or BREAK (teletypes)

5.65 READ TERMINAL

The READ TERMINAL statement requests a synchronous or asynchronous basic mode data transfer from the terminal to program variable storage.

5.65.1 Syntax

```
►►── READ TERMINAL ──────────────────────────────────────────────►
►┬─ WAIT ◄ ─┬┬─ BUFFER ──────────────────────────────┬──────────►
 │└─ NOWAIT ─┘│└─ MODIFIED FROM POSITION screen-position ─┘
►└─ GET STORAGE ─┘──────────────────────────────────────────────►
►── INTO input-data-location ─┬─ TO end-input-data-location ──┬──►
                              └─ MAX LENGTH input-data-max-length ─┘
►┬──────────────────────────────────────────────────────┬─ . ──►◄
 └─ RETURN LENGTH into input-data-actual-length ─┘
```

5.65.2 Parameters

WAIT

Specifies that the read operation will be synchronous; the issuing task will automatically relinquish control to CA-IDMS and must wait for completion of the read operation before processing can continue.

This is the default.

NOWAIT

Specifies that the read operation will be asynchronous; the issuing task will continue executing.

If you specify NOWAIT, the program must issue a CHECK TERMINAL request (described earlier in this chapter) before performing any other I/O operations.

MODIFIED

3270-type devices only. Reads all modified fields in the terminal buffer into variable storage without requiring the user to signal completion of data entry.

BUFFER

3270-type devices only. Executes a READ BUFFER command that reads the entire contents of the terminal buffer into variable storage without requiring the user to signal completion of data entry.

FROM POSITION

Defines the buffer address (screen position) at which the read will start.

screen-position

Either the symbolic name of a user-defined PIC S9(4) COMP SYNC (halfword) field or the address itself enclosed in quotation marks.

GET STORAGE

Synchronous requests only. Acquires an input buffer for the data being read into the program; CA-IDMS allocates the required storage when the read operation is complete.

INTO

Specifies the 01-level WORKING-STORAGE SECTION or LINKAGE SECTION entry of the input data stream.

If you also specify GET STORAGE, the data area reserved for the input data stream must be an unallocated 01-level LINKAGE SECTION entry.

If you do not specify GET STORAGE, the data area must be a WORKING STORAGE or previously allocated LINKAGE SECTION entry.

input-data-location

The symbolic name of a user-defined field.

TO

Indicates the end of the data area reserved for the input data stream.

end-input-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the data area reserved for the input data stream.

MAX LENGTH

Defines the length, in bytes, of the data area reserved for the input data stream.

If the input data stream is larger than the specified WORKING-STORAGE SECTION or LINKAGE SECTION entry, CA-IDMS truncates the data to fit the available space.

input-data-max-length

Either the symbolic name of a user-defined field that contains the length of the data area or the length itself expressed as a numeric constant.

RETURN LENGTH into

Indicates the location to which CA-IDMS will return the actual length of the input data stream.

input-data-actual-length

The symbolic name of a user-defined field. If the data stream has been truncated, *input-data-actual-length* contains the original length before truncation.

5.65.3 Example

The following statement illustrates a basic mode request to read data from the terminal to the specified location in variable storage:

```
READ TERMINAL
   WAIT
   INTO TERM-LINE TO END-TERM-LINE.
```

5.65.4 Status codes

After completion of the READ TERMINAL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4519	The input area specified for the return of data to the issuing program is too small; the returned data has been truncated to fit the available space
4527	A permanent I/O error has occurred during processing
4528	The dial-up line for the terminal has been disconnected
4531	The terminal request block (TRB) contains an invalid field, indicating a possible error in the program's parameters
4532	The derived length of the specified input data area is zero or negative
4535	Storage for the input buffer cannot be acquired because the specified 01-level LINKAGE SECTION entry has been previously allocated; no I/O has been performed
4537	Storage for the input buffer cannot be acquired because the specified entry is defined in the WORKING-STORAGE SECTION rather than in the LINKAGE SECTION; no I/O has been performed
4538	The specified 01-level LINKAGE SECTION entry has not been previously allocated and the GET STORAGE option has not been specified; no I/O has been performed
4539	The terminal device associated with the issuing task is out of service

5.66 READY

The READY statement prepares a database area for access by DML functions and specifies that area's usage mode.

The DBA can specify default usage modes in the subschema. Run units that use such a subschema need not issue any READY statements; the areas are automatically readied in the predefined usage modes. However, if a run unit issues a READY statement for one area, it must issue READY statements for all areas that it will access.

The specified usage mode can be qualified with a PROTECTED option to prevent concurrent update or an EXCLUSIVE option to prevent concurrent use of areas by other run units executing under the CA-IDMS central version. Each area can be readied in its own usage mode. Usage modes can be changed by executing a FINISH statement (see 5.42, “FINISH” on page 5-110 earlier in this chapter) then starting a new run unit by issuing a BIND RUN-UNIT statement, the appropriate BIND RECORD statements, and a READY statement specifying the new usage mode.

When the run unit readies database areas, all areas can be readied with a single READY statement or each area to be accessed can be readied individually. All areas affected explicitly or implicitly by the DML statements issued by the run unit must be readied. Other areas included in the subschema need not be readied.

The READY statement can appear anywhere within an application program; however, to avoid runtime deadlock, the best practice is to ready all areas before issuing any other DML statements.

The READY statement is used in both the navigational and the non-navigational environments.

5.66.1 Syntax

```
►►── READY ──┬──────────────┬──────────────────────────────────────►
           └─ area-name ─┘

►──┬──────────────────────────────────────────────────────────────┬──.──►◄
   └─ USAGE-MODE is ──┬──────────────┬──┬─ RETRIEVAL ◄ ─┬
                      ├─ PROTECTED ─┤  └─ UPDATE ──────┘
                      └─ EXCLUSIVE ─┘
```

5.66.2 Parameters

area-name

The name of an area included in the subschema.

By default, if you do not specify an area, READY will open all areas in the subschema.

USAGE-MODE IS

Specifies the usage mode in which the area will be opened.

PROTECTED

Prevents concurrent update of the area by run units executing under the same central version. Once a run unit has readied an area with the PROTECTED option, no other run unit can ready that area in any UPDATE usage mode until the first run unit releases it by means of the FINISH statement (see 5.42, "FINISH" on page 5-110 earlier in this chapter). A run unit cannot ready an area with the PROTECTED option if another run unit has readied the area in UPDATE usage mode or with the EXCLUSIVE option.

By default, if you do not specify PROTECTED or EXCLUSIVE, the areas will be opened in shared mode.

EXCLUSIVE

Prevents concurrent use of the area by any other run unit executing under the CA-IDMS central version. Once a run unit has readied an area with the EXCLUSIVE option, no other run unit can ready that area in any usage mode until the first run unit releases it.

By default, if you do not specify PROTECTED or EXCLUSIVE, the areas will be opened in shared mode.

RETRIEVAL

Opens the area for retrieval only and allows other concurrently executing run units to open the same area in any non-exclusive usage mode.

This is the default.

UPDATE

Opens the area for both retrieval and update and allows other concurrently executing run units to open the same area in any usage mode other than exclusive or protected.

Note: If a READY statement would result in a usage mode conflict for an area, while running under the CA-IDMS central version, the run unit issuing the READY is placed in a wait state on the first functional database call.

Note: Modification statements involving areas opened in one of the update usage modes are not valid if they affect sets that include records in an area opened in one of the retrieval usage modes.

5.66.3 Example

The following statement readies all subschema areas in a usage mode of PROTECTED UPDATE:

```
READY USAGE-MODE IS PROTECTED UPDATE.
```

5.66.4 Status codes

After completion of the READY function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
0910	The subschema specifies an access restriction that prohibits readying the area in the specified usage mode
0923	The named area is not in the subschema
0928	The run unit has attempted to ready an area that has been readied previously
0966	The area specified is not available for update. If the 0966 status code is ignored, subsequent attempts to access the area will return a 01 or 09 minor code. Probable causes for the return of the status code are as follows: ▪ If running in local mode, the area is locked against update ▪ If running under the central version, the area is not available to the program in the desired access mode
0970	The database or journal file will not ready properly; a JCL error is the probable cause
0971	The page range for the area being readied could not be found in the DMCL
0978	A wait for an area would cause a deadlock. It is recommended that all areas be readied either before the first functional call is issued or that all programs ready areas in the same order.

5.67 RETURN

The RETURN statement retrieves the database key for an indexed record without retrieving the record itself, thus establishing currency in the indexed set. The record's symbolic key is moved into the data fields within the record in program variable storage. The contents of all non-key fields for the record are unpredictable after the execution of the RETURN verb. Optionally, the program can indicate that the symbolic key can be moved into some other specified variable storage location.

Current of index is established by:

- Successful execution of the RETURN statement, which sets current of index at the index entry from which the database key was retrieved.
- A status code of 1707 (end of index), which set currency on the index owner. The DBMS returns the owner's db-key.
- An status code of 1726 (index entry not found), which sets current of index as follows:
  - Between the two entries that are higher and lower than the specified value
  - After the highest entry, if the specified value is higher than all index entries
  - Before the lowest entry, if the specified value is lower than all index entries

The RETURN statement is used in both the navigational and the non-navigational environments.

5.67.1 Syntax

```
►►── RETURN dbkey FROM index-set-name ─┬─ CURRENCY ──────────────────┬──►
                                       ├─ FIRST currency ────────────┤
                                       ├─ LAST currency ─────────────┤
                                       ├─ NEXT currency ─────────────┤
                                       ├─ PRIOR currency ────────────┤
                                       └─ USING index-key-value ─────┘

►─┬──────────────────────────┬─ . ──────────────────────────────────►◄
  └─ KEY into symbolic-key ──┘
```

5.67.2 Parameters

db-key
The symbolic name of a user-defined PIC S9(8) COMP SYNC (fullword) field.

FROM
Identifies the indexed set from which the specified database key is to be returned.

index-set-name
The name of the indexed set.

CURRENCY
Retrieves the database key for the current index entry.

FIRST currency
Retrieves the database key for the first index entry.

LAST currency
Retrieves the database key for the last index entry.

NEXT currency
Retrieves the database key for the index entry following current of index. If the current of index is the last entry, status code 1707 (end of index) is returned.

PRIOR currency
Retrieves the database key for the index entry preceding current of index. If the current of index is the first entry, status code 1707 (end of index) is returned.

USING
Retrieves the database key for the first index entry with the specified symbolic key.

index-key-value
The symbolic key to be used.

If no such entry exists, status code 1726 (index entry not found) is returned.

KEY into
Saves the symbolic key (CALC, sort, or index) of the specified record.

symbolic-key
The name of a user-defined alphanumeric field into which the symbolic key of the specified record will be returned. *Symbolic-key* must be large enough to contain the largest contiguous or noncontiguous symbolic key.

If the 'KEY into' clause is not specified, the symbolic key will be moved into the corresponding fields in the user record's storage.

The precompiler views an incorrectly formatted RETURN statement as a COBOL RETURN function and does not flag the error. The incorrect RETURN DML statement is passed to the COBOL compiler without expansion into a CALL statement, causing compile-time errors.

5.67.3 Example

The following RETURN statement retrieves the database key for the first index entry in the EMP-LNAME-NDX set and moves the record's symbolic key into the NDX-LNAME-SYM-KEY field.

```
RETURN INT-INDEX-KEY FROM EMP-LNAME-NDX
   FIRST CURRENCY
   KEY INTO NDX-LNAME-SYM-KEY.
```

5.67.4 Status codes

After completion of the RETURN function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
1701	The area in which the object record or its index owner record participates has not been readied
1707	Either the end of the indexed set has been reached or the indexed set is empty
1725	Currency has not been established for the specified indexed set
1726	The record cannot be found

5.68 ROLLBACK

The ROLLBACK statement requests recovery of the recovery unit (that part of a run unit that falls between two checkpoints) and, optionally, allows the run unit to continue database accessing activities without reissuing the necessary BIND and READY commands. Recovery restores database records, scratch areas, and queues to their condition at the most recent begin or commit checkpoint written for the issuing run unit or task.

The way in which recovery is effected depends on whether the run unit is running under the CA-IDMS central version or in local mode and whether the journal file is on disk or tape:

- **Recovery is effected automatically** if the run unit issuing the ROLLBACK statement is running under the central version and the central version is journaling to a disk file. The central version continues to process other applications during recovery. If the last checkpoint was not established by a COMMIT TASK or FINISH TASK function, the database, scratch, and queue areas may have checkpoints that do not occur at the same point in the journal file.
- **Recovery is not effected automatically** under the following circumstances:
  - If the run unit is running under the central version and the central version is journaling to a tape file
  - If the run unit is executing in local mode

  In these cases, the ROLLBACK statement causes the affected areas to be flagged for subsequent recovery by the IDMSRBCK utility, which rolls back the database. If the journal file is on disk, it must be archived to tape by using the IDMSAJNL utility before IDMSRBCK is run. (For detailed information on IDMSRBCK and IDMSAJNL, refer to *CA-IDMS Utilities*).

The ROLLBACK statement is used in both the navigational and the non-navigational environments.

5.68.1 Syntax

```
►►── ROLLBACK ──┬──────────┬──┬──────────────┬── . ──────────────────────►◄
                └─ TASK ───┘  └─ CONTINUE ───┘
```

5.68.2 Parameters

TASK
Restores all run units associated with the issuing task to the most recent checkpoint.

CONTINUE
Central version only. Rolls back the issuing run unit (ROLLBACK CONTINUE) or all run units associated with the issuing task (ROLLBACK TASK

CONTINUE), but does not terminate the run unit. Database access can be resumed without reissuing BIND and READY statements.

5.68.3 Example

The following statement clears the system buffers, writes an abort checkpoint to the journal file, restores database records to the most recent checkpoint, nullifies all currencies, and terminates the run unit:

```
ROLLBACK.
```

5.68.4 Status codes

After completion of the ROLLBACK function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
5031	The specified request is invalid; the program may contain a logic error
5096	Too many run units are currently active; check the DC system log for details
5097	An invalid status has been received from DBIO/DBMS; check the DC system log for details

5.69 SEND MESSAGE

The SEND MESSAGE statement sends a message to another terminal or user or to a group of terminals or users defined as a *destination* during system generation. The SEND MESSAGE function does not employ the data dictionary message area; instead, CA-IDMS places each message in a queue, sending the message to the appropriate terminal only when it is possible to do so without disrupting executing tasks. Typically, CA-IDMS sends queued messages to a terminal the next time the ENTER NEXT TASK CODE message is displayed.

5.69.1 Syntax

```
►►── SEND MESSAGE ──┬──────────┬──────────────────────────────────────►
                    ├─ ONLY ◄─┤
                    └─ ALWAYS ─┘

 ►── TO ──┬─ DEST ID destination-id ──┬──────────────────────────────►
          ├─ USER ID user-id ─────────┤
          └─ LTERM ID lterm-id ───────┘

 ►── FROM message-location ──┬─ TO end-message-location ─┬─ . ──────►◄
                             └─ LENGTH message-length ───┘
```

5.69.2 Parameters

ONLY

Sends the message immediately if the destination, user, or terminal is available, and does not queue the message for subsequent transmission if the destination, user, or terminal is not available.

This is the default.

Note: If ONLY is specified with the DEST ID option (described below) and if some, but not all, of a group of users or terminals in the destination are available, CA-IDMS will send the message to those available. The sender will not be aware of any unsuccessful transmissions.

ALWAYS

Sends the message immediately if the destination, user, or terminal is available, and queues the message for later transmission if the destination, user, or terminal is not available.

TO DEST ID

Identifies the recipient of the message as a destination. The specified destination must have been defined during system generation.

destination-id

Either the symbolic name of a user-defined field that contains the destination ID or the ID itself enclosed in quotation marks.

TO USER ID

Identifies the user to receive a message. The specified user can be signed on to any terminal.

user-id
Identifies the user to receive the message. The specified user can be signed on to any terminal. *User-id* is the symbolic name of a 32-byte user-defined field that contains the user-id.

TO LTERM ID
Identifies the logical terminal to receive the message.

lterm-id
Either the symbolic name of a user-defined field that contains the terminal ID or the ID itself enclosed in quotation marks.

FROM
Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the text of the message to be sent.

message-location
The symbolic name of a user-defined field.

TO
Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the message text.

end-message-location
The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the message text.

LENGTH
Defines the length in bytes of the message text.

message-length
Either the symbolic name of a user-defined field that contains the length or the length itself expressed as a numeric constant.

5.69.3 Examples

The following examples illustrate the use of the SEND MESSAGE statement.

Example 1: The following statement requests CA-IDMS to send the message in the TERM-MESS field to the logical terminal KENNEDYA:

```
SEND MESSAGE ALWAYS
   TO LTERM ID 'KENNEDYA'
   FROM TERM-MESS TO END-TERM-MESS.
```

Example 2: The following statement requests CA-IDMS to send the message in the TERM-MESS field to the user field:

```
MOVE 'KYJOE2' to USER32.
     SEND MESSAGE
        TO USER ID USER32
        FROM TERM-MESS TO END-TERM-MESS.
```

Example 3: The following statement requests CA-IDMS to send the message in the TERM-MESS field to the destination ALL:

```
SEND MESSAGE
   TO DEST ID 'ALL'
   FROM TERM-MESS TO END-TERM-MESS.
```

5.69.4 Status codes

After completion of the SEND MESSAGE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4907	An I/O error has occurred during processing.
4921	The specified message recipient has not been defined.
4931	The parameter list is invalid.
4932	The derived length of the specified message data area is zero or negative.
4934	The specified variable storage area that contains the message text is in the LINKAGE SECTION but is not an 01-level entry.
4938	The specified WORKING-STORAGE SECTION or LINKAGE SECTION entry has not been allocated, as required. A GET STORAGE request must be issued.

5.70 SET ABEND EXIT

The SET ABEND EXIT (STAE) statement establishes or cancels linkage to an abend routine to which CA-IDMS passes control if the issuing task terminates abnormally. Any program within a task can establish an abend exit; however, only one abend exit is in effect at any given time for each task level. If more than one abend exit has been established, CA-IDMS recognizes the exit associated with the last STAE request issued.

When a task terminates abnormally (following either a processing error or an ABEND request), abend exits for the program that was executing at the time of the abend and for all higher-level programs will be executed before the task is terminated. The program can prevent CA-IDMS from executing abend exits automatically either by coding the EXITS IGNORED clause in an ABEND request or by coding a DC RETURN request in the abend routine.

5.70.1 Syntax

```
►►── SET abend EXIT ─┬─ on PROGRAM program ─┬─ . ──────────────────────►◄
                    └─ OFF ────────────────┘
```

5.70.2 Parameters

on PROGRAM

Specifies the program to which control is to transfer if the issuing task terminates abnormally.

program

Either the symbolic name of a user-defined field that contains the program name or the name itself enclosed in quotation marks.

Note: CA-IDMS does not check to determine if the specified program name is valid when the STAE request is issued. Rather, if the program is not found or is otherwise unloadable when CA-IDMS attempts to execute it, the STAE request will be ignored.

OFF

Cancels any previously issued STAE request for the issuing task level.

5.70.3 Examples

The following examples illustrate the use of the SET ABEND EXIT statement.

Example 1: The following statement establishes an abend exit that will execute the program ABENDRTN if the issuing task terminates abnormally:

```
SET ABEND EXIT ON PROGRAM 'ABENDRTN'.
```

Example 2: The following statement cancels all abend exits previously established at the task level of the issuing program:

```
SET ABEND EXIT OFF.
```

5.70.4 Status codes

After completion of the SET ABEND EXIT function, the only possible value in the ERROR-STATUS field of the IDMS-DC communications block is 0000.

5.71 SET TIMER

The SET TIMER statement defines an event that is to occur after a specified time interval or cancels the effect of a previously issued SET TIMER request. Using the SET TIMER function, a program can:

- Delay task processing for a specified period of time
- Post an ECB at the end of a specified period of time
- Initiate a task at the end of a specified period of time

5.71.1 Syntax

```
►►── SET TIMER ─┬─ WAIT ──┬──────────────────────────────────────────►
               ├─ POST ──┤
               ├─ START ─┤
               └─ CANCEL ┘

►─┬──────────────────────────────────────┬──────────────────────────►
  └─ INTERVAL time-interval seconds ─────┘

►─┬──────────────────────────────────────┬──────────────────────────►
  └─ EVENT post-ecb ─────────────────────┘

►─┬──────────────────────────────────────┬──────────────────────────►
  └─ TASK CODE task-code ─┬──────────────┬┘
                          └─ PRIORITY priority ─┘

►─┬──────────────────────────────────────┬──────────────────────────►
  └─ TIMER ID ice-address ───────────────┘

►─┬──────────────────────────────────────────────────────────┬──────►◄
  └─ DATA FROM task-data-location ─┬─ TO end-task-data-location ─┬┘ .
                                   └─ LENGTH task-data-length ───┘
```

5.71.2 Parameters

WAIT

Places the issuing task in a wait state and instructs CA-IDMS to redispatch the issuing task after the specified time interval elapses. Because WAIT relinquishes control until the time interval has elapsed, a subsequent SET TIMER request cannot be used to cancel this WAIT request.

POST

Posts a user-specified ECB after the specified time interval elapses; the issuing task continues to run. If POST is specified, the EVENT parameter (described below) must also be specified.

START

Initiates a user-specified task after the specified time interval elapses. If START is specified, the TASK CODE parameter (described below) must also be specified.

CANCEL

Cancels the effect of a previously issued SET TIMER request.

INTERVAL ... seconds

WAIT, POST, START requests only. Specifies the time, in seconds, from the issuance of a SET TIMER request at which the requested event will occur.

time-interval

Either the symbolic name of a user-defined field that contains the time interval or the interval itself expressed as a numeric constant.

EVENT

POST requests only. Specifies the ECB to be posted.

post-ecb

The symbolic name of a user-defined area that contains three PIC S9(8) COMP SYNC (fullword) fields.

TASK CODE

START requests only. Specifies the task to be initiated.

task-code

Either the symbolic name of the user-defined field that contains the task code or the task code itself enclosed in quotation marks.

The specified task code must have been defined to CA-IDMS during system generation or at runtime with a DCMT VARY DYNAMIC TASK command.

PRIORITY

Specifies a dispatching priority for the task.

priority

Either the symbolic name of a user-defined field that contains the priority or the priority itself expressed as a numeric constant in the range 0 through 240. The new task's priority defaults to the priority defined for that task code.

TIMER ID

POST, START, CANCEL requests only. Specifies the address of the interval control element (ICE) associated with the timed event.

ice-address

The symbolic name of a user-defined PIC S9(8) COMP SYNC (fullword) field. If either POST or START has been specified, *ice-address* references a field to which CA-IDMS will return the ICE address. If CANCEL has been specified, *ice-address* references the field that contains the ICE address returned by CA-IDMS following a SET TIMER POST or SET TIMER START request.

Note: The TIMER ID parameter must be specified with SET TIMER POST and SET TIMER START requests if the program is to issue subsequent SET TIMER CANCEL requests.

DATA FROM

START requests only. Identifies the user data to be passed to the new task.

task-data-location

The symbolic name of a user-defined field that identifies the beginning of an area containing the data item(s) to be passed.

TO

Indicates the end of the data area being passed to the new task.

end-task-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the data area being passed.

LENGTH

Specifies the length in bytes of the data area.

task-data-length

Either the symbolic name of a user-defined WORKING-STORAGE SECTION or LINKAGE SECTION field that contains the length of the data area or the length itself expressed as a numeric constant.

Note: The data passed to the new task consists of two bytes containing the length of the original data followed by the original data itself. This may be accessed by means of a LINKAGE SECTION entry corresponding to the data and a USING clause in the PROCEDURE DIVISION header.

5.71.3 Examples

The following examples illustrate the use of the SET TIMER statement.

Example 1: The following statement requests CA-IDMS to place the issuing task in a wait state and redispatch it after nine seconds have elapsed:

```
SET TIMER WAIT
   INTERVAL 9 SECONDS.
```

Example 2: The following statement requests CA-IDMS to post the event PODB after five seconds have elapsed:

```
SET TIMER POST
   INTERVAL 5 SECONDS
   EVENT 'PODB'
   TIMER ID TMR-ID.
```

Example 3: The following statement requests CA-IDMS to start the SPSG task after five seconds have elapsed and to pass the specified data to that task:

```
SET TIMER START
   INTERVAL 5 SECONDS
   TASK CODE 'SPSG'
   TIMER ID TMR-ID
   DATA FROM PASSGR LENGTH REC-LENGTH.
```

Example 4: The following statement requests CA-IDMS to cancel the timed event referenced by TMR-ID:

```
SET TIMER CANCEL
   TIMER ID TMR-ID.
```

5.71.4 Status codes

After completion of the SET TIMER function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3516	The interval control element (ICE) specified for a SET TIMER CANCEL request cannot be found
3532	The derived length of the data area is negative

5.72 SNAP

The SNAP statement requests a memory snap of one or all of the following areas:

- **Task areas** — Includes all resources associated with the issuing task, as well as the task control element (TCE) and dispatch control element (DCE) for the task. Information displayed by the snap is formatted with headers.
- **System areas** — Includes areas for all tasks and CA-IDMS internal control blocks. Task areas are not itemized separately. Information displayed by the snap is formatted with headers.
- **Specified locations in memory** — Includes one or more areas of memory specifically requested by location and length. The information displayed is not formatted with headers.

The areas requested in the SNAP request are written to the DC system log file, which is defined during system generation as a sequential data set or a data dictionary area.

5.72.1 Syntax

```
►►── SNAP ──┬──────────────────────┬──┬──────────┬──────────────────────────────────►
              └─ TITLE is title ─┘  ├─ ALL ─────┤
                                    ├─ SYSTEM ──┤
                                    └─ TASK ────┘

►──┬──────────────────────────────────────────────────────────────────────┬── . ──►◄
   └─↓─ FROM begin-snap-location ──┬─ TO end-snap-location ─┬─┘
                                   └─ LENGTH snap-length ───┘
```

5.72.2 Parameters

TITLE is

Specifies the title to be printed at the beginning of each page of the snap.

title

The symbolic name of a user-defined field that contains the title.

A title must contain 134 characters; the first character is reserved for use by CA-IDMS, and the second character must be a valid ASA carriage control character (blank, 0, 1, +, or -).

ALL

Writes a snap of both task and system areas. Areas associated with the issuing task are formatted separately from the system areas. (Task areas are also included with the system areas but are not itemized by task.)

SYSTEM

Writes a snap of system areas.

TASK

Writes a snap of task areas.

FROM

Writes a snap of the specified memory location.

begin-snap-location

The symbolic name of a user-defined field indicating the starting location of the area to be snapped.

TO

Indicates the end of the area to be snapped.

end-snap-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the area requested in the snap.

LENGTH

Defines the length in bytes of the area to be included in the snap.

snap-length

Either the symbolic name of a user-defined field that contains the length of the data area or the length itself expressed as a numeric constant.

Note: If snap-length is greater than 100, some COBOL compilers may produce errors. In this case, either use a symbolic name that contains the length, or use the FROM/TO form of the verb.

5.72.3 Example

The following example illustrates a SNAP statement that requests CA-IDMS to write a memory snap of the specified memory location:

```
SNAP TITLE IS SNAP-TITLE
   FROM WS-START TO WS-END.
```

5.72.4 Status codes

After completion of the SNAP function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4032	The derived length of the specified snap storage area is zero or negative

5.73 STARTPAGE

The STARTPAGE statement initiates a paging session. It can be followed by any number of DML commands, including MAP IN and MAP OUT commands. The map paging session is terminated by an ENDPAGE command (or by another STARTPAGE command, if one is encountered before an ENDPAGE command).

Note: Only *one* pageable map can be handled by the statements enclosed by a given STARTPAGE/ENDPAGE pair.

5.73.1 Syntax

```
►►── STARTPAGE session map-name ──┬──────────┬──────────────────────────────►
                                 ├─ WAIT ───┤
                                 ├─ NOWAIT ◄─┤
                                 └─ RETURN ─┘

►──┬─ BACKPAGE ◄─┬──┬─ UPDATE ◄─┬──┬─ AUTODISPLAY ◄─┬──────────────────────►◄
   └─ NOBACKPAGE ─┘  └─ BROWSE ──┘  └─ NOAUTODISPLAY ─┘
```

5.73.2 Parameters

map-name

Specifies the pageable map to be used for the session.

WAIT

Specifies that runtime mapping automatically handles paging transactions that do not cause data to be updated. Control is passed to the program when the user presses a control key that requests an update or nonpaging operation.

NOWAIT

Specifies that runtime mapping automatically handles all paging and update transactions.

This is the default.

Control is passed to the program only when neither an update nor paging request is made when the operator presses a control key.

RETURN

Specifies that runtime mapping does not handle any terminal transactions in the paging session. Control is passed to the program whenever the operator presses a control key.

Note: Runtime mapping does not update program variable storage unless a MAP IN command is issued. In cases where the operator can update data, it is recommended that WAIT or RETURN be specified for the session so that data can be retrieved as it is updated.

BACKPAGE

Specifies that the operator can display previous pages of detail occurrences.

This is the default.

NOBACKPAGE

Specifies that the operator cannot display any page of detail occurrences with a page number lower than the current page number. Modifications made on a given page of the map must be requested by MAP IN statements in the application program before a MAP OUT RESUME command is issued. The previous page of detail occurrences is deleted from the session scratch record when a new map page is displayed.

Note: NOBACKPAGE cannot be assigned if UPDATE and NOWAIT are specified for the session.

UPDATE

Specifies that the user can modify variable map fields, subject to restrictions specified for the map either at map definition time or by statements in the program.

This is the default.

BROWSE

Specifies that the user can modify only the page and response fields (if any) of the map. The MDTs for variable fields on the map can be set on only according to specifications made either in the map definition or by statements in the program.

AUTODISPLAY

Enables automatic display of the pageable map's first page.

This is the default.

NOAUTODISPLAY

Disables automatic display of the pageable map's first page. You manually display the page by using a MAP OUT RESUME statement.

5.73.3 Examples

Initiating a paging session: The following statement initiates a paging session in which the operator can page forward and backward within the pageable map but can make no modifications:

```
STARTPAGE SESSION EMPMAPPG NOWAIT BACKPAGE BROWSE.
```

Overriding automatic display: Use STARTPAGE to override automatic display for the first page of pageable map EMPMAPPG:

```
STARTPAGE SESSION EMPMAPPG NOAUTODISPLAY.
```

5.73.4 Status codes

After completion of the STARTPAGE function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.

Status code	Meaning
4604	A paging session was already in progress when this STARTPAGE command was received. An implied ENDPAGE was processed before this STARTPAGE was successfully executed.

5.74 STORE

The STORE statement performs the following functions:

- Acquires space and a database key for a new record occurrence in the database
- Transfers the value of the appropriate elements from program variable storage to the specified record occurrence in the database
- Connects the new record occurrence to all sets for which it is defined as an automatic member

Before execution of the STORE statement, the following conditions must be satisfied:

- All areas affected either implicitly or explicitly must be readied in one of the update usage modes (see 5.66, "READY" on page 5-201, earlier in this chapter).
- All control elements (that is, CALC and sorted set control fields) must be initialized by the program.
- If the record being stored has a location mode of DIRECT, the contents of DIRECT-DBKEY (positions 197-200 of the IDMS communications block, as described in Chapter 3, "Communications Blocks and Error Detection" on page 3-1) must be initialized with a suggested db-key value or a null db-key value of -1.
- If the record is to be stored in a native VSAM relative-record data set (RRDS), the contents of DIRECT-DBKEY must be initialized with the relative record number that represents the location within the data set where the record is to be stored.
- All sets in which the named record is defined as an automatic member, and the owner record of each of those sets must be included in the subschema. Sets for which the named record is defined as a manual member need not be defined in the subschema since the STORE statement does not access those sets. (An automatic member is connected automatically to the selected set occurrence when the record is stored; a manual member is not connected automatically to the selected set occurrence.)
- If the record being stored has a location mode of VIA, currency must be established for that VIA set, regardless of whether the record being stored is an automatic or manual member of that set. Current of the VIA set provides the suggested page for the record being stored.
- Currency must be established for all set *occurrences* in which the stored record will participate as an *automatic* member. Depending on set order, the STORE statement uses currency as follows:
  - If the named record is defined as a member of a set that is ordered FIRST or LAST, the record that is current of set establishes the set occurrence to which the new record will be connected.
  - If the named record is defined as a member of a set that is ordered NEXT or PRIOR, the record that is current of set establishes the set occurrence into

which the new record will be connected *and* determines its position within the set.

– If the named record is defined as a member of a sorted set, the record that is current of set establishes the set occurrence into which the new record will be connected. The DBMS compares the sort key of the new record with the sort key of the current record of set to determine if the new record can be inserted into the set by movement in the next direction. If it can, the current of set remains positioned at the record that is current of set and the new record is inserted. If it cannot, the DBMS finds the owner of the current of set (not necessarily the current occurrence of the owner record type) and moves as far forward in the next direction as is necessary to determine the logical insertion point for the new record.

A record is stored in the database based on the location mode specified in the schema definition of the record. The location modes are as follows:

- **CALC** — The record being stored is placed on or near a page calculated by CA-IDMS from a control element (the CALC key) in the record.
- **VIA** — The record being stored is placed either as close as possible to the current of set (if current of set and member record occurrences share a common page range) or in the same relative position in the member record's page range as the current of set is in its associated page range (if current of set and member record occurrences do not share a common page range).
- **DIRECT** — The record being stored is placed on or near a user-specified page as determined by the value in the DIRECT-DBKEY field of the IDMS communications block. If DIRECT-DBKEY contains a valid db-key for the record being stored, the DBMS assigns a db-key on the same page if space is available to the new record occurrence. Otherwise, it assigns the next available db-key, subject to the page-range limits of the record being stored. If DIRECT-DBKEY contains a value of -1, the first db-key available in the page range in which the record is to be stored is assigned to the record. In any case, the db-key of the stored record occurrence is returned to DBKEY (positions 13-16 in the IDMS communications block). The contents of DIRECT-DBKEY remain unchanged.

Currency: Following successful execution of a STORE statement, the stored record becomes current of run unit, its record type, its area, and all sets in which it participates as owner or automatic member.

5.74.1 Syntax

```
►►── STORE record-name . ──────────────────────────────────────────►◄
```

5.74.2 Parameters

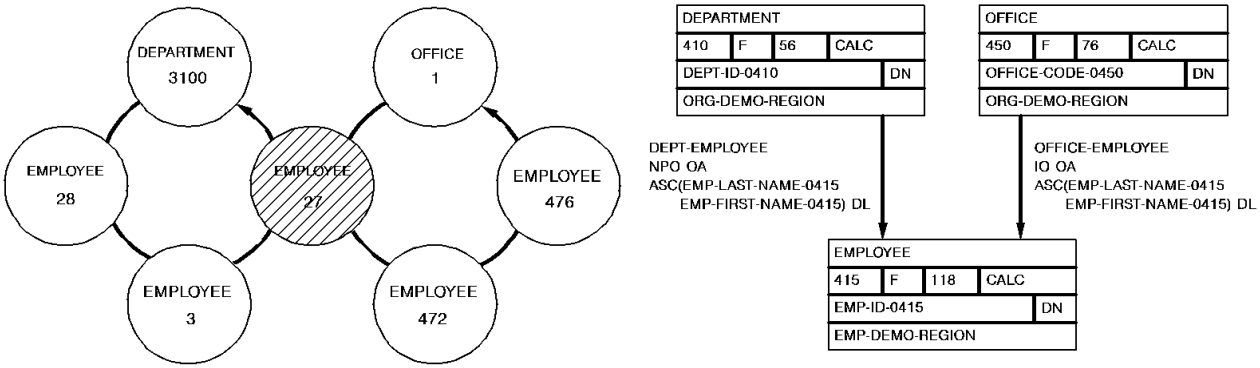
record-name

The name of a record type included in the subschema. The current occurrence of the record type will be moved from variable storage to the database, connected to an occurrence of each set for which it is defined as an automatic member, and established as the owner of a set occurrence for each set in which it is defined as an owner.

The ordering rules for each set govern the insertion point of the specified record in the set.

5.74.3 Example

The following figure illustrates the steps necessary to add a new EMPLOYEE record to the database. Since EMPLOYEE is defined as an automatic member of both the DEPT-EMPLOYEE and OFFICE-EMPLOYEE sets, currency must be established in each of those sets before issuing the STORE. The first two DML statements establish OFFICE 1 and DEPARTMENT 3100 as current of the OFFICE-EMPLOYEE and DEPT-EMPLOYEE sets, respectively. When EMPLOYEE 27 is stored, it is connected automatically to each set.



	CURRENCIES RUN UNIT, RECORD, SET, AREA							
	RUN UNIT	DEPARTMENT	EMPLOYEE	OFFICE	DEPT-EMPLOYEE	OFFICE-EMPLOYEE	ORG-DEMO-REGION	EMP-DEMO-REGION
MOVE OFFICE-CODE-IN TO OFFICE-CODE. FIND CALC OFFICE.	1			1		1	1	
MOVE DEPT-ID-IN TO DEPT-ID. FIND CALC DEPARTMENT.	3100	3100		1	3100	1	3100	
STORE EMPLOYEE.	27	3100	27	1	27	27	3100	27

5.74.4 Status codes

After completion of the STORE function, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
1201	The area in which the named record is to be stored has not been readied
1202	The suggested DIRECT-DBKEY value is not within the page range for the named record
1204	The OCCURS DEPENDING ON item is either less than zero or greater than the maximum number of occurrences of the control element
1205	Storage of the record would violate a duplicates-not-allowed option for a CALC record, a sorted set, or an index set
1208	The named record is not in the subschema; The program has probably invoked the wrong subschema.
1209	The named record's area has not been readied in one of the update usage modes

Status code	Meaning
1210	The subschema specifies an access restriction that prohibits storage of the named record
1211	The record cannot be stored in the area because of insufficient space
1212	The record cannot be stored because no db-key is available; this is a system internal error.
1218	The record has not been bound
1221	An area other than the area of the named record occurrence has been readied with an incorrect usage mode
1225	A set occurrence has not been established for each set in which the named record is to be stored
1233	At least one set in which the record participates as an automatic member has not been included in the subschema
1253	The subschema definition of an indexed set does not match the indexed set's physical structure in the database
1254	Either the prefix length of an SR51 record is less than zero or the data length is less than or equal to zero
1255	An invalid length has been defined for a variable length record
1260	A record occurrence encountered in the process of connecting automatic sets is inconsistent with the set named in the ERROR-SET field of the IDMS communications block; probable causes include: a broken chain and improper database description.
1261	The record cannot be stored because of broken chains in the database

5.75 STORE (LRF)

The STORE statement updates the database with field values for a logical-record occurrence. STORE does not necessarily result in storing new occurrences of all or any of the database records that participate in the logical record; the path selected to service a STORE logical-record request performs whatever database access operations the DBA has specified to service the request. For example, if an existing department gets a new employee, only the new employee information will be stored in the database; the department information need not be stored in the database because it already exists.

LRF uses field values present in the variable storage location reserved for the logical record to make the appropriate updates to the database. You can optionally name an alternative storage location from which the new field values are to be obtained to perform the requested store operation.

5.75.1 Syntax

```
►►── STORE logical-record-name ──────────────────────────────────►
►┬──────────────────────────────┬──────────────────────────────►
 └─ FROM alt-logical-record-location ─┘
►┬──────────────────────────┬──────────────────────────────────►
 └─ WHERE boolean-expression ─┘
►┬──────────────────────────────────┬─ . ──────────────────────►◄
 └─ ON path-status imperative-statement ─┘
```

5.75.2 Parameters

logical-record-name
The name of a logical record defined in the subschema.

FROM
Specifies an alternative variable storage location that contains the field values to be used to make appropriate updates to the database. When storing a logical record that has previously been retrieved into an alternative variable storage location, use the FROM clause to name the same area specified in the OBTAIN request.

alt-logical-record-location
A record location defined in the WORKING-STORAGE SECTION or LINKAGE SECTION.

WHERE
Specifies selection criteria to be applied to the object logical record. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

boolean-expression
The selection criteria to apply.

ON parameter

Specifies the action to be taken depending on the value returned to the LR-STATUS field in the LRC block. For details on coding this clause, see 5.84, “Logical-record clauses” on page 5-258 at the end of this chapter.

path-status

The value of the LR-STATUS field in the LRC block which triggers the specified action.

imperative-statement

The action to take.

5.75.3 Example

The following example illustrates the steps necessary to store a new EMP-INSURANCE-LR for a given employee:

```
MOVE EMP-ID-IN TO EMP-ID-0415.
MOVE INS-PLAN-IN TO INS-PLAN-CODE-0435.
MOVE S-DATE-IN TO SELECTION-DATE-0400
MOVE T-DATE-IN TO TERMINATION-DATE-0400
MOVE TYPE-IN TO TYPE-0400
MOVE PLAN-IN TO INS-PLAN-CODE-0400.
STORE EMP-INSURANCE-LR.
```

The following figure illustrates the new occurrence of the EMP-INSURANCE-LR, which consists of EMPLOYEE 149, INS-PLAN 001, and COVERAGE 'D'. The bottom COVERAGE occurrence represents the only data physically added to the database.

	EMPLOYEE	INS-PLAN	COVERAGE
NEW OCCURRENCE OF EMP-INS-LR	149	002	M
	149	002	F
	149	001	(D)

5.76 TRANSFER CONTROL

The TRANSFER CONTROL statement is used to:

- Establish linkage with a specified program and to pass control and an optional parameter list to that program. The program issuing the TRANSFER CONTROL RETURN request expects return of control at the instruction immediately following the TRANSFER CONTROL statement when the linked program terminates or issues a DC RETURN request.
- Transfer control and an optional parameter list to a specified program. The program issuing the TRANSFER CONTROL NORETURN request does *not* expect return of control.

5.76.1 Syntax

```
►►── TRANSFER control to program ─┬──────────────────────────────────►
                                  │┌─ RETURN ──────────┐
                                  │└─ LINK ────────────┘
                                  │
                                  └┬─ NORETURN ─┬─ ◄ ─┐
                                   └─ XCTL ─────┘

►──┬──────────────────────────┬──┬─ . ─┐──────────────────────────►◄
   │                          │  └─ ; ─┘
   └─ USING ─┬─ parameter ─┬─┘
             ↓             ↑
```

5.76.2 Parameters

program

Either the symbolic name of a user-defined field that contains the program name or the name itself enclosed in quotation marks.

RETURN (LINK)

Establishes linkage with the specified program, expecting return of control.

RETURN and LINK are synonyms and can be used interchangeably.

NORETURN (XCTL)

Transfers control to the specified program, not expecting return of control.

This is the default.

NORETURN and XCTL are synonyms and can be used interchangeably.

USING

Passes one or more parameters (data items) to the program receiving control.

parameter

The symbolic name of a user-defined field that contains the names of the data items to be passed. Multiple parameter specifications must be separated with a blank.

If the USING clause is specified with the RETURN option, the data items being passed are defined in either the WORKING-STORAGE SECTION or

the LINKAGE SECTION of the calling program, and in the LINKAGE SECTION of the linked program.

If the USING clause is specified with the NORETURN option, the data items being passed are defined in the LINKAGE SECTION of *both* the calling program and the program receiving control. In either case, the program receiving control must have a corresponding USING clause and parameter list as part of its PROCEDURE DIVISION header.

5.76.3 Examples

The following examples illustrate the use of the TRANSFER CONTROL statement.

Example 1: The following statement requests CA-IDMS to transfer control to the program in the PROGRAM-NAME field; the issuing program expects return of control:

```
TRANSFER CONTROL TO PROGRAM-NAME
   LINK.
```

Example 2: The following statement requests CA-IDMS to transfer control to PROGRAMD and passes three data items (FIELD-1, FIELD-2, and FIELD-3) to the program; the issuing program does not expect return of control:

```
TRANSFER CONTROL TO 'PROGRAMD'
   NORETURN
   USING FIELD-1 FIELD-2 FIELD-3.
```

5.76.4 Status codes

After completion of the TRANSFER CONTROL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3020	The request cannot be serviced because an I/O, program-not-found, or potential deadlock error has occurred

5.77 WAIT

The WAIT statement relinquishes control either to CA-IDMS, pending completion of one or more events, or to a higher priority ready-to-run task. If control is relinquished to wait for the completion of one or more events, an event control block (ECB) must be defined for each event. If an ECB is already posted when the WAIT is issued, the task is redispatched immediately and control does not pass to another task.

5.77.1 Syntax

```
►►── WAIT ┬──────────────────────────────────────┬── . ──────────────►◄
        │                                      │
        │   ┌──────────────────────────────┐   │
        │   ↓                              │   │
        ├─┬─ LONG ◄ ─┬─┬─ EVENT ecb ─────────┬─┤
        │ └─ SHORT ──┘ │                     │ │
        │              └─ EVENT NAME ecb-id ─┘ │
        │                                      │
        └─ REDISPATCH ─────────────────────────┘
```

5.77.2 Parameters

LONG

Specifies that the wait is expected to be long-term.

This is the default.

LONG should be specified for all waits expected to last a second or more (for example, terminal input).

SHORT

Specifies that the wait is expected to be short-term. SHORT should be specified for all waits expected to last less than a second (for example, a disk I/O).

EVENT

Defines one or more ECBs upon which the task will wait.

ecb

The symbolic name of a user-defined area that contains three PIC S9(8) COMP SYNC (fullword) fields.

Multiple EVENT parameters must be separated by at least one blank.

EVENT NAME

Specifies the ECB upon which the task will wait.

ecb-id

Either the symbolic name of a user-defined field that contains the ECB ID or the ID itself enclosed in quotation marks.

You cannot specify multiple EVENT NAMEs.

REDISPATCH

Specifies that the issuing task wishes to relinquish control to any higher priority ready-to-run task before being redispatched.

5.77.3 Example

The following example requests a short-term wait on the event PODB:

```
WAIT
   SHORT
   EVENT NAME 'PODB'.
```

5.77.4 Status codes

After completion of the WAIT function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
3101	To wait on the specified ECB would cause a deadlock

5.78 WRITE JOURNAL

The WRITE JOURNAL statement writes a task-defined record to the journal file. Records written to the journal file with the WRITE JOURNAL function will be available to user-defined exit routines during a task- or system-initiated rollback.

5.78.1 Syntax

```
►►── WRITE JOURNAL ─┬──────────┬──────────┬─────────┬──────────────────────►
                    ├─ WAIT ───┤          ├─ SPAN ◄─┤
                    └─ NOWAIT ◄┘          └─ NOSPAN ┘

 ►── FROM record-location ─┬─ TO end-record-location ─┬─ . ──────────────►◄
                           └─ LENGTH record-length ───┘
```

5.78.2 Parameters

WAIT

Specifies that the issuing task will wait for completion of the physical I/O associated with the WRITE JOURNAL function before resuming execution. This option will cause CA-IDMS to write a partially filled buffer to the journal file.

NOWAIT

Specifies that the issuing task will not wait for completion of the WRITE JOURNAL function; the journal record will remain in a storage buffer until a future request necessitates writing the buffer to the journal file.

This is the default.

SPAN

Indicates that CA-IDMS will write the record across several journal file blocks, if necessary.

This is the default.

NOSPAN

Indicates that CA-IDMS will write the record to a single journal file block; if it is longer than the journal block, the record will be split.

When a record is shorter than a journal file block, based on space available in the current journal block, CA-IDMS will either place the record in the block, split it across multiple blocks (SPAN), or write it to a new block after the current block is written (NOSPAN).

The following considerations apply to using an exit routine to retrieve journal file records during recovery:

- If a WRITE JOURNAL statement issued before a failure specified the SPAN option, records may have been written across several journal blocks. To retrieve these records, the exit routine will be invoked once for each segment of each record to be retrieved.
- If a WRITE JOURNAL statement issued before a failure specified the NOSPAN option and records written to the journal file are shorter than

journal blocks, the exit routine need only be concerned with the complete records.

Note: In general, the SPAN option provides better space utilization in the journal file because it increases the average fullness of each block.

FROM

Defines the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the record to be written to the journal file.

record-location

The symbolic name of a user-defined field.

TO

Indicates the end of the record area to be written to the journal file.

end-record-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the record being written to the journal file.

LENGTH

Defines the length in bytes of the record to be written to the journal file.

record-length

Either the symbolic name of the user-defined field that contains the length or the length itself expressed as a numeric constant.

5.78.3 Example

The following statement requests CA-IDMS to write the JOURNAL-DATA record to the journal file, spanning it across several blocks if necessary:

```
WRITE JOURNAL SPAN
   FROM JOURNAL-DATA TO END-JOURNAL-DATA.
```

5.78.4 Status codes

After completion of the WRITE JOURNAL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
5002	Storage is not available for the required control blocks
5032	The derived length of the specified journal record is zero or negative
5097	An invalid status has been received from DBIO/DBMS; check the DC system log for details

5.79 WRITE LINE TO TERMINAL

The WRITE LINE TO TERMINAL statement requests CA-IDMS to transfer data from program variable storage to a terminal. WRITE LINE TO TERMINAL also establishes, modifies, and deletes page header lines.

Data transfers requested by WRITE LINE TO TERMINAL statements can be synchronous or asynchronous:

- Following a **synchronous** request, control passes to CA-IDMS, which places the issuing task in an inactive state. For non-3270 devices, control does not return to the issuing program until the WRITE LINE TO TERMINAL request is complete. For 3270-type devices, all lines of output are saved in a buffer; the buffer is not transmitted to the terminal until it is full.

  The transfer of a line to the buffer will result in a processing delay; however, control returns to the program immediately following the request. If the line of data fills the buffer, the entire page of data must be transmitted to the terminal. In this case, control does not return to the issuing program until the user responds by pressing ENTER. Thus, the program is made conversational.

- Following an **asynchronous** request, control returns immediately to the issuing program. Thereafter, each time the program issues a line mode I/O request, CA-IDMS automatically checks to determine if the last asynchronous request has completed and, therefore, whether a new data transfer can be initiated.

  With asynchronous requests, programs can buffer all required pages of output without suspending task execution during the actual transmission of data. However, the task can optionally terminate itself, thereby freeing resources and allowing the user to review the buffered output.

CA-IDMS processes I/O requests in the sequence received from the task; thus, if a program issues a synchronous WRITE LINE TO TERMINAL request after issuing one or more asynchronous requests, CA-IDMS will complete all I/O requests before returning control to the issuing program.

The WRITE LINE TO TERMINAL request issued automatically by CA-IDMS to empty partially filled buffers upon completion of a task is synchronous; therefore, the user can view all screens and catch up with processing at that time. If an application allows the user to interrupt or terminate processing at some point within a task, a synchronous WRITE LINE TO TERMINAL request must be issued to suspend processing while awaiting an operator response.

5.79.1 Syntax

```
►►── WRITE LINE to TERMINAL ──────────────────────────────────────►
►─┬─ WAIT ◄ ─┬─┬─ NEWPAGE ─┬─┬─ NOBACKPAGE ─┬──────────────────────►
  └─ NOWAIT ─┘ └─ ERASE ───┘ └──────────────┘
►── FROM output-data-location ─┬─ TO end-output-data-location ─┬──────►
                               └─ LENGTH output-data-length ───┘
►─┬──────────────────────────┬─ . ──────────────────────────────►◄
  └─ HEADER header-line-number ─┘
```

5.79.2 Parameters

WAIT

Specifies that the write operation is synchronous; the issuing task automatically relinquishes control and must wait for completion of the output operation before processing can continue.

This is the default.

NOWAIT

Specifies that the write operation is asynchronous; the issuing task continues executing.

NEWPAGE (ERASE)

Write the output data line beginning on a new page. For 3270-type devices, the NEWPAGE option forces CA-IDMS to output the contents of the current buffer, even if the buffer is not full.

NEWPAGE and ERASE are synonyms and can be used interchangeably.

NOBACKPAGE

3270-type devices only. Does not keep pages output in a scratch area. If NOBACKPAGE is specified, the user can view only the current page of output. NOBACKPAGE is valid only with the first I/O request in a line mode session.

FROM

Identifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the data to be transferred to the terminal device or the page header line being created, modified, or deleted.

output-data-location

The symbolic name of a user-defined field.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the output data stream.

end-output-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the output data.

LENGTH

Defines the length in bytes of the output data area.

output-data-length
Either the symbolic name of a user-defined field that contains the length of the data area or the length itself expressed as a numeric constant.

Note: If the WRITE LINE TO TERMINAL statement is being used to delete a page header line, *output-data-length* must be zero.

HEADER
Specifies the number of the page header line being created, modified, or deleted.

header-line-number
Either the symbolic name of a user-defined field that contains the header line number or the header line number itself expressed as a numeric constant.

5.79.3 Examples

The following examples illustrate the use of the WRITE LINE TO TERMINAL statement.

Example 1: The following statement defines the value of a data area as a header to be displayed at the top of each new page written to the terminal:

```
WRITE LINE TO TERMINAL
  FROM EMPL-HEAD TO END-EMPL-HEAD
  HEADER 1.
```

Example 2: The following statement requests CA-IDMS to write the value in the specified data area to a new page on the terminal:

```
WRITE LINE TO TERMINAL
   NOWAIT
   FROM EMPL-RPT LENGTH 60.
```

5.79.4 Status codes

After completion of the WRITE LINE TO TERMINAL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4707	A logical or permanent I/O error has occurred during processing.
4731	The line request block (LRB) contains an invalid field, indicating a possible error in the program's parameters.
4732	The derived length of the specified line output area is zero or negative.
4738	The specified 01-level LINKAGE SECTION entry has not been allocated as required. A GET STORAGE request must be issued.

Status code	Meaning
4743	The line I/O session has been canceled; the user has pressed CLEAR (3270s), ATTENTION (2741s), or BREAK (teletypes).

5.80 WRITE LOG

The WRITE LOG statement retrieves a predefined message from the message area of the data dictionary and optionally writes the message to a specified location in program variable storage. Retrieved messages are sent to the destination specified in the message definition; typical destinations are the operator's console and the DC system log file. If the operator's console has been defined as the message destination, the WRITE LOG statement can request a reply. When a reply is requested, control is not returned to the issuing task until the reply is received. For more more information on global messages, refer to the *IDD DDDL Reference*.

The message ID specified in the WRITE LOG statement is a seven-digit number. The first six (most significant) digits make up the actual message ID used to retrieve the message from the data dictionary; the seventh digit is a severity code. This severity code is predefined in the dictionary and is retrieved along with the message text to indicate the action to be taken after the message is written to the log:

Severity Level	**CA-IDMS Action**
0	Return control to the issuing program and continue processing
1	Snap all task resources and return control to the issuing program
2	Snap all system areas and return control to the issuing program
3	Snap all task resources and abend the task with a task abend code of D002
4	Snap all system areas and abend the task with a task abend code of D002
5	Terminate the task with a task abend code of D002
6	Undefined
7	Undefined
8	Snap all system areas and abend the system with a system abend code of 3996
9	Terminate the system with a system abend code of 3996

If a WRITE LOG statement specifies a message ID that is not in the data dictionary, CA-IDMS will use a prototype message but will perform the action associated with the severity code specified in the WRITE LOG request.

Messages stored in the data dictionary can contain symbolic parameters. Symbolic parameters, identified by an ampersand (&) followed by a two-digit numeric identifier, can appear in any order within the message. The WRITE LOG statement can specify replacement values for one or more symbolic parameters; however, the position of replacement values within the WRITE LOG request must correspond exactly with the

two-digit numeric identifier in the message text. For example, the first value specified corresponds to &01, the second to &02, and so forth.

5.80.1 Syntax

```
►►── WRITE LOG MESSAGE ID message-id ──────────────────────────────►
►┬──────────────────────────────────────────────────────────────────►
 │
 └─ PARMS ─┬─ FROM parameter-location ─┬─ TO end-parameter-location ─┬─┘
           │                           └─ LENGTH parameter-length ───┘
           ↓
►┬──────────────────────────────────────────────────────────────────►
 │
 └─ REPLY INTO reply-location ─┬─ TO end-reply-location ──────────┐
                               └─ MAX LENGTH reply-max-length ────┘
►┬──────────────────────────────────────────────────────────────────►
 │
 └─ MESSAGE PREFIX is 'DC' ──┘
►┬──────────────────────────────────────────────────────────────────►─
 │
 └─ TEXT INTO text-return-location ─┬─ TO end-text-return-location ─┬──
                                    └─ MAX LENGTH text-max-length ──┘
►──────────────────────────────────────────────────────────┬─ . ──────►◄
 ┌────────────────────────────────────────────────────────┘
 │
 ├─ MESSAGE PREFIX is ─┬─ YES ◄ ──────────┬─┐
 │                     ├─ NO ─────────────┤ │
 │                     └─ message-prefix ─┘ │
 │                                          │
 └─ TEXT is ONLY ───────────────────────────┘
```

5.80.2 Parameters

MESSAGE ID

Specifies the message ID. The first six digits specify the ID of the message; the seventh digit specifies the message's severity code.

message-id

Either the symbolic name of a user-defined PIC S9(8) COMP (fullword) field that contains the message ID or the ID itself expressed as a numeric constant.

Message IDs 000001 through 900000 are reserved for use by CA-IDMS; the WRITE LOG statement can specify any number in the range 900001 through 999999.

Caution: The message length must be seven digits. The system will always interpret the last digit as the severity level. If you request message 987659 and do not code a severity level of zero (that is, 9876590) you are actually requesting that message 098765 be written to the log and that the system should be terminated with a 3996 abend code.

When messages are added to the data dictionary for use with the WRITE LOG statement, they are assigned an eight-character identification number; the first two characters are DC. A request for message 987654 retrieves DC987654.

PARMS FROM

Supplies replacement values for one or more symbolic parameters stored with the message text.

parameter-location

The symbolic name of a user-defined field that contains the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the replacement parameter.

This field must begin with a one-byte field into which the system places the length of the adjacent field. The value in the length does not include the length byte.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the replacement parameter.

end-parameter-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the replacement parameter.

LENGTH

Defines the length, in bytes, of the replacement parameter.

parameter-length

Either the symbolic name of a user-defined field that contains the length or the length itself expressed as a numeric constant.

REPLY INTO

Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the area reserved for a reply to the message issued by the WRITE LOG request.

reply-location

The symbolic name of a user-defined field.

This field must begin with a one-byte field into which the system places the length of the adjacent field. The value in the length does not include the length byte.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry reserved for the reply.

end-reply-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the reply.

MAX LENGTH

Defines the maximum length, in bytes, of the area reserved for the reply.

reply-max-length

Either the symbolic name of a user-defined field that contains the length or the length itself expressed as a numeric constant.

TEXT INTO

Specifies that the contents of the named message, along with any replacement parameters, are to be written to the issuing program.

text-return-location

The symbolic name of a user-defined 1 through 132 character alphanumeric field that contains the WORKING-STORAGE SECTION or LINKAGE SECTION entry to which the message text is to be returned.

This field must begin with a 1-byte field into which the system places the length of the adjacent field. The value in the length does not include the length byte.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry reserved for the text.

end-text-return-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the returned text.

MAX LENGTH

Defines the maximum length in bytes of the WORKING-STORAGE SECTION or LINKAGE SECTION entry reserved for the returned message text.

text-max-length

Either the symbolic name of a user-defined field that contains the text length or the length itself expressed as a numeric constant.

MESSAGE PREFIX is

Defines the format of the message prefix. To override the default DC prefix, specify any one or two characters for *message-prefix*. To suppress a prefix, specify blanks.

YES

Indicates that the message text is preceded by:

```
IDMS ppnnnnnnn Vssssss REPLYnn
```

- *pp* is the prefix specified in the MESSAGE PREFIX parameter
- *nnnnnnn* is the message number
- V*ssssss* is the system number
- REPLY*nn* is the message's system-supplied reply number (present only if the REPLY parameter is used)

This is the default.

NO

Indicates that the message text is preceded by:

```
ppnnnnnnn
```

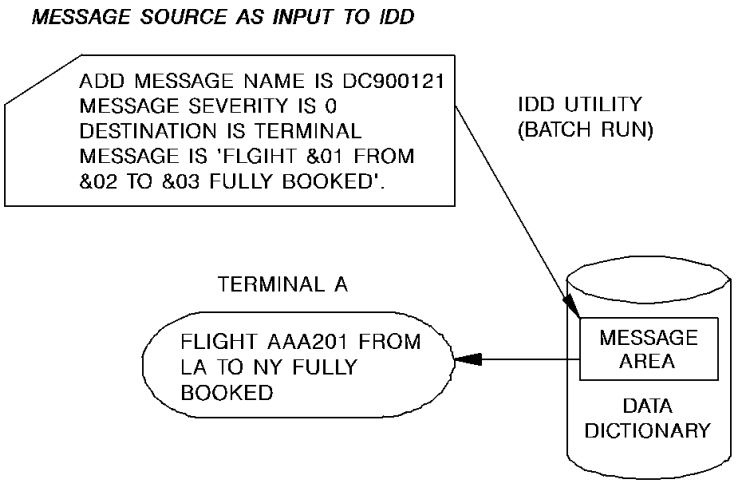
- *pp* is the prefix specified in the MESSAGE PREFIX parameter
- *nnnnnnn* is the message number

TEXT is ONLY
Indicates that the message text is output with no prefix.

5.80.3 Example

The following figure illustrates a WRITE LOG statement that supplies three replacement parameters.

WRITE LOG activities: Task A issues a WRITE LOG request for message 900121, specifying values to replace symbolic parameters &01, &02, and &03 stored with the message text. CA-IDMS sends the message to its destination, which has been defined as the logical terminal associated with the issuing task.



WRITE LOG REQUEST

```
WRITE LOG MESSAGE ID 9001210
PARMS FROM FLT-NO TO END-FLT-NO
      FROM DPT-CITY TO END-DEPT-CITY
      FROM ARV-CITY TO END-ARV-CITY.
```

```
WHERE:  FLT-NO    = AAA201
        DPT-CITY  = LA
        ARV-CITY  = NY
```

5.80.4 Status codes

After completion of the WRITE LOG function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully

Status code	Meaning
3623	No storage or resource control element (RCE) can be allocated for the specified reply area
3624	The maximum number of outstanding replies has been exceeded; a maximum of 98 messages can be awaiting reply at a given time
3631	The parameter list is invalid

5.81 WRITE PRINTER

The WRITE PRINTER statement requests CA-IDMS to transmit data from a task to a terminal defined to the system as a printer device during system generation. Any type of terminal can be designated as a printer; however, the terminal is usually a hard-copy device.

CA-IDMS does not transmit data directly from program variable storage to the terminal. Rather, data is passed to a queue maintained by CA-IDMS, and from the queue to the printer. The data stream passed to the queue by the WRITE PRINTER request contains only data; CA-IDMS adds the necessary line and device control characters when it writes the data to the printer.

Note: Native mode data streams (that is, those that contain device-control information as well as user data) can also be transmitted with a WRITE PRINTER request. This capability is useful in formatting reports for 3280-type printers.

Each line of data transmitted in a WRITE PRINTER request is considered a record. Each record is associated with a **report** in the print queue. A report consists of one or more records. Any task can have up to 256 active print reports. A program can issue multiple WRITE PRINTER requests, each specifying a different report. Because CA-IDMS maintains the records associated with each report individually, records associated with one report are not interspersed with records associated with other reports when printed.

The WRITE PRINTER request can direct reports to print classes and to destinations:

- During system generation, one or more **print classes** are associated with each terminal designated as a printer. For each report, the first record transmitted to the print queue by means of a WRITE PRINTER request establishes the print class for that report. The report will be printed on the first available printer that is assigned the same print class.
- **Destinations** are groups of terminals, printers, or users. If destinations have been defined during system generation, the WRITE PRINTER request can direct a report to a destination. Reports sent to printer destinations are printed on the first available printer for the destination, regardless of print class.

CA-IDMS prints a report only when that report is completed, either explicitly as part of a WRITE PRINTER request or implicitly when the issuing task terminates.

Note: Normal task termination, a FINISH TASK request, or a COMMIT TASK request will end all of the task's reports. Queued reports are made eligible for printing.

Abnormal task termination (abend) or a ROLLBACK TASK request will cause any queued reports belonging to the task to be deleted.

5.81.1 Syntax

```
►►── WRITE PRINTER ─┬─┬─ NEWPAGE ─┬─┬─ ENDRPT ─┬──────────────────────────►
                    │ └─ ERASE ───┘ │          │
►─┬─┬─────────── FROM message-location ─┬─ TO end-message-location ─┬──────►
  │ └─ NATIVE ─┘                        └─ LENGTH message-length ───┘
  │
  └─ SCREEN CONTENTS ──────────────────────────────────────────────────────┘

►─┬─ COPIES ─┬─ 1 ◄ ──────┬─┬─ REPORT ID ─┬─ 1 ◄ ──────┬──────────────────►
  │          └─ copy-count ─┘ │             └─ report-id ─┘
  │
►─┬─ CLASS printer-class ──────────────────┬──────────────────────────────►
  │                                        │
  └─ DESTINATION printer-destination ─┬─┬─┬─┘
                                      │ │ │
                                      └─ ALL ─┘

►─┬─ HOLD ─┬─┬─ KEEP ─┬─ . ──────────────────────────────────────────────►◄
  │        │ │        │
```

5.81.2 Parameters

NEWPAGE (ERASE)

Specifies that the data stream will be printed beginning on a new page.

NEWPAGE and ERASE are synonyms and can be used interchangeably.

ENDRPT

Indicates that the data stream constitutes the last record in the specified report. When ENDRPT is specified, the report can be printed before the issuing task has terminated. However, the program must issue a COMMIT TASK request to signal CA-IDMS to print the ended report. A subsequent WRITE PRINTER request with the same report ID will start a separate report.

NATIVE

Specifies that the data stream contains device control characters. If NATIVE is not specified, CA-IDMS automatically inserts the necessary characters.

FROM

Specifies the WORKING-STORAGE SECTION or LINKAGE SECTION entry of the data to be transmitted to the print queue.

message-location

The symbolic name of a user-defined field.

TO

Indicates the end of the WORKING-STORAGE SECTION or LINKAGE SECTION entry that contains the data to be transmitted to the print queue.

end-message-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the output data.

LENGTH

Defines the length, in bytes, of the data stream.

message-length

Either the symbolic name of a user-defined field that contains the length of the data or the length itself expressed as a numeric constant.

SCREEN CONTENTS

3270-type devices only. Specifies that the contents of the currently displayed screen are to be transmitted to the print queue.

COPIES

Specifies the number of copies of the report to be printed.

copy-count

Either the symbolic name of a user-defined field that contains the copy count or the count itself expressed as a numeric constant.

The count must be an integer in the range 1 through 255; the default is one.

REPORT ID

Specifies the identifier of the report to be printed.

report-id

Either the symbolic name of a user-defined field that contains the report ID or the ID itself expressed as a numeric constant. *Report-id* must be an integer in the range 1 through 255; the default is one.

CLASS

Specifies the print class to which the report will be assigned.

The CLASS or DESTINATION should be specified only for the first line of each report. If no class or destination is specified, the default print class assigned to the issuing task's physical terminal during system generation will be used.

printer-class

Either the symbolic name of a user-defined field that contains the print class or the class itself expressed as a numeric constant.

Valid print classes are 1 through 64; the default is 1.

DESTINATION

Specifies the destination to which the report will be routed.

The CLASS or DESTINATION should be specified only for the first line of each report. If no class or destination is specified, the default print class assigned to the issuing task's physical terminal during system generation will be used.

printer-destination

Either the symbolic name of a user-defined field that contains the destination or the destination itself enclosed in quotation marks.

The specified destination must have been defined during system generation.

ALL

Specifies that the report is to be printed on all of the printers belonging to the specified destination. The report will be printed, one printer at a time, and

saved until it has been printed on each of the printers associated with the destination.

HOLD
Specifies that a queued report will be held without being printed. The specified report will be held until a DCMT VARY REPORT *report-name* RELEASE command is issued at run time.

KEEP
Specifies that CA-IDMS will keep the report in the print queue after it has been printed. The report can be released for printing with a DCMT VARY REPORT *report-name* RELEASE command. In this way, the report can be printed several times. A KEPT report can be deleted from the print queue manually (using a DCMT VARY REPORT *report-name* DELETE command at run time) or automatically (when the queue retention period has been exceeded).

5.81.3 Examples

The following examples illustrate the use of the WRITE PRINTER statement.

Example 1: The following statement requests CA-IDMS to associate the data in the specified location with report 32 in the print queue and to print it beginning on a new page. Report 32 will print on the first terminal assigned to print class 3 when the program notifies CA-IDMS that the report is complete or when the task terminates.

```
WRITE PRINTER
   NEWPAGE
   FROM PASSGR-RPT TO END-PASSGR-RPT
   REPORT ID 32
   CLASS 3.
```

Example 2: The following statement requests CA-IDMS to print three copies of the current screen contents on a printer associated with destination A and to keep the contents of the report in the print queue after it has printed:

```
WRITE PRINTER
   SCREEN CONTENTS
   COPIES 3
   DESTINATION 'A'
   KEEP.
```

5.81.4 Status codes

After completion of the WRITE PRINTER function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully.
4807	An I/O error has occurred while placing the record in the print queue.

Status code	Meaning
4818	The current DC system definition contains no logical terminal-printer associations.
4821	The specified printer destination is undefined or is not a printer.
4831	The parameter list is invalid.
4832	The derived length of the specified printer output data area is zero or negative.
4838	The specified 01-level LINKAGE SECTION entry has not been allocated, as required. A GET STORAGE request for the specified variable must be issued before the WRITE PRINTER statement.
4845	A WRITE PRINTER SCREEN CONTENTS request cannot be serviced because the terminal associated with the issuing task is not a 3270-type device or is a remote 3270 device running under TCAM.
4846	A terminal I/O error has occurred.

5.82 WRITE TERMINAL

The WRITE TERMINAL statement requests a synchronous or asynchronous data transfer from program variable storage to the terminal buffer.

5.82.1 Syntax

```
►►── WRITE TERMINAL ──┬──────────┬──────────────────────────────────►
                      ├─ WAIT ◄─┤
                      └─ NOWAIT ─┘

►──┬──────────────────────────┬──┬──────────────────────────────────►
   ├─┬─ NEWPAGE ─────────────┬─┤  └─ FREE STORAGE ─┘
   │ └─ ERASE ─────────────┘ │
   │                         │
   └─ EAU ─────────────────┬─┘
     └─ ERASE ALL UNPROTECTED ─┘

►── FROM output-data-location ─┬─ TO end-output-data-location ─┬─ . ──────►◄
                                └─ LENGTH output-data-length ───┘
```

5.82.2 Parameters

WAIT

Specifies that the write operation will be synchronous; the issuing task will automatically relinquish control to CA-IDMS and wait for completion of the write operation before continuing processing.

This is the default.

NOWAIT

Specifies that the write operation will be asynchronous; the issuing task will continue executing.

Note: If NOWAIT is specified, the program must issue a CHECK TERMINAL request (described earlier in this chapter) before performing any other I/O operation.

NEWPAGE (ERASE)

Activates the page-eject (SYSINOUT devices) or erase-write (3270-type devices) mechanism to erase the contents of a screen. If NEWPAGE is not specified, the WRITE TERMINAL request will write over rather than erase data displayed on the terminal.

NEWPAGE and ERASE are synonyms and can be used interchangeably.

EAU (ERASE ALL UNPROTECTED)

3270-type devices only. Activates the erase-all-unprotected mechanism. Following a WRITE TERMINAL EAU function, only protected fields remain on the terminal. If EAU is specified, the FROM clause (described below) need not be specified.

EAU and ERASE ALL UNPROTECTED are synonyms and can be used interchangeably.

FREE STORAGE

Releases the output buffer associated with the data being written to the terminal. The storage area being freed must have been acquired by a GET STORAGE statement (described earlier in this chapter) or the GET STORAGE option of a previously issued READ TERMINAL or WRITE THEN READ TERMINAL request. If FREE STORAGE is not specified, CA-IDMS does not free the storage associated with the output buffer until the issuing task terminates.

FROM

Specifies the 01-level WORKING-STORAGE SECTION or LINKAGE SECTION entry of the output data stream.

output-data-location

The symbolic name of a user-defined field. If FREE STORAGE is specified, *output-data-location* must be an 01-level LINKAGE SECTION entry.

TO

Indicates the end of the output data stream and is specified following the last data-item entry in *output-data-location*.

end-output-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the output data stream.

LENGTH

Defines the length in bytes of the output data stream.

output-data-length

Either the symbolic name of a user-defined field that contains the length of the data area or the length itself expressed as a numeric constant.

5.82.3 Example

The following statement illustrates an asynchronous basic mode request to write data to the terminal from the specified location in program variable storage:

```
WRITE TERMINAL
   NOWAIT
   FROM TERM-LINE LENGTH 72.
```

5.82.4 Status codes

After completion of the WRITE TERMINAL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4525	The output operation has been interrupted; the user has pressed ATTENTION or BREAK
4526	A logical error (for example, an invalid control character) has been encountered in the output data stream

Status code	Meaning
4527	A permanent I/O error has occurred during processing
4528	The dial-up line for the terminal has been disconnected
4531	The terminal request block (TRB) contains an invalid field, indicating a possible error in the program's parameters
4532	The derived length of the specified output data area is zero or negative
4537	Storage for the output buffer cannot be freed because the specified data area is defined in the WORKING-STORAGE SECTION rather than in the LINKAGE SECTION.
4539	The terminal associated with the issuing task is out of service

5.83 WRITE THEN READ TERMINAL

The WRITE THEN READ TERMINAL statement requests a transfer of data from program variable storage to the terminal buffer and, when the user has completed entering data, a transfer of that data back to program variable storage.

5.83.1 Syntax

```
►►── WRITE then READ TERMINAL ──┬─ WAIT ◄ ─┬──────────────────────────────►
                               ├─ NOWAIT ─┘
                               └─ WAIT ◄ ─┘

►──┬──────────────────────────┬──────────────────────────────────────────►
   │┌─ NEWPAGE ──────────────┐│   └─ FREE STORAGE ─┘
   │└─ ERASE ─────────────────┘│
   │                           │
   └─ EAU ──────────────────────┘
     └─ ERASE ALL UNPROTECTED ─┘

►── FROM output-data-location ─┬─ TO end-output-data-location ─┬──────────►
                               └─ LENGTH output-data-length ───┘

►──┬──────────────────────────────────────────────────────────────────────►
   │┌─ MODIFIED ─┬──────────────────────────────────────────┐
   │└─ BUFFER ───┘└─ FROM POSITION screen-position ─────────┘

►──┬──────────────────────────────────────────────────────────────────────►
   └─ GET STORAGE ─┘

►── INTO input-data-location ─┬─ TO end-input-data-location ─────────────►
                               └─ MAX LENGTH input-data-max-length ─┘

►──┬──────────────────────────────────────────────────────────────────────►
   └─ RETURN LENGTH into input-data-actual-length ─┘ .
```

5.83.2 Parameters

WAIT

Specifies that the I/O operation will be synchronous; the issuing task will automatically relinquish control to CA-IDMS and must wait for completion of the I/O operation before processing can continue.

This is the default.

NOWAIT

Specifies that the I/O operation will be asynchronous; the issuing task will continue executing.

Note: If NOWAIT is specified, the program must issue a CHECK TERMINAL request (described earlier in this chapter) before performing any other I/O operation.

NEWPAGE (ERASE)

Activates the page-eject (SYSINOUT devices) or erase-write (3270-type devices) mechanism to erase the contents of a screen. If NEWPAGE is not specified, the

WRITE TERMINAL request will write over rather than erase data displayed on the terminal.

NEWPAGE and ERASE are synonyms and can be used interchangeably.

EAU (ERASE ALL UNPROTECTED)

3270-type devices only. Activates the erase-all-unprotected mechanism. Following a WRITE TERMINAL EAU function, only protected fields remain on the terminal. If EAU is specified, the FROM clause (described below) need not be specified.

EAU and ERASE ALL UNPROTECTED are synonyms and can be used interchangeably.

FREE STORAGE

Releases the output buffer associated with the data being written to the terminal. The storage area being freed must have been acquired by a GET STORAGE statement (described earlier in this chapter) or the GET STORAGE option of a previously issued READ TERMINAL or WRITE THEN READ TERMINAL request. If FREE STORAGE is not specified, CA-IDMS does not free the storage associated with the output buffer until the issuing task terminates.

FROM

Specifies the 01-level WORKING-STORAGE SECTION or LINKAGE SECTION entry of the output data stream.

output-data-location

The symbolic name of a user-defined field. If FREE STORAGE has been specified, *output-data-location* must be an 01-level LINKAGE SECTION entry.

TO

Indicates the end of the output data stream.

end-output-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the output data stream.

LENGTH

Defines the length in bytes of the output data stream.

output-data-length

Either the symbolic name of a user-defined field that contains the length of the data stream or the length itself expressed as a numeric constant.

MODIFIED

3270-type devices only. Reads all modified fields in the terminal buffer into program variable storage.

BUFFER

3270-type devices only. Executes a READ BUFFER command that reads the entire contents of the terminal buffer into the program variable storage.

FROM POSITION

Defines the buffer address (screen position) at which the read will start.

screen-position

Either the symbolic name of a user-defined PIC S9(8) COMP SYNC (fullword) field or the address itself enclosed in quotation marks.

GET STORAGE

Synchronous requests only. Acquires an input buffer for the data being read into the program; CA-IDMS allocates the required storage when the read operation is complete.

INTO

Specifies the 01-level WORKING-STORAGE SECTION or LINKAGE SECTION entry of the data area reserved for the input data stream.

input-data-location

The symbolic name of a user-defined field.

If GET STORAGE is specified, the data area reserved for the input data stream must be an unallocated 01-level LINKAGE SECTION entry. If GET STORAGE is not specified, the data area must be a WORKING-STORAGE SECTION or previously allocated LINKAGE SECTION entry.

TO

Indicates the end of the data area reserved for the input data stream.

end-input-data-location

The symbolic name of either a user-defined dummy byte field or a field that contains a data item not associated with the data area reserved for the input data stream.

MAX LENGTH

Defines the length, in bytes, of the data area reserved for the input data stream.

input-data-max-length

Either the symbolic name of a user-defined field that contains the length of the data stream or the length itself expressed as a numeric constant.

If the input data stream is larger than the data area reserved in the WORKING-STORAGE SECTION or LINKAGE SECTION, CA-IDMS truncates the data stream to fit the available space.

RETURN LENGTH into

Indicates the location to which CA-IDMS will return the actual length of the input data stream.

input-data-actual-length

The symbolic name of a user-defined field. If the data stream has been truncated, *input-data-actual-length* contains the original length before truncation.

5.83.3 Example

The following statement illustrates a basic mode request to write data from the program (OUTPUT-LINE) to the terminal, read the data from the terminal to the specified location (INPUT-LINE) in the program, and return the actual length of the input data stream (LINE-LENGTH) to variable storage:

```
WRITE THEN READ TERMINAL
   WAIT
   FROM OUTPUT-LINE TO END-INPUT-LINE
   INTO INPUT-LINE MAX LENGTH 80
   RETURN LENGTH INTO LINE-LENGTH.
```

5.83.4 Status codes

After completion of the WRITE THEN READ TERMINAL function, the ERROR-STATUS field in the IDMS-DC communications block indicates the outcome of the operation:

Status code	Meaning
0000	The request has been serviced successfully
4519	The input area specified for the return of data is too small; the returned data has been truncated to fit the available space
4525	The output operation has been interrupted; the terminal operator has pressed ATTENTION or BREAK
4526	A logical error (for example, an invalid control character) has been encountered in the output data stream
4527	A permanent I/O error has occurred
4528	The dial-up line for the terminal has been disconnected
4531	The terminal request block (TRB) contains an invalid field, indicating a possible error in the program's parameters .
4532	The derived length of the specified I/O data area is zero or negative.
4535	Storage for the input buffer cannot be acquired because the specified 01-level LINKAGE SECTION entry has been allocated
4537	A storage area cannot be acquired or freed because the specified data area is defined in the WORKING-STORAGE SECTION rather than in the LINKAGE SECTION
4538	The specified 01-level LINKAGE SECTION entry has not been allocated and the GET STORAGE option has not been specified
4539	The terminal device associated with the issuing task is out of service

5.84 Logical-record clauses

Logical-record clauses are used with any of the four DML statements that access logical records (that is, OBTAIN, MODIFY, STORE, or ERASE). The logical-record clauses are as follows:

- **WHERE** specifies criteria used to select and/or criteria used to limit the selection of logical-record occurrences.
- **ON** tests for a specific path status returned to indicate the result of a logical-record DML statement.

5.84.1 WHERE

The WHERE clause has two major functions:

- **To direct the program to a path**, predefined in the subschema by the DBA and transparent to the application program. This allows you to access the database without issuing specific instructions for navigating the database.

  You need not be concerned about path selection; LRF automatically picks the most appropriate path to efficiently service the request.

- **To specify selection criteria to be applied to a logical record**. This allows the program to specify attributes of the desired logical record, thereby reducing the need for the program to inspect multiple logical records to isolate the logical record of interest.

The WHERE clause is issued in the form of a boolean expression that consists of comparisons and keywords connected by boolean operators (AND, OR, and NOT). The format of the WHERE clause follows COBOL syntax rules (that is, operands or operators are separated by a blank).

5.84.2 Syntax

```
►►── WHERE ┬──────┬─┬─ dba-designated-keyword ─┬──────────────────────►
         └─ NOT ─┘ └─ comparison ────────────┘

►──┬──────────────────────────────────────────────────────────────►◄
   │ ┌──────────────────────────────────────────────────────────┐
   └─↓─┬─ AND ─┬──┬──────┬─┬─ dba-designated-keyword ─┬──┘
       └─ OR ──┘  └─ NOT ─┘ └─ comparison ────────────┘
```

Expansion of comparison

```
►─┬─ 'logical-record-field-name' ──────────────────────────────────►
  │
  ├─ idd-defined-variable-field-name ─┬──────────────────────────┬─┤
  │                                   │   ┌──────────────────┐   │
  │                                   └─┬─┴─ OF group-id ────┴───┘
  │
  ├─ logical-record-field-name ─┬──────────────────────────┬─┬─────┬─┤
  │                             │   ┌──────────────────┐   │ │ OF LR │
  │                             └─┬─┴─ OF group-id ────┴───┘ └───────┘
  │
  └─ arithmetic-expression ──────────────────────────────────────┘

►─┬─ CONTAINS ─┬──────────────────────────────────────────────────►
  ├─ MATCHES ──┤
  ├─ EQ ───────┤
  │  └─ = ─────┘
  ├─ NE ───────┤
  ├─ GT ───────┤
  │  └─ > ─────┘
  ├─ LT ───────┤
  │  └─ < ─────┘
  ├─ GE ───────┤
  └─ LE ───────┘
```

5.84.3 Parameters

dba-designated-keyword

Specifies a DBA-designated keyword to be applied to the logical record that is the object of the command. *Dba-designated-keyword* is a keyword specified by the DBA that is applicable to the logical record named in the command; it can be no longer than 32 characters. The keyword represents an operation to be performed at the path level and serves only to route the logical-record request to the appropriate, predetermined path.

A path must exist to service a request that issues *dba-designated-keyword.* If no such path exists, the precompiler flags this condition by issuing an error message.

comparison

Specifies a comparison operation to be performed, using the indicated operands and operators. It also serves to direct the logical-record request to a path.

Individual comparisons and keywords are connected by the boolean operators AND, OR, and NOT. Parentheses can be used to clarify a multiple-comparison boolean expression or to override the precedence of operators.

5.84.4 Parameters

logical-record-field-name

Specifies a data field that participates in the named logical record.

CONTAINS/MATCHES/EQ/NE/GT/LT/GE/LE

Specifies the comparison operator:

- **CONTAINS** — Is true if the value of the right operand occurs in the value of the left operand. Both operands included with the CONTAINS operator must be alphanumeric values.
- **MATCHES** — Is true if each character in the left operand matches a corresponding character in the right operand (the mask). When MATCHES is specified, LRF compares the left operand with the mask, one character at a

time, moving from left to right. The result of the match is either true or false: the result is true if the end of the mask is reached before encountering a character in the left operand that does not match a corresponding character in the mask; the result is false if LRF encounters a character in the left operand that does not match a mask character.

Three special characters can be used in the mask to perform pattern matching: **@**, which matches any alphabetic character; **#**, which matches any numeric character; and *****, which matches any alphabetic or numeric character. Both the left operand and the mask must be alphanumeric values and elementary elements.

- **EQ** — Is true if the value of the left operand is equal to the value of the right operand.
- **NE** — Is true if the value of the left operand is not equal to the value of the right operand.
- **GT** — Is true if the value of the left operand is greater than the value of the right operand.
- **LT** — Is true if the value of the left operand is less than the value of the right operand.
- **GE** — Is true if the value of the left operand is greater than or equal to the value of the right operand.
- **LE** — Is true if the value of the left operand is less than or equal to the value of the right operand.

'literal'

Any alphanumeric or numeric literal. Alphanumeric literals must be enclosed in quotation marks.

idd-defined-variable-field-name

The name of a program variable storage field predefined in the data dictionary.

OF

Uniquely identifies the named variable field.

This qualifier is required if *idd-defined-variable-field-name* is not unique within program variable storage.

A maximum of 15 different OF *group-id* qualifiers can be specified to identify as many as 15 levels of group elements.

group-id

The name of the group element that contains the field.

logical-record-field-name

Specifies a data field that participates in the named logical record.

OF

Uniquely identifies the named logical-record field.

This qualifier is required if *logical-record-field-name* is not unique within all

subschema records, including those not part of the logical record, and including all non-CA-IDMS database records copied into the program.

A maximum of 15 different OF *group-id* qualifiers can be specified to identify as many as 15 levels of group elements.

group-id
The name of the group element or database record that contains the field.

OF LR
Specifies that the value of the named field at the time the request is issued will be used throughout processing of the request.

If the value of the field changes during request processing, LRF will continue to use the original value. If you do not specify OF LR, and the value of the field changes during request processing, the new field value in variable storage will be used if the field is required for further processing.

arithmetic-expression
Specifies an arithmetic expression designated as a unary minus (-), unary plus (+), simple arithmetic operation, or compound arithmetic operation. Arithmetic operators permitted in an arithmetic expression are add (+), subtract (-), multiply (*), and divide (/). Operands can be literals, variable-storage fields, and logical-record fields as described above.

If the WHERE clause compares a CALC-key field to a *literal*, the literal's format must correspond exactly to the CALC-key definition. Enclose the literal in quotation marks if the CALC key has a usage of DISPLAY and use leading zeros if the literal consists of fewer characters than the field's picture. For example, if the *calc-key-field* CALC key is defined as PIC 9(3) USAGE DISPLAY, code the WHERE clause as follows:

```
WHERE calc-key-field EQ '054'
```

The WHERE clause can contain as many comparisons and keywords as required to specify the criteria to be applied to the logical record. If necessary, the value of the SIZE parameter on the COPY IDMS SUBSCHEMA-LR-CTRL statement can be increased to accommodate very large and complex WHERE clause specifications. Processing efficiency is not affected by the composition of the WHERE clause (other than the logical order of the operators, as noted below), since LRF automatically uses the most efficient path to process the logical-record request.

Operators in a WHERE clause are evaluated in the following order:

1. Comparisons enclosed in parentheses
2. Arithmetic, comparison, and boolean operators by order of precedence, from highest to lowest:
   a. Unary plus or minus in an arithmetic expression
   b. Multiplication or division in an arithmetic expression
   c. Addition or subtraction in an arithmetic expression
   d. MATCHES or CONTAINS comparison operators

e. EQ, NE, GT, LT, GE, LE comparison operators

f. NOT boolean operator

g. AND boolean operator

h. OR boolean operator

3. From left to right within operators of equal precedence

5.84.5 Examples

The following examples illustrate the use of the WHERE clause.

Example 1: The following logical-record request uses a DBA-designated keyword (PROGRAMMER-ANALYSTS) to direct LRF to a DBA-defined access path:

```
OBTAIN NEXT EMP-JOB-LR
    WHERE PROGRAMMER-ANALYSTS.
```

Example 2: The following logical-record request uses boolean selection criteria to specify the desired occurrence of EMP-JOB-LR:

```
OBTAIN EMP-JOB-LR
    WHERE OFFICE-CODE-0450 EQ '001'.
```

5.84.6 ON clause

The ON clause tests for a specific path status returned to indicate the result of the statement. If LRF returns the specified path status, the imperative statement included in the ON clause is executed; if the specified path status is not returned, the imperative statement included in the ON clause is ignored and IDMS-STATUS is performed.

If the DML statement with the ON clause is the object of a PERFORM, then the user should avoid scope problems by using the THROUGH option of the PERFORM statement.

A logical-record DML statement can include an ON clause only if the AUTOSTATUS protocol is in effect for the program. AUTOSTATUS automatically invokes an error-checking routine after every DML statement except IF. For more details, see 3.4, “Error detection” on page 3-29.

The ON clause tests for a standard or DBA-defined path status, which is in the form of a 1- through 16-character unquoted string. Path statuses are issued during execution of the path selected to service the request. The standard path statuses are:

- **LR-FOUND** is returned when the logical-record request has been successfully executed. This status can be returned as the result of any of the four LRF DML statements. When LR-FOUND is returned, the ERROR-STATUS field of the IDMS communications block contains 0000.
- **LR-NOT-FOUND** is returned when the logical record specified cannot be found, either because no such record exists or because all such occurrences have already been retrieved. This status can be returned as the result of any of the four LRF

DML statements, provided that the path to which LRF is directed includes retrieval logic. When LR-NOT-FOUND is returned, the ERROR-STATUS field of the IDMS communications block contains 0000.

Note: A successful STORE can return LR-NOT-FOUND if its WHERE clause references a logical-record field and the STORE path performs no OBTAINs.

- **LR-ERROR** is returned when a logical-record request is issued incorrectly or when an error occurs in the processing of the path selected to service the request. When LR-ERROR is returned, the type of status code returned to the program in the ERROR-STATUS field of the IDMS communications block differs according to the type of error:
  - When the error occurs in the **logical-record request**, the ERROR-STATUS field contains a status code issued by LRF (major code of 20).
  - When an error occurs in the **logical-record path processing**, the ERROR-STATUS field contains a status code issued by the DBMS (major code from 00 to 19). For more information on status codes, refer to Chapter 3, “Communications Blocks and Error Detection” on page 3-1.

When accessing ASF-defined data tables, you should always check for all of the following path statuses:

- **INVALID-DATA** is returned when the data violates the definition-time selection criteria (for example, WHERE STATE EQ 'MA' and the program tries to replace the state with 'NY'). When INVALID-DATA is returned, the ERROR-STATUS field in the IDMS communications block is set to 0000.
- **DEFN-MISSING** is returned when the record definition cannot be found. When DEFN-MISSING is returned, the ERROR-STATUS field in the IDMS communications block is set to 0000.
- **OOAK-MISSING** is returned when a one-of-a-kind record cannot be found. When OOAK-MISSING is returned, the ERROR-STATUS field in the IDMS communications block is set to 0000.
- **SYNC-ERROR** is returned when the time stamp in the catalog and the table definition do not match. When SYNC-ERROR is returned, the ERROR-STATUS field in the IDMS communications block is set to 0000. This applies to ASF tables only.

The return of any of these statuses indicates a fatal error; for more information, consult your DBA.

5.84.7 Syntax

```
►── ON path-status imperative-statement ──────────────────────────►
```

5.84.8 Parameters

ON parameter

Tests for a path status returned as the result of the logical-record request issued by the program.

path-status

A 1 through 16 character alphanumeric value.

imperative-statement

Specifies the program action to be taken if the indicated path status results from the logical-record request.

5.84.9 Examples

The following statements use the path status LR-NOT-FOUND in two different ways. If LR-NOT-FOUND occurs following the initial statement, a LR-MISSING message is output; if LR-NOT-FOUND occurs in subsequent statements, an END-OF-LR message is output.

```
OBTAIN-FIRST-LR.
    OBTAIN FIRST EMP-JOB-LR
        WHERE OFFICE-CODE-450 EQ OFFICE-CODE-IN
        ON LR-NOT-FOUND
           GO TO LR-MISSING.
      .
      .
      .
OBTAIN-REST-LR.
    OBTAIN NEXT EMP-JOB-LR
        WHERE OFFICE-CODE-0450 EQ OFFICE-CODE-IN
        ON LR-NOT-FOUND
           GO TO END-OF-LR.
      .
      .
      .
      .
      .
    GO TO OBTAIN-REST-LR.
```

5.84.10 Status codes

The following codes are returned to the ERROR-STATUS field in the IDMS or IDMS-DC communications block when an LR-ERROR path status is returned to the LR-STATUS field in the LRC block:

Status code	Meaning
2001	The requested logical record was not found in the subschema. (The path DML statement, EVALUATE, returns 0000 if true, and 2001 if false.)
2008	The named record is not in the subschema, or the specified request is not permitted for the named record.

Status code	Meaning
2010	The subschema prohibits access to logical records.
2018	A path command has attempted to access a database record that has not been bound.
2040	The WHERE clause in an OBTAIN NEXT command directed LRF to a different processing path than did the WHERE clause in the preceding OBTAIN command for the same logical record.
2041	The request's WHERE clause cannot be matched to a path in the subschema.
2042	The logical-record path for the request specifies return of the LR-ERROR status.
2043	Bad or inconsistent data was encountered in the logical-record buffer during evaluation of the request's WHERE clause.
2044	The request's WHERE clause does not include data required by the logical-record path.
2045	A subscript value in a WHERE clause is either less than zero or greater than its maximum allowed value.
2046	A program check has revealed an arithmetic exception (for example, overflow, underflow, significance, divide) during evaluation of a WHERE clause.
2063	The request's WHERE clause contains a keyword that exceeds the 16-character maximum.
2064	The path command has attempted to access a CALC data item that has not been defined properly in the subschema.
2072	The request's WHERE clause is too long to be evaluated in the available work area.

Appendix A. DML Precompile, COBOL Compile, and Link-Edit JCL

A.1 About this appendix A-3
A.2 Compiling a COBOL program A-4
A.3 MVS JCL A-6
A.4 VSE JCL A-9
A.4.1 Local mode A-10
A.4.2 IDMSLBLS Procedure A-11
A.5 BS2000 JCL A-17
A.6 CMS commands A-19
A.7 Link-edit considerations A-22
A.8 Passing Parameters to the Precompiler A-23

A.1 About this appendix

This appendix contains the JCL used to prepare COBOL source code that contains DML statements for execution. Link-edit considerations are also discussed. Samples of MVS, VSE, BS2000, and VM/CMS JCL are included.

A.2 Compiling a COBOL program

To compile a COBOL program under the DML precompiler:

1. Execute the program IDMSDMLC
2. Execute the COBOL compiler
3. Link edit

Input to IDMSDMLC consists of source code written in COBOL/DML, protocol/control information, and data dictionary record descriptions. Output from IDMSDMLC is as follows:

- A source COBOL program
- A DML source listing and diagnostics

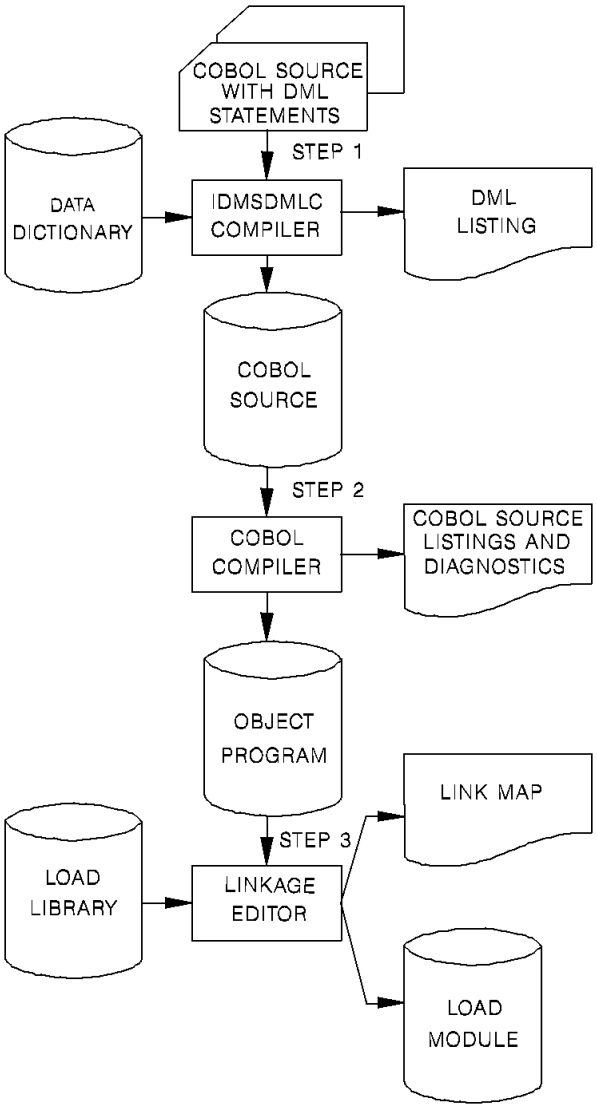
Input to the COBOL compiler consists of the source program produced by IDMSDMLC. Output is as follows:

- An object program
- COBOL listings

Input to the linkage editor consists of the object program produced by the COBOL compiler. Output is as follows:

- A load module (phase)
- A link-edit map

The following figure illustrates the steps involved in compiling a COBOL program.


COBOL SOURCE WITH DML STATEMENTS
STEP 1
DATA DICTIONARY
IDMSDMLC COMPILER
DML LISTING
COBOL SOURCE
STEP 2
COBOL COMPILER
COBOL SOURCE LISTINGS AND DIAGNOSTICS
OBJECT PROGRAM
LINK MAP
STEP 3
LOAD LIBRARY
LINKAGE EDITOR
LOAD MODULE

A.3 MVS JCL

Sample JCL for MVS and OS/390 operating systems is shown below, followed by a description of statements that need tailoring for site-specific conditions.

```
//*****************************************************************
//**                  PRECOMPILE COBOL PROGRAM                   **
//*****************************************************************
//precomp  EXEC PGM=IDMSDMLC,REGION=4096K,
//             PARM='precompiler-options'
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYS001   DD   UNIT=sysda,SPACE=(TRK,(10,10)),
//              DCB=(RECFM=VB,LRECL=133,BLKSIZE=1334,DSORG=PS)
//SYS002   DD   UNIT=sysda,SPACE=(TRK,(10,10)),
//              DCB=(RECFM=VB,LRECL=133,BLKSIZE=1334,DSORG=PS)
//SYS003   DD   UNIT=sysda,SPACE=(TRK,(10,10)),
//              DCB=(RECFM=VB,LRECL=133,BLKSIZE=1334,DSORG=PS)
//SYSPCH   DD   DSN=&&SOURCE,DISP=(NEW,PASS),
//             UNIT=sysda,SPACE=(TRK,(10,5),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSLST   DD   SYSOUT=A
//SYSIDMS  DD   *
DMCL=dmcl-name
sysidms-input-parms
/*
//SYSIPT   DD   *
COBOL DML source statements
/*
//*****************************************************************
//**                  COMPILE COBOL PROGRAM                      **
//*****************************************************************
//cblcmp   EXEC PGM=igycrctl,REGION=4096K,
//             PARM='compiler-options'
//STEPLIB  DD   DSN=cobol.loadlib,DISP=SHR
//SYSUT1   DD   UNIT=sysda,SPACE=(TRK,(10,5))
//SYSUT2   DD   UNIT=sysda,SPACE=(TRK,(10,5))
//SYSUT3   DD   UNIT=sysda,SPACE=(TRK,(10,5))
//SYSUT4   DD   UNIT=sysda,SPACE=(TRK,(10,5))
//SYSUT5   DD   UNIT=sysda,SPACE=(TRK,(10,5))
//SYSUT6   DD   UNIT=sysda,SPACE=(TRK,(10,5))
//SYSUT7   DD   UNIT=sysda,SPACE=(TRK,(10,5))
//syslin   DD   DSN=&&OBJECT,DISP=(NEW,PASS),
//             UNIT=sysda,SPACE=(TRK,(10,5),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSN=&&SOURCE,DISP=(OLD,DELETE)
```

```
//*****************************************************************
//**                  LINK PROGRAM MODULE                        **
//*****************************************************************
//link     EXEC PGM=IEWL,REGION=1024K,PARM='LET,LIST,MAP,XREF'
//SYSUT1   DD   UNIT=sysda,SPACE=(TRK,(20,5))
//SYSLIB   DD   DSN=cobol.linklib,DISP=SHR
//loadlib  DD   DSN=idms.loadlib,DISP=SHR
//SYSLMOD  DD   DSN=user.loadlib,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSLIN   DD   DSN=&&OBJECT,DISP=(OLD,DELETE)
//         DD   *
 INCLUDE loadlib(IDMS)      required, except omit for CICS
 INCLUDE loadlib(IDMSCANC)  required for BATCH and DC-BATCH if using IDMS-STATUS module
 INCLUDE loadlib(IDMSOPTI)  optional; BATCH and DC-BATCH only
 INCLUDE loadlib(idmscint)  required for CICS, otherwise omit
 ENTRY   userentry
 NAME    userprog(R)
/*
//*
```

Note: If using the IDMSOPTI module, you must assemble and link edit it before using the JCL above.

precompiler-options	options that control various aspects of the precompile process. See A.8, “Passing Parameters to the Precompiler” on page A-23 for a complete description of the options.
idms.dba.loadlib	data set name of the load library containing the DMCL and database name table
idms.loadlib	data set name of the load library containing the CA-IDMS executable modules
sysctl	ddname of SYSCTL file
idms.sysctl	data set name of SYSCTL file
dcmsg	ddname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	data set name of the system message (DDLDCMSG) area
sysda	symbolic device name for work files
sysidms-input-parms	parameters that specify physical requirements of the environment, runtime directives, or operating system-dependent file information. For a complete description of all SYSIDMS parameters and syntax, see *CA-IDMS Database Administration*. Also see A.8, “Passing Parameters to the Precompiler” on page A-23 for a discussion of parameters that can be passed via the PARM=SYSIDMS input statement.
dmcl-name	specifies the name of the DMCL that the precompiler should use to access the message dictionary

igycrctl	program name of the COBOL compiler
compiler-options	parameters that specify options that are appropriate to your version of the COBOL compiler. See Chapter 1, "Introduction to CA-IDMS Data Manipulation Language" on page 1-1, Appendix H, "VS COBOL II Support" on page H-1, and Appendix I, "Considerations for IBM Language Environment" on page I-1 for restrictions and recommendations specific to CA-IDMS access. Also see the IBM documentation for your compiler.
cobol.loadlib	load library that contains COBOL compiler
syslin	ddname of the object data set output by the COBOL compiler
cobol.linklib	load library that contains COBOL support modules
user.loadlib	user application load library
idmscint	load module created by compiling IDMSCINT or IDMSCINL. See the *CA-IDMS System Operations* guide for more information.
userentry	name of program entry point
userprog	name of program in load library

Note: Depending on the central version operating environment, an IDMSOPTI module link edited with IDMSDMLC can be used in place of or in addition to the SYSCTL file.

Local mode JCL: To execute the compiler in local mode, remove the SYSCTL statement from the precompile step and replace it with the following:

```
//dictdb   DD  DSN=idms.appldict.ddldml,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,DISP=(NEW,CATLG),UNIT=tape
```

dictdb	ddname of the application dictionary DDLDML area
idms.appldict.ddldml	data set name of application dictionary
sysjrnl	ddname of the tape journal file
idms.tapejrnl	data set name of the tape journal file
tape	symbolic device name of the tape journal file

A.4 VSE JCL

IDMSDMLC ('VSE')

```
/*****************************************************************
/**                    PRECOMPILE PROGRAM                       **
/*****************************************************************
* step1
// EXEC PROC=IDMSLBLS
// UPSI b                 if specified in IDMSOPTI module
// DLBL      idmspch,'temp.dmlc',0
// EXTENT    SYS020,nnnnnn,,,ssss,llll
// ASSGN     SYS020,DISK,VOL=nnnnnn,SHR
// DLBL      SYS001,'wkfile1',0
// EXTENT    SYS001,nnnnnn,,,ssss,llll
// ASSGN     SYS001,DISK,VOL=nnnnnn,SHR
// DLBL      SYS002,'wkfile2',0
// EXTENT    SYS002,nnnnnn,,,ssss,llll
// ASSGN     SYS002,DISK,VOL=nnnnnn,SHR
// DLBL      SYS003,'wkfile3',0
// EXTENT    SYS003,nnnnnn,,,ssss,llll
// ASSGN     SYS003,DISK,VOL=nnnnnn,SHR
// EXEC      IDMSDMLC,PARM='precompiler-options'

Input SYSIDMS parameters here, as required

/*

COBOL/DML source statements

/*****************************************************************
/**                    COMPILE PROGRAM                          **
/*****************************************************************
/*
* step2
// DLBL      IJSYSIN,'temp.dmlc',0
// EXTENT    SYSIPT,nnnnnn
   ASSGN     SYSIPT,DISK,VOL=nnnnnn,SHR
// OPTION    CATAL,NODECK,NOSYM
  PHASE userprog,*
// EXEC      FCOBOL
/*****************************************************************
/**                    LINK PROGRAM MODULE                      **
/*****************************************************************
* step3
   CLOSE     SYSIPT,SYSRDR
ENTRY (dmlc)
// EXEC      LNKEDT
/*
```

IDMSLBLS	Name of the procedure provided at installation that contains the file definitions for CA-IDMS dictionaries and databases. ▸▸ For a complete listing of IDMSLBLS, see A.4.2, “IDMSLBLS Procedure” on page A-11 later in this chapter.

b	appropriate UPSI switch, 1 through 8 characters, if specified in the IDMSOPTI module
idmspch	filename of data set output from the IDMSDMLC precompiler
temp.dmlc	file ID of data set output from the IDMSDMLC precompiler
SYS020	logical unit assignment of the DMLC output
nnnnnn	volume serial identifier of appropriate disk volume
ssss	starting track (CKD) or block (FBA) of disk extent
llll	number of tracks (CKD) or blocks (FBA) of disk extent
userprog	name of program in the library
precompiler-options	options that control various aspects of the precompile process. See A.8, “Passing Parameters to the Precompiler” on page A-23 for a complete description of the options.
dmlc	name of COBOL/DML module

You can use SYSIDMS parameters to specify information about your runtime environment.

►► See *CA-IDMS Database Administration* for more information on SYSIDMS parameters.

A.4.1 Local mode

To execute the IDMSDMLC precompiler in local mode:

- Remove the UPSI specification
- Add the following statements in step 1 (the IDMSDMLC step):

```
// TLBL     sysjrnl,'idms.tapejrnl',,nnnnnn,,f
// ASSGN    SYS009,TAPE,VOL=nnnnnn
```

idms.tapejrnl	file ID of tape journal file
f	file number of tape journal file
sys009	logical unit assignment for journal file

INCLUDE statements should be provided in local mode or central version JCL as follows:

- For programs that include a COBOL internal sort, place the following statements in the second step, before EXEC FCOBOL:

```
ACTION NOAUTO            Prevents multiple inclusions of IDMS

INCLUDE IDMS             IDMS interface for use with COMRG

INCLUDE IDMSOPTI         IDMSOPTI module

INCLUDE IDMSCANC         Local mode abort entry point
                         (omit IDMSCANC if TP application)

INCLUDE IDMSCINT         For CICS only, replaces INCLUDE IDMS
```

INCLUDE *IDMSOPTI* can be omitted for programs executed in local mode.

Note: COBOL overlay programs must resolve references to IDMS within their root segment; care must be taken to prevent the overlaying of the IDMS interface. Use of IDMS and IDMSLDPT is recommended for these programs.

A.4.2 IDMSLBLS Procedure

IDMSLBLS is a procedure provided during a CA-IDMS VSE installation. It contains file definitions for the CA-IDMS components listed below. These components are provided during installation:

- Dictionaries
- Sample databases
- Disk journal files
- SYSIDMS file

Tailor the IDMSLBLS procedure to reflect the filenames and definitions in use at your site and include this procedure in VSE JCL job streams.

The sample VSE JCL provided in this document includes the IDMSLBLS procedure. Therefore, individual file definitions for CA-IDMS dictionaries, sample databases, disk journal files, and SYSIDMS files are not included in the sample JCL.

IDMSLBLS procedure (VSE)

```
* -------- LIBDEFS --------
// LIBDEF  *,SEARCH=idmslib.sublib
// LIBDEF  *,CATALOG=user.sublib
/*  ------------------------- LABELS -------------------------
// DLBL    idmslib,'idms.library',1999/365
// EXTENT  ,nnnnnn,,,ssss,1500
// DLBL    dccat,'idms.system.dccat',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,31
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dccatl,'idms.system.dccatlod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dccatx,'idms.system.dccatx',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcdml,'idms.system.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclod,'idms.system.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,21
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclog,'idms.system.ddldclog',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,401
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcrun,'idms.system.ddldcrun',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,68
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcscr,'idms.system.ddldcscr',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,135
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcmsg,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclscr,'idms.sysloc.ddlocscr',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirldb,'idms.sysdirl.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirllod,'idms.sysdirl.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empdemo,'idms.empdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    insdemo,'idms.insdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    orgdemo,'idms.orgdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empldem,'idms.sqldemo.empldemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    infodem,'idms.sqldemo.infodemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    projdem,'idms.projseg.projdemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
```

```
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    indxdem,'idms.sqldemo.indxdemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sysctl,'idms.sysctl',1999/365,SD
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    secdd,'idms.sysuser.ddlsec',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dictdb,'idms.appldict.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dloddb,'idms.appldict.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqldd,'idms.syssql.ddlcat',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqllod,'idms.syssql.ddlcatl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqlxdd,'idms.syssql.ddlcatx',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asfdml,'idms.asfdict.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asflod,'idms.asfdict.asflod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,401
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asfdata,'idms.asfdict.asfdata',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    ASFDEFN,'idms.asfdict.asfdefn',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j1jrnl,'idms.j1jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j2jrnl,'idms.j2jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j3jrnl,'idms.j3jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    SYSIDMS,'#SYSIPT',0,SD
/+
/*
```

idmslib.sublib	name of the sublibrary within the library containing CA-IDMS modules
user.sublib	name of the sublibrary within the library containing user modules
idmslib	filename of the file containing CA-IDMS modules
idms.library	file-ID associated with the file containing CA-IDMS modules

SYSnnn	logical unit of the volume for which the extent is effective
nnnnnn	volume serial identifier of appropriate disk volume
ssss	starting track (CKD) or block (FBA) of disk extent
dccat	filename of the system dictionary catalog (DDLCAT) area
idms.system.dccat	file-ID of the system dictionary catalog (DDLCAT) area
dccatl	filename of the system dictionary catalog load (DDLCATLOD) area
idms.system.dccatlod	file-ID of the system dictionary catalog load (DDLCATLOD) area
dccatx	filename of the system dictionary catalog index (DDLCATX) area
idms.system.dccatx	file-ID of the system dictionary catalog index (DDLCATX) area
dcdml	filename of the system dictionary definition (DDLDML) area
idms.system.ddldml	file-ID of the system dictionary definition (DDLDML) area
dclod	filename of the system dictionary definition load (DDLDCLOD) area
idms.system.ddldclod	file-ID of the system dictionary definition load (DDLDCLOD) area
dclog	filename of the system log area (DDLDCLOG) area
idms.system.ddldclog	file-ID of the system log (DDLDCLOG) area
dcrun	filename of the system queue (DDLDCRUN) area
idms.system.ddldcrun	file-ID of the system queue (DDLDCRUN) area
dcscr	filename of the system scratch (DDLDCSCR) area
idms.system.ddldcscr	file-ID of the system scratch (DDLDCSCR) area
dcmsg	filename of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	file-ID of the system message (DDLDCMSG) area
dclscr	filename of the local mode system scratch (DDLOCSCR) area
idms.sysloc.ddlocscr	file-ID of the local mode system scratch (DDLOCSCR) area

dirldb	filename of the IDMSDIRL definition (DDLDML) area
idms.sysdirl.ddldml	file-ID of the IDMSDIRL definition (DDLDML) area
dirllod	filename of the IDMSDIRL definition load (DDLDCLOD) area
idms.sysdirl.dirllod	file-ID of the IDMSDIRL definition load (DDLDCLOD) area
empdemo	filename of the EMPDEMO area
idms.empdemo1	file-ID of the EMPDEMO area
insdemo	filename of the INSDEMO area
idms.insdemo1	file-ID of the INSDEMO area
orgdemo	filename of the ORGDEMO area
idms.orgdemo1	file-ID of the ORDDEMO area
empldem	filename of the EMPLDEMO area
idms.sqldemo.empldemo	file-ID of the EMPLDEMO area
infodem	filename of the INFODEMO area
idms.sqldemo.infodemo	file-ID of the INFODEMO area
projdem	filename of the PROJDEMO area
idms.projseg.projdemo	file-ID of the PROJDEMO area
indxdem	filename of the INDXDEMO area
idms.sqldemo.indxdemo	file-ID of the INDXDEMO area
sysctl	filename of the SYSCTL file
idms.sysctl	file-ID of the SYSCTL file
secdd	filename of the system user catalog (DDLSEC) area
idms.sysuser.ddlsec	file-ID of the system user catalog (DDLSEC) area
dictdb	filename of the application dictionary definition area
idms.appldict.ddldml	file-ID of the application dictionary definition (DDLDML) area
dloddb	filename of the application dictionary definition load area
idms.appldict.ddldclod	file-ID of the application dictionary definition load (DDLDCLOD) area
sqldd	filename of the SQL catalog (DDLCAT) area
idms.syssql.ddlcat	file-ID of the SQL catalog (DDLCAT) area
sqllod	filename of the SQL catalog load (DDLCATL) area

idms.syssql.ddlcatl	file-ID of SQL catalog load (DDLCATL) area
sqlxdd	filename of the SQL catalog index (DDLCATX) area
idms.syssql.ddlcatx	file-ID of the SQL catalog index (DDLCATX) area
asfdml	filename of the asf dictionary definition (DDLDML) area
idms.asfdict.ddldml	file-ID of the asf dictionary definition (DDLDML) area
asflod	filename of the asf dictionary definition load (ASFLOD) area
idms.asfdict.asflod	file-ID of the asf dictionary definition load (ASFLOD) area
asfdata	filename of the asf data (ASFDATA) area
idms.asfdict.asfdata	file-ID of the asf data area (ASFDATA) area
ASFDEFN	filename of the asf data definition (ASFDEFN) area
idms.asfdict.asfdefn	file-ID of the asf data definition area (ASFDEFN) area
j1jrnl	filename of the first disk journal file
idms.j1jrnl	file-ID of the first disk journal file
j2jrnl	filename of the second disk journal file
idms.j2jrnl	file-ID of the second disk journal file
j3jrnl	filename of the third disk journal file
idms.j3jrnl	file-ID of the third disk journal file
SYSIDMS	filename of the SYSIDMS parameter file

A.5 BS2000 JCL

IDMSDMLC ('BS2000')

```
/ADD-FILE-LINK L-NAME=CDMSLIB,F-NAME=idms.dba.loadlib
/ADD-FILE-LINK L-NAME=CDMSLIB1,F-NAME=idms.loadlib
/ADD-FILE-LINK L-NAME=CDMSLODR,F-NAME=idms.loadlib
/ADD-FILE-LINK L-NAME=sysctl,F-NAME=idms.sysctl,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=SYSIDMS,F-NAME=*DUMMY
/ASSIGN-SYSOPT TO=temp.punch
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSDMLC,LIB=idms.dba.loadlib,RUN-MODE=*ADV)
DICTNAME=dictionary-name
DMCL=dmcl-name
sysidms-input-parms
PARM='precompiler-options'
END-SYSIDMS

COBOL/DML source statements

/ASSIGN-SYSOPT TO=*PRIMARY
/ASSIGN-SYSDTA TO=temp.punch
/START-COBOL85-COMPILER                                          -
/   MODULE-OUTPUT=LIB-ELEM(LIB=idms.objlib.user,ELEM=userprog,   -
/   COMPILER-ACTION=MODULE-GENERATION(MODULE-FORMAT=OM),         -
/   LISTING=(SOURCE=YES,DIAGNOSTICS=YES,OUTPUT=SYSLIST)
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=userprog
//INC-MOD LIB=idms.objlib.user,ELEM=userprog
//INC-MOD LIB=idms.loadlib,ELEM=IDMSPBS2      For DC, BATCH and DCBATCH
//INC-MOD LIB=idms.loadlib,ELEM=IDMSTCM       UTM only
//RESOLVE-BY-AUTOLINK LIB=cobol.objlib
//SAVE-LLM LIB=idms.loadlib.user,ELEM=userprog(VER=@),OVER=YES
//END
/DELETE-FILE temp.punch
```

idms.loadlib	filename of the load library containing the CA-IDMS executable modules
idms.dba.loadlib	filename of the load library containing the DMCL and database name table load modules
sysctl	linkname of SYSCTL file
idms.sysctl	filename of SYSCTL file
temp.punch	filename of temporary file that contains DML compiler output
sysidms-input-parms	parameters that specify physical requirements of the environment, runtime directives, or operating system-dependent file information. For a complete description of all SYSIDMS parameters and syntax, see *CA-IDMS Database Administration*

precompiler-options	options that control various aspects of the precompile process. See A.8, “Passing Parameters to the Precompiler” on page A-23 for a complete description of the options.
idms.objlib.user	filename of user object library
userprog	name of user application program
cobol.objlib	filename of the COBOL run-time object library
idms.loadlib.user	filename of the user load library

Note: Depending on the CV operating environment, an IDMSOPTI module link edited with the DML compiler can be used in place of or in addition to the SYSCTL file.

Local mode: To execute the compiler in local mode:

- Remove the SYSCTL ADD-FILE-LINK command
- Add:

```
/ADD-FILE-LINK L-NAME=dictdb,F-NAME=idms.appldict.ddldml,SHARED-UPD=*YES
[/CREATE-FILE F-NAME=idms.tapejrnl,SUPPRESS-ERRORS=*FILE-EXIST,         -
/    SUP=*TAPE(VOLUME=nnnnnn,DEVICE=tape)]
/ADD-FILE-LINK L-NAME=sysjrnl,F-NAME=idms.tapejrnl [,BUF-LEN=bbbb,      -
/    SUP=*TAPE(F-SEQ=1)]
```

Statements and parameters between brackets must be specified only when using the journal file on tape.

dictdb	linkname of the data dictionary file
idms.appldict.ddldml	filename of the data dictionary file
sysjrnl	linkname of the tape journal file
idms.tapejrnl	filename of the tape journal file
bbbb	page size of the file
nnnnnn	volume serial number of the tape archive file
tape	device name for the tape journal file

A.6 CMS commands

IDMSDMLC ('CMS')

```
FILEDEF SYSIPT DISK sysipt data a (RECFM F LRECL ppp BLKSIZE nnn
FILEDEF SYSPCH DISK prgnme cobol a
FILEDEF SYSIDMS DISK sysidms parms a (RECFM F LRECL ppp. BLKSIZE nnn
EXEC IDMSFD
OSRUN IDMSDMLC PARM='CVMACH=vmid,precompiler-options'
FILEDEF TEXT DISK prgnme TEXT A
GLOBAL TXTLIB coblibvs IDMSLIB1
COBOL prgnme (OSDECK APOST LIB               COBOL compile step
TXTLIB DEL utextlib prgnme
TXTLIB ADD utextlib prgnme
FILEDEF SYSLMOD uloadlib LOADLIB a (RECFM V LRECL 1024 BLKSIZE 10 24
FILEDEF objlibl DISK IDMSLIB1 TXTLIB A
FILEDEF objlib DISK utextlib TXTLIB a
FILEDEF SYSLIB DISK coblibvs TXTLIB p
FILEDEF SYS001 DISK wfn wft wfm
LKED linkctl (LIST XREF LET MAP RENT NOTERM PRINT SIZE 512K 64K
                                             Link edit step
```

sysipt data a	Filename, type, and mode of the file containing the COBOL/DML source statements
ppp	Record length of the data file
nnn	Block size of the data file
prgnme cobol a	Filename of the COBOL program
sysidms parms a	Filename, filetype, and filemode of the file that contains SYSIDMS parameters (parameters that define your runtime environment)
vmid	ID of the virtual machine running the central version
precompiler-options	options that control various aspects of the precompile process. See A.8, “Passing Parameters to the Precompiler” on page A-23 for a complete description of the options.
coblibvs	Filename of the library that contains COBOL logic modules
utextlib	Filename of the user text library
uloadlib LOADLIB a	Filename, filetype, and filemode of the user load library
objlib1	DDname of the first CA-IDMS object library
objlib	DDname of the user object library
coblibvs TXTLIB p	Filename, filetype, and filemode of the library that contains COBOL logic modules
wfn wft wfm	Filename, type, and mode of the files to be used as intermediate work files by IDMSDMLC

linkctl	Filename of the file that contains the linkage editor control statements

How to edit the SYSIDMS file: To create the SYSIDMS file, enter these CMS commands:

```
XEDIT sysidms parms a (NOPROF
INPUT
 .
 .
 .
SYSIDMS parameters
 .
 .
 .
FILE
```

To run IDMSDMLC, you must include the NODENAME and DICTNAME SYSIDMS parameters.

▶▶ See *CA-IDMS Database Administration* for more information on SYSIDMS parameters.

How to create the SYSIPT file: To create the SYSIPT file, enter these CMS commands:

```
XEDIT sysipt data a (NOPROF
INPUT
 .
 .
 .
DML source statements
 .
 .
 .
FILE
```

How to create the LINKCTL file: To create the LINKCTL file, enter these CMS commands:

```
XEDIT linkctl data a (NOPROF
INPUT
 .
 .
 .
INCLUDE objlib(prgnme)
INCLUDE objlib1(IDMS)     IDMS is required, omit for CICS
INCLUDE objlib1(IDMSCINT) for CICS only
INCLUDE objlib1(IDMSCANC) IDMSCANC for BATCH and DC_BATCH
ENTRY prgnme
NAME prgnme(R)
 .
 .
 .
FILE
```

Executing in local mode: To execute IDMSDMLC in local mode, remove the CVMACH parameter from OSRUN, and do *one* of the following:

- Link IDMSDMLC with an IDMSOPTI program that specifies local execution mode
- Specify *LOCAL* as the first input parameter in the file specified in the FILEDEF SYSIPT statement
- Modify the OSRUN statement, as follows:

  ```
  OSRUN IDMSDMLC PARM='*LOCAL*'
  ```

  Note: This option is valid only if the OSRUN command is issued from a System Product Interpreter or from an EXEC2 file.

A.7 Link-edit considerations

The modules involved in the link edit of an application program contain three external references. Some must be resolved, others can be left unresolved depending on the mode of operation. The table below lists and explains the external references.

Reference	Referenced by	Resolved by	Comments
ABORT	Application program	IDMSCANC	Should be resolved ONLY in a batch environment; should NOT be included in a tp environment.
IDMS	Application program	IDMS	Must be resolved
IDMSOPTI*	IDMS	IDMSOPTI module	Must be resolved if using the central version without a SYSCTL file

* IDMSOPTI is a weak external reference (WXTRN).

A.8 Passing Parameters to the Precompiler

A number of parameters can be provided to control the action taken by the precompiler. The parameters can be specified in one of three ways:

- An EDBPPARM CSECT can be compiled with parameter values that are always appropriate to a particular operating system or client site. This CSECT is then linked as part of the IDMSDMLC load module.
- A PARM= clause can be coded on the EXEC statement that invokes IDMSDMLC in an MVS, OS/390, or VSE environment or on the OSRUN statement that invokes IDMSDMLC in a CMS environment. Any option that is specified on the EXEC or OSRUN statement will take precedence over the same parameter if it is coded with a different value in the EDBPPARM CSECT.
- A PARM= statement can be coded as a SYSIDMS input parameter. See *CA-IDMS Database Administration* for more information on using SYSIDMS. Any option that is specified in the PARM= statement will take precedence over the same parameter if it is coded with a different value in the EDBPPARM CSECT. Note that if PARM= is specified both as a SYSIDMS input statement and on an EXEC or OSRUN statement, the PARM= clause on the EXEC or OSRUN statement will be ignored entirely.

Precompiler options: Parameter options available to code in the EXEC statement of the precompile step are:

- RCM=*rcm-name*
  - Specifies the name of the RCM created for the program by the precompiler if the program uses SQL access.
- RCMVERSION=*rcm-version-number*
  - Specifies the version number of the RCM created for the program by the precompiler.
- AM=*access-module-name*
  - Specifies the name of the access module to be executed for the program at runtime if the program uses SQL access.
- SCHEMA=*schema-name*
  - Specifies the default schema-name qualifier for the precompiler to use when processing an INCLUDE TABLE statement that does not supply a qualifier.
- NOINSTALL
  - Specifies that the precompiler should only check syntax.
- DICTNAME=*dictionary-name*
  - Specifies the name of the dictionary the precompiler should access.
- SQL=NO/89/FIPS

- Specifies the SQL syntax standard that the precompiler should apply when checking the validity of SQL statements in the program.
  - Option NO, the default, means that compliance with a named SQL standard is not checked or enforced, and all CA-IDMS extensions are permitted.
  - Option 89 directs the precompiler to use ANSI X3.135-1989 (Rev), Database Language SQL with integrity enhancement as the standard for compliance.
  - Option FIPS directs the precompiler to use FIPS PUB 127-1, *Database Language SQL* as the standard for compliance.

- LIST/NOList
  - LIST directs the precompiler to create a listing of the program with precompiler messages. NOList directs the compiler not to create a listing of the program with precompiler messages.
- DATE=ISO/USA/EUR/JIS
  - Specifies the format of the DATE data type to be used for communication between the program and the database when the access module is executed.
- TIME=ISO/USA/EUR/JIS
  - Specifies the format of the TIME data type to be used for communication between the program and the database when the access module is executed.
- COBOL=1/2/85
  - Specifies the version of COBOL with which COBOL statements generated by the precompiler must comply.
    - Option 1 directs the precompiler to generate statements that comply with any of the following:
      - Versions of VS-COBOL that precede VS-COBOL II for MVS, OS/390, or VSE operating systems
      - all CBOL compiler versions for BS2000 operating systems
    - Option 2, the default, directs the precompiler to generate statements that comply with VS-COBOL II or LE-compliant COBOL compilers.
    - Option 85 directs the precompiler to comply with COBOL85, the version of COBOL required for the Fujitsu and Hitachi compilers.
- SR1SR7 = YES/NO
  - If YES is specified then SR1 and SR7 will be emitted in SUBSCHEMA-RECNAMES. NO is the default.

►► For more information on SQL-related parameter options, see *CA-IDMS SQL Programming*.

Site-specific parameters: The following sample EDBPPARM CSECT will change the default for the COBOL parameter from COBOL=2 to COBOL=1 and will direct the precompiler not to produce a listing of the source program.

```
EDBPPARM  CSECT
DC C'COBOL=1,NOLIST'
DC X'00'
END
```

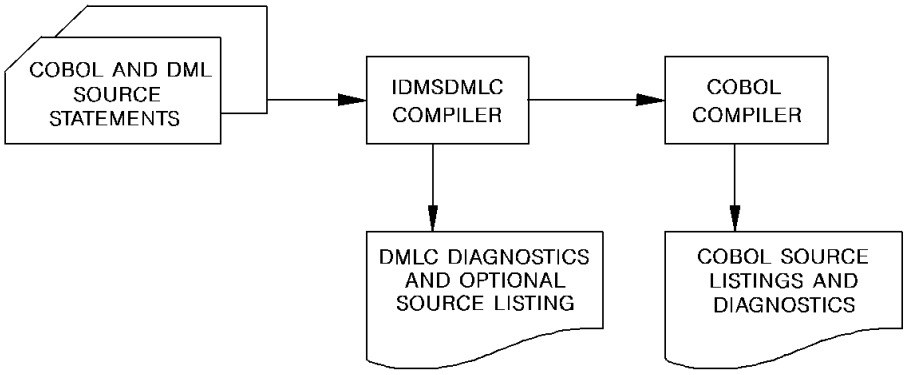
Appendix B. Sample Batch Program

B.1 About this appendix B-3
B.2 Sample batch program as input to the DML compiler B-4
B.3 Sample batch program as output from the DML compiler B-10
B.4 Sample batch program from the COBOL precompiler B-21

B.1 About this appendix

This appendix contains a sample batch COBOL program that accesses database records using navigational DML statements. The following figure shows the program as it appears in the various stages of the compilation process. You create a program using COBOL and DML statements. This program is input to the DML compiler, which produces a listing that contains diagnostics and, optionally, DML source statements. The expanded code is input to the COBOL compiler, which generates a listing of the fully expanded code and diagnostics.

Compilation process



B.2 Sample batch program as input to the DML compiler

The sample program contains COBOL and DML source statements.

```
*RETRIEVAL
*DMLIST
*NO-ACTIVITY-LOG
*SCHEMA-COMMENTS

 IDENTIFICATION DIVISION.

    PROGRAM-ID.                  DEPTRPT.

    AUTHOR.                      COMPUTER ASSOCIATES INTERNATIONAL.

    DATE-WRITTEN.                APRIL 1995.

    REMARKS.                     THIS PROGRAM DEMONSTRATES
                                 CA-IDMS DATABASE ACCESS USING
                                 COBOL DML STATEMENTS. IT READS
                                 DEPARTMENT ID NUMBERS AND RETRIEVES
                                 RELATED RECORD OCCURRENCES,
                                 PRINTING A REPORT THAT INCLUDES
                                 DEPARTMENT, EMPLOYEE, JOB, AND
                                 OFFICE INFORMATION.
*****************************************************************
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT DEPT-FILE-IN           ASSIGN TO INFILE.
     SELECT DEPT-FILE-OUT          ASSIGN TO OUTFILE.
     SELECT ERR-FILE-OUT           ASSIGN TO ERRFILE.
*****************************************************************
 IDMS-CONTROL SECTION.

 PROTOCOL.                      MODE IS BATCH DEBUG
                                IDMS-RECORDS MANUAL.
      SKIP3
*****************************************************************
 DATA DIVISION.

 SCHEMA SECTION.

 DB EMPSS01 WITHIN EMPSCHM.

*****************************************************************
 FILE SECTION.

 FD  DEPT-FILE-IN
     RECORD CONTAINS 80
     BLOCK  CONTAINS 80 CHARACTERS
     RECORDING MODE IS F
     LABEL RECORDS ARE OMITTED.

 01  DEPT-REC-IN.
     02 DEPT-ID-IN             PIC 9(4).
     02 DEPT-IN-FILLER         PIC X(76).

 FD  DEPT-FILE-OUT
     RECORD CONTAINS 133
     BLOCK  CONTAINS 133 CHARACTERS
     RECORDING MODE IS F
     LABEL RECORDS ARE OMITTED.

 01  DEPT-REC-OUT.
     02 CC                     PIC X.
     02 PRINT-LINE             PIC X(132).

 FD  ERR-FILE-OUT
     RECORD CONTAINS 133
     BLOCK  CONTAINS 133 CHARACTERS
     RECORDING MODE IS F
```

```
     LABEL RECORDS ARE OMITTED.

 01  ERR-REC-OUT.
     02 ERR-CC                 PIC X.
     02 ERR-LINE               PIC X(132).

*************************************************************
 WORKING-STORAGE SECTION.
 01  EOF-SW             PIC X      VALUE 'N'.
     88 END-OF-FILE                VALUE 'Y'.
 01  LINE-COUNT        PIC 99      VALUE 0.
 01  ERR-LINE-COUNT    PIC 99      VALUE 0.
 01  LINE-MAX          PIC 99      VALUE 50.
*************************************************************
 01  DEPT-HEADER.
     05  FILLER        PIC X(30)    VALUE SPACES.
     05  FILLER        PIC X(13)    VALUE 'DEPARTMENT ID'.
     05  FILLER        PIC X(10)    VALUE SPACES.
     05  FILLER        PIC X(9)     VALUE 'DEPT NAME'.
     05  FILLER        PIC X(70)    VALUE SPACES.
 01  DEPT-DETAIL-LINE.
     05  FILLER        PIC X(33)    VALUE SPACES.
     05  DEPT-ID-OUT   PIC X(4).
     05  FILLER        PIC X(16)    VALUE SPACES.
     05  DEPT-NAME-OUT PIC X(45).
     05  FILLER        PIC X(34)    VALUE SPACES.
 01  EMP-HEADER.
     05  FILLER        PIC X(5)     VALUE SPACES.
     05  FILLER        PIC X(6)     VALUE 'EMP ID'.
     05  FILLER        PIC X(2)     VALUE SPACES.
     05  FILLER        PIC X(9)     VALUE 'LAST NAME'.
     05  FILLER        PIC X(8)     VALUE SPACES.
     05  FILLER        PIC X(10)    VALUE 'FIRST NAME'.
     05  FILLER        PIC X(3)     VALUE SPACES.
     05  FILLER        PIC X(10)    VALUE 'START DATE'.
     05  FILLER        PIC X(2)     VALUE SPACES.
     05  FILLER        PIC X(9)     VALUE 'JOB TITLE'.
     05  FILLER        PIC X(13)    VALUE SPACES.
     05  FILLER        PIC X(14)    VALUE 'OFFICE ADDRESS'.
     05  FILLER        PIC X(42)    VALUE SPACES.
 01  EMP-DETAIL-LINE.
     05  FILLER        PIC X(5)     VALUE SPACES.
     05  ID-OUT        PIC X(4).
     05  FILLER        PIC X(4)     VALUE SPACES.
     05  LAST-OUT      PIC X(15).
     05  FILLER        PIC X(2)     VALUE SPACES.
     05  FIRST-OUT     PIC X(10).
     05  FILLER        PIC X(3)     VALUE SPACES.
     05  SD-OUT.
         10 SD-MM      PIC XX.
         10 FILLER     PIC X        VALUE '/'.
         10 SD-DD      PIC XX.
         10 FILLER     PIC X        VALUE '/'.
         10 SD-YY      PIC XX.
     05  FILLER        PIC X(4)     VALUE SPACES.
     05  TITLE-OUT     PIC X(20).
     05  FILLER        PIC X(2)     VALUE SPACES.
     05  OFF-ADDRESS-OUT.
         10 STREET-OUT PIC X(20).
         10 FILLER     PIC XX       VALUE SPACES.
         10 CITY-OUT   PIC X(15).
         10 FILLER     PIC XX       VALUE SPACES.
         10 STATE-OUT  PIC XX.
         10 FILLER     PIC XX       VALUE SPACES.
         10 ZIP-OUT    PIC X(5).
     05  FILLER        PIC X(8)     VALUE SPACES.
 01  ERR-HEADER-1.
     05  FILLER        PIC X(40)    VALUE SPACES.
     05  FILLER        PIC X(12)    VALUE 'ERROR REPORT'.
     05  FILLER        PIC X(80)    VALUE SPACES.
 01  ERR-HEADER-2.
     05  FILLER        PIC X(10)    VALUE SPACES.
     05  FILLER        PIC X(4)     VALUE '*** '.
     05  FILLER        PIC X(51)    VALUE
         'THIS REPORT LISTS EMPTY AND NONEXISTENT DEPARTMENTS'.
     05  FILLER        PIC X(4)     VALUE ' ***'.
```

```
     05  FILLER        PIC X(63)    VALUE SPACES.
 01  ERR-HEADER-3.
     05  FILLER        PIC X(20)    VALUE SPACES.
     05  FILLER        PIC X(7)     VALUE 'DEPT ID'.
     05  FILLER        PIC X(9)     VALUE SPACES.
     05  FILLER        PIC X(7)     VALUE 'MESSAGE'.
     05  FILLER        PIC X(89)    VALUE SPACES.
 01  ERR-DETAIL-LINE.
     05  FILLER        PIC X(20)    VALUE SPACES.
     05  ERR-ID-OUT    PIC X(4).
     05  FILLER        PIC X(12)    VALUE SPACES.
     05  ERR-MESS-OUT  PIC X(15).
     05  FILLER        PIC X(79)    VALUE SPACES.
****************************************************************
 01  MESSAGES.
     05 NO-JOB-MESSAGE.
        10 FILLER          PIC X(20)  VALUE 'NO JOB ASSIGNED'.
     05 NO-OFFICE-MESSAGE.
        10 FILLER          PIC X(20)
                          VALUE 'NO OFFICE ASSIGNED'.
     05 NO-DEPT-MESSAGE.
        10 FILLER          PIC X(15)  VALUE 'DOES NOT EXIST'.
     05 NO-EMP-MESSAGE.
        10 FILLER          PIC X(15)  VALUE 'IS EMPTY'.
     05 NO-INPUT-MESSAGE.
        10 FILLER          PIC XX     VALUE SPACES.
        10 FILLER          PIC X(11)  VALUE '========>> '.
        10 FILLER          PIC X(8)   VALUE 'NO INPUT'.
        10 FILLER          PIC X(11)  VALUE ' <<========'.
        10 FILLER          PIC X(100) VALUE SPACES.

 01  COPY IDMS SUBSCHEMA-CTRL.

 01  COPY IDMS SUBSCHEMA-SSNAME.

 01  COPY IDMS SUBSCHEMA-RECNAMES.

 01  COPY IDMS SUBSCHEMA-SETNAMES.

 01  COPY IDMS RECORD EMPLOYEE.

 01  COPY IDMS RECORD DEPARTMENT.

 01  COPY IDMS RECORD JOB.

 01  COPY IDMS RECORD EMPOSITION.

 01  COPY IDMS RECORD OFFICE.
     EJECT
 PROCEDURE DIVISION.

*    ********************************************************
*    *  PROCEDURE DIVISION GENERAL STRATEGY:                *
*    *      1) READ DEPT-ID-IN, WHICH CONTAINS THE          *
*    *         DEPT-ID NUMBER                               *
*    *      2) ACCESS THE DATABASE USING THE DEPT-ID NUMBER  *
*    *         WITH AN OBTAIN CALC ON THE DEPARTMENT RECORD  *
*    *      3) ACCESS ALL EMPLOYEES IN THE DEPT-EMPLOYEE SET *
*    *         AND RETRIEVE RELATED JOB AND OFFICE DATA      *
*    *      4) PRINT A REPORT FOR EACH DEPARTMENT           *
*    *      5) PRINT AN ERROR REPORT FOR EMPTY DEPARTMENTS   *
*    *         AND NONEXISTENT DEPARTMENTS (NO MATCHING      *
*    *         DEPT-ID)                                     *
*    ********************************************************

 MAIN-LINE.
     PERFORM INIT-FILES.
     IF END-OF-FILE
        PERFORM EMPTY-INPUT-PROCESSING
     ELSE
        PERFORM INIT-BIND-READY
        PERFORM U220-ERR-HEADER
        PERFORM DEPT-PROCESSING THRU DEPT-PROCESSING-EXIT
                     UNTIL END-OF-FILE.
     PERFORM END-PROCESSING.
     GOBACK.
```

```
 INIT-BIND-READY.
*************************************************************
*  THE BIND STATEMENTS ARE PERFORMED INDIVIDUALLY (RATHER     *
* THAN BY USING A COPY IDMS SUBSCHEMA-BINDS) IN ORDER TO      *
* CHECK EACH ERROR-STATUS BY PERFORMING THE IDMS-STATUS       *
* ROUTINE.                                                    *
*************************************************************
     MOVE 'DEPTRPT' TO PROGRAM-NAME.
     BIND RUN-UNIT.
     PERFORM IDMS-STATUS.
     BIND EMPLOYEE.
     PERFORM IDMS-STATUS.
     BIND DEPARTMENT.
     PERFORM IDMS-STATUS.
     BIND JOB.
     PERFORM IDMS-STATUS.
     BIND EMPOSITION.
     PERFORM IDMS-STATUS.
     BIND OFFICE.
     PERFORM IDMS-STATUS.
     READY.
     PERFORM IDMS-STATUS.

 INIT-FILES.
     OPEN INPUT DEPT-FILE-IN.
     OPEN OUTPUT DEPT-FILE-OUT.
     OPEN OUTPUT ERR-FILE-OUT.
     MOVE SPACES TO PRINT-LINE.
     MOVE SPACES TO ERR-LINE.
     READ DEPT-FILE-IN AT END MOVE 'Y' TO EOF-SW.

 EMPTY-INPUT-PROCESSING.
     MOVE NO-INPUT-MESSAGE TO PRINT-LINE.
     MOVE '1' TO CC.
     PERFORM U000-WRITE-LINE.

*************************************************************
* THIS PARAGRAPH ACCESSES THE DATABASE USING THE DEPT-ID-0415 *
* CALCKEY VALUE.                                              *
*************************************************************
 DEPT-PROCESSING.
     MOVE DEPT-ID-IN TO DEPT-ID-0410.
     OBTAIN CALC DEPARTMENT.
     IF DB-REC-NOT-FOUND THEN
         PERFORM NO-DEPT-PROCESSING
     ELSE
         PERFORM IDMS-STATUS
         IF DEPT-EMPLOYEE IS NOT EMPTY THEN
             PERFORM U020-VALID-HEADER
             MOVE DEPT-ID-0410  TO DEPT-ID-OUT
             MOVE DEPT-NAME-0410  TO DEPT-NAME-OUT
             MOVE DEPT-DETAIL-LINE TO PRINT-LINE
             PERFORM U000-WRITE-LINE
             PERFORM U030-EMP-HEADERS
             PERFORM SET-WALK THRU SET-WALK-EXIT
                          UNTIL DB-END-OF-SET
         ELSE
              PERFORM EMPTY-SET.
     READ DEPT-FILE-IN AT END MOVE 'Y' TO EOF-SW.
 DEPT-PROCESSING-EXIT.
     EXIT.

*************************************************************
* THIS PARAGRAPH RETRIEVES EMPLOYEE, JOB, AND OFFICE DATA     *
* FOR EACH EMPLOYEE IN THE DEPT-EMPLOYEE SET.                 *
*************************************************************
 SET-WALK.
     OBTAIN NEXT EMPLOYEE WITHIN DEPT-EMPLOYEE.
     IF DB-END-OF-SET
        GO TO SET-WALK-EXIT
     ELSE
        PERFORM IDMS-STATUS.
     MOVE EMP-ID-0415 TO ID-OUT.
     MOVE EMP-LAST-NAME-0415 TO LAST-OUT.
     MOVE EMP-FIRST-NAME-0415 TO FIRST-OUT.
```

```
    MOVE START-YEAR-0415 TO SD-YY.
    MOVE START-MONTH-0415 TO SD-MM.
    MOVE START-DAY-0415 TO SD-DD.
    IF EMP-EMPOSITION IS EMPTY
        MOVE NO-JOB-MESSAGE TO TITLE-OUT
    ELSE
        FIND FIRST WITHIN EMP-EMPOSITION
        PERFORM IDMS-STATUS
        IF NOT JOB-EMPOSITION MEMBER
            MOVE NO-JOB-MESSAGE TO TITLE-OUT
        ELSE
            OBTAIN OWNER WITHIN JOB-EMPOSITION
            PERFORM IDMS-STATUS
            MOVE TITLE-0440 TO TITLE-OUT.
    IF OFFICE-EMPLOYEE IS EMPTY
        MOVE NO-OFFICE-MESSAGE TO STREET-OUT
        MOVE SPACES TO CITY-OUT
        MOVE SPACES TO STATE-OUT
        MOVE SPACES TO ZIP-OUT
    ELSE
        OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
        PERFORM IDMS-STATUS
        MOVE OFFICE-STREET-0450 TO STREET-OUT
        MOVE OFFICE-CITY-0450 TO CITY-OUT
        MOVE OFFICE-STATE-0450 TO STATE-OUT
        MOVE OFFICE-ZIP-FIRST-FIVE-0450 TO ZIP-OUT
        MOVE EMP-DETAIL-LINE TO PRINT-LINE.
    PERFORM U000-WRITE-LINE.
SET-WALK-EXIT.
    EXIT.

END-PROCESSING.
    FINISH.
    PERFORM IDMS-STATUS.
    CLOSE DEPT-FILE-OUT.
    CLOSE ERR-FILE-OUT.
    CLOSE DEPT-FILE-IN.

EMPTY-SET.
    MOVE SPACES TO ERR-LINE.
    MOVE DEPT-ID-0410 TO ERR-ID-OUT.
    MOVE NO-EMP-MESSAGE TO ERR-MESS-OUT.
    MOVE ERR-DETAIL-LINE TO ERR-LINE.
    PERFORM U200-WRITE-ERR-LINE.

NO-DEPT-PROCESSING.
    MOVE DEPT-ID-IN TO ERR-ID-OUT.
    MOVE NO-DEPT-MESSAGE TO ERR-MESS-OUT.
    MOVE ERR-DETAIL-LINE TO ERR-LINE.
    PERFORM U200-WRITE-ERR-LINE.

U000-WRITE-LINE.
    WRITE DEPT-REC-OUT AFTER POSITIONING CC.
    IF CC = '1' THEN MOVE 0 TO LINE-COUNT
      ELSE IF CC = ' ' THEN ADD 1 TO LINE-COUNT
           ELSE IF CC = '0' THEN ADD 2 TO LINE-COUNT.
    IF LINE-COUNT > LINE-MAX
                  THEN PERFORM U010-NEW-PAGE-ROUTINE.
U010-NEW-PAGE-ROUTINE.
    PERFORM U020-VALID-HEADER.
    MOVE DEPT-DETAIL-LINE TO PRINT-LINE.
    PERFORM U000-WRITE-LINE.
    PERFORM U030-EMP-HEADERS.
U020-VALID-HEADER.
    MOVE DEPT-HEADER TO PRINT-LINE.
    MOVE '1' TO CC.
    PERFORM U000-WRITE-LINE
    MOVE ' ' TO CC.
U030-EMP-HEADERS.
    MOVE '0' TO CC.
    MOVE EMP-HEADER TO PRINT-LINE.
    PERFORM U000-WRITE-LINE.
    MOVE SPACES TO PRINT-LINE.
    MOVE ' ' TO CC.
    PERFORM U000-WRITE-LINE.
```

```
U200-WRITE-ERR-LINE.
    WRITE ERR-REC-OUT AFTER POSITIONING ERR-CC.
    IF ERR-CC = '1' THEN MOVE 0 TO ERR-LINE-COUNT
      ELSE IF ERR-CC = ' ' THEN ADD 1 TO ERR-LINE-COUNT
          ELSE IF ERR-CC = '0' THEN ADD 2 TO ERR-LINE-COUNT.
    IF ERR-LINE-COUNT > LINE-MAX THEN
                        PERFORM U220-ERR-HEADER.
U220-ERR-HEADER.
    MOVE ERR-HEADER-1 TO ERR-LINE.
    MOVE '1' TO ERR-CC.
    PERFORM U200-WRITE-ERR-LINE
    MOVE '0' TO ERR-CC.
    MOVE ERR-HEADER-2 TO ERR-LINE.
    PERFORM U200-WRITE-ERR-LINE.
    MOVE ERR-HEADER-3 TO ERR-LINE.
    PERFORM U200-WRITE-ERR-LINE.
    MOVE SPACES TO ERR-LINE.
    MOVE ' ' TO ERR-CC.
    PERFORM U200-WRITE-ERR-LINE.
IDMS-ABORT.
    EXIT.
IDMS-ABORT-EXIT.
    COPY IDMS IDMS-STATUS.
```

B.3 Sample batch program as output from the DML compiler

Since the *DMLIST option is specified in the program's IDENTIFICATION DIVISION, printed output consists of expanded code as well as diagnostics. This output is in the following format:

- **Heading** — The top of each page of the listing contains the name of the DML compiler being used (IDMSDMLC), the release number of the processor (Release 10.0), the name of the listing (Listing of Messages), the date, the time, and the page number.
- **Input listing and DML compiler-generated code** — The body of the printout contains the program input listing along with the DML compiler-generated code, formatted as follows:

Column	Explanation
1	Sequence numbers generated by the DML compiler
12	Line numbers generated by the DML compiler
19	Line numbers generated by the user program
26	Text of the COBOL source code including text generated by the DML compiler

- **Warnings and Status Messages** — Diagnostics are imbedded in the input listing and DML compiler-generated code following the errant lines of source code. For a complete description of DML compiler status messages, refer to *CA-IDMS Messages and Codes*.

This listing contains the sample batch program and partially expanded code generated by the DML compiler.

```
00001          *RETRIEVAL
00002          *DMLIST
00003          *NO-ACTIVITY-LOG
00004          *SCHEMA-COMMENTS
00005
00006           IDENTIFICATION DIVISION.
00007
00008              PROGRAM-ID.                 DEPTRPT.
00009
00010              AUTHOR.                     COMPUTER ASSOCIATES INTERNATIONAL.
00011
00012              DATE-WRITTEN.               APRIL 1995.
00013
00014              REMARKS.                    THIS PROGRAM DEMONSTRATES
00015                                          CA-IDMS DATABASE ACCESS USING
00016                                          COBOL DML STATEMENTS. IT READS
00017                                          DEPARTMENT ID NUMBERS AND RETRIEVES
00018                                          RELATED RECORD OCCURRENCES,
00019                                          PRINTING A REPORT THAT INCLUDES
00020                                          DEPARTMENT, EMPLOYEE, JOB, AND
00021                                          OFFICE INFORMATION.
00022          ************************************************************
00023           ENVIRONMENT DIVISION.
```

```
         00024          INPUT-OUTPUT SECTION.
         00025          FILE-CONTROL.
         00026              SELECT DEPT-FILE-IN           ASSIGN TO INFILE.
         00027              SELECT DEPT-FILE-OUT          ASSIGN TO OUTFILE.
         00028              SELECT ERR-FILE-OUT           ASSIGN TO ERRFILE.
         00029         ***************************************************************
DMLC     00030          IDMS-CONTROL SECTION.
         00031
         00032          PROTOCOL.                      MODE IS BATCH DEBUG
         00033                                         IDMS-RECORDS MANUAL.
         00034               SKIP3
         00035         ***************************************************************
         00036          DATA DIVISION.
         00037
DMLC     00038          SCHEMA SECTION.
         00039
         00040          DB EMPSS01 WITHIN EMPSCHM.
         00041
         00042         ***************************************************************
         00043          FILE SECTION.
         00044
         00045          FD  DEPT-FILE-IN
         00046              RECORD CONTAINS 80
         00047              BLOCK  CONTAINS 80 CHARACTERS
         00048              RECORDING MODE IS F
         00049              LABEL RECORDS ARE OMITTED.
         00050
         00051          01  DEPT-REC-IN.
         00052              02 DEPT-ID-IN             PIC 9(4).
         00053              02 DEPT-IN-FILLER         PIC X(76).
         00054
         00055          FD  DEPT-FILE-OUT
         00056              RECORD CONTAINS 133
         00057              BLOCK  CONTAINS 133 CHARACTERS
         00058              RECORDING MODE IS F
         00059              LABEL RECORDS ARE OMITTED.
         00060
         00061          01  DEPT-REC-OUT.
         00062              02 CC                     PIC X.
         00063              02 PRINT-LINE             PIC X(132).
         00064
         00065          FD  ERR-FILE-OUT
         00066              RECORD CONTAINS 133
         00067              BLOCK  CONTAINS 133 CHARACTERS
         00068              RECORDING MODE IS F
         00069              LABEL RECORDS ARE OMITTED.
         00070
         00071          01  ERR-REC-OUT.
         00072              02 ERR-CC                 PIC X.
         00073              02 ERR-LINE               PIC X(132).
         00074
         00075         ***************************************************************
         00076          WORKING-STORAGE SECTION.
         00077          01  EOF-SW             PIC X      VALUE 'N'.
         00078              88 END-OF-FILE               VALUE 'Y'.
         00079          01  LINE-COUNT        PIC 99     VALUE 0.
         00080          01  ERR-LINE-COUNT    PIC 99     VALUE 0.
         00081          01  LINE-MAX          PIC 99     VALUE 50.
         00082         ***************************************************************
         00083          01  DEPT-HEADER.
         00084              05  FILLER        PIC X(30)    VALUE SPACES.
         00085              05  FILLER        PIC X(13)    VALUE 'DEPARTMENT ID'.
         00086              05  FILLER        PIC X(10)    VALUE SPACES.
         00087              05  FILLER        PIC X(9)     VALUE 'DEPT NAME'.
         00088              05  FILLER        PIC X(70)    VALUE SPACES.
         00089          01  DEPT-DETAIL-LINE.
         00090              05  FILLER        PIC X(33)    VALUE SPACES.
         00091              05  DEPT-ID-OUT   PIC X(4).
         00092              05  FILLER        PIC X(16)    VALUE SPACES.
         00093              05  DEPT-NAME-OUT PIC X(45).
         00094              05  FILLER        PIC X(34)    VALUE SPACES.
         00095          01  EMP-HEADER.
         00096              05  FILLER        PIC X(5)     VALUE SPACES.
         00097              05  FILLER        PIC X(6)     VALUE 'EMP ID'.
         00098              05  FILLER        PIC X(2)     VALUE SPACES.
         00099              05  FILLER        PIC X(9)     VALUE 'LAST NAME'.
```

```
00100               05  FILLER        PIC X(8)     VALUE SPACES.
00101               05  FILLER        PIC X(10)    VALUE 'FIRST NAME'.
00102               05  FILLER        PIC X(3)     VALUE SPACES.
00103               05  FILLER        PIC X(10)    VALUE 'START DATE'.
00104               05  FILLER        PIC X(2)     VALUE SPACES.
00105               05  FILLER        PIC X(9)     VALUE 'JOB TITLE'.
00106               05  FILLER        PIC X(13)    VALUE SPACES.
00107               05  FILLER        PIC X(14)    VALUE 'OFFICE ADDRESS'.
00108               05  FILLER        PIC X(42)    VALUE SPACES.
00109           01  EMP-DETAIL-LINE.
00110               05  FILLER        PIC X(5)     VALUE SPACES.
00111               05  ID-OUT        PIC X(4).
00112               05  FILLER        PIC X(4)     VALUE SPACES.
00113               05  LAST-OUT      PIC X(15).
00114               05  FILLER        PIC X(2)     VALUE SPACES.
00115               05  FIRST-OUT     PIC X(10).
00116               05  FILLER        PIC X(3)     VALUE SPACES.
00117               05  SD-OUT.
00118                   10 SD-MM      PIC XX.
00119                   10 FILLER     PIC X        VALUE '/'.
00120                   10 SD-DD      PIC XX.
00121                   10 FILLER     PIC X        VALUE '/'.
00122                   10 SD-YY      PIC XX.
00123               05  FILLER        PIC X(4)     VALUE SPACES.
00124               05  TITLE-OUT     PIC X(20).
00125               05  FILLER        PIC X(2)     VALUE SPACES.
00126               05  OFF-ADDRESS-OUT.
00127                   10 STREET-OUT PIC X(20).
00128                   10 FILLER     PIC XX       VALUE SPACES.
00129                   10 CITY-OUT   PIC X(15).
00130                   10 FILLER     PIC XX       VALUE SPACES.
00131                   10 STATE-OUT  PIC XX.
00132                   10 FILLER     PIC XX       VALUE SPACES.
00133                   10 ZIP-OUT    PIC X(5).
00134               05  FILLER        PIC X(8)     VALUE SPACES.
00135           01  ERR-HEADER-1.
00136               05  FILLER        PIC X(40)    VALUE SPACES.
00137               05  FILLER        PIC X(12)    VALUE 'ERROR REPORT'.
00138               05  FILLER        PIC X(80)    VALUE SPACES.
00139           01  ERR-HEADER-2.
00140               05  FILLER        PIC X(10)    VALUE SPACES.
00141               05  FILLER        PIC X(4)     VALUE '*** '.
00142               05  FILLER        PIC X(51)    VALUE
00143                   'THIS REPORT LISTS EMPTY AND NONEXISTENT DEPARTMENTS'.
00144               05  FILLER        PIC X(4)     VALUE ' ***'.
00145               05  FILLER        PIC X(63)    VALUE SPACES.
00146           01  ERR-HEADER-3.
00147               05  FILLER        PIC X(20)    VALUE SPACES.
00148               05  FILLER        PIC X(7)     VALUE 'DEPT ID'.
00149               05  FILLER        PIC X(9)     VALUE SPACES.
00150               05  FILLER        PIC X(7)     VALUE 'MESSAGE'.
00151               05  FILLER        PIC X(89)    VALUE SPACES.
00152           01  ERR-DETAIL-LINE.
00153               05  FILLER        PIC X(20)    VALUE SPACES.
00154               05  ERR-ID-OUT    PIC X(4).
00155               05  FILLER        PIC X(12)    VALUE SPACES.
00156               05  ERR-MESS-OUT  PIC X(15).
00157               05  FILLER        PIC X(79)    VALUE SPACES.
00158          ****************************************************************
00159           01  MESSAGES.
00160               05 NO-JOB-MESSAGE.
00161                  10 FILLER          PIC X(20)  VALUE 'NO JOB ASSIGNED'.
00162               05 NO-OFFICE-MESSAGE.
00163                  10 FILLER          PIC X(20)
00164                                    VALUE 'NO OFFICE ASSIGNED'.
00165               05 NO-DEPT-MESSAGE.
00166                  10 FILLER          PIC X(15)  VALUE 'DOES NOT EXIST'.
00167               05 NO-EMP-MESSAGE.
00168                  10 FILLER          PIC X(15)  VALUE 'IS EMPTY'.
00169               05 NO-INPUT-MESSAGE.
00170                  10 FILLER          PIC XX     VALUE SPACES.
00171                  10 FILLER          PIC X(11)  VALUE '========>> '.
00172                  10 FILLER          PIC X(8)   VALUE 'NO INPUT'.
00173                  10 FILLER          PIC X(11)  VALUE ' <<========'.
00174                  10 FILLER          PIC X(100) VALUE SPACES.
```

```
                 00175
DMLC             00176          01  COPY IDMS SUBSCHEMA-CTRL.
                 00177          01  SUBSCHEMA-CTRL.
                 00178               03  PROGRAM-NAME           PIC X(8)
                 00179                                           VALUE  SPACES .
                 00180               03  ERROR-STATUS           PIC X(4)
                 00181                                           VALUE  '1400' .
                 00182                                       88  DB-STATUS-OK
                 00183                                           VALUE  '0000' .
                 00184                                       88  ANY-STATUS
                 00185                                           VALUE  '    ' THRU '9999'  .
                 00186                                       88  ANY-ERROR-STATUS
                 00187                                           VALUE  '0001' THRU '9999'  .
                 00188                                       88  DB-END-OF-SET
                 00189                                           VALUE  '0307' .
                 00190                                       88  DB-REC-NOT-FOUND
                 00191                                           VALUE  '0326' .
                 00192               03  DBKEY                  PIC S9(8) COMP SYNC.
                 00193               03  RECORD-NAME            PIC X(16)
                 00194                                           VALUE  SPACES .
                 00195               03  RRECORD-NAME           REDEFINES RECORD-NAME.
                 00196                 05  SSC-NODN             PIC X(8).
                 00197                 05  SSC-DBN              PIC X(8).
                 00198               03  AREA-NAME              PIC X(16)
                 00199                                           VALUE  SPACES .
                 00200               03  AREA-RNAME             REDEFINES AREA-NAME.
                 00201                 05  SSC-DNO              PIC X(8).
                 00202                 05  SSC-DNA              PIC X(8).
                 00203               03  ERROR-SET              PIC X(16)
                 00204                                           VALUE  SPACES .
                 00205               03  ERROR-RECORD           PIC X(16)
                 00206                                           VALUE  SPACES .
                 00207               03  ERROR-AREA             PIC X(16)
                 00208                                           VALUE  SPACES .
                 00209               03  IDBMSCOM-AREA          PIC X(100)
                 00210                                           VALUE  LOW-VALUE .
                 00211               03  IDBMSCOM               REDEFINES IDBMSCOM-AREA
                 00212                                           PIC X
                 00213                                           OCCURS 100.
                 00214               03  RIDBMSCOM              REDEFINES IDBMSCOM-AREA.
                 00215                 05  DB-SUB-ADDR          PIC X(4).
                 00216                 05  FILLER               PIC X(96).
                 00217               03  R1DBMSCOM              REDEFINES IDBMSCOM-AREA.
                 00218                 05  PAGE-INFO.
                 00219                   07  PAGE-INFO-GROUP    PIC S9(4) COMP.
                 00220                   07  PAGE-INFO-DBK-FORMAT
                 00221                                           PIC 9(4) COMP.
                 00222                 05  FILLER               PIC X(96).
                 00223               03  DIRECT-DBKEY           PIC S9(8) COMP SYNC.
                 00224               03  DIRECT-DBK             REDEFINES DIRECT-DBKEY
                 00225                                           PIC S9(8) COMP SYNC.
                 00226               03  DATABASE-STATUS.
                 00227                 05  DBSTATMENT-CODE      PIC X(2).
                 00228                 05  DBSTATUS-CODE        PIC X(5).
                 00229               03  FILLER                 PIC X.
                 00230               03  RECORD-OCCUR           PIC S9(8) COMP SYNC.
                 00231               03  DML-SEQUENCE           PIC S9(8) COMP SYNC.
                 00232
DMLC             00233          01  COPY IDMS SUBSCHEMA-SSNAME.
                 00234          01  SUBSCHEMA-SSNAME            PIC X(8)
                 00235                                           VALUE  'EMPSS01 ' .
                 00236
DMLC             00237          01  COPY IDMS SUBSCHEMA-RECNAMES.
                 00238          01  SUBSCHEMA-RECNAMES.
                 00239               03  SR460                  PIC X(16)
                 00240                                           VALUE  'STRUCTURE       ' .
                 00241               03  SR455                  PIC X(16)
                 00242                                           VALUE  'SKILL           ' .
                 00243               03  SR450                  PIC X(16)
                 00244                                           VALUE  'OFFICE          ' .
                 00245               03  SR445                  PIC X(16)
                 00246                                           VALUE  'NON-HOSP-CLAIM  ' .
                 00247               03  SR440                  PIC X(16)
                 00248                                           VALUE  'JOB             ' .
                 00249               03  SR435                  PIC X(16)
                 00250                                           VALUE  'INSURANCE-PLAN  ' .
```

```
                 00251              03  SR430                PIC X(16)
                 00252                                       VALUE  'HOSPITAL-CLAIM  ' .
                 00253              03  SR425                PIC X(16)
                 00254                                       VALUE  'EXPERTISE       ' .
                 00255              03  SR420                PIC X(16)
                 00256                                       VALUE  'EMPOSITION      ' .
                 00257              03  SR415                PIC X(16)
                 00258                                       VALUE  'EMPLOYEE        ' .
                 00259              03  SR410                PIC X(16)
                 00260                                       VALUE  'DEPARTMENT      ' .
                 00261              03  SR405                PIC X(16)
                 00262                                       VALUE  'DENTAL-CLAIM    ' .
                 00263              03  SR400                PIC X(16)
                 00264                                       VALUE  'COVERAGE        ' .
                 00265
DMLC             00266          01  COPY IDMS SUBSCHEMA-SETNAMES.
                 00267          01  SUBSCHEMA-SETNAMES.
                 00268              03  COVERAGE-CLAIMS        PIC X(16)
                 00269                                       VALUE  'COVERAGE-CLAIMS ' .
                 00270              03  DEPT-EMPLOYEE          PIC X(16)
                 00271                                       VALUE  'DEPT-EMPLOYEE   ' .
                 00272              03  EMP-COVERAGE           PIC X(16)
                 00273                                       VALUE  'EMP-COVERAGE    ' .
                 00274              03  EMP-EXPERTISE          PIC X(16)
                 00275                                       VALUE  'EMP-EXPERTISE   ' .
                 00276              03  EMP-NAME-NDX           PIC X(16)
                 00277                                       VALUE  'EMP-NAME-NDX    ' .
                 00278              03  EMP-EMPOSITION         PIC X(16)
                 00279                                       VALUE  'EMP-EMPOSITION  ' .
                 00280              03  JOB-EMPOSITION         PIC X(16)
                 00281                                       VALUE  'JOB-EMPOSITION  ' .
                 00282              03  JOB-TITLE-NDX          PIC X(16)
                 00283                                       VALUE  'JOB-TITLE-NDX   ' .
                 00284              03  MANAGES                PIC X(16)
                 00285                                       VALUE  'MANAGES         ' .
                 00286              03  OFFICE-EMPLOYEE        PIC X(16)
                 00287                                       VALUE  'OFFICE-EMPLOYEE ' .
                 00288              03  REPORTS-TO             PIC X(16)
                 00289                                       VALUE  'REPORTS-TO      ' .
                 00290              03  SKILL-EXPERTISE        PIC X(16)
                 00291                                       VALUE  'SKILL-EXPERTISE ' .
                 00292              03  SKILL-NAME-NDX         PIC X(16)
                 00293                                       VALUE  'SKILL-NAME-NDX  ' .
                 00294              03  CALC                   PIC X(16)
                 00295                                       VALUE  'CALC            ' .
                 00296
DMLC             00297          01  COPY IDMS RECORD EMPLOYEE.
                 00298          01  EMPLOYEE.
                 00299              02  EMP-ID-0415             PIC 9(4).
                 00300              02  EMP-NAME-0415.
                 00301               03  EMP-FIRST-NAME-0415    PIC X(10).
                 00302               03  EMP-LAST-NAME-0415     PIC X(15).
                 00303              02  EMP-ADDRESS-0415.
                 00304               03  EMP-STREET-0415        PIC X(20).
                 00305               03  EMP-CITY-0415          PIC X(15).
                 00306               03  EMP-STATE-0415         PIC X(2).
                 00307               03  EMP-ZIP-0415.
                 00308                04  EMP-ZIP-FIRST-FIVE-0415
                 00309                                       PIC X(5).
                 00310                04  EMP-ZIP-LAST-FOUR-0415
                 00311                                       PIC X(4).
                 00312              02  EMP-PHONE-0415          PIC 9(10).
                 00313              02  STATUS-0415             PIC X(2).
                 00314                                   88  ACTIVE-0415
                 00315                                       VALUE  '01' .
                 00316                                   88  ST-DISABIL-0415
                 00317                                       VALUE  '02' .
                 00318                                   88  LT-DISABIL-0415
                 00319                                       VALUE  '03' .
                 00320                                   88  LEAVE-OF-ABSENCE-0415
                 00321                                       VALUE  '04' .
                 00322                                   88  TERMINATED-0415
                 00323                                       VALUE  '05' .
                 00324              02  SS-NUMBER-0415          PIC 9(9).
                 00325              02  START-DATE-0415.
                 00326               03  START-YEAR-0415        PIC 9(4).
```

```
00327                 03  START-MONTH-0415       PIC 9(2).
00328                 03  START-DAY-0415         PIC 9(2).
00329                02  TERMINATION-DATE-0415.
00330                 03  TERMINATION-YEAR-0415  PIC 9(4).
00331                 03  TERMINATION-MONTH-0415 PIC 9(2).
00332                 03  TERMINATION-DAY-0415   PIC 9(2).
00333                02  BIRTH-DATE-0415.
00334                 03  BIRTH-YEAR-0415        PIC 9(4).
00335                 03  BIRTH-MONTH-0415       PIC 9(2).
00336                 03  BIRTH-DAY-0415         PIC 9(2).
00337
DMLC        00338           01  COPY IDMS RECORD DEPARTMENT.
00339           01  DEPARTMENT.
00340               02  DEPT-ID-0410            PIC 9(4).
00341               02  DEPT-NAME-0410          PIC X(45).
00342               02  DEPT-HEAD-ID-0410       PIC 9(4).
00343               02  FILLER                  PIC XXX.
00344
DMLC        00345           01  COPY IDMS RECORD JOB.
00346           01  JOB.
00347               02  JOB-ID-0440             PIC 9(4).
00348               02  TITLE-0440              PIC X(20).
00349               02  DESCRIPTION-0440.
00350                03  DESCRIPTION-LINE-0440  PIC X(60)
00351                                            OCCURS 2.
00352               02  REQUIREMENTS-0440.
00353                03  REQUIREMENT-LINE-0440  PIC X(60)
00354                                            OCCURS 2.
00355               02  MINIMUM-SALARY-0440     PIC S9(6)V99.
00356               02  MAXIMUM-SALARY-0440     PIC S9(6)V99.
00357               02  SALARY-GRADES-0440      PIC 9(2)
00358                                            OCCURS 4.
00359               02  NUMBER-OF-POSITIONS-0440
00360                                            PIC 9(3).
00361               02  NUMBER-OPEN-0440        PIC 9(3).
00362               02  FILLER                  PIC XX.
00363
DMLC        00364           01  COPY IDMS RECORD EMPOSITION.
00365           01  EMPOSITION.
00366               02  START-DATE-0420.
00367                03  START-YEAR-0420        PIC 9(4).
00368                03  START-MONTH-0420       PIC 9(2).
00369                03  START-DAY-0420         PIC 9(2).
00370               02  FINISH-DATE-0420.
00371                03  FINISH-YEAR-0420       PIC 9(4).
00372                03  FINISH-MONTH-0420      PIC 9(2).
00373                03  FINISH-DAY-0420        PIC 9(2).
00374               02  SALARY-GRADE-0420       PIC 9(2).
00375               02  SALARY-AMOUNT-0420      PIC S9(7)V99 COMP-3.
00376               02  BONUS-PERCENT-0420      PIC SV999 COMP-3.
00377               02  COMMISSION-PERCENT-0420 PIC SV999 COMP-3.
00378               02  OVERTIME-RATE-0420      PIC S9V99 COMP-3.
00379               02  FILLER                  PIC XXX.
00380
DMLC        00381           01  COPY IDMS RECORD OFFICE.
00382           01  OFFICE.
00383               02  OFFICE-CODE-0450        PIC X(3).
00384               02  OFFICE-ADDRESS-0450.
00385                03  OFFICE-STREET-0450     PIC X(20).
00386                03  OFFICE-CITY-0450       PIC X(15).
00387                03  OFFICE-STATE-0450      PIC X(2).
00388                03  OFFICE-ZIP-0450.
00389                 04  OFFICE-ZIP-FIRST-FIVE-0450
00390                                            PIC X(5).
00391                 04  OFFICE-ZIP-LAST-FOUR-0450
00392                                            PIC X(4).
00393               02  OFFICE-PHONE-0450       PIC 9(7)
00394                                            OCCURS 3.
00395               02  OFFICE-AREA-CODE-0450   PIC X(3).
00396               02  SPEED-DIAL-0450         PIC X(3).
00397               02  FILLER                  PIC X(4).
00398               EJECT
00399           PROCEDURE DIVISION.
00400
00401          *    ********************************************************
00402          *    *  PROCEDURE DIVISION GENERAL STRATEGY:                *
```

```
                 00403          *    *      1) READ DEPT-ID-IN, WHICH CONTAINS THE           *
                 00404          *    *         DEPT-ID NUMBER                                *
                 00405          *    *      2) ACCESS THE DATABASE USING THE DEPT-ID NUMBER  *
                 00406          *    *         WITH AN OBTAIN CALC ON THE DEPARTMENT RECORD  *
                 00407          *    *      3) ACCESS ALL EMPLOYEES IN THE DEPT-EMPLOYEE SET *
                 00408          *    *         AND RETRIEVE RELATED JOB AND OFFICE DATA      *
                 00409          *    *      4) PRINT A REPORT FOR EACH DEPARTMENT            *
                 00410          *    *      5) PRINT AN ERROR REPORT FOR EMPTY DEPARTMENTS   *
                 00411          *    *         AND NONEXISTENT DEPARTMENTS (NO MATCHING      *
                 00412          *    *         DEPT-ID)                                      *
                 00413          *    *******************************************************
                 00414
                 00415           MAIN-LINE.
                 00416               PERFORM INIT-FILES.
                 00417               IF END-OF-FILE
                 00418                  PERFORM EMPTY-INPUT-PROCESSING
                 00419               ELSE
                 00420                  PERFORM INIT-BIND-READY
                 00421                  PERFORM U220-ERR-HEADER
                 00422                  PERFORM DEPT-PROCESSING THRU DEPT-PROCESSING-EXIT
                 00423                                UNTIL END-OF-FILE.
                 00424               PERFORM END-PROCESSING.
                 00425               GOBACK.
                 00426
                 00427           INIT-BIND-READY.
                 00428          ***************************************************************
                 00429          *  THE BIND STATEMENTS ARE PERFORMED INDIVIDUALLY (RATHER     *
                 00430          * THAN BY USING A COPY IDMS SUBSCHEMA-BINDS) IN ORDER TO      *
                 00431          * CHECK EACH ERROR-STATUS BY PERFORMING THE IDMS-STATUS       *
                 00432          * ROUTINE.                                                    *
                 00433          ***************************************************************
                 00434               MOVE 'DEPTRPT' TO PROGRAM-NAME.
DMLC0001         00435               BIND RUN-UNIT.
                 00436                     MOVE 1 TO DML-SEQUENCE
                 00437                     CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00438                             IDBMSCOM (59)
                 00439                             SUBSCHEMA-CTRL
                 00440                             SUBSCHEMA-SSNAME.
                 00441               PERFORM IDMS-STATUS.
DMLC0002         00442               BIND EMPLOYEE.
                 00443                     MOVE 2 TO DML-SEQUENCE
                 00444                     CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00445                             IDBMSCOM (48)
                 00446                             SR415
                 00447                             EMPLOYEE.
                 00448               PERFORM IDMS-STATUS.
DMLC0003         00449               BIND DEPARTMENT.
                 00450                     MOVE 3 TO DML-SEQUENCE
                 00451                     CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00452                             IDBMSCOM (48)
                 00453                             SR410
                 00454                             DEPARTMENT.
                 00455               PERFORM IDMS-STATUS.
DMLC0004         00456               BIND JOB.
                 00457                     MOVE 4 TO DML-SEQUENCE
                 00458                     CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00459                             IDBMSCOM (48)
                 00460                             SR440
                 00461                             JOB.
                 00462               PERFORM IDMS-STATUS.
DMLC0005         00463               BIND EMPOSITION.
                 00464                     MOVE 5 TO DML-SEQUENCE
                 00465                     CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00466                             IDBMSCOM (48)
                 00467                             SR420
                 00468                             EMPOSITION.
                 00469               PERFORM IDMS-STATUS.
DMLC0006         00470               BIND OFFICE.
                 00471                     MOVE 6 TO DML-SEQUENCE
                 00472                     CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00473                             IDBMSCOM (48)
                 00474                             SR450
                 00475                             OFFICE.
                 00476               PERFORM IDMS-STATUS.
DMLC0007         00477               READY.
                 00478                     MOVE 7 TO DML-SEQUENCE
```

```
                 00479                 CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00480                         IDBMSCOM (37).
                 00481            PERFORM IDMS-STATUS.
                 00482
                 00483        INIT-FILES.
                 00484            OPEN INPUT DEPT-FILE-IN.
                 00485            OPEN OUTPUT DEPT-FILE-OUT.
                 00486            OPEN OUTPUT ERR-FILE-OUT.
                 00487            MOVE SPACES TO PRINT-LINE.
                 00488            MOVE SPACES TO ERR-LINE.
                 00489            READ DEPT-FILE-IN AT END MOVE 'Y' TO EOF-SW.
                 00490
                 00491        EMPTY-INPUT-PROCESSING.
                 00492            MOVE NO-INPUT-MESSAGE TO PRINT-LINE.
                 00493            MOVE '1' TO CC.
                 00494            PERFORM U000-WRITE-LINE.
                 00495
                 00496       ***************************************************************
                 00497       * THIS PARAGRAPH ACCESSES THE DATABASE USING THE DEPT-ID-0415 *
                 00498       * CALCKEY VALUE.                                             *
                 00499       ***************************************************************
                 00500        DEPT-PROCESSING.
                 00501            MOVE DEPT-ID-IN TO DEPT-ID-0410.
DMLC0008         00502            OBTAIN CALC DEPARTMENT.
                 00503                  MOVE 8 TO DML-SEQUENCE
                 00504                  CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00505                          IDBMSCOM (32)
                 00506                          SR410
                 00507                          IDBMSCOM (43).
                 00508            IF DB-REC-NOT-FOUND THEN
                 00509                PERFORM NO-DEPT-PROCESSING
                 00510            ELSE
                 00511                PERFORM IDMS-STATUS
DMLC0009         00512                IF DEPT-EMPLOYEE IS NOT EMPTY
                 00513                  MOVE 9 TO DML-SEQUENCE
                 00514                  CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00515                          IDBMSCOM (65)
                 00516                          DEPT-EMPLOYEE;
                 00517                  IF ERROR-STATUS EQUAL TO '1601'
                 00518                                          THEN
                 00519                    PERFORM U020-VALID-HEADER
                 00520                    MOVE DEPT-ID-0410  TO DEPT-ID-OUT
                 00521                    MOVE DEPT-NAME-0410  TO DEPT-NAME-OUT
                 00522                    MOVE DEPT-DETAIL-LINE TO PRINT-LINE
                 00523                    PERFORM U000-WRITE-LINE
                 00524                    PERFORM U030-EMP-HEADERS
                 00525                    PERFORM SET-WALK THRU SET-WALK-EXIT
                 00526                                 UNTIL DB-END-OF-SET
                 00527                ELSE
                 00528                     PERFORM EMPTY-SET.
                 00529            READ DEPT-FILE-IN AT END MOVE 'Y' TO EOF-SW.
                 00530        DEPT-PROCESSING-EXIT.
                 00531            EXIT.
                 00532
                 00533       ***************************************************************
                 00534       * THIS PARAGRAPH RETRIEVES EMPLOYEE, JOB, AND OFFICE DATA     *
                 00535       * FOR EACH EMPLOYEE IN THE DEPT-EMPLOYEE SET.                *
                 00536       ***************************************************************
                 00537        SET-WALK.
DMLC0010         00538            OBTAIN NEXT EMPLOYEE WITHIN DEPT-EMPLOYEE.
                 00539                  MOVE 10 TO DML-SEQUENCE
                 00540                  CALL 'IDMS' USING SUBSCHEMA-CTRL
                 00541                          IDBMSCOM (10)
                 00542                          SR415
                 00543                          DEPT-EMPLOYEE
                 00544                          IDBMSCOM (43).
                 00545            IF DB-END-OF-SET
                 00546               GO TO SET-WALK-EXIT
                 00547            ELSE
                 00548               PERFORM IDMS-STATUS.
                 00549            MOVE EMP-ID-0415 TO ID-OUT.
                 00550            MOVE EMP-LAST-NAME-0415 TO LAST-OUT.
                 00551            MOVE EMP-FIRST-NAME-0415 TO FIRST-OUT.
                 00552            MOVE START-YEAR-0415 TO SD-YY.
                 00553            MOVE START-MONTH-0415 TO SD-MM.
```

```
            00554                MOVE START-DAY-0415 TO SD-DD.
DMLC0011    00555                IF EMP-EMPOSITION IS EMPTY
            00556                      MOVE 11 TO DML-SEQUENCE
            00557                      CALL 'IDMS' USING SUBSCHEMA-CTRL
            00558                              IDBMSCOM (64)
            00559                              EMP-EMPOSITION;
            00560                      IF ERROR-STATUS EQUAL TO '0000'
            00561                    MOVE NO-JOB-MESSAGE TO TITLE-OUT
            00562                ELSE
DMLC0012    00563                    FIND FIRST WITHIN EMP-EMPOSITION
            00564                      MOVE 12 TO DML-SEQUENCE
            00565                      CALL 'IDMS' USING SUBSCHEMA-CTRL
            00566                              IDBMSCOM (20)
            00567                              EMP-EMPOSITION;
            00568                    PERFORM IDMS-STATUS
DMLC0013    00569                    IF NOT JOB-EMPOSITION MEMBER
            00570                      MOVE 13 TO DML-SEQUENCE
            00571                      CALL 'IDMS' USING SUBSCHEMA-CTRL
            00572                              IDBMSCOM (62)
            00573                              JOB-EMPOSITION;
            00574                      IF ERROR-STATUS EQUAL TO '1601'
            00575                        MOVE NO-JOB-MESSAGE TO TITLE-OUT
            00576                    ELSE
DMLC0014    00577                        OBTAIN OWNER WITHIN JOB-EMPOSITION
            00578                      MOVE 14 TO DML-SEQUENCE
            00579                      CALL 'IDMS' USING SUBSCHEMA-CTRL
            00580                              IDBMSCOM (31)
            00581                              JOB-EMPOSITION
            00582                              IDBMSCOM (43);
            00583                        PERFORM IDMS-STATUS
            00584                        MOVE TITLE-0440 TO TITLE-OUT.
DMLC0015    00585                IF OFFICE-EMPLOYEE IS EMPTY
            00586                      MOVE 15 TO DML-SEQUENCE
            00587                      CALL 'IDMS' USING SUBSCHEMA-CTRL
            00588                              IDBMSCOM (64)
            00589                              OFFICE-EMPLOYEE;
            00590                      IF ERROR-STATUS EQUAL TO '0000'
            00591                    MOVE NO-OFFICE-MESSAGE TO STREET-OUT
            00592                    MOVE SPACES TO CITY-OUT
            00593                    MOVE SPACES TO STATE-OUT
            00594                    MOVE SPACES TO ZIP-OUT
            00595                ELSE
DMLC0016    00596                    OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
            00597                      MOVE 16 TO DML-SEQUENCE
            00598                      CALL 'IDMS' USING SUBSCHEMA-CTRL
            00599                              IDBMSCOM (31)
            00600                              OFFICE-EMPLOYEE
            00601                              IDBMSCOM (43);
            00602                    PERFORM IDMS-STATUS
            00603                    MOVE OFFICE-STREET-0450 TO STREET-OUT
            00604                    MOVE OFFICE-CITY-0450 TO CITY-OUT
            00605                    MOVE OFFICE-STATE-0450 TO STATE-OUT
            00606                    MOVE OFFICE-ZIP-FIRST-FIVE-0450 TO ZIP-OUT
            00607                    MOVE EMP-DETAIL-LINE TO PRINT-LINE.
            00608                PERFORM U000-WRITE-LINE.
            00609            SET-WALK-EXIT.
            00610                EXIT.
            00611
            00612            END-PROCESSING.
DMLC0017    00613                FINISH.
            00614                      MOVE 17 TO DML-SEQUENCE
            00615                      CALL 'IDMS' USING SUBSCHEMA-CTRL
            00616                              IDBMSCOM (2).
            00617                PERFORM IDMS-STATUS.
            00618                CLOSE DEPT-FILE-OUT.
            00619                CLOSE ERR-FILE-OUT.
            00620                CLOSE DEPT-FILE-IN.
            00621
            00622            EMPTY-SET.
            00623                MOVE SPACES TO ERR-LINE.
            00624                MOVE DEPT-ID-0410 TO ERR-ID-OUT.
            00625                MOVE NO-EMP-MESSAGE TO ERR-MESS-OUT.
            00626                MOVE ERR-DETAIL-LINE TO ERR-LINE.
            00627                PERFORM U200-WRITE-ERR-LINE.
            00628
            00629            NO-DEPT-PROCESSING.
```

```
                 00630              MOVE DEPT-ID-IN TO ERR-ID-OUT.
                 00631              MOVE NO-DEPT-MESSAGE TO ERR-MESS-OUT.
                 00632              MOVE ERR-DETAIL-LINE TO ERR-LINE.
                 00633              PERFORM U200-WRITE-ERR-LINE.
                 00634
                 00635          U000-WRITE-LINE.
                 00636              WRITE DEPT-REC-OUT AFTER POSITIONING CC.
                 00637              IF CC = '1' THEN MOVE 0 TO LINE-COUNT
                 00638                ELSE IF CC = ' ' THEN ADD 1 TO LINE-COUNT
                 00639                    ELSE IF CC = '0' THEN ADD 2 TO LINE-COUNT.
                 00640              IF LINE-COUNT > LINE-MAX
                 00641                          THEN PERFORM U010-NEW-PAGE-ROUTINE.
                 00642          U010-NEW-PAGE-ROUTINE.
                 00643              PERFORM U020-VALID-HEADER.
                 00644              MOVE DEPT-DETAIL-LINE TO PRINT-LINE.
                 00645              PERFORM U000-WRITE-LINE.
                 00646              PERFORM U030-EMP-HEADERS.
                 00647          U020-VALID-HEADER.
                 00648              MOVE DEPT-HEADER TO PRINT-LINE.
                 00649              MOVE '1' TO CC.
                 00650              PERFORM U000-WRITE-LINE
                 00651              MOVE ' ' TO CC.
                 00652          U030-EMP-HEADERS.
                 00653              MOVE '0' TO CC.
                 00654              MOVE EMP-HEADER TO PRINT-LINE.
                 00655              PERFORM U000-WRITE-LINE.
                 00656              MOVE SPACES TO PRINT-LINE.
                 00657              MOVE ' ' TO CC.
                 00658              PERFORM U000-WRITE-LINE.
                 00659
                 00660          U200-WRITE-ERR-LINE.
                 00661              WRITE ERR-REC-OUT AFTER POSITIONING ERR-CC.
                 00662              IF ERR-CC = '1' THEN MOVE 0 TO ERR-LINE-COUNT
                 00663                ELSE IF ERR-CC = ' ' THEN ADD 1 TO ERR-LINE-COUNT
                 00664                   ELSE IF ERR-CC = '0' THEN ADD 2 TO ERR-LINE-COUNT.
                 00665              IF ERR-LINE-COUNT > LINE-MAX THEN
                 00666                              PERFORM U220-ERR-HEADER.
                 00667          U220-ERR-HEADER.
                 00668              MOVE ERR-HEADER-1 TO ERR-LINE.
                 00669              MOVE '1' TO ERR-CC.
                 00670              PERFORM U200-WRITE-ERR-LINE
                 00671              MOVE '0' TO ERR-CC.
                 00672              MOVE ERR-HEADER-2 TO ERR-LINE.
                 00673              PERFORM U200-WRITE-ERR-LINE.
                 00674              MOVE ERR-HEADER-3 TO ERR-LINE.
                 00675              PERFORM U200-WRITE-ERR-LINE.
                 00676              MOVE SPACES TO ERR-LINE.
                 00677              MOVE ' ' TO ERR-CC.
                 00678              PERFORM U200-WRITE-ERR-LINE.
                 00679          IDMS-ABORT.
                 00680              EXIT.
                 00681          IDMS-ABORT-EXIT.
DMLC             00682              COPY IDMS IDMS-STATUS.
                 00683         ******************************************************************
                 00684          IDMS-STATUS                                              SECTION.
                 00685         ******************************************************************
                 00686          IDMS-STATUS-PARAGRAPH.
                 00687                  IF DB-STATUS-OK GO TO ISABEX.
                 00688                  PERFORM IDMS-ABORT.
                 00689                  DISPLAY '**************************'
                 00690                          ' ABORTING - ' PROGRAM-NAME
                 00691                          ', '           ERROR-STATUS
                 00692                          ', '           ERROR-RECORD
                 00693                          ' **** RECOVER IDMS ****'
                 00694                          UPON CONSOLE.
                 00695                  DISPLAY 'PROGRAM NAME ------ ' PROGRAM-NAME.
                 00696                  DISPLAY 'ERROR STATUS ------ ' ERROR-STATUS.
                 00697                  DISPLAY 'ERROR RECORD ------ ' ERROR-RECORD.
                 00698                  DISPLAY 'ERROR SET --------- ' ERROR-SET.
                 00699                  DISPLAY 'ERROR AREA -------- ' ERROR-AREA.
                 00700                  DISPLAY 'LAST GOOD RECORD -- ' RECORD-NAME.
                 00701                  DISPLAY 'LAST GOOD AREA ---- ' AREA-NAME.
                 00702                  DISPLAY 'DML SEQUENCE ------ ' DML-SEQUENCE.
DMLC0018         00703                  ROLLBACK.
                 00704                    MOVE 18 TO DML-SEQUENCE
                 00705                    CALL 'IDMS' USING SUBSCHEMA-CTRL
```

```
00706                                IDBMSCOM (67).
00707                      CALL 'ABORT'.
00708            ISABEX. EXIT.

NO MESSAGES FOR PROGRAM DEPTRPT
```

B.4 Sample batch program from the COBOL precompiler

The following listing illustrates the sample batch program after precompilation by the COBOL precompiler. The original code is further expanded and includes the following:

- Line numbers generated by the COBOL compiler
- CA-IDMS call statements for the requested DML functions
- Diagnostic messages

For details on the expanded code generated by the DML compiler, refer to Appendix D, “CA-IDMS Call Formats” on page D-1.

This listing contains the sample program output from the COBOL compiler with the fully expanded code (including the calls to CA-IDMS) generated by the DML compiler.

```
00001         *DMLIST
00002         *NO-ACTIVITY-LOG
00003         *SCHEMA-COMMENTS
00004
00005          IDENTIFICATION DIVISION.
00006
00007             PROGRAM-ID.                  DEPTRPT.
00008
00009             AUTHOR.                      COMPUTER ASSOCIATES INTERNATIONAL.
00010
00011             DATE-WRITTEN.                APRIL 1995.
00012
00013             REMARKS.                     THIS PROGRAM DEMONSTRATES
00014                                          CA-IDMS DATABASE ACCESS USING
00015                                          COBOL DML STATEMENTS. IT READS
00016                                          DEPARTMENT ID NUMBERS AND RETRIEVES
00017                                          RELATED RECORD OCCURRENCES,
00018                                          PRINTING A REPORT THAT INCLUDES
00019                                          DEPARTMENT, EMPLOYEE, JOB, AND
00020                                          OFFICE INFORMATION.
00021         ***************************************************************
00022          ENVIRONMENT DIVISION.
00023          INPUT-OUTPUT SECTION.
00024          FILE-CONTROL.
00025              SELECT DEPT-FILE-IN           ASSIGN TO INFILE.
00026              SELECT DEPT-FILE-OUT          ASSIGN TO OUTFILE.
00027              SELECT ERR-FILE-OUT           ASSIGN TO ERRFILE.
00028         ***************************************************************
00029         *IDMS-CONTROL SECTION.
00030         *
00031         *PROTOCOL.                      MODE IS BATCH DEBUG
00032         *                               IDMS-RECORDS MANUAL.

00034         ***************************************************************
00035          DATA DIVISION.
00036
00037         *SCHEMA SECTION.
00038         *
00039         *DB EMPSS01 WITHIN EMPSCHM.
00040
00041         ***************************************************************
00042          FILE SECTION.
00043
00044          FD  DEPT-FILE-IN
00045              RECORD CONTAINS 80
00046              BLOCK  CONTAINS 80 CHARACTERS
00047              RECORDING MODE IS F
```

```
00048             LABEL RECORDS ARE OMITTED.
00049
00050         01  DEPT-REC-IN.
00051             02 DEPT-ID-IN             PIC 9(4).
00052             02 DEPT-IN-FILLER         PIC X(76).
00053
00054         FD  DEPT-FILE-OUT
00055             RECORD CONTAINS 133
00056             BLOCK  CONTAINS 133 CHARACTERS
00057             RECORDING MODE IS F
00058             LABEL RECORDS ARE OMITTED.
00059
00060         01  DEPT-REC-OUT.
00061             02 CC                      PIC X.
00062             02 PRINT-LINE              PIC X(132).
00063
00064         FD  ERR-FILE-OUT
00065             RECORD CONTAINS 133
00066             BLOCK  CONTAINS 133 CHARACTERS
00067             RECORDING MODE IS F
00068             LABEL RECORDS ARE OMITTED.
00069
00070         01  ERR-REC-OUT.
00071             02 ERR-CC                  PIC X.
00072             02 ERR-LINE                PIC X(132).
00073
00074        ***************************************************************
00075         WORKING-STORAGE SECTION.
00076         01  EOF-SW              PIC X      VALUE 'N'.
00077             88 END-OF-FILE                 VALUE 'Y'.
00078         01  LINE-COUNT        PIC 99       VALUE 0.
00079         01  ERR-LINE-COUNT    PIC 99       VALUE 0.
00080         01  LINE-MAX          PIC 99       VALUE 50.
00081        ***************************************************************
00082         01  DEPT-HEADER.
00083             05  FILLER        PIC X(30)    VALUE SPACES.
00084             05  FILLER        PIC X(13)    VALUE 'DEPARTMENT ID'.
00085             05  FILLER        PIC X(10)    VALUE SPACES.
00086             05  FILLER        PIC X(9)     VALUE 'DEPT NAME'.
00087             05  FILLER        PIC X(70)    VALUE SPACES.
00088         01  DEPT-DETAIL-LINE.
00089             05  FILLER        PIC X(33)    VALUE SPACES.
00090             05  DEPT-ID-OUT   PIC X(4).
00091             05  FILLER        PIC X(16)    VALUE SPACES.
00092             05  DEPT-NAME-OUT PIC X(45).
00093             05  FILLER        PIC X(34)    VALUE SPACES.
00094         01  EMP-HEADER.
00095             05  FILLER        PIC X(5)     VALUE SPACES.
00096             05  FILLER        PIC X(6)     VALUE 'EMP ID'.
00097             05  FILLER        PIC X(2)     VALUE SPACES.
00098             05  FILLER        PIC X(9)     VALUE 'LAST NAME'.
00099             05  FILLER        PIC X(8)     VALUE SPACES.
00100             05  FILLER        PIC X(10)    VALUE 'FIRST NAME'.
00101             05  FILLER        PIC X(3)     VALUE SPACES.
00102             05  FILLER        PIC X(10)    VALUE 'START DATE'.
00103             05  FILLER        PIC X(2)     VALUE SPACES.
00104             05  FILLER        PIC X(9)     VALUE 'JOB TITLE'.
00105             05  FILLER        PIC X(13)    VALUE SPACES.
00106             05  FILLER        PIC X(14)    VALUE 'OFFICE ADDRESS'.
00107             05  FILLER        PIC X(42)    VALUE SPACES.
00108         01  EMP-DETAIL-LINE.
00109             05  FILLER        PIC X(5)     VALUE SPACES.
00110             05  ID-OUT        PIC X(4).
00111             05  FILLER        PIC X(4)     VALUE SPACES.
00112             05  LAST-OUT      PIC X(15).
00113             05  FILLER        PIC X(2)     VALUE SPACES.
00114             05  FIRST-OUT     PIC X(10).
00115             05  FILLER        PIC X(3)     VALUE SPACES.
00116             05  SD-OUT.
00117                 10 SD-MM      PIC XX.
00118                 10 FILLER     PIC X        VALUE '/'.
00119                 10 SD-DD      PIC XX.
00120                 10 FILLER     PIC X        VALUE '/'.
00121                 10 SD-YY      PIC XX.
00122             05  FILLER        PIC X(4)     VALUE SPACES.
00123             05  TITLE-OUT     PIC X(20).
```

```
00124             05  FILLER        PIC X(2)     VALUE SPACES.
00125             05  OFF-ADDRESS-OUT.
00126                 10 STREET-OUT PIC X(20).
00127                 10 FILLER     PIC XX       VALUE SPACES.
00128                 10 CITY-OUT   PIC X(15).
00129                 10 FILLER     PIC XX       VALUE SPACES.
00130                 10 STATE-OUT  PIC XX.
00131                 10 FILLER     PIC XX       VALUE SPACES.
00132                 10 ZIP-OUT    PIC X(5).
00133             05  FILLER        PIC X(8)     VALUE SPACES.
00134         01  ERR-HEADER-1.
00135             05  FILLER        PIC X(40)    VALUE SPACES.
00136             05  FILLER        PIC X(12)    VALUE 'ERROR REPORT'.
00137             05  FILLER        PIC X(80)    VALUE SPACES.
00138         01  ERR-HEADER-2.
00139             05  FILLER        PIC X(10)    VALUE SPACES.
00140             05  FILLER        PIC X(4)     VALUE '*** '.
00141             05  FILLER        PIC X(51)    VALUE
00142                 'THIS REPORT LISTS EMPTY AND NONEXISTENT DEPARTMENTS'.
00143             05  FILLER        PIC X(4)     VALUE ' ***'.
00144             05  FILLER        PIC X(63)    VALUE SPACES.
00145         01  ERR-HEADER-3.
00146             05  FILLER        PIC X(20)    VALUE SPACES.
00147             05  FILLER        PIC X(7)     VALUE 'DEPT ID'.
00148             05  FILLER        PIC X(9)     VALUE SPACES.
00149             05  FILLER        PIC X(7)     VALUE 'MESSAGE'.
00150             05  FILLER        PIC X(89)    VALUE SPACES.
00151         01  ERR-DETAIL-LINE.
00152             05  FILLER        PIC X(20)    VALUE SPACES.
00153             05  ERR-ID-OUT    PIC X(4).
00154             05  FILLER        PIC X(12)    VALUE SPACES.
00155             05  ERR-MESS-OUT  PIC X(15).
00156             05  FILLER        PIC X(79)    VALUE SPACES.
00157        ************************************************************
00158         01  MESSAGES.
00159             05 NO-JOB-MESSAGE.
00160                10 FILLER          PIC X(20)  VALUE 'NO JOB ASSIGNED'.
00161             05 NO-OFFICE-MESSAGE.
00162                10 FILLER          PIC X(20)
00163                                   VALUE 'NO OFFICE ASSIGNED'.
00164             05 NO-DEPT-MESSAGE.
00165                10 FILLER          PIC X(15)  VALUE 'DOES NOT EXIST'.
00166             05 NO-EMP-MESSAGE.
00167                10 FILLER          PIC X(15)  VALUE 'IS EMPTY'.
00168             05 NO-INPUT-MESSAGE.
00169                10 FILLER          PIC XX     VALUE SPACES.
00170                10 FILLER          PIC X(11)  VALUE '========>> '.
00171                10 FILLER          PIC X(8)   VALUE 'NO INPUT'.
00172                10 FILLER          PIC X(11)  VALUE ' <<========'.
00173                10 FILLER          PIC X(100) VALUE SPACES.
00174
00175        *01  COPY IDMS SUBSCHEMA-CTRL.
00176         01  SUBSCHEMA-CTRL.
00177              03  PROGRAM-NAME           PIC X(8)
00178                                          VALUE  SPACES .
00179              03  ERROR-STATUS           PIC X(4)
00180                                          VALUE  '1400' .
00181                                     88  DB-STATUS-OK
00182                                          VALUE  '0000' .
00183                                     88  ANY-STATUS
00184                                          VALUE  '    ' THRU '9999'  .
00185                                     88  ANY-ERROR-STATUS
00186                                          VALUE  '0001' THRU '9999'  .
00187                                     88  DB-END-OF-SET
00188                                          VALUE  '0307' .
00189                                     88  DB-REC-NOT-FOUND
00190                                          VALUE  '0326' .
00191              03  DBKEY                  PIC S9(8) COMP SYNC.
00192              03  RECORD-NAME            PIC X(16)
00193                                          VALUE  SPACES .
00194              03  RRECORD-NAME           REDEFINES RECORD-NAME.
00195                05  SSC-NODN             PIC X(8).
00196                05  SSC-DBN              PIC X(8).
00197              03  AREA-NAME              PIC X(16)
00198                                          VALUE  SPACES .
00199              03  AREA-RNAME             REDEFINES AREA-NAME.
```

```
00200                05  SSC-DNO              PIC X(8).
00201                05  SSC-DNA              PIC X(8).
00202              03  ERROR-SET              PIC X(16)
00203                                          VALUE  SPACES .
00204              03  ERROR-RECORD           PIC X(16)
00205                                          VALUE  SPACES .
00206              03  ERROR-AREA             PIC X(16)
00207                                          VALUE  SPACES .
00208              03  IDBMSCOM-AREA          PIC X(100)
00209                                          VALUE  LOW-VALUE .
00210              03  IDBMSCOM               REDEFINES IDBMSCOM-AREA
00211                                          PIC X
00212                                          OCCURS 100.
00213              03  RIDBMSCOM              REDEFINES IDBMSCOM-AREA.
00214                05  DB-SUB-ADDR          PIC X(4).
00215                05  FILLER               PIC X(96).
00216              03  R1DBMSCOM              REDEFINES IDBMSCOM-AREA.
00217                05  PAGE-INFO.
00218                  07  PAGE-INFO-GROUP    PIC S9(4) COMP.
00219                  07  PAGE-INFO-DBK-FORMAT
00220                                          PIC 9(4) COMP.
00221                05  FILLER               PIC X(96).
00222              03  DIRECT-DBKEY           PIC S9(8) COMP SYNC.
00223              03  DIRECT-DBK             REDEFINES DIRECT-DBKEY
00224                                          PIC S9(8) COMP SYNC.
00225              03  DATABASE-STATUS.
00226                05  DBSTATMENT-CODE      PIC X(2).
00227                05  DBSTATUS-CODE        PIC X(5).
00228              03  FILLER                 PIC X.
00229              03  RECORD-OCCUR           PIC S9(8) COMP SYNC.
00230              03  DML-SEQUENCE           PIC S9(8) COMP SYNC.
00231
00232         *01  COPY IDMS SUBSCHEMA-SSNAME.
00233          01  SUBSCHEMA-SSNAME            PIC X(8)
00234                                          VALUE  'EMPSS01 ' .
00235
00236         *01  COPY IDMS SUBSCHEMA-RECNAMES.
00237          01  SUBSCHEMA-RECNAMES.
00238               03  SR460                  PIC X(16)
00239                                          VALUE  'STRUCTURE       ' .
00240               03  SR455                  PIC X(16)
00241                                          VALUE  'SKILL           ' .
00242               03  SR450                  PIC X(16)
00243                                          VALUE  'OFFICE          ' .
00244               03  SR445                  PIC X(16)
00245                                          VALUE  'NON-HOSP-CLAIM  ' .
00246               03  SR440                  PIC X(16)
00247                                          VALUE  'JOB             ' .
00248               03  SR435                  PIC X(16)
00249                                          VALUE  'INSURANCE-PLAN  ' .
00250               03  SR430                  PIC X(16)
00251                                          VALUE  'HOSPITAL-CLAIM  ' .
00252               03  SR425                  PIC X(16)
00253                                          VALUE  'EXPERTISE       ' .
00254               03  SR420                  PIC X(16)
00255                                          VALUE  'EMPOSITION      ' .
00256               03  SR415                  PIC X(16)
00257                                          VALUE  'EMPLOYEE        ' .
00258               03  SR410                  PIC X(16)
00259                                          VALUE  'DEPARTMENT      ' .
00260               03  SR405                  PIC X(16)
00261                                          VALUE  'DENTAL-CLAIM    ' .
00262               03  SR400                  PIC X(16)
00263                                          VALUE  'COVERAGE        ' .
00264
00265         *01  COPY IDMS SUBSCHEMA-SETNAMES.
00266          01  SUBSCHEMA-SETNAMES.
00267               03  COVERAGE-CLAIMS        PIC X(16)
00268                                          VALUE  'COVERAGE-CLAIMS ' .
00269               03  DEPT-EMPLOYEE          PIC X(16)
00270                                          VALUE  'DEPT-EMPLOYEE   ' .
00271               03  EMP-COVERAGE           PIC X(16)
00272                                          VALUE  'EMP-COVERAGE    ' .
00273               03  EMP-EXPERTISE          PIC X(16)
00274                                          VALUE  'EMP-EXPERTISE   ' .
00275               03  EMP-NAME-NDX           PIC X(16)
```

```
                                        VALUE  'EMP-NAME-NDX    ' .
             03  EMP-EMPOSITION         PIC X(16)
                                        VALUE  'EMP-EMPOSITION  ' .
             03  JOB-EMPOSITION         PIC X(16)
                                        VALUE  'JOB-EMPOSITION  ' .
             03  JOB-TITLE-NDX          PIC X(16)
                                        VALUE  'JOB-TITLE-NDX   ' .
             03  MANAGES                PIC X(16)
                                        VALUE  'MANAGES         ' .
             03  OFFICE-EMPLOYEE        PIC X(16)
                                        VALUE  'OFFICE-EMPLOYEE ' .
             03  REPORTS-TO             PIC X(16)
                                        VALUE  'REPORTS-TO      ' .
             03  SKILL-EXPERTISE        PIC X(16)
                                        VALUE  'SKILL-EXPERTISE ' .
             03  SKILL-NAME-NDX         PIC X(16)
                                        VALUE  'SKILL-NAME-NDX  ' .
             03  CALC                   PIC X(16)
                                        VALUE  'CALC            ' .

        *01  COPY IDMS RECORD EMPLOYEE.
         01  EMPLOYEE.
             02  EMP-ID-0415            PIC 9(4).
             02  EMP-NAME-0415.
              03  EMP-FIRST-NAME-0415    PIC X(10).
              03  EMP-LAST-NAME-0415     PIC X(15).
             02  EMP-ADDRESS-0415.
              03  EMP-STREET-0415        PIC X(20).
              03  EMP-CITY-0415          PIC X(15).
              03  EMP-STATE-0415         PIC X(2).
              03  EMP-ZIP-0415.
               04  EMP-ZIP-FIRST-FIVE-0415
                                        PIC X(5).
               04  EMP-ZIP-LAST-FOUR-0415
                                        PIC X(4).
             02  EMP-PHONE-0415         PIC 9(10).
             02  STATUS-0415            PIC X(2).
                                   88  ACTIVE-0415
                                        VALUE  '01' .
                                   88  ST-DISABIL-0415
                                        VALUE  '02' .
                                   88  LT-DISABIL-0415
                                        VALUE  '03' .
                                   88  LEAVE-OF-ABSENCE-0415
                                        VALUE  '04' .
                                   88  TERMINATED-0415
                                        VALUE  '05' .
             02  SS-NUMBER-0415         PIC 9(9).
             02  START-DATE-0415.
              03  START-YEAR-0415        PIC 9(4).
              03  START-MONTH-0415       PIC 9(2).
              03  START-DAY-0415         PIC 9(2).
             02  TERMINATION-DATE-0415.
              03  TERMINATION-YEAR-0415  PIC 9(4).
              03  TERMINATION-MONTH-0415 PIC 9(2).
              03  TERMINATION-DAY-0415   PIC 9(2).
             02  BIRTH-DATE-0415.
              03  BIRTH-YEAR-0415        PIC 9(4).
              03  BIRTH-MONTH-0415       PIC 9(2).
              03  BIRTH-DAY-0415         PIC 9(2).

        *01  COPY IDMS RECORD DEPARTMENT.
         01  DEPARTMENT.
             02  DEPT-ID-0410           PIC 9(4).
             02  DEPT-NAME-0410         PIC X(45).
             02  DEPT-HEAD-ID-0410      PIC 9(4).
             02  FILLER                 PIC XXX.

        *01  COPY IDMS RECORD JOB.
         01  JOB.
             02  JOB-ID-0440            PIC 9(4).
             02  TITLE-0440             PIC X(20).
             02  DESCRIPTION-0440.
              03  DESCRIPTION-LINE-0440  PIC X(60)
                                        OCCURS 2.
             02  REQUIREMENTS-0440.
```

```
00352              03  REQUIREMENT-LINE-0440  PIC X(60)
00353                                          OCCURS 2.
00354             02  MINIMUM-SALARY-0440     PIC S9(6)V99.
00355             02  MAXIMUM-SALARY-0440     PIC S9(6)V99.
00356             02  SALARY-GRADES-0440      PIC 9(2)
00357                                          OCCURS 4.
00358             02  NUMBER-OF-POSITIONS-0440
00359                                          PIC 9(3).
00360             02  NUMBER-OPEN-0440        PIC 9(3).
00361             02  FILLER                  PIC XX.
00362
00363         *01  COPY IDMS RECORD EMPOSITION.
00364          01  EMPOSITION.
00365             02  START-DATE-0420.
00366              03  START-YEAR-0420        PIC 9(4).
00367              03  START-MONTH-0420       PIC 9(2).
00368              03  START-DAY-0420         PIC 9(2).
00369             02  FINISH-DATE-0420.
00370              03  FINISH-YEAR-0420       PIC 9(4).
00371              03  FINISH-MONTH-0420      PIC 9(2).
00372              03  FINISH-DAY-0420        PIC 9(2).
00373             02  SALARY-GRADE-0420       PIC 9(2).
00374             02  SALARY-AMOUNT-0420      PIC S9(7)V99 COMP-3.
00375             02  BONUS-PERCENT-0420      PIC SV999 COMP-3.
00376             02  COMMISSION-PERCENT-0420 PIC SV999 COMP-3.
00377             02  OVERTIME-RATE-0420      PIC S9V99 COMP-3.
00378             02  FILLER                  PIC XXX.
00379
00380         *01  COPY IDMS RECORD OFFICE.
00381          01  OFFICE.
00382             02  OFFICE-CODE-0450        PIC X(3).
00383             02  OFFICE-ADDRESS-0450.
00384              03  OFFICE-STREET-0450     PIC X(20).
00385              03  OFFICE-CITY-0450       PIC X(15).
00386              03  OFFICE-STATE-0450      PIC X(2).
00387              03  OFFICE-ZIP-0450.
00388               04  OFFICE-ZIP-FIRST-FIVE-0450
00389                                          PIC X(5).
00390               04  OFFICE-ZIP-LAST-FOUR-0450
00391                                          PIC X(4).
00392             02  OFFICE-PHONE-0450       PIC 9(7)
00393                                          OCCURS 3.
00394             02  OFFICE-AREA-CODE-0450   PIC X(3).
00395             02  SPEED-DIAL-0450         PIC X(3).
00396             02  FILLER                  PIC X(4).
00398          PROCEDURE DIVISION.
00399
00400         *    ********************************************************
00401         *    *  PROCEDURE DIVISION GENERAL STRATEGY:                *
00402         *    *      1) READ DEPT-ID-IN, WHICH CONTAINS THE          *
00403         *    *         DEPT-ID NUMBER                               *
00404         *    *      2) ACCESS THE DATABASE USING THE DEPT-ID NUMBER  *
00405         *    *         WITH AN OBTAIN CALC ON THE DEPARTMENT RECORD  *
00406         *    *      3) ACCESS ALL EMPLOYEES IN THE DEPT-EMPLOYEE SET *
00407         *    *         AND RETRIEVE RELATED JOB AND OFFICE DATA      *
00408         *    *      4) PRINT A REPORT FOR EACH DEPARTMENT           *
00409         *    *      5) PRINT AN ERROR REPORT FOR EMPTY DEPARTMENTS   *
00410         *    *         AND NONEXISTENT DEPARTMENTS (NO MATCHING      *
00411         *    *         DEPT-ID)                                      *
00412         *    ********************************************************
00413
00414          MAIN-LINE.
00415             PERFORM INIT-FILES.
00416             IF END-OF-FILE
00417                PERFORM EMPTY-INPUT-PROCESSING
00418             ELSE
00419                PERFORM INIT-BIND-READY
00420                PERFORM U220-ERR-HEADER
00421                PERFORM DEPT-PROCESSING THRU DEPT-PROCESSING-EXIT
00422                            UNTIL END-OF-FILE.
00423             PERFORM END-PROCESSING.
00424             GOBACK.
00425
00426          INIT-BIND-READY.
00427         ***************************************************************
00428         *  THE BIND STATEMENTS ARE PERFORMED INDIVIDUALLY (RATHER     *
```

```
00429        * THAN BY USING A COPY IDMS SUBSCHEMA-BINDS) IN ORDER TO      *
00430        * CHECK EACH ERROR-STATUS BY PERFORMING THE IDMS-STATUS       *
00431        * ROUTINE.                                                    *
00432        ***************************************************************
00433             MOVE 'DEPTRPT' TO PROGRAM-NAME.
00434        *    BIND RUN-UNIT.                                            DMLC0001
00435                   MOVE 1 TO DML-SEQUENCE
00436                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00437                           IDBMSCOM (59)
00438                           SUBSCHEMA-CTRL
00439                           SUBSCHEMA-SSNAME.
00440             PERFORM IDMS-STATUS.
00441        *    BIND EMPLOYEE.                                            DMLC0002
00442                   MOVE 2 TO DML-SEQUENCE
00443                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00444                           IDBMSCOM (48)
00445                           SR415
00446                           EMPLOYEE.
00447             PERFORM IDMS-STATUS.
00448        *    BIND DEPARTMENT.                                          DMLC0003
00449                   MOVE 3 TO DML-SEQUENCE
00450                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00451                           IDBMSCOM (48)
00452                           SR410
00453                           DEPARTMENT.
00454             PERFORM IDMS-STATUS.
00455        *    BIND JOB.                                                 DMLC0004
00456                   MOVE 4 TO DML-SEQUENCE
00457                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00458                           IDBMSCOM (48)
00459                           SR440
00460                           JOB.
00461             PERFORM IDMS-STATUS.
00462        *    BIND EMPOSITION.                                          DMLC0005
00463                   MOVE 5 TO DML-SEQUENCE
00464                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00465                           IDBMSCOM (48)
00466                           SR420
00467                           EMPOSITION.
00468             PERFORM IDMS-STATUS.
00469        *    BIND OFFICE.                                              DMLC0006
00470                   MOVE 6 TO DML-SEQUENCE
00471                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00472                           IDBMSCOM (48)
00473                           SR450
00474                           OFFICE.
00475             PERFORM IDMS-STATUS.
00476        *    READY.                                                    DMLC0007
00477                   MOVE 7 TO DML-SEQUENCE
00478                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00479                           IDBMSCOM (37).
00480             PERFORM IDMS-STATUS.
00481
00482         INIT-FILES.
00483             OPEN INPUT DEPT-FILE-IN.
00484             OPEN OUTPUT DEPT-FILE-OUT.
00485             OPEN OUTPUT ERR-FILE-OUT.
00486             MOVE SPACES TO PRINT-LINE.
00487             MOVE SPACES TO ERR-LINE.
00488             READ DEPT-FILE-IN AT END MOVE 'Y' TO EOF-SW.
00489
00490         EMPTY-INPUT-PROCESSING.
00491             MOVE NO-INPUT-MESSAGE TO PRINT-LINE.
00492             MOVE '1' TO CC.
00493             PERFORM U000-WRITE-LINE.
00494
00495        ***************************************************************
00496        * THIS PARAGRAPH ACCESSES THE DATABASE USING THE DEPT-ID-0415 *
00497        * CALCKEY VALUE.                                              *
00498        ***************************************************************
00499         DEPT-PROCESSING.
00500             MOVE DEPT-ID-IN TO DEPT-ID-0410.
00501        *    OBTAIN CALC DEPARTMENT.                                   DMLC0008
00502                   MOVE 8 TO DML-SEQUENCE
00503                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00504                           IDBMSCOM (32)
```

```
00505                           SR410
00506                           IDBMSCOM (43).
00507             IF DB-REC-NOT-FOUND THEN
00508                 PERFORM NO-DEPT-PROCESSING
00509             ELSE
00510                 PERFORM IDMS-STATUS
00511        *        IF DEPT-EMPLOYEE IS NOT EMPTY                          DMLC0009
00512                   MOVE 9 TO DML-SEQUENCE
00513                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00514                           IDBMSCOM (65)
00515                           DEPT-EMPLOYEE;
00516                   IF ERROR-STATUS EQUAL TO '1601'
00517                                                THEN
00518                     PERFORM U020-VALID-HEADER
00519                     MOVE DEPT-ID-0410  TO DEPT-ID-OUT
00520                     MOVE DEPT-NAME-0410  TO DEPT-NAME-OUT
00521                     MOVE DEPT-DETAIL-LINE TO PRINT-LINE
00522                     PERFORM U000-WRITE-LINE
00523                     PERFORM U030-EMP-HEADERS
00524                     PERFORM SET-WALK THRU SET-WALK-EXIT
00525                                 UNTIL DB-END-OF-SET
00526                 ELSE
00527                      PERFORM EMPTY-SET.
00528             READ DEPT-FILE-IN AT END MOVE 'Y' TO EOF-SW.
00529         DEPT-PROCESSING-EXIT.
00530             EXIT.
00531
00532        ***************************************************************
00533        * THIS PARAGRAPH RETRIEVES EMPLOYEE, JOB, AND OFFICE DATA     *
00534        * FOR EACH EMPLOYEE IN THE DEPT-EMPLOYEE SET.                 *
00535        ***************************************************************
00536         SET-WALK.
00537        *    OBTAIN NEXT EMPLOYEE WITHIN DEPT-EMPLOYEE.                DMLC0010
00538                   MOVE 10 TO DML-SEQUENCE
00539                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00540                           IDBMSCOM (10)
00541                           SR415
00542                           DEPT-EMPLOYEE
00543                           IDBMSCOM (43).
00544             IF DB-END-OF-SET
00545                GO TO SET-WALK-EXIT
00546             ELSE
00547                PERFORM IDMS-STATUS.
00548             MOVE EMP-ID-0415 TO ID-OUT.
00549             MOVE EMP-LAST-NAME-0415 TO LAST-OUT.
00550             MOVE EMP-FIRST-NAME-0415 TO FIRST-OUT.
00551             MOVE START-YEAR-0415 TO SD-YY.
00552             MOVE START-MONTH-0415 TO SD-MM.
00553             MOVE START-DAY-0415 TO SD-DD.
00554        *    IF EMP-EMPOSITION IS EMPTY                                DMLC0011
00555                   MOVE 11 TO DML-SEQUENCE
00556                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00557                           IDBMSCOM (64)
00558                           EMP-EMPOSITION;
00559                   IF ERROR-STATUS EQUAL TO '0000'
00560                 MOVE NO-JOB-MESSAGE TO TITLE-OUT
00561             ELSE
00562        *        FIND FIRST WITHIN EMP-EMPOSITION                      DMLC0012
00563                   MOVE 12 TO DML-SEQUENCE
00564                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00565                           IDBMSCOM (20)
00566                           EMP-EMPOSITION;
00567                 PERFORM IDMS-STATUS
00568        *        IF NOT JOB-EMPOSITION MEMBER                          DMLC0013
00569                   MOVE 13 TO DML-SEQUENCE
00570                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00571                           IDBMSCOM (62)
00572                           JOB-EMPOSITION;
00573                   IF ERROR-STATUS EQUAL TO '1601'
00574                     MOVE NO-JOB-MESSAGE TO TITLE-OUT
00575                 ELSE
00576        *            OBTAIN OWNER WITHIN JOB-EMPOSITION                DMLC0014
00577                   MOVE 14 TO DML-SEQUENCE
00578                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00579                           IDBMSCOM (31)
00580                           JOB-EMPOSITION
```

```
00581                           IDBMSCOM (43);
00582                     PERFORM IDMS-STATUS
00583                     MOVE TITLE-0440 TO TITLE-OUT.
00584         *    IF OFFICE-EMPLOYEE IS EMPTY                                DMLC0015
00585                    MOVE 15 TO DML-SEQUENCE
00586                    CALL 'IDMS' USING SUBSCHEMA-CTRL
00587                           IDBMSCOM (64)
00588                           OFFICE-EMPLOYEE;
00589                    IF ERROR-STATUS EQUAL TO '0000'
00590                 MOVE NO-OFFICE-MESSAGE TO STREET-OUT
00591                 MOVE SPACES TO CITY-OUT
00592                 MOVE SPACES TO STATE-OUT
00593                 MOVE SPACES TO ZIP-OUT
00594             ELSE
00595         *       OBTAIN OWNER WITHIN OFFICE-EMPLOYEE                     DMLC0016
00596                    MOVE 16 TO DML-SEQUENCE
00597                    CALL 'IDMS' USING SUBSCHEMA-CTRL
00598                           IDBMSCOM (31)
00599                           OFFICE-EMPLOYEE
00600                           IDBMSCOM (43);
00601                 PERFORM IDMS-STATUS
00602                 MOVE OFFICE-STREET-0450 TO STREET-OUT
00603                 MOVE OFFICE-CITY-0450 TO CITY-OUT
00604                 MOVE OFFICE-STATE-0450 TO STATE-OUT
00605                 MOVE OFFICE-ZIP-FIRST-FIVE-0450 TO ZIP-OUT
00606                 MOVE EMP-DETAIL-LINE TO PRINT-LINE.
00607             PERFORM U000-WRITE-LINE.
00608         SET-WALK-EXIT.
00609             EXIT.
00610
00611         END-PROCESSING.
00612         *    FINISH.                                                    DMLC0017
00613                    MOVE 17 TO DML-SEQUENCE
00614                    CALL 'IDMS' USING SUBSCHEMA-CTRL
00615                           IDBMSCOM (2).
00616             PERFORM IDMS-STATUS.
00617             CLOSE DEPT-FILE-OUT.
00618             CLOSE ERR-FILE-OUT.
00619             CLOSE DEPT-FILE-IN.
00620
00621         EMPTY-SET.
00622             MOVE SPACES TO ERR-LINE.
00623             MOVE DEPT-ID-0410 TO ERR-ID-OUT.
00624             MOVE NO-EMP-MESSAGE TO ERR-MESS-OUT.
00625             MOVE ERR-DETAIL-LINE TO ERR-LINE.
00626             PERFORM U200-WRITE-ERR-LINE.
00627
00628         NO-DEPT-PROCESSING.
00629             MOVE DEPT-ID-IN TO ERR-ID-OUT.
00630             MOVE NO-DEPT-MESSAGE TO ERR-MESS-OUT.
00631             MOVE ERR-DETAIL-LINE TO ERR-LINE.
00632             PERFORM U200-WRITE-ERR-LINE.
00633
00634         U000-WRITE-LINE.
00635             WRITE DEPT-REC-OUT AFTER POSITIONING CC.
00636             IF CC = '1' THEN MOVE 0 TO LINE-COUNT
00637               ELSE IF CC = ' ' THEN ADD 1 TO LINE-COUNT
00638                    ELSE IF CC = '0' THEN ADD 2 TO LINE-COUNT.
00639             IF LINE-COUNT > LINE-MAX
00640                           THEN PERFORM U010-NEW-PAGE-ROUTINE.
00641         U010-NEW-PAGE-ROUTINE.
00642             PERFORM U020-VALID-HEADER.
00643             MOVE DEPT-DETAIL-LINE TO PRINT-LINE.
00644             PERFORM U000-WRITE-LINE.
00645             PERFORM U030-EMP-HEADERS.
00646         U020-VALID-HEADER.
00647             MOVE DEPT-HEADER TO PRINT-LINE.
00648             MOVE '1' TO CC.
00649             PERFORM U000-WRITE-LINE
00650             MOVE ' ' TO CC.
00651         U030-EMP-HEADERS.
00652             MOVE '0' TO CC.
00653             MOVE EMP-HEADER TO PRINT-LINE.
00654             PERFORM U000-WRITE-LINE.
00655             MOVE SPACES TO PRINT-LINE.
00656             MOVE ' ' TO CC.
```

```
00657             PERFORM U000-WRITE-LINE.
00658
00659         U200-WRITE-ERR-LINE.
00660             WRITE ERR-REC-OUT AFTER POSITIONING ERR-CC.
00661             IF ERR-CC = '1' THEN MOVE 0 TO ERR-LINE-COUNT
00662               ELSE IF ERR-CC = ' ' THEN ADD 1 TO ERR-LINE-COUNT
00663                   ELSE IF ERR-CC = '0' THEN ADD 2 TO ERR-LINE-COUNT.
00664             IF ERR-LINE-COUNT > LINE-MAX THEN
00665                             PERFORM U220-ERR-HEADER.
00666         U220-ERR-HEADER.
00667             MOVE ERR-HEADER-1 TO ERR-LINE.
00668             MOVE '1' TO ERR-CC.
00669             PERFORM U200-WRITE-ERR-LINE
00670             MOVE '0' TO ERR-CC.
00671             MOVE ERR-HEADER-2 TO ERR-LINE.
00672             PERFORM U200-WRITE-ERR-LINE.
00673             MOVE ERR-HEADER-3 TO ERR-LINE.
00674             PERFORM U200-WRITE-ERR-LINE.
00675             MOVE SPACES TO ERR-LINE.
00676             MOVE ' ' TO ERR-CC.
00677             PERFORM U200-WRITE-ERR-LINE.
00678         IDMS-ABORT.
00679             EXIT.
00680         IDMS-ABORT-EXIT.
00681        *    COPY IDMS IDMS-STATUS.
00682        ****************************************************************
00683         IDMS-STATUS                                            SECTION.
00684        ****************************************************************
00685         IDMS-STATUS-PARAGRAPH.
00686                 IF DB-STATUS-OK GO TO ISABEX.
00687                 PERFORM IDMS-ABORT.
00688                 DISPLAY '**************************'
00689                         ' ABORTING - ' PROGRAM-NAME
00690                         ', '           ERROR-STATUS
00691                         ', '           ERROR-RECORD
00692                         ' **** RECOVER IDMS ****'
00693                         UPON CONSOLE.
00694                 DISPLAY 'PROGRAM NAME ------ ' PROGRAM-NAME.
00695                 DISPLAY 'ERROR STATUS ------ ' ERROR-STATUS.
00696                 DISPLAY 'ERROR RECORD ------ ' ERROR-RECORD.
00697                 DISPLAY 'ERROR SET --------- ' ERROR-SET.
00698                 DISPLAY 'ERROR AREA -------- ' ERROR-AREA.
00699                 DISPLAY 'LAST GOOD RECORD -- ' RECORD-NAME.
00700                 DISPLAY 'LAST GOOD AREA ---- ' AREA-NAME.
00701                 DISPLAY 'DML SEQUENCE ------ ' DML-SEQUENCE.
00702        *        ROLLBACK.                                          DMLC0018
00703                   MOVE 18 TO DML-SEQUENCE
00704                   CALL 'IDMS' USING SUBSCHEMA-CTRL
00705                            IDBMSCOM (67).
00706                 CALL 'ABORT'.
00707         ISABEX. EXIT.
```

Appendix C. Sample Online Program

C.1 About this appendix C-3
C.2 Application components C-4
C.3 Application runtime requirements C-6
C.3.1 Sample online COBOL program as input to the DML precompiler . . C-6
C.3.2 Sample online COBOL program as output from the DML precompiler C-11
C.3.3 Sample online COBOL program from the COBOL compiler C-25

C.1 About this appendix

This appendix contains a sample CA-IDMS online application that illustrates the structure of CA-IDMS programs that accept data from a terminal operator and retrieve information from the database. The application program highlights the following CA-IDMS features:

- Mapping mode input and output
- Automatic editing and error handling
- Pseudo-conversational transactions
- LRF DML statements

The application's components, runtime requirements, and DML code are described below.

C.2 Application components

The application comprises a program, two tasks, a map, and a subschema:

- **Program**— The EMPDISP program either performs a MAP OUT to start a session or performs a MAP IN, error checking, database access, and a MAP OUT.
- **Tasks** — The task codes TSK01 and TSK02 affect the program flow of control:
  - **TSK01** causes the program to perform the INITIAL-MAPOUT portion of the program, mapping out the empty screen with an initial input message.
  - **TSK02** causes the program to perform the GET-EMP portion of the program, mapping in the data, checking the AID byte, performing the error checking and database access portion of the program, and mapping out either an error message or employee data.
- **Map** — The application uses a map named EMPMAPLR to communicate with the terminal operator. The EMPMAPLR map is illustrated below. Its map definition specifies:
  - Eight literal fields including the title *** EMPLOYEE INFORMATION SCREEN ***.
  - Ten variable data fields, to contain: employee ID, last name, first name, job title, start date, department name, and office address (street, city, state, and zip code). All data is contained in the EMP-JOB-LR logical record.
  - Automatic editing for the employee ID field specifies that the field is in error if the ID entered by the terminal operator does not comply with the field's external picture (PIC 9(4)).
  - Messages are output in the $MESSAGE field.
- **Subschema** — The application uses the EMPSS09 subschema, which specifies a usage mode of LR. The program uses LRF DML statements to retrieve the EMP-JOB-LR logical record.

```
                 *** EMPLOYEE INFORMATION SCREEN ***

               EMPLOYEE ID:
               LAST NAME :
               FIRST NAME:

     JOB TITLE:                                   START DATE:

          DEPARTMENT NAME:
                   OFFICE:
                        :
                        :




    ENTER AN EMP ID AND PRESS ENTER ** CLEAR TO EXIT
```

C.3 Application runtime requirements

The following requirements must be met to execute the sample online application under CA-IDMS:

- Define and generate the EMPMAPLR map.
- Compile and link edit the EMPDISP program into a load library that is identified to CA-IDMS.
- Define the EMPDISP program to the CA-IDMS system either by submitting PROGRAM statements to the system generation compiler or by using the DCMT VARY DYNAMIC PROGRAM command at runtime.
- Define the EMPMAPLR map and the EMPSS09 subschema to the CA-IDMS system by submitting PROGRAM statements to the system generation compiler. Maps and subschemas are defined automatically at system startup if null program definition elements (PDEs) have been allocated for them at system generation.

C.3.1 Sample online COBOL program as input to the DML precompiler

```
*NO-ACTIVITY-LOG
*DMLIST
 IDENTIFICATION DIVISION.

 PROGRAM-ID.                    EMPDISP.
 AUTHOR.                        COMPUTER ASSOCIATES INTERNATIONAL.

 DATE-WRITTEN.                  APRIL 1995.

 REMARKS.                       THIS PROGRAM DEMONSTRATES
                                CA-IDMS PROGRAMMING USING
                                THE LOGICAL RECORD FACILITY.

****************************************************************
 ENVIRONMENT DIVISION.
****************************************************************
 IDMS-CONTROL SECTION.

 PROTOCOL.                     MODE IS IDMS-DC DEBUG
                                     IDMS-RECORDS MANUAL.
     SKIP3
 DATA DIVISION.

 SCHEMA SECTION.

     DB EMPSS09 WITHIN EMPSCHM.

 MAP SECTION.
 MAX FIELD LIST IS 5.
 MAP EMPMAPLR VERSION 1 TYPE IS STANDARD.
```

```
WORKING-STORAGE SECTION.
01  TASK-CODE          PIC X(8).
01  TSK01              PIC X(8)     VALUE 'TSK01'.
01  TSK02              PIC X(8)     VALUE 'TSK02'.

01  MESSAGES.
    05 INITIAL-MESSAGE         PIC X(80) VALUE
       'ENTER AN EMP ID AND PRESS ENTER ** CLEAR TO EXIT'.
    05 EDIT-ERROR-MESSAGE      PIC X(80) VALUE
       'EMP-ID EITHER NOT ENTERED OR NOT NUMERIC'.
    05 EMP-NOT-FOUND-MESSAGE   PIC X(80) VALUE
       'SPECIFIED EMPLOYEE COULD NOT BE FOUND'.
    05 DISPLAY-MESSAGE         PIC X(80) VALUE
       'CLEAR TO EXIT ** NEW EMP-ID AND ENTER TO CONTINUE'.

01  COPY IDMS DC-AID-CONDITION-NAMES.

01  COPY IDMS EMP-DATE-WORK-REC.

01  COPY IDMS SUBSCHEMA-LR-CONTROL.

01  COPY IDMS SUBSCHEMA-LR-RECORDS.
    03 SUBSCHEMA-LR-CTRL-END          PIC X.

01  COPY IDMS MAP-CONTROLS.

    EJECT
PROCEDURE DIVISION.

*    ********************************************************
*    *  PROCEDURE DIVISION GENERAL STRATEGY:                *
*    *      RETRIEVE INFORMATION FOR A SPECIFIED EMPLOYEE.  *
*    *      DISPLAYED DATA INCLUDES EMPLOYEE, DEPARTMENT,   *
*    *                    JOB, AND OFFICE INFORMATION.      *
*    * ==> THIS PROGRAM USES THE EMP-JOB-LR LOGICAL RECORD<= *
*    *  PROGRAM STRATEGY:                                   *
*    *         ** CHECK FOR TASK CODE: TSK01= INITIAL MAPOUT *
*    *                        ANYTHING ELSE = RETRIEVE LR   *
*    *         ** CLEAR TO EXIT APPLICATION                 *
*    *         ** ENTER AND NEW EMP-ID TO CONTINUE          *
*    ********************************************************
```

```
 MAIN-LINE.
*****************************************************************
*  THE BIND MAP STATEMENTS ADVISE IDMS-DC OF THE LOCATION OF  *
* THE MRB AND THE MAP RECORDS.                                *
*****************************************************************
     BIND MAP EMPMAPLR.
     BIND MAP EMPMAPLR RECORD EMPLOYEE.
     BIND MAP EMPMAPLR RECORD DEPARTMENT.
     BIND MAP EMPMAPLR RECORD JOB.
     BIND MAP EMPMAPLR RECORD OFFICE.
     BIND MAP EMPMAPLR RECORD EMP-DATE-WORK-REC.
*    ACCEPT TASK CODE INTO TASK-CODE.
     IF TASK-CODE = TSK01
          GO TO INITIAL-MAPOUT
     ELSE
          GO TO GET-EMP.
*****************************************************************
*****************************************************************
*  THE INITIAL-MAPOUT PARAGRAPH IS PERFORMED IF THE CALLING   *
* TASK CODE IS TSK01.                                         *
*****************************************************************
*  THE MODIFY MAP STATEMENT ASSIGNS THE PROTECTED             *
* ATTRIBUTE TO ALL MAP FIELDS EXCEPT EMP-ID-0415.             *
*****************************************************************
*  THE MAP OUT STATEMENT TRANSMITS THE EMPMAPLR MAP           *
* TO THE TERMINAL.                                            *
*****************************************************************
*  THE DC RETURN STATEMENT SPECIFIES THAT THE NEXT            *
* TASK THAT WILL BE INITIATED ON THE SAME TERMINAL WHEN THE   *
* OPERATOR PRESSES A CONTROL KEY WILL BE TSK02.               *
*****************************************************************
 INITIAL-MAPOUT.
```

```
     MODIFY MAP EMPMAPLR TEMPORARY
        FOR ALL EXCEPT EMP-ID-0415
          ATTRIBUTES PROTECTED.
*
     MOVE ZERO TO EMP-ID-0415.
     MAP OUT USING EMPMAPLR
        OUTPUT DATA IS YES NEWPAGE
        MESSAGE IS INITIAL-MESSAGE LENGTH 80.

     DC RETURN
        NEXT TASK CODE TSK02.
 INITIAL-MAPOUT-EXIT.
     EXIT.
*****************************************************************
*****************************************************************
*  THE GET-EMP PARAGRAPH IS PERFORMED IF THE CALLING TASK     *
* CODE IS NOT TSK01.                                          *
*****************************************************************
*  THE MAP IN STATEMENT TRANSMITS DATA FROM THE TERMINAL TO   *
* VARIABLE STORAGE DATA FIELDS.                               *
*****************************************************************
*  THIS FIRST INQUIRE MAP STATEMENT IS USED TO DETERMINE      *
* THE AID KEY PRESSED.                                        *
*****************************************************************
*  THIS SECOND INQUIRE MAP STATEMENT USES AUTOMATIC EDITING   *
* TO DETERMINE IF THE DATA ENTERED IS CONSISTENT WITH         *
* THE EXTERNAL PICTURE OF THE NAMED DATA ELEMENT.             *
*****************************************************************
*  THE MAP OUT STATEMENT TRANSMITS DATA FROM THE              *
* EMP-JOB-LR LOGICAL RECORD IN VARIABLE STORAGE TO MAP        *
* FIELDS.                                                     *
*****************************************************************
 GET-EMP.
```

```
     MAP IN USING EMPMAPLR.
*
     INQUIRE MAP EMPMAPLR
        MOVE AID TO DC-AID-IND-V.
     IF CLEAR-HIT
        DC RETURN.

*
     INQUIRE MAP EMPMAPLR
        IF DFLD EMP-ID-0415 EDIT IS ERROR
          THEN GO TO EDIT-ERROR.
*
     COPY IDMS SUBSCHEMA-BINDS.
     READY USAGE-MODE IS RETRIEVAL.
*****************************************************************
*  SINCE THE MAP FIELD IS ASSOCIATED WITH THE EMP-ID-0415     *
* FIELD, THE PROGRAM USES THE "OF LR" RETRIEVAL.  NOTE THAT   *
* AUTOSTATUS IMPLICITLY CHECKS FOR THE LR-ERROR PATH STATUS.  *
*****************************************************************
     OBTAIN EMP-JOB-LR
         WHERE EMP-ID-0415 = EMP-ID-0415 OF LR
         ON LR-NOT-FOUND
             GO TO NOT-FOUND.
     FINISH.
*****************************************************************
*  REFORMAT DATE TO MMDDYY; OUTPUT AS MM/DD/YY USING THE OLM  *
*  EXTERNAL PICTURE SPECIFICATION (XX/XX/XX).                 *
*****************************************************************
     MOVE START-YEAR-0415 TO WORK-YY.
     MOVE START-MONTH-0415 TO WORK-MM.
     MOVE START-DAY-0415 TO WORK-DD.

     MAP OUT USING EMPMAPLR
        OUTPUT DATA IS YES
        MESSAGE IS DISPLAY-MESSAGE LENGTH 80.
*
     DC RETURN NEXT TASK CODE TSK02.
 GET-EMP-EXIT.
     EXIT.
```

```
*****************************************************************
*****************************************************************
*  THE MODIFY MAP STATEMENT SPECIFIES THAT ALL MAP              *
* FIELDS EXCEPT THE INCORRECT EMP-ID-0415 FIELD WILL BE         *
* ERASED ON THE NEXT MAP OUT.                                   *
*****************************************************************
 EDIT-ERROR.
     MODIFY MAP EMPMAPLR TEMPORARY
        FOR ALL EXCEPT DFLD EMP-ID-0415
          OUTPUT DATA IS ERASE.
*
     MAP OUT USING EMPMAPLR
        MESSAGE IS EDIT-ERROR-MESSAGE LENGTH 80.
*
     DC RETURN
        NEXT TASK CODE TSK02.
 EDIT-ERROR-EXIT.
     EXIT.
*****************************************************************
*****************************************************************
*  THE FOLLOWING MODIFY MAP STATEMENT SPECIFIES THAT ALL        *
* MAP FIELDS EXCEPT THE EMP-ID-0415 FIELD WILL BE ERASED        *
* ON THE NEXT MAP OUT.                                          *
*****************************************************************
 NOT-FOUND.
     MODIFY MAP EMPMAPLR TEMPORARY
        FOR ALL EXCEPT DFLD EMP-ID-0415
          OUTPUT DATA IS ERASE.
*
     MAP OUT USING EMPMAPLR
        MESSAGE IS EMP-NOT-FOUND-MESSAGE LENGTH 80.
*
     DC RETURN
        NEXT TASK CODE TSK02.
 NOT-FOUND-EXIT.
     EXIT.

*****************************************************************
 IDMS-ABORT.
     MOVE ERROR-STATUS TO SSC-ERRSTAT-SAVE.
     MOVE DML-SEQUENCE TO SSC-DMLSEQ-SAVE.
     SNAP FROM SUBSCHEMA-LR-CTRL TO SUBSCHEMA-LR-CTRL-END
          ON ANY-STATUS NEXT SENTENCE.
     MOVE SSC-ERRSTAT-SAVE TO ERROR-STATUS.
     MOVE SSC-DMLSEQ-SAVE TO DML-SEQUENCE.
 IDMS-ABORT-EXIT.
     EXIT.
     COPY IDMS IDMS-STATUS.
```

C.3.2 Sample online COBOL program as output from the DML precompiler

```
        00002       *DMLIST
        00003
        00004        IDENTIFICATION DIVISION.
        00005
        00006        PROGRAM-ID.                     EMPDISP.
        00007
        00008        AUTHOR.                         COMPUTER ASSOCIATES INTERNATIONAL.
        00009
        00010        DATE-WRITTEN.                   APRIL 1995.
        00011
        00012        REMARKS.                        THIS PROGRAM DEMONSTRATES
        00013                                        CA-IDMS PROGRAMMING USING
        00014                                        THE LOGICAL RECORD FACILITY.
        00015
        00016       ****************************************************************
        00017        ENVIRONMENT DIVISION.
        00018       ****************************************************************
DMLC    00019        IDMS-CONTROL SECTION.
        00020
DMLC    00021        PROTOCOL.                      MODE IS IDMS-DC DEBUG
DMLC    00022                                             IDMS-RECORDS MANUAL.
        00023            SKIP3
        00024        DATA DIVISION.
        00025
DMLC    00026        SCHEMA SECTION.
        00027
DMLC    00028            DB EMPSS09 WITHIN EMPSCHM.
        00029
DMLC    00030        MAP SECTION.
DMLC    00031        MAX FIELD LIST IS 5.
DMLC    00032        MAP EMPMAPLR VERSION 1 TYPE IS STANDARD.
        00033
        00034
        00035
        00036        WORKING-STORAGE SECTION.
        00037        01  TASK-CODE          PIC X(8).
        00038        01  TSK01              PIC X(8)     VALUE 'TSK01'.
        00039        01  TSK02              PIC X(8)     VALUE 'TSK02'.
        00040
        00041        01  MESSAGES.
        00042            05 INITIAL-MESSAGE         PIC X(80) VALUE
        00043               'ENTER AN EMP ID AND PRESS ENTER ** CLEAR TO EXIT'.
        00044            05 EDIT-ERROR-MESSAGE      PIC X(80) VALUE
        00045               'EMP-ID EITHER NOT ENTERED OR NOT NUMERIC'.
        00046            05 EMP-NOT-FOUND-MESSAGE   PIC X(80) VALUE
        00047               'SPECIFIED EMPLOYEE COULD NOT BE FOUND'.
        00048            05 DISPLAY-MESSAGE         PIC X(80) VALUE
        00049               'CLEAR TO EXIT ** NEW EMP-ID AND ENTER TO CONTINUE'.
        00050
DMLC    00051        01  COPY IDMS DC-AID-CONDITION-NAMES.
        00052        01  DC-AID-CONDITION-NAMES.
        00053             03  DC-AID-IND-V           PIC X.
        00054                                  88  ENTER-HIT VALUE QUOTE.
        00055                                  88  CLEAR-HIT VALUE '_'.
        00056                                  88  PF01-HIT VALUE '1'.
        00057                                  88  PF02-HIT VALUE '2'.
        00058                                  88  PF03-HIT VALUE '3'.
        00059                                  88  PF04-HIT VALUE '4'.
        00060                                  88  PF05-HIT VALUE '5'.
        00061                                  88  PF06-HIT VALUE '6'.
        00062                                  88  PF07-HIT VALUE '7'.
        00063                                  88  PF08-HIT VALUE '8'.
        00064                                  88  PF09-HIT VALUE '9'.
        00065                                  88  PF10-HIT VALUE ':'.
        00066                                  88  PF11-HIT VALUE '#'.
        00067                                  88  PF12-HIT VALUE '@'.
```

```
        00068                                  88  PF13-HIT VALUE 'A'.
        00069                                  88  PF14-HIT VALUE 'B'.
        00070                                  88  PF15-HIT VALUE 'C'.
        00071                                  88  PF16-HIT VALUE 'D'.
        00072                                  88  PF17-HIT VALUE 'E'.
        00073                                  88  PF18-HIT VALUE 'F'.
        00074                                  88  PF19-HIT VALUE 'G'.
        00075                                  88  PF20-HIT VALUE 'H'.
        00076                                  88  PF21-HIT VALUE 'I'.
        00077                                  88  PF22-HIT VALUE '•'.
        00078                                  88  PF23-HIT VALUE '.'.
        00079                                  88  PF24-HIT VALUE '<'.
        00080                                  88  PA01-HIT VALUE '%'.
        00081                                  88  PA02-HIT VALUE '>'.
        00082                                  88  PA03-HIT VALUE ','.
        00083                                  88  PEN-ATTN-SPACE-NULL VALUE '='.
        00084                                  88  PEN-ATTN VALUE QUOTE.
        00085
DMLC    00086          01  COPY IDMS EMP-DATE-WORK-REC.
        00087          01  EMP-DATE-WORK-REC.
        00088              02  WORK-DATE.
        00089               03  WORK-MM                PIC 9(2).
        00090               03  WORK-DD                PIC 9(2).
        00091               03  WORK-YY                PIC 9(2).
        00092
DMLC    00093          01  COPY IDMS SUBSCHEMA-LR-CONTROL.
        00094          01  SUBSCHEMA-CTRL.
        00095               03  PROGRAM-NAME           PIC X(8) VALUE SPACES.
        00096               03  ERROR-STATUS           PIC X(4) VALUE '1400'.
        00097                                  88  DB-STATUS-OK VALUE '0000'.
        00098                                  88  ANY-STATUS
        00099                                      VALUE '0000' THRU '9999'.
        00100                                  88  ANY-ERROR-STATUS
        00101                                      VALUE '0001' THRU '9999'.


        00102                                  88  DB-END-OF-SET VALUE '0307'.
        00103                                  88  DB-REC-NOT-FOUND VALUE '0326'.
        00104                                  88  DC-DEADLOCK VALUE '3101'
        00105                                      '3201' '3401' '3901'.
        00106                                  88  DC-NO-STORAGE VALUE '3202'
        00107                                      '3402'.
        00108                                  88  DC-AREA-ID-UNK VALUE '4303'.
        00109                                  88  DC-QUEUE-ID-UNK VALUE '4404'.
        00110                                  88  DC-REC-NOT-FOUND VALUE '4305'
        00111                                      '4405'.
        00112                                  88  DC-RESOURCE-NOT-AVAIL
        00113                                      VALUE '3908'.
        00114                                  88  DC-RESOURCE-AVAIL
        00115                                      VALUE '3909'.
        00116                                  88  DC-NEW-STORAGE VALUE '3210'.
        00117                                  88  DC-MAX-TASKS VALUE '3711'.
        00118                                  88  DC-REC-REPLACED VALUE '4317'.
        00119                                  88  DC-TRUNCATED-DATA
        00120                                      VALUE '4319' '4419'
        00121                                      '4519' '4719'.
        00122                                  88  DC-ATTN-INT VALUE '4525'
        00123                                      '4625'.
        00124                                  88  DC-OPER-CANCEL VALUE '4743'.
        00125                                  88  DC-FIRST-PAGE-SENT
        00126                                      VALUE '4676'.
        00127                                  88  DC-SECOND-STARTPAGE
        00128                                      VALUE '4604'.
        00129                                  88  DC-DETAIL-NOT-FOUND
```

```
00130                                   VALUE '4664'.
00131             03  DBKEY            PIC S9(8)
00132                                   USAGE COMP.
00133             03  RECORD-NAME      PIC X(16) VALUE SPACES.
00134             03  RRECORD-NAME     REDEFINES RECORD-NAME.
00135               05  SSC-NODN       PIC X(8).
00136               05  SSC-DBN        PIC X(8).
00137             03  AREA-NAME        PIC X(16) VALUE SPACES.
00138             03  ERROR-SET        PIC X(16) VALUE SPACES.
00139             03  ERROR-RECORD     PIC X(16) VALUE SPACES.
00140             03  ERROR-AREA       PIC X(16) VALUE SPACES.
00141             03  IDBMSCOM-AREA    PIC X(100) VALUE LOW-VALUE.
00142             03  IDBMSCOM         REDEFINES IDBMSCOM-AREA
00143                                   PIC X
00144                                   OCCURS 100.
00145             03  RIDBMSCOM        REDEFINES IDBMSCOM-AREA.
00146               05  DB-SUB-ADDR    PIC X(4).
00147               05  FILLER         PIC X(0096).
00148             03  DIRECT-DBKEY     PIC S9(8)
00149                                   USAGE COMP.
00150             03  DIRECT-DBK       REDEFINES DIRECT-DBKEY
00151                                   PIC S9(8)
00152                                   USAGE COMP.
00153             03  DCBMSCOM-AREA    PIC X(100) VALUE LOW-VALUE.
00154             03  DCBMSCOM         REDEFINES DCBMSCOM-AREA
00155                                   PIC X
00156                                   OCCURS 100.
```

```
00157             03  R1DCBMSCOM       REDEFINES DCBMSCOM-AREA.
00158               05  R2DCBMSCOM     PIC S9(8)
00159                                   OCCURS 11
00160                                   USAGE COMP.
00161               05  DCSTR1         PIC X(16).
00162               05  R3DCBMSCOM     REDEFINES DCSTR1.
00163                 07  DCSTR2       PIC X(8).
00164                 07  R4DCBMSCOM   REDEFINES DCSTR2.
00165                   09  DCSTR4     PIC X(4).
00166                   09  DCSTR5     PIC X(4).
00167                 07  DCSTR3       PIC X(8).
00168               05  R5DCBMSCOM     REDEFINES DCSTR1.
00169                 07  DCPNUM1      PIC S9(15)
00170                                   USAGE COMP-3.
00171               05  DCNUM1         PIC S9(8)
00172                                   USAGE COMP.
00173               05  R6DCBMSCOM     REDEFINES DCNUM1.
00174                 07  DCPNUM2      PIC S9(7)
00175                                   USAGE COMP-3.
00176               05  DCNUM2         PIC S9(8)
00177                                   USAGE COMP.
00178               05  DCNUM3         PIC S9(8)
00179                                   USAGE COMP.
00180               05  DCFLG1         PIC S9(4)
00181                                   USAGE COMP.
00182               05  DCFLG2         PIC S9(4)
00183                                   USAGE COMP.
```

```
00184                05  DCFLG3               PIC S9(4)
00185                                          USAGE COMP.
00186                05  DCFLG4               PIC S9(4)
00187                                          USAGE COMP.
00188              03  SSC-ERRSTAT-SAVE       PIC X(4) VALUE SPACES.
00189              03  SSC-DMLSEQ-SAVE        PIC S9(8)
00190                                          USAGE COMP.
00191              03  DML-SEQUENCE           PIC S9(8)
00192                                          USAGE COMP.
00193              03  RECORD-OCCUR           PIC S9(8)
00194                                          USAGE COMP.
00195              03  SUBSCHEMA-CTRL-END     PIC X(4) VALUE SPACES.
00196          01  SUBSCHEMA-LR-CTRL.
00197              03  LRC-LRPXELNG           PIC S9(4)
00198                                          USAGE COMP.
00199              03  LRC-MAXVXP             PIC S9(4)
00200                                          USAGE COMP.
00201              03  LRIDENT                PIC X(4) VALUE 'LRC '.
00202              03  LRVERB                 PIC X(8).
00203              03  LRNAME                 PIC X(16).
00204              03  LR-STATUS              PIC X(16).
00205              03  FILLER                 PIC X(16).
00206              03  LRPXE                  PIC X
00207                                          OCCURS 0 TO 512
00208                                          DEPENDING ON LRC-LRPXELNG.
00209              03  PXE.
00210                05  PXENEXT              PIC S9(8)
00211                                          USAGE COMP.
```

```
00212                05  PXETABO              PIC S9(4)
00213                                          USAGE COMP.
00214                05  PXEDSPL              PIC S9(4)
00215                                          USAGE COMP.
00216                05  PXEDYN               PIC S9(4)
00217                                          USAGE COMP.
00218                05  PXEDLEN              PIC S9(4)
00219                                          USAGE COMP.
00220                05  PXENDEC              PIC X.
00221                05  PXEDTYP              PIC X.
00222                05  PXEOTYP              PIC X.
00223                05  PXEFLAG              PIC X.
00224                05  FILLER               PIC X(240).
00225              03  PXEDSP256              REDEFINES PXE
00226                                          PIC X(256).
00227              03  PXEDSP248              REDEFINES PXE
00228                                          PIC X(248).
00229              03  PXEDSP240              REDEFINES PXE
00230                                          PIC X(240).
00231              03  PXEDSP232              REDEFINES PXE
00232                                          PIC X(232).
00233              03  PXEDSP224              REDEFINES PXE
00234                                          PIC X(224).
00235              03  PXEDSP216              REDEFINES PXE
00236                                          PIC X(216).
00237              03  PXEDSP208              REDEFINES PXE
00238                                          PIC X(208).
```

```
00239          03  PXEDSP200          REDEFINES PXE
00240                                  PIC X(200).
00241          03  PXEDSP192          REDEFINES PXE
00242                                  PIC X(192).
00243          03  PXEDSP184          REDEFINES PXE
00244                                  PIC X(184).
00245          03  PXEDSP176          REDEFINES PXE
00246                                  PIC X(176).
00247          03  PXEDSP168          REDEFINES PXE
00248                                  PIC X(168).
00249          03  PXEDSP160          REDEFINES PXE
00250                                  PIC X(160).
00251          03  PXEDSP152          REDEFINES PXE
00252                                  PIC X(152).
00253          03  PXEDSP144          REDEFINES PXE
00254                                  PIC X(144).
00255          03  PXEDSP136          REDEFINES PXE
00256                                  PIC X(136).
00257          03  PXEDSP128          REDEFINES PXE
00258                                  PIC X(128).
00259          03  PXEDSP120          REDEFINES PXE
00260                                  PIC X(120).
00261          03  PXEDSP112          REDEFINES PXE
00262                                  PIC X(112).
00263          03  PXEDSP104          REDEFINES PXE
```

```
00264                                  PIC X(104).
00265          03  PXEDSP96           REDEFINES PXE
00266                                  PIC X(96).
00267          03  PXEDSP88           REDEFINES PXE
00268                                  PIC X(88).
00269          03  PXEDSP80           REDEFINES PXE
00270                                  PIC X(80).
00271          03  PXEDSP72           REDEFINES PXE
00272                                  PIC X(72).
00273          03  PXEDSP64           REDEFINES PXE
00274                                  PIC X(64).
00275          03  PXEDSP56           REDEFINES PXE
00276                                  PIC X(56).
00277          03  PXEDSP48           REDEFINES PXE
00278                                  PIC X(48).
00279          03  PXEDSP40           REDEFINES PXE
00280                                  PIC X(40).
00281          03  PXEDSP32           REDEFINES PXE
00282                                  PIC X(32).
00283          03  PXEDSP24           REDEFINES PXE
00284                                  PIC X(24).
00285          03  PXEDSP16           REDEFINES PXE
00286                                  PIC X(16).
00287          03  PXEDSP8            REDEFINES PXE
00288                                  PIC X(8).
00289          03  PXECOMP-1          REDEFINES PXE
```

```
00290                                   USAGE COMP-1.
00291             03  PXECOMP-2        REDEFINES PXE
00292                                   USAGE COMP-2.
00293             03  PXECOMP-30       REDEFINES PXE
00294                                   PIC S9(18)
00295                                   USAGE COMP-3.
00296             03  PXECOMP-31       REDEFINES PXE
00297                                   PIC S9(17)V9(1)
00298                                   USAGE COMP-3.
00299             03  PXECOMP-32       REDEFINES PXE
00300                                   PIC S9(16)V9(2)
00301                                   USAGE COMP-3.
00302             03  PXECOMP-33       REDEFINES PXE
00303                                   PIC S9(15)V9(3)
00304                                   USAGE COMP-3.
00305             03  PXECOMP-34       REDEFINES PXE
00306                                   PIC S9(14)V9(4)
00307                                   USAGE COMP-3.
00308             03  PXECOMP-35       REDEFINES PXE
00309                                   PIC S9(13)V9(5)
00310                                   USAGE COMP-3.
00311             03  PXECOMP-36       REDEFINES PXE
00312                                   PIC S9(12)V9(6)
00313                                   USAGE COMP-3.
00314             03  PXECOMP-37       REDEFINES PXE
00315                                   PIC S9(11)V9(7)
00316                                   USAGE COMP-3.
00317             03  PXECOMP-38       REDEFINES PXE
00318                                   PIC S9(10)V9(8)
```

```
00319                                   USAGE COMP-3.
00320             03  PXECOMP-39       REDEFINES PXE
00321                                   PIC S9(9)V9(9)
00322                                   USAGE COMP-3.
00323             03  PXECOMP-310      REDEFINES PXE
00324                                   PIC S9(8)V9(10)
00325                                   USAGE COMP-3.
00326             03  PXECOMP-311      REDEFINES PXE
00327                                   PIC S9(7)V9(11)
00328                                   USAGE COMP-3.
00329             03  PXECOMP-312      REDEFINES PXE
00330                                   PIC S9(6)V9(12)
00331                                   USAGE COMP-3.
00332             03  PXECOMP-313      REDEFINES PXE
00333                                   PIC S9(5)V9(13)
00334                                   USAGE COMP-3.
00335             03  PXECOMP-314      REDEFINES PXE
00336                                   PIC S9(4)V9(14)
00337                                   USAGE COMP-3.
00338             03  PXECOMP-315      REDEFINES PXE
00339                                   PIC S9(3)V9(15)
00340                                   USAGE COMP-3.
00341             03  PXECOMP-316      REDEFINES PXE
00342                                   PIC S9(2)V9(16)
00343                                   USAGE COMP-3.
00344             03  PXECOMP-317      REDEFINES PXE
00345                                   PIC S9(1)V9(17)
00346                                   USAGE COMP-3.
```

```
00347          03  PXECOMP-318          REDEFINES PXE
00348                                    PIC SV9(18)
00349                                    USAGE COMP-3.
00350          03  PXECOMP20            REDEFINES PXE
00351                                    PIC S9(4)
00352                                    USAGE COMP.
00353          03  PXECOMP21            REDEFINES PXE
00354                                    PIC S9(3)V9(1)
00355                                    USAGE COMP.
00356          03  PXECOMP22            REDEFINES PXE
00357                                    PIC S9(2)V9(2)
00358                                    USAGE COMP.
00359          03  PXECOMP23            REDEFINES PXE
00360                                    PIC S9(1)V9(3)
00361                                    USAGE COMP.
00362          03  PXECOMP24            REDEFINES PXE
00363                                    PIC SV9(4)
00364                                    USAGE COMP.
00365          03  PXECOMP40            REDEFINES PXE
00366                                    PIC S9(9)
00367                                    USAGE COMP.
00368          03  PXECOMP41            REDEFINES PXE
00369                                    PIC S9(8)V9(1)
00370                                    USAGE COMP.
00371          03  PXECOMP42            REDEFINES PXE
00372                                    PIC S9(7)V9(2)
00373                                    USAGE COMP.
00374          03  PXECOMP43            REDEFINES PXE
```

```
00375                                    PIC S9(6)V9(3)
00376                                    USAGE COMP.
00377          03  PXECOMP44            REDEFINES PXE
00378                                    PIC S9(5)V9(4)
00379                                    USAGE COMP.
00380          03  PXECOMP45            REDEFINES PXE
00381                                    PIC S9(4)V9(5)
00382                                    USAGE COMP.
00383          03  PXECOMP46            REDEFINES PXE
00384                                    PIC S9(3)V9(6)
00385                                    USAGE COMP.
00386          03  PXECOMP47            REDEFINES PXE
00387                                    PIC S9(2)V9(7)
00388                                    USAGE COMP.
00389          03  PXECOMP48            REDEFINES PXE
00390                                    PIC S9(1)V9(8)
00391                                    USAGE COMP.
00392          03  PXECOMP49            REDEFINES PXE
00393                                    PIC SV9(9)
00394                                    USAGE COMP.
00395          03  PXECOMP80            REDEFINES PXE
00396                                    PIC S9(18)
00397                                    USAGE COMP.
00398          03  PXECOMP81            REDEFINES PXE
00399                                    PIC S9(17)V9(1)
00400                                    USAGE COMP.
00401          03  PXECOMP82            REDEFINES PXE
00402                                    PIC S9(16)V9(2)
00403                                    USAGE COMP.
00404          03  PXECOMP83            REDEFINES PXE
00405                                    PIC S9(15)V9(3)
00406                                    USAGE COMP.
00407          03  PXECOMP84            REDEFINES PXE
```

```
00408                                   PIC S9(14)V9(4)
00409                                   USAGE COMP.
00410             03  PXECOMP85        REDEFINES PXE
00411                                   PIC S9(13)V9(5)
00412                                   USAGE COMP.
00413             03  PXECOMP86        REDEFINES PXE
00414                                   PIC S9(12)V9(6)
00415                                   USAGE COMP.
00416             03  PXECOMP87        REDEFINES PXE
00417                                   PIC S9(11)V9(7)
00418                                   USAGE COMP.
00419             03  PXECOMP88        REDEFINES PXE
00420                                   PIC S9(10)V9(8)
00421                                   USAGE COMP.
00422             03  PXECOMP89        REDEFINES PXE
00423                                   PIC S9(9)V9(9)
00424                                   USAGE COMP.
00425             03  PXECOMP810       REDEFINES PXE
00426                                   PIC S9(8)V9(10)
00427                                   USAGE COMP.
00428             03  PXECOMP811       REDEFINES PXE
00429                                   PIC S9(7)V9(11)
00430                                   USAGE COMP.
00431             03  PXECOMP812       REDEFINES PXE
00432                                   PIC S9(6)V9(12)
00433                                   USAGE COMP.
00434             03  PXECOMP813       REDEFINES PXE
00435                                   PIC S9(5)V9(13)
00436                                   USAGE COMP.
00437             03  PXECOMP814       REDEFINES PXE
```

```
00438                                   PIC S9(4)V9(14)
00439                                   USAGE COMP.
00440             03  PXECOMP815       REDEFINES PXE
00441                                   PIC S9(3)V9(15)
00442                                   USAGE COMP.
00443             03  PXECOMP816       REDEFINES PXE
00444                                   PIC S9(2)V9(16)
00445                                   USAGE COMP.
00446             03  PXECOMP817       REDEFINES PXE
00447                                   PIC S9(1)V9(17)
00448                                   USAGE COMP.
00449             03  PXECOMP818       REDEFINES PXE
00450                                   PIC SV9(18)
00451                                   USAGE COMP.
00452         01  SUBSCHEMA-SSNAME     PIC X(8) VALUE 'EMPSS09 '.
00453         01  SUBSCHEMA-AREANAMES.
00454             03  EMP-DEMO-REGION  PIC X(16)
00455                                   VALUE 'EMP-DEMO-REGION '.
00456             03  INS-DEMO-REGION  PIC X(16)
00457                                   VALUE 'INS-DEMO-REGION '.
00458             03  ORG-DEMO-REGION  PIC X(16)
00459                                   VALUE 'ORG-DEMO-REGION '.
```

```
        00460
DMLC    00461        01  COPY IDMS SUBSCHEMA-LR-RECORDS.
        00462        01  EMP-JOB-LR.
        00463            02  EMPLOYEE.
        00464             03  EMP-ID-0415            PIC 9(4).
        00465             03  EMP-NAME-0415.
        00466              04  EMP-FIRST-NAME-0415   PIC X(10).
        00467              04  EMP-LAST-NAME-0415    PIC X(15).
        00468             03  STATUS-0415            PIC X(2).
        00469                                  88  ACTIVE-0415 VALUE '01'.
        00470                                  88  ST-DISABIL-0415 VALUE '02'.
        00471                                  88  LT-DISABIL-0415 VALUE '03'.
        00472                                  88  LEAVE-OF-ABSENCE-0415
        00473                                      VALUE '04'.
        00474                                  88  TERMINATED-0415 VALUE '05'.
        00475             03  SS-NUMBER-0415         PIC 9(9).
        00476             03  START-DATE-0415.
        00477              04  START-YEAR-0415       PIC 9(2).
        00478              04  START-MONTH-0415      PIC 9(2).
        00479              04  START-DAY-0415        PIC 9(2).
        00480             03  FILLER                 PIC X(2).
        00481            02  DEPARTMENT.
        00482             03  DEPT-ID-0410           PIC 9(4).
        00483             03  DEPT-NAME-0410         PIC X(45).
        00484             03  DEPT-HEAD-ID-0410      PIC 9(4).
        00485             03  FILLER                 PIC XXX.
        00486            02  JOB.
        00487             03  JOB-ID-0440            PIC 9(4).
        00488             03  TITLE-0440             PIC X(20).
        00489            02  OFFICE.



        00490             03  OFFICE-CODE-0450       PIC X(3).
        00491             03  OFFICE-ADDRESS-0450.
        00492              04  OFFICE-STREET-0450    PIC X(20).
        00493              04  OFFICE-CITY-0450      PIC X(15).
        00494              04  OFFICE-STATE-0450     PIC X(2).
        00495              04  OFFICE-ZIP-0450.
        00496               05  OFFICE-ZIP-FIRST-FIVE-0450
        00497                                      PIC X(5).
        00498               05  OFFICE-ZIP-LAST-FOUR-0450
        00499                                      PIC X(4).
        00500             03  OFFICE-PHONE-0450      PIC 9(7)
        00501                                      OCCURS 3.
        00502             03  OFFICE-AREA-CODE-0450  PIC X(3).
        00503             03  SPEED-DIAL-0450        PIC X(3).
        00504             03  FILLER                 PIC X(4).
        00505            03 SUBSCHEMA-LR-CTRL-END          PIC X.
        00506
DMLC    00507        01  COPY IDMS MAP-CONTROLS.
        00508        01  MRB-EMPMAPLR.
        00509             03  MRB-EMPMAPLR-ID        PIC X(8).
        00510             03  MRB-EMPMAPLR-MCOMP-VER.
        00511               05  MRB-EMPMAPLR-MCOMP-DATE
        00512                                      PIC X(8).
        00513               05  MRB-EMPMAPLR-MCOMP-TIME
        00514                                      PIC X(6).
        00515               05  MRB-EMPMAPLR-MCOMP-VERID
        00516                                      PIC X(2).
        00517             03  MRB-EMPMAPLR-SUBSCHEMA PIC X(8).
        00518             03  MRB-EMPMAPLR-FLGS      PIC X
        00519                                      OCCURS 4.
        00520             03  FILLER                 PIC X(6).
        00521             03  MRB-EMPMAPLR-NFLDS     PIC S9(4)
        00522                                      USAGE COMP.
        00523             03  MRB-EMPMAPLR-NRECS     PIC S9(4)
```

```
                                        USAGE COMP.
            03  MRB-EMPMAPLR-RECOF     PIC S9(4)
                                        USAGE COMP.
            03  MRB-EMPMAPLR-PERM-CURSOR
                                        PIC XX.
            03  MRB-EMPMAPLR-TEMP-CURSOR
                                        PIC XX.
            03  MRB-EMPMAPLR-PERM-WCC  PIC X.
            03  MRB-EMPMAPLR-TEMP-WCC  PIC X.
            03  MRB-EMPMAPLR-CURSOR    PIC XX.
            03  MRB-EMPMAPLR-AID       PIC X.
            03  MRB-EMPMAPLR-INPUT-FLGS
                                        PIC X.
            03  MRB-EMPMAPLR-SEGVIEW   PIC X.
            03  FILLER                 PIC X.
            03  MRB-EMPMAPLR-MREO      PIC S9(4)
                                        USAGE COMP.
            03  MRB-EMPMAPLR-ERR-CNT   PIC S9(4)
                                        USAGE COMP.
            03  MRB-EMPMAPLR-ATTR-FLGS PIC X
                                        OCCURS 4.
            03  MRB-EMPMAPLR-CURR-MFLD PIC S9(4)
                                        USAGE COMP.
            03  MRB-EMPMAPLR-XTYP      PIC X.
            03  MRB-EMPMAPLR-FILLER    PIC X.
            03  MRB-EMPMAPLR-MRE-XLEN  PIC S9(4)
                                        USAGE COMP.
```

```
            03  MRB-EMPMAPLR-MRB-XLEN  PIC S9(4)
                                        USAGE COMP.
            03  MRB-EMPMAPLR-MRE       OCCURS 11.
              05  MRB-EMPMAPLR-MRE-FLGS
                                        PIC X
                                        OCCURS 8.
              05  MRB-EMPMAPLR-MRE-INLEN
                                        PIC S9(4)
                                        USAGE COMP.
              05  MRB-EMPMAPLR-MRE-PAD-CHAR
                                        PIC X
                                        OCCURS 2.
              05  MRB-EMPMAPLR-MRE-FLG2
                                        PIC X
                                        OCCURS 2.
            03  MRB-EMPMAPLR-RECS      PIC S9(8)
                                        OCCURS 5
                                        USAGE COMP
                                        SYNC.
            03  MRB-EMPMAPLR-END       PIC X.
            03  MRB-EMPMAPLR-MRE-SUB   PIC S9(4)
                                        USAGE COMP.

        EJECT
    01  MRB-FLDLST.
        02  FLDLST                     PIC S9(8)
                                        OCCURS 6
                                        USAGE COMP.
```

```
           00579         PROCEDURE DIVISION.
           00580
           00581       *    ********************************************************
           00582       *    *  PROCEDURE DIVISION GENERAL STRATEGY:                 *
           00583       *    *      RETRIEVE INFORMATION FOR A SPECIFIED EMPLOYEE.   *
           00584       *    *      DISPLAYED DATA INCLUDES EMPLOYEE, DEPARTMENT,    *
           00585       *    *                     JOB, AND OFFICE INFORMATION.      *
           00586       *    * ==> THIS PROGRAM USES THE EMP-JOB-LR LOGICAL RECORD<= *
           00587       *    *  PROGRAM STRATEGY:                                    *
           00588       *    *         ** CHECK FOR TASK CODE: TSK01= INITIAL MAPOUT *
           00589       *    *                       ANYTHING ELSE = RETRIEVE LR    *
           00590       *    *         ** CLEAR TO EXIT APPLICATION                 *
           00591       *    *         ** ENTER AND NEW EMP-ID TO CONTINUE          *
           00592       *    ********************************************************
           00593
           00594         MAIN-LINE.
           00595       ****************************************************************
           00596       *  THE BIND MAP STATEMENTS ADVISE IDMS-DC OF THE LOCATION OF  *
           00597       * THE MRB AND THE MAP RECORDS.                                *
           00598       ****************************************************************
DMLC0001   00599            BIND MAP EMPMAPLR.
DMLC0002   00628            BIND MAP EMPMAPLR RECORD EMPLOYEE.
DMLC0003   00635            BIND MAP EMPMAPLR RECORD DEPARTMENT.
DMLC0004   00642            BIND MAP EMPMAPLR RECORD JOB.
DMLC0005   00649            BIND MAP EMPMAPLR RECORD OFFICE.
DMLC0006   00656            BIND MAP EMPMAPLR RECORD EMP-DATE-WORK-REC.
           00663       *
DMLC0007   00664            ACCEPT TASK CODE INTO TASK-CODE.
           00671            IF TASK-CODE = TSK01
           00672                 GO TO INITIAL-MAPOUT
           00673            ELSE
           00674                 GO TO GET-EMP.
```

```
           00675       ****************************************************************
           00676       ****************************************************************
           00677       *  THE INITIAL-MAPOUT PARAGRAPH IS PERFORMED IF THE CALLING   *
           00678       * TASK CODE IS TSK01.                                         *
           00679       ****************************************************************
           00680       *  THE MODIFY MAP STATEMENT ASSIGNS THE PROTECTED             *
           00681       * ATTRIBUTE TO ALL MAP FIELDS EXCEPT EMP-ID-0415.             *
           00682       ****************************************************************
           00683       *  THE MAP OUT STATEMENT TRANSMITS THE EMPMAPLR MAP           *
           00684       * TO THE TERMINAL.                                            *
           00685       ****************************************************************
           00686       *  THE DC RETURN STATEMENT SPECIFIES THAT THE NEXT            *
           00687       * TASK THAT WILL BE INITIATED ON THE SAME TERMINAL WHEN THE   *
           00688       * OPERATOR PRESSES A CONTROL KEY WILL BE TSK02.               *
           00689       ****************************************************************
           00690        INITIAL-MAPOUT.
DMLC0008   00691            MODIFY MAP EMPMAPLR TEMPORARY
DMLC0008   00692               FOR ALL EXCEPT EMP-ID-0415
DMLC0008   00693                 ATTRIBUTES PROTECTED.
           00707       *
           00708            MOVE ZERO TO EMP-ID-0415.
DMLC0009   00709            MAP OUT USING EMPMAPLR
DMLC0009   00710               OUTPUT DATA IS YES NEWPAGE
DMLC0009   00711               MESSAGE IS INITIAL-MESSAGE LENGTH 80.
           00722
DMLC0010   00723            DC RETURN
DMLC0010   00724               NEXT TASK CODE TSK02.
           00731        INITIAL-MAPOUT-EXIT.
           00732            EXIT.
```

```
           00733        ****************************************************************
           00734        ****************************************************************
           00735        *  THE GET-EMP PARAGRAPH IS PERFORMED IF THE CALLING TASK     *
           00736        * CODE IS NOT TSK01.                                           *
           00737        ****************************************************************
           00738        *  THE MAP IN STATEMENT TRANSMITS DATA FROM THE TERMINAL TO   *
           00739        * VARIABLE STORAGE DATA FIELDS.                                 *
           00740        ****************************************************************
           00741        *  THIS FIRST INQUIRE MAP STATEMENT IS USED TO DETERMINE      *
           00742        * THE AID KEY PRESSED.                                          *
           00743        ****************************************************************
           00744        *  THIS SECOND INQUIRE MAP STATEMENT USES AUTOMATIC EDITING   *
           00745        * TO DETERMINE IF THE DATA ENTERED IS CONSISTENT WITH         *
           00746        * THE EXTERNAL PICTURE OF THE NAMED DATA ELEMENT.              *
           00747        ****************************************************************
           00748        *  THE MAP OUT STATEMENT TRANSMITS DATA FROM THE              *
           00749        * EMP-JOB-LR LOGICAL RECORD IN VARIABLE STORAGE TO MAP        *
           00750        * FIELDS.                                                      *
           00751        ****************************************************************
           00752         GET-EMP.


DMLC0011   00753             MAP IN USING EMPMAPLR.
           00763        *
DMLC0012   00764             INQUIRE MAP EMPMAPLR
DMLC0012   00765                MOVE AID TO DC-AID-IND-V.
           00773             IF CLEAR-HIT
DMLC0013   00774                DC RETURN.
           00780
           00781        *
DMLC0014   00782             INQUIRE MAP EMPMAPLR
DMLC0014   00783                IF DFLD EMP-ID-0415 EDIT IS ERROR
           00795                  THEN GO TO EDIT-ERROR.
           00796        *
DMLC       00797             COPY IDMS SUBSCHEMA-BINDS.
           00798             MOVE 'EMPDISP ' TO PROGRAM-NAME
DMLC0015   00799             BIND RUN-UNIT.
DMLC0016   00810             READY USAGE-MODE IS RETRIEVAL.
           00815        ****************************************************************
           00816        *  SINCE THE MAP FIELD IS ASSOCIATED WITH THE EMP-ID-0415     *
           00817        * FIELD, THE PROGRAM USES THE "OF LR" RETRIEVAL.  NOTE THAT   *
           00818        * AUTOSTATUS IMPLICITLY CHECKS FOR THE LR-ERROR PATH STATUS.  *
           00819        ****************************************************************
DMLC       00820             OBTAIN EMP-JOB-LR
DMLC       00821                 WHERE EMP-ID-0415 = EMP-ID-0415 OF LR
DMLC0017   00822                 ON LR-NOT-FOUND
           00845                     GO TO NOT-FOUND.
DMLC0018   00846             FINISH.
           00851
```

```
           00852        ***************************************************************
           00853        *  REFORMAT DATE TO MMDDYY; OUTPUT AS MM/DD/YY USING THE OLM  *
           00854        *  EXTERNAL PICTURE SPECIFICATION (XX/XX/XX).                 *
           00855        ***************************************************************
           00856             MOVE START-YEAR-0415 TO WORK-YY.
           00857             MOVE START-MONTH-0415 TO WORK-MM.
           00858             MOVE START-DAY-0415 TO WORK-DD.
           00859
DMLC0019   00860             MAP OUT USING EMPMAPLR
DMLC0019   00861                OUTPUT DATA IS YES
DMLC0019   00862                MESSAGE IS DISPLAY-MESSAGE LENGTH 80.
           00873        *
DMLC0020   00874             DC RETURN NEXT TASK CODE TSK02.
           00881         GET-EMP-EXIT.
           00882             EXIT.
           00883        ***************************************************************
           00884        ***************************************************************
           00885        *  THE MODIFY MAP STATEMENT SPECIFIES THAT ALL MAP            *
           00886        * FIELDS EXCEPT THE INCORRECT EMP-ID-0415 FIELD WILL BE       *
           00887        * ERASED ON THE NEXT MAP OUT.                                 *
           00888        ***************************************************************
           00889         EDIT-ERROR.
DMLC0021   00890             MODIFY MAP EMPMAPLR TEMPORARY
DMLC0021   00891                FOR ALL EXCEPT DFLD EMP-ID-0415
DMLC0021   00892                  OUTPUT DATA IS ERASE.
           00906        *
DMLC0022   00907             MAP OUT USING EMPMAPLR
DMLC0022   00908                MESSAGE IS EDIT-ERROR-MESSAGE LENGTH 80.
           00919        *
DMLC0023   00920             DC RETURN
DMLC0023   00921                NEXT TASK CODE TSK02.
           00928         EDIT-ERROR-EXIT.
           00929             EXIT.



           00930        ***************************************************************
           00931        ***************************************************************
           00932        *  THE FOLLOWING MODIFY MAP STATEMENT SPECIFIES THAT ALL      *
           00933        * MAP FIELDS EXCEPT THE EMP-ID-0415 FIELD WILL BE ERASED      *
           00934        * ON THE NEXT MAP OUT.                                        *
           00935        ***************************************************************
           00936         NOT-FOUND.
DMLC0024   00937             MODIFY MAP EMPMAPLR TEMPORARY
DMLC0024   00938                FOR ALL EXCEPT DFLD EMP-ID-0415
DMLC0024   00939                  OUTPUT DATA IS ERASE.
           00953        *
DMLC0025   00954             MAP OUT USING EMPMAPLR
DMLC0025   00955                MESSAGE IS EMP-NOT-FOUND-MESSAGE LENGTH 80.
           00966        *
DMLC0026   00967             DC RETURN
DMLC0026   00968                NEXT TASK CODE TSK02.
           00975         NOT-FOUND-EXIT.
           00976             EXIT.
           00977        ***************************************************************
           00978         IDMS-ABORT.
           00979             MOVE ERROR-STATUS TO SSC-ERRSTAT-SAVE.
           00980             MOVE DML-SEQUENCE TO SSC-DMLSEQ-SAVE.
DMLC       00981             SNAP FROM SUBSCHEMA-LR-CTRL TO SUBSCHEMA-LR-CTRL-END
DMLC0027   00982                  ON ANY-STATUS
           00993                              NEXT SENTENCE.
           00994             MOVE SSC-ERRSTAT-SAVE TO ERROR-STATUS.
           00995             MOVE SSC-DMLSEQ-SAVE TO DML-SEQUENCE.
           00996         IDMS-ABORT-EXIT.
           00997             EXIT.
DMLC       00998             COPY IDMS IDMS-STATUS.
```

```
          00999        ****************************************************************01617000
          01000         IDMS-STATUS                                              SECTION.01618000
          01001        ******************** IDMS-STATUS FOR IDMS-DC *******************01619000
          01002                 IF DB-STATUS-OK GO TO ISABEX.                          01620000
          01003                 PERFORM IDMS-ABORT.                                    01621000
          01004                 MOVE ERROR-STATUS TO SSC-ERRSTAT-SAVE                  01622000
          01005                 MOVE DML-SEQUENCE TO SSC-DMLSEQ-SAVE                   01623000
DMLC      01006                 SNAP FROM SUBSCHEMA-CTRL TO SUBSCHEMA-CTRL-END         01624000
DMLC0028  01007                           ON ANY-STATUS                                01625000
          01018                                     NEXT SENTENCE.
DMLC      01019                 ABEND CODE SSC-ERRSTAT-SAVE                            01626000
DMLC0029  01020                           ON ANY-STATUS                                01627000
          01028                                     NEXT SENTENCE.
          01029         ISABEX. EXIT.                                                  01628000
          01030         DMLC-DC-GEN-GOBACK SECTION.
          01031             GOBACK.
***2000           *  W BIND RECORD NOT ISSUED
***2400           *  W WAS MOST SEVERE ERROR FOUND

0002 MESSAGES FOR PROGRAM EMPDISP
```

C.3.3 Sample online COBOL program from the COBOL compiler

```
00001         *NO-ACTIVITY-LOG
00002         *DMLIST
00003
00004          IDENTIFICATION DIVISION.
00005
00006          PROGRAM-ID.                     EMPDISP.
00007
00008          AUTHOR.                         COMPUTER ASSOCIATES.
00009
00010          DATE-WRITTEN.                   APRIL 1995.
00011
00012          REMARKS.                        THIS PROGRAM DEMONSTRATES
00013                                          CA-IDMS PROGRAMMING USING
00014                                          THE LOGICAL RECORD FACILITY.
00015
00016         ***************************************************************
00017          ENVIRONMENT DIVISION.
00018         ***************************************************************
00019         *IDMS-CONTROL SECTION.
00020
00021         *PROTOCOL.                      MODE IS IDMS-DC DEBUG
00022         *                                     IDMS-RECORDS MANUAL.
00024          DATA DIVISION.
00025
00026         *SCHEMA SECTION.
00027
00028         *    DB EMPSS09 WITHIN EMPSCHM.
00029
00030         *MAP SECTION.
00031         *MAX FIELD LIST IS 5.
00032         *MAP EMPMAPLR VERSION 1 TYPE IS STANDARD.
```

```
00033
00034
00035
00036          WORKING-STORAGE SECTION.
00037          01  TASK-CODE          PIC X(8).
00038          01  TSK01              PIC X(8)     VALUE 'TSK01'.
00039          01  TSK02              PIC X(8)     VALUE 'TSK02'.
00040
00041          01  MESSAGES.
00042              05 INITIAL-MESSAGE         PIC X(80) VALUE
00043                 'ENTER AN EMP ID AND PRESS ENTER ** CLEAR TO EXIT'.
00044              05 EDIT-ERROR-MESSAGE      PIC X(80) VALUE
00045                 'EMP-ID EITHER NOT ENTERED OR NOT NUMERIC'.
00046              05 EMP-NOT-FOUND-MESSAGE   PIC X(80) VALUE
00047                 'SPECIFIED EMPLOYEE COULD NOT BE FOUND'.
00048              05 DISPLAY-MESSAGE         PIC X(80) VALUE
00049                 'CLEAR TO EXIT ** NEW EMP-ID AND ENTER TO CONTINUE'.
00050
00051         *01  COPY IDMS DC-AID-CONDITION-NAMES.
00052          01  DC-AID-CONDITION-NAMES.
00053               03  DC-AID-IND-V           PIC X.
00054                                   88  ENTER-HIT VALUE QUOTE.
00055                                   88  CLEAR-HIT VALUE '_'.
00056                                   88  PF01-HIT VALUE '1'.
00057                                   88  PF02-HIT VALUE '2'.
00058                                   88  PF03-HIT VALUE '3'.
00059                                   88  PF04-HIT VALUE '4'.
00060                                   88  PF05-HIT VALUE '5'.
00061                                   88  PF06-HIT VALUE '6'.
00062                                   88  PF07-HIT VALUE '7'.
00063                                   88  PF08-HIT VALUE '8'.
00064                                   88  PF09-HIT VALUE '9'.
00065                                   88  PF10-HIT VALUE ':'.
00066                                   88  PF11-HIT VALUE '#'.
00067                                   88  PF12-HIT VALUE '@'.
00068                                   88  PF13-HIT VALUE 'A'.
00069                                   88  PF14-HIT VALUE 'B'.
00070                                   88  PF15-HIT VALUE 'C'.
00071                                   88  PF16-HIT VALUE 'D'.
00072                                   88  PF17-HIT VALUE 'E'.
00073                                   88  PF18-HIT VALUE 'F'.
00074                                   88  PF19-HIT VALUE 'G'.
00075                                   88  PF20-HIT VALUE 'H'.
00076                                   88  PF21-HIT VALUE 'I'.
00077                                   88  PF22-HIT VALUE '•'.
00078                                   88  PF23-HIT VALUE '.'.
00079                                   88  PF24-HIT VALUE '<'.
00080                                   88  PA01-HIT VALUE '%'.
00081                                   88  PA02-HIT VALUE '>'.
00082                                   88  PA03-HIT VALUE ','.
00083                                   88  PEN-ATTN-SPACE-NULL VALUE '='.
00084                                   88  PEN-ATTN VALUE QUOTE.
00085
00086         *01  COPY IDMS EMP-DATE-WORK-REC.
00087          01  EMP-DATE-WORK-REC.
00088              02  WORK-DATE.
00089               03  WORK-MM                PIC 9(2).
00090               03  WORK-DD                PIC 9(2).
00091               03  WORK-YY                PIC 9(2).
```

```
00092
00093         *01  COPY IDMS SUBSCHEMA-LR-CONTROL.
00094          01  SUBSCHEMA-CTRL.
00095               03  PROGRAM-NAME           PIC X(8) VALUE SPACES.
00096               03  ERROR-STATUS           PIC X(4) VALUE '1400'.
00097                                     88  DB-STATUS-OK VALUE '0000'.
00098                                     88  ANY-STATUS
00099                                         VALUE '0000' THRU '9999'.
00100                                     88  ANY-ERROR-STATUS
00101                                         VALUE '0001' THRU '9999'.
00102                                     88  DB-END-OF-SET VALUE '0307'.
00103                                     88  DB-REC-NOT-FOUND VALUE '0326'.
00104                                     88  DC-DEADLOCK VALUE '3101'
00105                                         '3201' '3401' '3901'.
00106                                     88  DC-NO-STORAGE VALUE '3202'
00107                                         '3402'.
00108                                     88  DC-AREA-ID-UNK VALUE '4303'.
00109                                     88  DC-QUEUE-ID-UNK VALUE '4404'.
00110                                     88  DC-REC-NOT-FOUND VALUE '4305'
00111                                         '4405'.
00112                                     88  DC-RESOURCE-NOT-AVAIL
00113                                         VALUE '3908'.
00114                                     88  DC-RESOURCE-AVAIL
00115                                         VALUE '3909'.
00116                                     88  DC-NEW-STORAGE VALUE '3210'.
00117                                     88  DC-MAX-TASKS VALUE '3711'.
00118                                     88  DC-REC-REPLACED VALUE '4317'.
00119                                     88  DC-TRUNCATED-DATA
00120                                         VALUE '4319' '4419'
00121                                         '4519' '4719'.
00122                                     88  DC-ATTN-INT VALUE '4525'
```

```
00123                                         '4625'.
00124                                     88  DC-OPER-CANCEL VALUE '4743'.
00125                                     88  DC-FIRST-PAGE-SENT
00126                                         VALUE '4676'.
00127                                     88  DC-SECOND-STARTPAGE
00128                                         VALUE '4604'.
00129                                     88  DC-DETAIL-NOT-FOUND
00130                                         VALUE '4664'.
00131               03  DBKEY                PIC S9(8)
00132                                         USAGE COMP.
00133               03  RECORD-NAME          PIC X(16) VALUE SPACES.
00134               03  RRECORD-NAME         REDEFINES RECORD-NAME.
00135                 05  SSC-NODN           PIC X(8).
00136                 05  SSC-DBN            PIC X(8).
00137               03  AREA-NAME            PIC X(16) VALUE SPACES.
00138               03  ERROR-SET            PIC X(16) VALUE SPACES.
00139               03  ERROR-RECORD         PIC X(16) VALUE SPACES.
00140               03  ERROR-AREA           PIC X(16) VALUE SPACES.
00141               03  IDBMSCOM-AREA        PIC X(100) VALUE LOW-VALUE.
00142               03  IDBMSCOM             REDEFINES IDBMSCOM-AREA
00143                                         PIC X
00144                                         OCCURS 100.
00145               03  RIDBMSCOM            REDEFINES IDBMSCOM-AREA.
00146                 05  DB-SUB-ADDR        PIC X(4).
00147                 05  FILLER             PIC X(0096).
00148               03  DIRECT-DBKEY         PIC S9(8)
00149                                         USAGE COMP.
```

```
00150              03  DIRECT-DBK             REDEFINES DIRECT-DBKEY
00151                                          PIC S9(8)
00152                                          USAGE COMP.
00153              03  DCBMSCOM-AREA          PIC X(100) VALUE LOW-VALUE.
00154              03  DCBMSCOM               REDEFINES DCBMSCOM-AREA
00155                                          PIC X
00156                                          OCCURS 100.
00157              03  R1DCBMSCOM             REDEFINES DCBMSCOM-AREA.
00158                05  R2DCBMSCOM           PIC S9(8)
00159                                          OCCURS 11
00160                                          USAGE COMP.
00161                05  DCSTR1               PIC X(16).
00162                05  R3DCBMSCOM           REDEFINES DCSTR1.
00163                  07  DCSTR2             PIC X(8).
00164                  07  R4DCBMSCOM         REDEFINES DCSTR2.
00165                    09  DCSTR4           PIC X(4).
00166                    09  DCSTR5           PIC X(4).
00167                  07  DCSTR3             PIC X(8).
00168                05  R5DCBMSCOM           REDEFINES DCSTR1.
00169                  07  DCPNUM1            PIC S9(15)
00170                                          USAGE COMP-3.
00171                05  DCNUM1               PIC S9(8)
00172                                          USAGE COMP.
00173                05  R6DCBMSCOM           REDEFINES DCNUM1.
00174                  07  DCPNUM2            PIC S9(7)
00175                                          USAGE COMP-3.
00176                05  DCNUM2               PIC S9(8)
00177                                          USAGE COMP.
00178                05  DCNUM3               PIC S9(8)
00179                                          USAGE COMP.
00180                05  DCFLG1               PIC S9(4)
```

```
00181                                          USAGE COMP.
00182                05  DCFLG2               PIC S9(4)
00183                                          USAGE COMP.
00184                05  DCFLG3               PIC S9(4)
00185                                          USAGE COMP.
00186                05  DCFLG4               PIC S9(4)
00187                                          USAGE COMP.
00188              03  SSC-ERRSTAT-SAVE       PIC X(4) VALUE SPACES.
00189              03  SSC-DMLSEQ-SAVE        PIC S9(8)
00190                                          USAGE COMP.
00191              03  DML-SEQUENCE           PIC S9(8)
00192                                          USAGE COMP.
00193              03  RECORD-OCCUR           PIC S9(8)
00194                                          USAGE COMP.
00195              03  SUBSCHEMA-CTRL-END     PIC X(4) VALUE SPACES.
00196          01  SUBSCHEMA-LR-CTRL.
00197              03  LRC-LRPXELNG           PIC S9(4)
00198                                          USAGE COMP.
00199              03  LRC-MAXVXP             PIC S9(4)
00200                                          USAGE COMP.
00201              03  LRIDENT                PIC X(4) VALUE 'LRC '.
00202              03  LRVERB                 PIC X(8).
00203              03  LRNAME                 PIC X(16).
00204              03  LR-STATUS              PIC X(16).
00205              03  FILLER                 PIC X(16).
00206              03  LRPXE                  PIC X
00207                                          OCCURS 0 TO 512
00208                                          DEPENDING ON LRC-LRPXELNG.
00209              03  PXE.
00210                05  PXENEXT              PIC S9(8)
00211                                          USAGE COMP.
00212                05  PXETABO              PIC S9(4)
00213                                          USAGE COMP.
```

```
00214                 05  PXEDSPL              PIC S9(4)
00215                                           USAGE COMP.
00216                 05  PXEDYN               PIC S9(4)
00217                                           USAGE COMP.
00218                 05  PXEDLEN              PIC S9(4)
00219                                           USAGE COMP.
00220                 05  PXENDEC              PIC X.
00221                 05  PXEDTYP              PIC X.
00222                 05  PXEOTYP              PIC X.
00223                 05  PXEFLAG              PIC X.
00224                 05  FILLER               PIC X(240).
00225               03  PXEDSP256              REDEFINES PXE
00226                                           PIC X(256).
00227               03  PXEDSP248              REDEFINES PXE
00228                                           PIC X(248).
00229               03  PXEDSP240              REDEFINES PXE
00230                                           PIC X(240).
00231               03  PXEDSP232              REDEFINES PXE
00232                                           PIC X(232).
00233               03  PXEDSP224              REDEFINES PXE
00234                                           PIC X(224).
00235               03  PXEDSP216              REDEFINES PXE
00236                                           PIC X(216).
00237               03  PXEDSP208              REDEFINES PXE
00238                                           PIC X(208).
00239               03  PXEDSP200              REDEFINES PXE
00240                                           PIC X(200).
00241               03  PXEDSP192              REDEFINES PXE
00242                                           PIC X(192).
00243               03  PXEDSP184              REDEFINES PXE
00244                                           PIC X(184).
00245               03  PXEDSP176              REDEFINES PXE
00246                                           PIC X(176).
00247               03  PXEDSP168              REDEFINES PXE
00248                                           PIC X(168).
```

```
00249              03  PXEDSP160             REDEFINES PXE
00250                                        PIC X(160).
00251              03  PXEDSP152             REDEFINES PXE
00252                                        PIC X(152).
00253              03  PXEDSP144             REDEFINES PXE
00254                                        PIC X(144).
00255              03  PXEDSP136             REDEFINES PXE
00256                                        PIC X(136).
00257              03  PXEDSP128             REDEFINES PXE
00258                                        PIC X(128).
00259              03  PXEDSP120             REDEFINES PXE
00260                                        PIC X(120).
00261              03  PXEDSP112             REDEFINES PXE
00262                                        PIC X(112).
00263              03  PXEDSP104             REDEFINES PXE
00264                                        PIC X(104).
00265              03  PXEDSP96              REDEFINES PXE
00266                                        PIC X(96).
00267              03  PXEDSP88              REDEFINES PXE
00268                                        PIC X(88).
00269              03  PXEDSP80              REDEFINES PXE
00270                                        PIC X(80).
00271              03  PXEDSP72              REDEFINES PXE
00272                                        PIC X(72).
00273              03  PXEDSP64              REDEFINES PXE
00274                                        PIC X(64).
00275              03  PXEDSP56              REDEFINES PXE
00276                                        PIC X(56).
00277              03  PXEDSP48              REDEFINES PXE
00278                                        PIC X(48).
00279              03  PXEDSP40              REDEFINES PXE
00280                                        PIC X(40).
00281              03  PXEDSP32              REDEFINES PXE
00282                                        PIC X(32).
00283              03  PXEDSP24              REDEFINES PXE
00284                                        PIC X(24).
00285              03  PXEDSP16              REDEFINES PXE
00286                                        PIC X(16).
00287              03  PXEDSP8               REDEFINES PXE
00288                                        PIC X(8).
00289              03  PXECOMP-1             REDEFINES PXE
00290                                        USAGE COMP-1.
00291              03  PXECOMP-2             REDEFINES PXE
00292                                        USAGE COMP-2.
00293              03  PXECOMP-30            REDEFINES PXE
00294                                        PIC S9(18)
```

```
00295                                   USAGE COMP-3.
00296              03  PXECOMP-31      REDEFINES PXE
00297                                   PIC S9(17)V9(1)
00298                                   USAGE COMP-3.
00299              03  PXECOMP-32      REDEFINES PXE
00300                                   PIC S9(16)V9(2)
00301                                   USAGE COMP-3.
00302              03  PXECOMP-33      REDEFINES PXE
00303                                   PIC S9(15)V9(3)
00304                                   USAGE COMP-3.
00305              03  PXECOMP-34      REDEFINES PXE
00306                                   PIC S9(14)V9(4)
00307                                   USAGE COMP-3.
00308              03  PXECOMP-35      REDEFINES PXE
00309                                   PIC S9(13)V9(5)
00310                                   USAGE COMP-3.
00311              03  PXECOMP-36      REDEFINES PXE
00312                                   PIC S9(12)V9(6)
00313                                   USAGE COMP-3.
00314              03  PXECOMP-37      REDEFINES PXE
00315                                   PIC S9(11)V9(7)
00316                                   USAGE COMP-3.
00317              03  PXECOMP-38      REDEFINES PXE
00318                                   PIC S9(10)V9(8)
00319                                   USAGE COMP-3.
00320              03  PXECOMP-39      REDEFINES PXE
00321                                   PIC S9(9)V9(9)
00322                                   USAGE COMP-3.
00323              03  PXECOMP-310     REDEFINES PXE
00324                                   PIC S9(8)V9(10)
00325                                   USAGE COMP-3.
00326              03  PXECOMP-311     REDEFINES PXE
00327                                   PIC S9(7)V9(11)
00328                                   USAGE COMP-3.
00329              03  PXECOMP-312     REDEFINES PXE
00330                                   PIC S9(6)V9(12)
00331                                   USAGE COMP-3.


00332              03  PXECOMP-313     REDEFINES PXE
00333                                   PIC S9(5)V9(13)
00334                                   USAGE COMP-3.
00335              03  PXECOMP-314     REDEFINES PXE
00336                                   PIC S9(4)V9(14)
00337                                   USAGE COMP-3.
00338              03  PXECOMP-315     REDEFINES PXE
00339                                   PIC S9(3)V9(15)
00340                                   USAGE COMP-3.
00341              03  PXECOMP-316     REDEFINES PXE
00342                                   PIC S9(2)V9(16)
00343                                   USAGE COMP-3.
00344              03  PXECOMP-317     REDEFINES PXE
00345                                   PIC S9(1)V9(17)
00346                                   USAGE COMP-3.
00347              03  PXECOMP-318     REDEFINES PXE
00348                                   PIC SV9(18)
00349                                   USAGE COMP-3.
00350              03  PXECOMP20       REDEFINES PXE
00351                                   PIC S9(4)
00352                                   USAGE COMP.
00353              03  PXECOMP21       REDEFINES PXE
00354                                   PIC S9(3)V9(1)
00355                                   USAGE COMP.
00356              03  PXECOMP22       REDEFINES PXE
00357                                   PIC S9(2)V9(2)
00358                                   USAGE COMP.
00359              03  PXECOMP23       REDEFINES PXE
```

```
                                        PIC S9(1)V9(3)
                                        USAGE COMP.
                03  PXECOMP24          REDEFINES PXE
                                        PIC SV9(4)
                                        USAGE COMP.
                03  PXECOMP40          REDEFINES PXE
                                        PIC S9(9)
                                        USAGE COMP.
                03  PXECOMP41          REDEFINES PXE
                                        PIC S9(8)V9(1)
                                        USAGE COMP.
                03  PXECOMP42          REDEFINES PXE
                                        PIC S9(7)V9(2)
                                        USAGE COMP.
                03  PXECOMP43          REDEFINES PXE
                                        PIC S9(6)V9(3)
                                        USAGE COMP.
                03  PXECOMP44          REDEFINES PXE
                                        PIC S9(5)V9(4)
                                        USAGE COMP.
                03  PXECOMP45          REDEFINES PXE
                                        PIC S9(4)V9(5)
                                        USAGE COMP.
                03  PXECOMP46          REDEFINES PXE
                                        PIC S9(3)V9(6)
                                        USAGE COMP.
                03  PXECOMP47          REDEFINES PXE
                                        PIC S9(2)V9(7)
                                        USAGE COMP.
                03  PXECOMP48          REDEFINES PXE
                                        PIC S9(1)V9(8)
                                        USAGE COMP.
```

```
                03  PXECOMP49          REDEFINES PXE
                                        PIC SV9(9)
                                        USAGE COMP.
                03  PXECOMP80          REDEFINES PXE
                                        PIC S9(18)
                                        USAGE COMP.
                03  PXECOMP81          REDEFINES PXE
                                        PIC S9(17)V9(1)
                                        USAGE COMP.
                03  PXECOMP82          REDEFINES PXE
                                        PIC S9(16)V9(2)
                                        USAGE COMP.
                03  PXECOMP83          REDEFINES PXE
                                        PIC S9(15)V9(3)
                                        USAGE COMP.
                03  PXECOMP84          REDEFINES PXE
                                        PIC S9(14)V9(4)
                                        USAGE COMP.
                03  PXECOMP85          REDEFINES PXE
                                        PIC S9(13)V9(5)
                                        USAGE COMP.
                03  PXECOMP86          REDEFINES PXE
                                        PIC S9(12)V9(6)
                                        USAGE COMP.
                03  PXECOMP87          REDEFINES PXE
                                        PIC S9(11)V9(7)
                                        USAGE COMP.
                03  PXECOMP88          REDEFINES PXE
                                        PIC S9(10)V9(8)
                                        USAGE COMP.
                03  PXECOMP89          REDEFINES PXE
```

```
00423                                   PIC S9(9)V9(9)
00424                                   USAGE COMP.
00425            03  PXECOMP810        REDEFINES PXE
00426                                   PIC S9(8)V9(10)
00427                                   USAGE COMP.
00428            03  PXECOMP811        REDEFINES PXE
00429                                   PIC S9(7)V9(11)
00430                                   USAGE COMP.
00431            03  PXECOMP812        REDEFINES PXE
00432                                   PIC S9(6)V9(12)
00433                                   USAGE COMP.
00434            03  PXECOMP813        REDEFINES PXE
00435                                   PIC S9(5)V9(13)
00436                                   USAGE COMP.
00437            03  PXECOMP814        REDEFINES PXE
00438                                   PIC S9(4)V9(14)
00439                                   USAGE COMP.
00440            03  PXECOMP815        REDEFINES PXE
00441                                   PIC S9(3)V9(15)
00442                                   USAGE COMP.
00443            03  PXECOMP816        REDEFINES PXE
00444                                   PIC S9(2)V9(16)
00445                                   USAGE COMP.
00446            03  PXECOMP817        REDEFINES PXE
00447                                   PIC S9(1)V9(17)
00448                                   USAGE COMP.
00449            03  PXECOMP818        REDEFINES PXE
00450                                   PIC SV9(18)
00451                                   USAGE COMP.
00452        01  SUBSCHEMA-SSNAME      PIC X(8) VALUE 'EMPSS09 '.
```

```
00453        01  SUBSCHEMA-AREANAMES.
00454            03  EMP-DEMO-REGION   PIC X(16)
00455                                   VALUE 'EMP-DEMO-REGION '.
00456            03  INS-DEMO-REGION   PIC X(16)
00457                                   VALUE 'INS-DEMO-REGION '.
00458            03  ORG-DEMO-REGION   PIC X(16)
00459                                   VALUE 'ORG-DEMO-REGION '.
00460
00461       *01  COPY IDMS SUBSCHEMA-LR-RECORDS.
00462        01  EMP-JOB-LR.
00463            02  EMPLOYEE.
00464            03  EMP-ID-0415       PIC 9(4).
00465            03  EMP-NAME-0415.
00466             04  EMP-FIRST-NAME-0415   PIC X(10).
00467             04  EMP-LAST-NAME-0415    PIC X(15).
00468            03  STATUS-0415       PIC X(2).
00469                              88  ACTIVE-0415 VALUE '01'.
00470                              88  ST-DISABIL-0415 VALUE '02'.
00471                              88  LT-DISABIL-0415 VALUE '03'.
00472                              88  LEAVE-OF-ABSENCE-0415
00473                                  VALUE '04'.
00474                              88  TERMINATED-0415 VALUE '05'.
00475            03  SS-NUMBER-0415    PIC 9(9).
00476            03  START-DATE-0415.
00477             04  START-YEAR-0415  PIC 9(2).
00478             04  START-MONTH-0415 PIC 9(2).
00479             04  START-DAY-0415   PIC 9(2).
00480            03  FILLER            PIC X(2).
00481            02  DEPARTMENT.
00482            03  DEPT-ID-0410      PIC 9(4).
00483            03  DEPT-NAME-0410    PIC X(45).
00484            03  DEPT-HEAD-ID-0410 PIC 9(4).
```

```
00485              03  FILLER                PIC XXX.
00486             02  JOB.
00487              03  JOB-ID-0440           PIC 9(4).
00488              03  TITLE-0440            PIC X(20).
00489             02  OFFICE.
00490              03  OFFICE-CODE-0450      PIC X(3).
00491              03  OFFICE-ADDRESS-0450.
00492               04  OFFICE-STREET-0450   PIC X(20).
00493               04  OFFICE-CITY-0450     PIC X(15).
00494               04  OFFICE-STATE-0450    PIC X(2).
00495               04  OFFICE-ZIP-0450.
00496                05  OFFICE-ZIP-FIRST-FIVE-0450
00497                                        PIC X(5).
00498                05  OFFICE-ZIP-LAST-FOUR-0450
00499                                        PIC X(4).
00500              03  OFFICE-PHONE-0450     PIC 9(7)
00501                                        OCCURS 3.
00502              03  OFFICE-AREA-CODE-0450 PIC X(3).
00503              03  SPEED-DIAL-0450       PIC X(3).
00504              03  FILLER                PIC X(4).
00505             03 SUBSCHEMA-LR-CTRL-END         PIC X.
00506
00507         *01  COPY IDMS MAP-CONTROLS.
00508          01  MRB-EMPMAPLR.
00509              03  MRB-EMPMAPLR-ID       PIC X(8).
00510              03  MRB-EMPMAPLR-MCOMP-VER.
00511                05  MRB-EMPMAPLR-MCOMP-DATE
00512                                        PIC X(8).
00513                05  MRB-EMPMAPLR-MCOMP-TIME
00514                                        PIC X(6).
00515                05  MRB-EMPMAPLR-MCOMP-VERID
00516                                        PIC X(2).
```

```
00517              03  MRB-EMPMAPLR-SUBSCHEMA PIC X(8).
00518              03  MRB-EMPMAPLR-FLGS     PIC X
00519                                        OCCURS 4.
00520              03  FILLER                PIC X(6).
00521              03  MRB-EMPMAPLR-NFLDS    PIC S9(4)
00522                                        USAGE COMP.
00523              03  MRB-EMPMAPLR-NRECS    PIC S9(4)
00524                                        USAGE COMP.
00525              03  MRB-EMPMAPLR-RECOF    PIC S9(4)
00526                                        USAGE COMP.
00527              03  MRB-EMPMAPLR-PERM-CURSOR
00528                                        PIC XX.
00529              03  MRB-EMPMAPLR-TEMP-CURSOR
00530                                        PIC XX.
00531              03  MRB-EMPMAPLR-PERM-WCC PIC X.
00532              03  MRB-EMPMAPLR-TEMP-WCC PIC X.
00533              03  MRB-EMPMAPLR-CURSOR   PIC XX.
00534              03  MRB-EMPMAPLR-AID      PIC X.
00535              03  MRB-EMPMAPLR-INPUT-FLGS
00536                                        PIC X.
00537              03  MRB-EMPMAPLR-SEGVIEW  PIC X.
00538              03  FILLER                PIC X.
00539              03  MRB-EMPMAPLR-MREO     PIC S9(4)
00540                                        USAGE COMP.
00541              03  MRB-EMPMAPLR-ERR-CNT  PIC S9(4)
00542                                        USAGE COMP.
00543              03  MRB-EMPMAPLR-ATTR-FLGS PIC X
00544                                        OCCURS 4.
00545              03  MRB-EMPMAPLR-CURR-MFLD PIC S9(4)
00546                                        USAGE COMP.
00547              03  MRB-EMPMAPLR-XTYP     PIC X.
00548              03  MRB-EMPMAPLR-FILLER   PIC X.
```

```
00549              03  MRB-EMPMAPLR-MRE-XLEN  PIC S9(4)
00550                                          USAGE COMP.
00551              03  MRB-EMPMAPLR-MRB-XLEN  PIC S9(4)
00552                                          USAGE COMP.
00553              03  MRB-EMPMAPLR-MRE       OCCURS 11.
00554                05  MRB-EMPMAPLR-MRE-FLGS
00555                                          PIC X
00556                                          OCCURS 8.
00557                05  MRB-EMPMAPLR-MRE-INLEN
00558                                          PIC S9(4)
00559                                          USAGE COMP.
00560                05  MRB-EMPMAPLR-MRE-PAD-CHAR
00561                                          PIC X
00562                                          OCCURS 2.
00563                05  MRB-EMPMAPLR-MRE-FLG2
00564                                          PIC X
00565                                          OCCURS 2.
00566              03  MRB-EMPMAPLR-RECS      PIC S9(8)
00567                                          OCCURS 5
00568                                          USAGE COMP
00569                                          SYNC.
00570              03  MRB-EMPMAPLR-END       PIC X.
00571              03  MRB-EMPMAPLR-MRE-SUB   PIC S9(4)
00572                                          USAGE COMP.
00573
00574
00575          01  MRB-FLDLST.
00576              02  FLDLST                 PIC S9(8)



00577                                          OCCURS 6
00578                                          USAGE COMP.
00579          PROCEDURE DIVISION.
00580
00581         *    ********************************************************
00582         *    *  PROCEDURE DIVISION GENERAL STRATEGY:                *
00583         *    *      RETRIEVE INFORMATION FOR A SPECIFIED EMPLOYEE.   *
00584         *    *      DISPLAYED DATA INCLUDES EMPLOYEE, DEPARTMENT,    *
00585         *    *                  JOB, AND OFFICE INFORMATION.         *
00586         *    * ==> THIS PROGRAM USES THE EMP-JOB-LR LOGICAL RECORD<= *
00587         *    *  PROGRAM STRATEGY:                                   *
00588         *    *         ** CHECK FOR TASK CODE: TSK01= INITIAL MAPOUT *
00589         *    *                          ANYTHING ELSE = RETRIEVE LR    *
00590         *    *         ** CLEAR TO EXIT APPLICATION                 *
00591         *    *         ** ENTER AND NEW EMP-ID TO CONTINUE          *
00592         *    ********************************************************
00593
00594          MAIN-LINE.
00595         ***************************************************************
00596         *  THE BIND MAP STATEMENTS ADVISE IDMS-DC OF THE LOCATION OF  *
00597         * THE MRB AND THE MAP RECORDS.                                *
00598         ***************************************************************
00599         *    BIND MAP EMPMAPLR.
00600                        MOVE 0001 TO DML-SEQUENCE                      DMLC0001
00601                        CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00602                               DCBMSCOM (90)
00603                               MRB-EMPMAPLR
00604                               MRB-EMPMAPLR-END
00605                        MOVE '08/12/85112414R2'
00606                             TO MRB-EMPMAPLR-MCOMP-VER
00607                        MOVE 'EMPSS09 '
00608                             TO MRB-EMPMAPLR-SUBSCHEMA
00609                        MOVE 'EMPMAPLR'
00610                             TO MRB-EMPMAPLR-ID
00611                        MOVE 11
```

```
00612                           TO MRB-EMPMAPLR-NFLDS
00613                      MOVE 5
00614                           TO MRB-EMPMAPLR-NRECS
00615                      MOVE 156
00616                           TO MRB-EMPMAPLR-RECOF
00617                      MOVE 76
00618                           TO MRB-EMPMAPLR-MREO
00619                      MOVE '0'
00620                           TO MRB-EMPMAPLR-XTYP
00621                      MOVE 0
00622                           TO MRB-EMPMAPLR-MRE-XLEN
00623                      MOVE 0
00624                           TO MRB-EMPMAPLR-MRB-XLEN
00625                      MOVE 'Y'
00626                           TO MRB-EMPMAPLR-SEGVIEW
00627                        PERFORM IDMS-STATUS.
00628        *    BIND MAP EMPMAPLR RECORD EMPLOYEE.
00629                      MOVE 0002 TO DML-SEQUENCE                      DMLC0002
00630                      CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00631                             DCBMSCOM (91)
00632                             MRB-EMPMAPLR-RECS (1)
00633                             EMPLOYEE
00634                        PERFORM IDMS-STATUS.
00635        *    BIND MAP EMPMAPLR RECORD DEPARTMENT.
00636                      MOVE 0003 TO DML-SEQUENCE                      DMLC0003
00637                      CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00638                             DCBMSCOM (91)
00639                             MRB-EMPMAPLR-RECS (2)
00640                             DEPARTMENT
00641                        PERFORM IDMS-STATUS.
```

```
00642        *    BIND MAP EMPMAPLR RECORD JOB.
00643                      MOVE 0004 TO DML-SEQUENCE                      DMLC0004
00644                      CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00645                             DCBMSCOM (91)
00646                             MRB-EMPMAPLR-RECS (3)
00647                             JOB
00648                        PERFORM IDMS-STATUS.
00649        *    BIND MAP EMPMAPLR RECORD OFFICE.
00650                      MOVE 0005 TO DML-SEQUENCE                      DMLC0005
00651                      CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00652                             DCBMSCOM (91)
00653                             MRB-EMPMAPLR-RECS (4)
00654                             OFFICE
00655                        PERFORM IDMS-STATUS.
00656        *    BIND MAP EMPMAPLR RECORD EMP-DATE-WORK-REC.
00657                      MOVE 0006 TO DML-SEQUENCE                      DMLC0006
00658                      CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00659                             DCBMSCOM (91)
00660                             MRB-EMPMAPLR-RECS (5)
00661                             EMP-DATE-WORK-REC
00662                        PERFORM IDMS-STATUS.
00663        *
00664        *    ACCEPT TASK CODE INTO TASK-CODE.
00665                      MOVE 0007 TO DML-SEQUENCE                      DMLC0007
00666                      MOVE 1 TO DCNUM1
00667                      CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00668                             DCBMSCOM (2)
00669                             TASK-CODE
00670                        PERFORM IDMS-STATUS.
```

```
00671             IF TASK-CODE = TSK01
00672                  GO TO INITIAL-MAPOUT
00673             ELSE
00674                  GO TO GET-EMP.
00675        ************************************************************
00676        ************************************************************
00677        *  THE INITIAL-MAPOUT PARAGRAPH IS PERFORMED IF THE CALLING   *
00678        * TASK CODE IS TSK01.                                        *
00679        ************************************************************
00680        *  THE MODIFY MAP STATEMENT ASSIGNS THE PROTECTED             *
00681        * ATTRIBUTE TO ALL MAP FIELDS EXCEPT EMP-ID-0415.             *
00682        ************************************************************
00683        *  THE MAP OUT STATEMENT TRANSMITS THE EMPMAPLR MAP           *
00684        * TO THE TERMINAL.                                           *
00685        ************************************************************
00686        *  THE DC RETURN STATEMENT SPECIFIES THAT THE NEXT            *
00687        * TASK THAT WILL BE INITIATED ON THE SAME TERMINAL WHEN THE   *
00688        * OPERATOR PRESSES A CONTROL KEY WILL BE TSK02.               *
00689        ************************************************************
00690         INITIAL-MAPOUT.
00691        *    MODIFY MAP EMPMAPLR TEMPORARY
00692        *       FOR ALL EXCEPT EMP-ID-0415
00693        *         ATTRIBUTES PROTECTED.
00694                       MOVE 0008 TO DML-SEQUENCE                      DMLC0008
00695                       MOVE 8 TO DCNUM1
00696                       MOVE 2561 TO DCFLG1
00697                       MOVE 0 TO DCFLG2
00698                       MOVE 0 TO DCFLG3
00699                       MOVE 0 TO DCFLG4
00700                       MOVE 1 TO FLDLST (2)
00701                       MOVE 1 TO FLDLST (1)
00702                       CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00703                              DCBMSCOM (93)
00704                              MRB-EMPMAPLR
00705                              MRB-FLDLST
```

```
00706                         PERFORM IDMS-STATUS.
00707        *
00708             MOVE ZERO TO EMP-ID-0415.
00709        *    MAP OUT USING EMPMAPLR
00710        *       OUTPUT DATA IS YES NEWPAGE
00711        *       MESSAGE IS INITIAL-MESSAGE LENGTH 80.
00712                       MOVE 0009 TO DML-SEQUENCE                      DMLC0009
00713                       MOVE 5 TO DCFLG1
00714                       MOVE 16 TO DCFLG2
00715                       MOVE 1 TO DCFLG3
00716                       MOVE 4 TO DCFLG4
00717                       CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00718                              DCBMSCOM (34)
00719                              MRB-EMPMAPLR
00720                 INITIAL-MESSAGE DCBMSCOM (80)
00721                         PERFORM IDMS-STATUS.
00722
00723        *    DC RETURN
00724        *       NEXT TASK CODE TSK02.
00725                       MOVE 0010 TO DML-SEQUENCE                      DMLC0010
00726                       MOVE TSK02 TO DCSTR2
00727                       MOVE 128 TO DCFLG1
00728                       CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00729                              DCBMSCOM (19)
00730                         PERFORM IDMS-STATUS.
```

```
00731          INITIAL-MAPOUT-EXIT.
00732              EXIT.
00733         ***************************************************************
00734         ***************************************************************
00735         *  THE GET-EMP PARAGRAPH IS PERFORMED IF THE CALLING TASK     *
00736         * CODE IS NOT TSK01.                                          *
00737         ***************************************************************
00738         *  THE MAP IN STATEMENT TRANSMITS DATA FROM THE TERMINAL TO   *
00739         * VARIABLE STORAGE DATA FIELDS.                               *
00740         ***************************************************************
00741         *  THIS FIRST INQUIRE MAP STATEMENT IS USED TO DETERMINE      *
00742         * THE AID KEY PRESSED.                                        *
00743         ***************************************************************
00744         *  THIS SECOND INQUIRE MAP STATEMENT USES AUTOMATIC EDITING   *
00745         * TO DETERMINE IF THE DATA ENTERED IS CONSISTENT WITH         *
00746         * THE EXTERNAL PICTURE OF THE NAMED DATA ELEMENT.             *
00747         ***************************************************************
00748         *  THE MAP OUT STATEMENT TRANSMITS DATA FROM THE              *
00749         * EMP-JOB-LR LOGICAL RECORD IN VARIABLE STORAGE TO MAP        *
00750         * FIELDS.                                                     *
00751         ***************************************************************
00752          GET-EMP.
00753         *    MAP IN USING EMPMAPLR.
```

```
00754                         MOVE 0011 TO DML-SEQUENCE                       DMLC0011
00755                         MOVE 6 TO DCFLG1
00756                         MOVE 0 TO DCFLG2
00757                         MOVE 0 TO DCFLG3
00758                         MOVE 0 TO DCFLG4
00759                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00760                                DCBMSCOM (34)
00761                                MRB-EMPMAPLR
00762                           PERFORM IDMS-STATUS.
00763         *
00764         *    INQUIRE MAP EMPMAPLR
00765         *       MOVE AID TO DC-AID-IND-V.
00766                         MOVE 0012 TO DML-SEQUENCE                       DMLC0012
00767                         MOVE 7 TO DCNUM1
00768                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00769                                DCBMSCOM (92)
00770                                MRB-EMPMAPLR
00771                         MOVE DCSTR2 TO DC-AID-IND-V
00772                           PERFORM IDMS-STATUS.
00773              IF CLEAR-HIT
00774         *       DC RETURN.
00775                         MOVE 0013 TO DML-SEQUENCE                       DMLC0013
00776                         MOVE 0 TO DCFLG1
00777                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00778                                DCBMSCOM (19)
00779                           PERFORM IDMS-STATUS.
00780
00781         *
00782         *    INQUIRE MAP EMPMAPLR
00783         *       IF DFLD EMP-ID-0415 EDIT IS ERROR
00784                         MOVE 0014 TO DML-SEQUENCE                       DMLC0014
00785                         MOVE 17 TO DCNUM1
00786                         MOVE 5 TO DCNUM2
```

```
00787                       MOVE 2048 TO DCFLG1
00788                       MOVE 1 TO FLDLST (2)
00789                       MOVE 1 TO FLDLST (1)
00790                       CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00791                              DCBMSCOM (92)
00792                              MRB-EMPMAPLR
00793                              MRB-FLDLST;
00794                       IF ERROR-STATUS EQUAL TO '4641'
00795                 THEN GO TO EDIT-ERROR.
00796         *
00797         *    COPY IDMS SUBSCHEMA-BINDS.
00798              MOVE 'EMPDISP ' TO PROGRAM-NAME
00799         *    BIND RUN-UNIT.
00800                       MOVE 0015 TO DML-SEQUENCE                      DMLC0015
00801                       MOVE 576  TO LRC-LRPXELNG
00802                       MOVE 6    TO LRC-MAXVXP
00803                       MOVE 'LRF-BIND' TO LR-STATUS
00804                       CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00805                              IDBMSCOM (59)
00806                              SUBSCHEMA-CTRL
00807                              SUBSCHEMA-SSNAME
00808                              SUBSCHEMA-LR-CTRL
00809                         PERFORM IDMS-STATUS.
00810         *    READY USAGE-MODE IS RETRIEVAL.
00811                       MOVE 0016 TO DML-SEQUENCE                      DMLC0016
00812                       CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00813                              IDBMSCOM (37)
00814                         PERFORM IDMS-STATUS.


00815         ************************************************************
00816         *  SINCE THE MAP FIELD IS ASSOCIATED WITH THE EMP-ID-0415     *
00817         * FIELD, THE PROGRAM USES THE "OF LR" RETRIEVAL.  NOTE THAT   *
00818         * AUTOSTATUS IMPLICITLY CHECKS FOR THE LR-ERROR PATH STATUS.  *
00819         ************************************************************
00820         *    OBTAIN EMP-JOB-LR
00821         *        WHERE EMP-ID-0415 = EMP-ID-0415 OF LR
00822         *        ON LR-NOT-FOUND
00823                       MOVE 0017 TO DML-SEQUENCE                      DMLC0017
00824                       MOVE 0    TO LRC-LRPXELNG
00825                       MOVE 0036 TO LRC-MAXVXP
00826                       MOVE 'LR-ERROR' TO LR-STATUS
00827                       MOVE 'OBTAIN N' TO LRVERB
00828                       MOVE 'EMP-JOB-LR' TO LRNAME
00856              MOVE START-YEAR-0415 TO WORK-YY.
00857              MOVE START-MONTH-0415 TO WORK-MM.
00858              MOVE START-DAY-0415 TO WORK-DD.
00859
00860         *    MAP OUT USING EMPMAPLR
00861         *       OUTPUT DATA IS YES
00862         *       MESSAGE IS DISPLAY-MESSAGE LENGTH 80.
00863                       MOVE 0019 TO DML-SEQUENCE                      DMLC0019
00864                       MOVE 5 TO DCFLG1
00865                       MOVE 16 TO DCFLG2
00866                       MOVE 0 TO DCFLG3
00867                       MOVE 4 TO DCFLG4
00868                       CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00869                              DCBMSCOM (34)
00870                              MRB-EMPMAPLR
00871                 DISPLAY-MESSAGE DCBMSCOM (80)
00872                         PERFORM IDMS-STATUS.
```

```
00873         *
00874         *    DC RETURN NEXT TASK CODE TSK02.
00875                         MOVE 0020 TO DML-SEQUENCE                       DMLC0020
00876                         MOVE TSK02 TO DCSTR2
00877                         MOVE 128 TO DCFLG1
00878                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00879                                DCBMSCOM (19)
00880                           PERFORM IDMS-STATUS.
00881          GET-EMP-EXIT.
00882              EXIT.
00883         ***************************************************************
00884         ***************************************************************
00885         *  THE MODIFY MAP STATEMENT SPECIFIES THAT ALL MAP            *
00886         * FIELDS EXCEPT THE INCORRECT EMP-ID-0415 FIELD WILL BE       *
00887         * ERASED ON THE NEXT MAP OUT.                                 *
00888         ***************************************************************
00889          EDIT-ERROR.
00890         *    MODIFY MAP EMPMAPLR TEMPORARY
00891         *       FOR ALL EXCEPT DFLD EMP-ID-0415
00892         *         OUTPUT DATA IS ERASE.
00893                         MOVE 0021 TO DML-SEQUENCE                       DMLC0021
00894                         MOVE 0 TO DCNUM1
00895                         MOVE 2561 TO DCFLG1
00896                         MOVE 16 TO DCFLG2
00897                         MOVE 0 TO DCFLG3
00898                         MOVE 0 TO DCFLG4
00899                         MOVE 1 TO FLDLST (2)
00900                         MOVE 1 TO FLDLST (1)
00901                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00902                                DCBMSCOM (93)
00903                                MRB-EMPMAPLR
00904                                MRB-FLDLST
00905                           PERFORM IDMS-STATUS.
```

```
00906         *
00907         *    MAP OUT USING EMPMAPLR
00908         *       MESSAGE IS EDIT-ERROR-MESSAGE LENGTH 80.
00909                         MOVE 0022 TO DML-SEQUENCE                       DMLC0022
00910                         MOVE 5 TO DCFLG1
00911                         MOVE 0 TO DCFLG2
00912                         MOVE 0 TO DCFLG3
00913                         MOVE 4 TO DCFLG4
00914                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00915                                DCBMSCOM (34)
00916                                MRB-EMPMAPLR
00917                  EDIT-ERROR-MESSAGE DCBMSCOM (80)
00918                           PERFORM IDMS-STATUS.
00919         *
00920         *    DC RETURN
00921         *       NEXT TASK CODE TSK02.
00922                         MOVE 0023 TO DML-SEQUENCE                       DMLC0023
00923                         MOVE TSK02 TO DCSTR2
00924                         MOVE 128 TO DCFLG1
00925                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00926                                DCBMSCOM (19)
00927                           PERFORM IDMS-STATUS.
00928          EDIT-ERROR-EXIT.
00929              EXIT.
00930         ***************************************************************
00931         ***************************************************************
00932         *  THE FOLLOWING MODIFY MAP STATEMENT SPECIFIES THAT ALL      *
00933         * MAP FIELDS EXCEPT THE EMP-ID-0415 FIELD WILL BE ERASED      *
00934         * ON THE NEXT MAP OUT.                                        *
00935         ***************************************************************
```

```
00936          NOT-FOUND.
00937         *    MODIFY MAP EMPMAPLR TEMPORARY
00938         *       FOR ALL EXCEPT DFLD EMP-ID-0415
00939         *         OUTPUT DATA IS ERASE.
00940                        MOVE 0024 TO DML-SEQUENCE                       DMLC0024
00941                        MOVE 0 TO DCNUM1
00942                        MOVE 2561 TO DCFLG1
00943                        MOVE 16 TO DCFLG2
00944                        MOVE 0 TO DCFLG3
00945                        MOVE 0 TO DCFLG4
00946                        MOVE 1 TO FLDLST (2)
00947                        MOVE 1 TO FLDLST (1)
00948                        CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00949                               DCBMSCOM (93)
00950                               MRB-EMPMAPLR
00951                               MRB-FLDLST
00952                          PERFORM IDMS-STATUS.
00953         *
00954         *    MAP OUT USING EMPMAPLR
00955         *       MESSAGE IS EMP-NOT-FOUND-MESSAGE LENGTH 80.
00956                        MOVE 0025 TO DML-SEQUENCE                       DMLC0025
00957                        MOVE 5 TO DCFLG1
00958                        MOVE 0 TO DCFLG2
00959                        MOVE 0 TO DCFLG3
00960                        MOVE 4 TO DCFLG4
00961                        CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00962                               DCBMSCOM (34)
00963                               MRB-EMPMAPLR
00964                 EMP-NOT-FOUND-MESSAGE DCBMSCOM (80)
00965                          PERFORM IDMS-STATUS.



00966         *
00967         *    DC RETURN
00968         *       NEXT TASK CODE TSK02.
00969                        MOVE 0026 TO DML-SEQUENCE                       DMLC0026
00970                        MOVE TSK02 TO DCSTR2
00971                        MOVE 128 TO DCFLG1
00972                        CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00973                               DCBMSCOM (19)
00974                          PERFORM IDMS-STATUS.
00975          NOT-FOUND-EXIT.
00976              EXIT.
00977         ************************************************************
00978          IDMS-ABORT.
00979              MOVE ERROR-STATUS TO SSC-ERRSTAT-SAVE.
00980              MOVE DML-SEQUENCE TO SSC-DMLSEQ-SAVE.
00981         *    SNAP FROM SUBSCHEMA-LR-CTRL TO SUBSCHEMA-LR-CTRL-END
00982         *         ON ANY-STATUS
00983                        MOVE 0027 TO DML-SEQUENCE                       DMLC0027
00984                        MOVE 0 TO DCFLG1
00985                        CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
00986                               DCBMSCOM (22)
00987                               DCSTR1
00988                               DCSTR1
00989                               DCSTR1
00990                 SUBSCHEMA-LR-CTRL SUBSCHEMA-LR-CTRL-END DCBMSCOM (1)
00991                        IF NOT ANY-STATUS PERFORM IDMS-STATUS;
00992                        ELSE
00993                               NEXT SENTENCE.
00994              MOVE SSC-ERRSTAT-SAVE TO ERROR-STATUS.
00995              MOVE SSC-DMLSEQ-SAVE TO DML-SEQUENCE.
00996          IDMS-ABORT-EXIT.
00997              EXIT.
00998         *    COPY IDMS IDMS-STATUS.
```

```
00999         ****************************************************************01617000
01000          IDMS-STATUS                                            SECTION.01618000
01001         ******************** IDMS-STATUS FOR IDMS-DC ********************01619000
01002                 IF DB-STATUS-OK GO TO ISABEX.                            01620000
01003                 PERFORM IDMS-ABORT.                                      01621000
01004                 MOVE ERROR-STATUS TO SSC-ERRSTAT-SAVE                    01622000
01005                 MOVE DML-SEQUENCE TO SSC-DMLSEQ-SAVE                     01623000
01006         *       SNAP FROM SUBSCHEMA-CTRL TO SUBSCHEMA-CTRL-END           01624000
01007         *                  ON ANY-STATUS                                 01625000
01008                         MOVE 0028 TO DML-SEQUENCE                        DMLC0028
01009                         MOVE 0 TO DCFLG1
01010                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
01011                                DCBMSCOM (22)
01012                                DCSTR1
01013                                DCSTR1
01014                                DCSTR1
01015                 SUBSCHEMA-CTRL SUBSCHEMA-CTRL-END DCBMSCOM (1)
01016                         IF NOT ANY-STATUS PERFORM IDMS-STATUS;
01017                         ELSE
01018                                         NEXT SENTENCE.
01019         *       ABEND CODE SSC-ERRSTAT-SAVE                              01626000
01020         *                  ON ANY-STATUS                                 01627000
01021                         MOVE 0029 TO DML-SEQUENCE                        DMLC0029
01022                         MOVE SSC-ERRSTAT-SAVE TO DCSTR4
01023                         MOVE 2 TO DCFLG1
01024                         CALL 'IDMSCOBI' USING SUBSCHEMA-CTRL
01025                                DCBMSCOM (1)
01026                         IF NOT ANY-STATUS PERFORM IDMS-STATUS;
01027                         ELSE
01028                                         NEXT SENTENCE.
01029          ISABEX. EXIT.                                                   01628000
01030          DMLC-DC-GEN-GOBACK SECTION.
```

Appendix D. CA-IDMS Call Formats

D.1 About this appendix D-3
D.2 DB call formats D-4
D.3 DC call formats D-12
D.4 Sample COBOL program calling IDMSIN01 . . . . . D-15

D.1 About this appendix

This appendix contains the call formats used by CA-IDMS to execute DML commands. Each DML function can be coded using standard CALL statements.

The two tables in this appendix present the function codes and arguments that are passed to CA-IDMS for execution of a DML command. Argument 0, which contains SUBSCHEMA-CTRL (the CA-IDMS communications block), is passed for all functions.

The program listing presented in this appendix illustrates the use of standard COBOL IDMS calling conventions.

The following example shows the expanded CA-IDMS call format for a BIND RECORD statement (BIND EMPLOYEE):

```
CALL 'IDMS' USING SUBSCHEMA-CTRL
                  IDBMSCOM (48)
                  SR415
                  EMPLOYEE.
```

The call expansions are presented in two tables; the first table lists the DB expansions and the second table lists the DC expansions.

D.2 DB call formats

The CA-IDMS DBMS passes the IDBMSCOM array as argument 1. Arguments marked with asterisks have default values.

		CALLING ARGUMENTS (argument 0 contains SUBSCHEMA-CTRL)				
Major Function Code	**Database Service (in COBOL DML)**	**(1) IDBMSCOM (nn)**	**(2)**	**(3)**	**(4)**	**(5)**
	CONTROL STATEMENTS					
14	BIND RUN-UNIT	59	IDMS Communications Block*	subschema-name*		
	BIND RUN-UNIT FOR subschema-name	59	IDMS Communications Block*	subschema-name		
	BIND RUN-UNIT NODENAME node-name	59	IDMS Communications Block*	subschema-name*	subschema-ctrl* OR subschema-lr-ctrl*	node-name
	BIND RUN-UNIT FOR subschema-name NODENAME node-name	59	IDMS Communications Block*	subschema-name	subschema-ctrl* OR subschema-lr-ctrl*	node-name
	BIND RUN-UNIT FOR subschema-name DBNAME database-name	59	IDMS Communications Block*	subschema-name	subschema-ctrl* OR subschema-lr-ctrl*	node-name
	BIND RUN-UNIT NODENAME node-name DBNAME database-name	59	IDMS Communications Block*	subschema-name*	subschema-ctrl* OR subschema-lr-ctrl*	node-name
	BIND RUN-UNIT FOR subschema-name NODENAME node-name DBNAME database-name	59	IDMS Communications Block*	subschema-name	subschema-ctrl* OR subschema-lr-ctrl*	node-name
	BIND record-name	48	record-id	record-location*		
	BIND record-name TO record-location	48	record-id	record-location		
	BIND record-location WITH record-name	48	record-id	record-location		
	BIND PROCEDURE FOR procedure-name TO procedure-control-location	73	procedure-name	procedure-control-location		

(Continued)

09	READY	37				
	READY area-name	37	area-name			
	READY area-name USAGE-MODE IS RETRIEVAL	37	area-name			
	READY area-name USAGE-MODE IS PROTECTED RETRIEVAL	39	area-name			
	READY area-name USAGE-MODE IS EXCLUSIVE RETRIEVAL	40	area-name			
	READY area-name USAGE-MODE IS UPDATE	36	area-name			
	READY area-name USAGE-MODE IS PROTECTED UPDATE	38	area-name			
	READY area-name USAGE-MODE IS EXCLUSIVE UPDATE	41	area-name			
	READY USAGE-MODE IS ...	**				
	**Choose function code from 36-41 as shown above.					
01	FINISH	02				
	FINISH TASK	113				
18	COMMIT	66				
	COMMIT ALL	95				
	COMMIT TASK	114				
	COMMIT TASK ALL	115				
19	ROLLBACK	67				
	ROLLBACK ALL	96				
	ROLLBACK TASK	116				
	ROLLBACK TASK CONTINUE	117				
06	KEEP CURRENT	87				
	KEEP EXCLUSIVE CURRENT	90				
	KEEP CURRENT record-name	89	record-name			
	KEEP EXCLUSIVE CURRENT record-name	90	record-name			
	KEEP CURRENT WITHIN set-name	91	set-name			
	KEEP EXCLUSIVE CURRENT WITHIN set-name	92	set-name			

(Continued)

	KEEP CURRENT WITHIN area-name	93	area-name			
	KEEP EXCLUSIVE CURRENT WITHIN area-name	92	area-name			
16	IF set-name IS EMPTY...	64	set-name			
	IF set-name IS NOT EMPTY...	65	set-name			
	(Upon return to user run-unit, the status indicator = '0000' if set is empty; '1601' if not empty.)					
	IF set-name MEMBER...	60	set-name			
	IF NOT set-name MEMBER...	62	set-name			
	(Upon return to user run-unit, the status indicator = '0000' if the record (current of run unit) is linked into the specified set; '1601' if it is not a member.)					
	MODIFICATION STATEMENTS					
12	STORE record-name	42	record-name			
07	CONNECT record-name TO set-name	44	record-name	set-name		
08	MODIFY record-name	35	record-name			
11	DISCONNECT record-name FROM set-name	46	record-name	set-name		
02	ERASE record-name	52	record-name			
	ERASE record-name PERMANENT MEMBERS	03	record-name			
	ERASE record-name SELECTIVE MEMBERS	53	record-name			
	ERASE record-name ALL MEMBERS	4	record-name			
	RETRIEVAL STATEMENTS					
03	FIND DB-KEY db-key	75	db-key			
	FIND record-name DB-KEY IS db-key	06	record-name	db-key		
	FIND CURRENT	30				
	FIND CURRENT record-name	07	record-name			

(Continued)

	FIND CURRENT WITHIN set-name	08	set-name			
	FIND CURRENT WITHIN area-name	09	area-name			
	FIND NEXT WITHIN set-name	14	set-name			
	FIND NEXT record-name WITHIN set-name	10	record-name	set-name		
	FIND PRIOR WITHIN set-name	16	set-name			
	FIND PRIOR record-name WITHIN set-name	12	record-name	set-name		
	FIND FIRST WITHIN set-name	20	set-name			
	FIND FIRST record-name WITHIN set-name	18	record-name	set-name		
	FIND LAST WITHIN set-name	24	set-name			
	FIND LAST record-name WITHIN set-name	22	record-name	set-name		
	FIND number WITHIN set-name	78	set-name	number		
	FIND number record-name WITHIN set-name	76	record-name	set-name	number	
	FIND NEXT WITHIN area-name	15	area-name			
	FIND NEXT record-name WITHIN area-name	11	record-name	area-name		
	FIND PRIOR WITHIN area-name	17	area-name			
	FIND PRIOR record-name WITHIN area-name	13	record-name	area-name		
	FIND FIRST WITHIN area-name	21	area-name			
	FIND FIRST record-name WITHIN area-name	19	record-name	area-name		
	FIND LAST WITHIN area-name	25	area-name			
	FIND LAST record-name WITHIN area-name	23	record-name	area-name		
	FIND number WITHIN area-name	79	area-name	number		

(Continued)

	FIND number record-name WITHIN area-name	77	record-name	area-name	number	
	FIND OWNER WITHIN set-name	31	set-name			
	FIND CALC (ANY) record-name	32	record-name			
	FIND DUPLICATE record-name	50	record-name			
	FIND record-name WITHIN set-name USING sort-key	33	record-name	set-name	sort-key	
	FIND record-name WITHIN set-name CURRENT USING sort-key	51	record-name	set-name	sort-key	
	OBTAIN					
	Any of the above FIND record selection expressions. Call generated consists of arguments described above for the FIND in question plus an additional argument of IDBMSCOM(43) function. For example:					
	OBTAIN CALC record-name	32	record-name	IDBMSCOM(43)		
	OBTAIN PRIOR record-name WITHIN set-name	12	record-name			
	KEEP KEEP EXCLUSIVE					
	Any of the above FIND/OBTAIN record selection expressions. Call generated consists of arguments described above for the FIND/OBTAIN in question plus one of the following additional IDBMSCOM functions: KEEP ..... IDBMSCOM(87) KEEP EXCLUSIVE .... IDBMSCOM(88) For example:					
	OBTAIN KEEP CALC record-name	32	record-name	IDBMSCOM(43)	IDBMSCOM(87)	
	FIND KEEP EXCLUSIVE CURRENT	30	IDBMSCOM(88)			
05	GET	43				
	GET record-name	34	record-name			

(Continued)

17	RETURN db-key FROM index-set-name CURRENCY KEY INTO symbolic-key	81	index-set-name	db-key	symbolic-key	
	RETURN db-key FROM index-set-name FIRST KEY INTO symbolic-key	82	index-set-name	db-key	symbolic-key	
	RETURN db-key FROM index-set-name LAST KEY INTO symbolic-key	83	index-set-name	db-key	symbolic-key	
	RETURN db-key FROM index-set-name NEXT KEY INTO symbolic-key	84	index-set-name	db-key	symbolic-key	
	RETURN db-key FROM index-set-name PRIOR KEY INTO symbolic-key	85	index-set-name	db-key	symbolic-key	
	RETURN db-key FROM index-set-name USING index-key-value KEY INTO symbolic-key	86	index-set-name	db-key	index-key-value	symbolic-key
	ACCEPT STATEMENTS					
15	ACCEPT db-key FROM CURRENCY	54	db-key			
	ACCEPT db-key FROM record-name CURRENCY	55	record-name	db-key		
	ACCEPT db-key FROM set-name CURRENCY	57	set-name	db-key		
	ACCEPT db-key FROM area-name CURRENCY	56	area-name	db-key		
	ACCEPT db-key FROM set-name NEXT CURRENCY	68	set-name	db-key		

(Continued)

	ACCEPT db-key FROM set-name PRIOR CURRENCY	69	set-name	db-key		
	ACCEPT db-key FROM set-name OWNER CURRENCY	70	set-name	db-key		
	ACCEPT db-statistics FROM IDMS-STATISTICS	71	db-statistics			
	ACCEPT bind-address FROM record-name BIND	72	record-name	bind-address		
	ACCEPT procedure-control-location FROM procedure-name PROCEDURE	74	procedure-name	procedure-control-location		
	LRF DML STATEMENTS					
20	OBTAIN FIRST logical-record-name	99	subschema-lr-ctrl*	logical-record-location*		
	OBTAIN FIRST logical-record-name INTO alt-logical-record-location	99	subschema-lr-ctrl*	logical-record-location		
	OBTAIN NEXT logical-record-name	99	subschema-lr-ctrl*	logical-record-location*		
	OBTAIN NEXT logical-record-name INTO alt-logical-record-location	99	subschema-lr-ctrl*	logical-record-location		
	MODIFY logical-record-name	99	subschema-lr-ctrl*	logical-record-location*		
	MODIFY logical-record-name FROM alt-logical-record-location	99	subschema-lr-ctrl*	logical-record-location		
	STORE logical-record-name	99	subschema-lr-ctrl*	logical-record-location*		

(Continued)

	STORE logical-record-name FROM alt-logical-record-location	99	subschema-lr-ctrl*	logical-record-location		
	ERASE logical-record-name	99	subschema-lr-ctrl*	logical-record-location*		
	ERASE logical-record-name FROM alt-logical-record-location	99	subschema-lr-ctrl*	logical-record-location		
	To differentiate between the LRF DML statements, the DML compiler places the name of the verb issued into the LRC Block (subschema-lr-ctrl).					

D.3 DC call formats

The DC system passes the DCBMSCOM array as argument 1. Note that the DC system also passes information in the DCSTR, DCFLG, and DCNUM fields of the SUBSCHEMA-CTRL block.

		CALLING ARGUMENTS (argument 0 contains SUBSCHEMA-CTRL)				
Major Function Code	**DC System Service (in COBOL DML)**	**(1) DCBMSCOM (nn)**	**(2)**	**(3)**	**(4)**	**(5)**
	PROGRAM MANAGEMENT STATEMENTS					
30	TRANSFER CONTROL	23	DCFLG1	DCSTR2	parameter	
30	DC RETURN	19				
34	LOAD TABLE	15	01-level-program-location	end-program-location		
34	DELETE TABLE	5	01-level-program-location			
33	SET ABEND EXIT (STAE)	20				
33	ABEND	1				
	STORAGE MANAGEMENT STATEMENTS					
32	GET STORAGE	13	01-level-storage-data-location	end-storage-data location		
32	FREE STORAGE	10	01-level-storage-data-location	start-free-storage-location		
	TASK MANAGEMENT STATEMENTS					
37	ATTACH	3				
37	CHANGE PRIORITY	4				
39	ENQUEUE	9	DCFLG1	DCBMSCOM (mode)	DCBMSCOM (length)	resource-id..
39	DEQUEUE	8	DCFLG1	DCBMSCOM (length)	resource-id..	
31	WAIT	24	ecb			
31	POST	16	ecb			
	TIME MANAGEMENT STATEMENTS					
35	GET TIME	14	return-time	return-date		
35	SET TIMER	21	task-data-location	end-task-data-location		
35	SET TIMER (post)	21	post-ecb			
	SCRATCH MANAGEMENT STATEMENTS					
43	PUT SCRATCH	18	scratch-data-location	end-scratch-data-location		
43	GET SCRATCH	12	return-scratch-data-location	end-scratch-data-location		
43	DELETE SCRATCH	6				

(Continued)

	QUEUE MANAGEMENT STATEMENTS					
44	PUT QUEUE	17	queue-data-location	end-queue-data-location		
44	GET QUEUE	11	return-queue-data-location	end-queue-data-location		
44	DELETE QUEUE	6				
	TERMINAL MANAGEMENT STATEMENTS					
45	READ TERMINAL	30	input-data-location	end-input-data-location		
45	WRITE TERMINAL	30	output-data-location	end-output-data-location		
45	WRITE THEN READ TERMINAL	30	output-data-location	end-output-data-location	input-data-location	end-input-data-location
45	CHECK TERMINAL	31	input-data-location	end-input-data-location		
47	READ LINE FROM TERMINAL	32	input-data-location	end-input-data-location		
47	WRITE LINE TO TERMINAL	32	output-data-location	end-output-data-location		
47	END LINE TERMINAL SESSION	32				
48	WRITE PRINTER	37	message-location	end-message-location		
46	MAP IN (IO)	34	MRB-mapname			
46	MAP IN (NOIO)	34	MRB-mapname	mapped-data-location	end-mapped-data-location	
46	MAP IN (paging) (a)	34	MRB-mapname	data-field-name	sequence-field-name	page-number
46	MAP IN (paging) (b)	34	MRB-mapname	key	page-number	
46	MAP OUT (IO)	34	MRB-mapname	message-text	end-message-data-location OR DCBMSCOM (length)	
46	MAP OUT (NOIO)	34	MRB-mapname	mapped-data-location	end-mapped-data-location	
46	MAP OUT (paging)	34	MRB-mapname	message-text	end-message-data-location OR DCBMSCOM (length)	key
46	MAP OUTIN	34	MRB-mapname	message-text	end-message-data-location OR DCBMSCOM (length)	
46	MODIFY MAP	93	MRB-mapname	MRE	MRB-FLDLST	
46	INQUIRE MAP (a)	92	MRB-mapname	MRE		
46	INQUIRE MAP (b)	92	MRB-mapname			
46	INQUIRE MAP (c)	92	MRB-mapname	MRE		
46	INQUIRE MAP (d)	92	MRB-mapname	MRB-FLDLST		
46	STARTPAGE	40	MRB-mapname			
46	ENDPAGE	41				

(Continued)

	UTILITY STATEMENTS					
48	ACCEPT	2	return-location			
40	SNAP	22	DCSTR1	DCSTR1 (6) begin-dump-location	DCSTR1 (7) end-dump-location	title (8) DCBMSCOM(1)
49	SEND MESSAGE	38	user-id	message-location	end-message-location	
38	BIND TRANSACTION STATISTICS	28				
38	ACCEPT TRANSACTION STATISTICS	28	return-stat-data-location			
38	END TRANSACTION STATISTICS	28	return-stat-data-location			
51	KEEP LONGTERM	29	record-name set-name area-name			
36	WRITE LOG	25	text-return-location	end-text-return-location	reply-location (6) parameter-location	end-reply-location (7) end-parameter-location
	RECOVERY STATEMENTS					
50	COMMIT	66				
50	COMMIT TASK	27				
50	FINISH	2				
50	FINISH TASK	27				
50	ROLLBACK	67				
50	ROLLBACK TASK	27				
50	WRITE JOURNAL	26	record-location	end-record-location		
	DC-BATCH					
14	BIND-TASK	28	DCSTR2			

D.4 Sample COBOL program calling IDMSIN01

The IDMS module contains an IDMSIN01 entry point which provides miscellaneous IDMS functions to user programs. The first two parms passed are always the address of an RPB block and the address of the function REQUEST-CODE and RETURN-CODE fields. The rest of the parms depend on what service is being called.

Currently, there are five IDMSIN01 service functions available to COBOL applications:

- Deactivate the DML trace
- Reactivate the DML trace
- Retrieve user profile information
- Establish user profile information
- Translate an internal 8-byte DATETIME stamp to displayable form

COBOL programs can use standard calling conventions. The following is a brief example on how to call IDMSIN01.

```
 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.


 01  RPB.
     02  FILLER             PIC X(36).

 01  REQ-WK.
     02  REQUEST-CODE       PIC S9(8) COMP.
     02  REQUEST-RETURN     PIC S9(8) COMP.

 01  WORK-FIELDS.
     02  WK-DTS-FORMAT      PIC S9(8) COMP VALUE 0.
     02  WK-CDTS            PIC X(26).
     02  WK-KEYWD           PIC X(8).
     02  WK-VALUE           PIC X(32).
     02  WK-DBNAME          PIC X(8).

 01  USER-WORK-DATA.
     02  WK-SCHEMA              PIC X(18).
     02  WK-CUSER               PIC X(18).
     02  WK-DTS                 PIC X(8).


*********************************************************
 PROCEDURE DIVISION.
*********************************************************


*********************************************************
*  Call IDMSIN01 to deactivate the DML trace
*  which was originally activated by the corresponding
*  SYSIDMS parm (DMLTRACE=ON).
*
*    Parm 1 is the address of the RPB.
*    Parm 2 is the address of the REQUEST-CODE and RETURN-CODE.
*********************************************************

     MOVE 1 TO REQUEST-CODE
     CALL 'IDMSIN01' USING RPB REQ-WK.


*********************************************************
*  Call IDMSIN01 to request a 'GETPROF' to get the user
*  profile default DBNAME, which was established by the
*  SYSIDMS parm DBNAME=xxxxxxxx when running 'Mini CV', or
*  by the DCUF SET DBNAME xxxxxxxx when running under CV.
*
*    Parm 1 is the address of the RPB.
*    Parm 2 is the address of the REQUEST-CODE and RETURN-CODE.
*    Parm 3 is the address of the 8-byte GETPROF keyword.
*    Parm 4 is the address of the 32-byte GETPROF returned value.
*********************************************************

     MOVE 2 TO REQUEST-CODE
     MOVE 'DBNAME' TO WK-KEYWD
```

```
     CALL 'IDMSIN01' USING RPB REQ-WK WK-KEYWD
       WK-VALUE.
     MOVE WK-VALUE TO WK-DBNAME.
     IF WK-DBNAME = SPACES
       DISPLAY 'DBNAME is set to BLANKS'
              ELSE
       DISPLAY 'DBNAME is set to ' WK-DBNAME.


*********************************************************
*  Call IDMSIN01 to request a 'SETPROF' to set the user
*  profile default SCHEMA to the value 'SYSTEM'.
*
*    Parm 1 is the address of the RPB.
*    Parm 2 is the address of the REQUEST-CODE and RETURN-CODE.
*    Parm 3 is the address of the 8-byte SETPROF keyword.
*    Parm 4 is the address of the 32-byte SETPROF value.
*********************************************************

     MOVE 3 TO REQUEST-CODE
     MOVE 'SCHEMA' TO WK-KEYWD
     MOVE 'SYSTEM' TO WK-VALUE
     CALL 'IDMSIN01' USING RPB REQ-WK WK-KEYWD
       WK-VALUE.
     IF REQUEST-RETURN NOT = 0
       DISPLAY 'SETPROF returned error ' REQUEST-RETURN.


*********************************************************
*  Call IDMSIN01 to have an 8-byte internal DATETIME stamp
*  returned as a displayable 26 character DATE/TIME display.
*
*    Parm 1 is the address of the RPB.
*    Parm 2 is the address of the REQUEST-CODE and RETURN-CODE.
*    Parm 3 is the address of the 4-byte format indicator (0's).
*    Parm 4 is the address of the 8-byte internal DATETIME stamp.
*    Parm 5 is the address of the 26-byte DATE/TIME returned.
*********************************************************

     MOVE 5 TO REQUEST-CODE
     MOVE 'UNKNOWN' TO WK-CDTS
     CALL 'IDMSIN01' USING RPB REQ-WK
           WK-DTS-FORMAT WK-DTS WK-CDTS
        DISPLAY WK-SCHEMA ' ' WK-CUSER ' ' WK-CDTS.


*********************************************************
*  Call IDMSIN01 to reactivate the DML trace
*  which was originally activated by the corresponding
*  SYSIDMS parm (DMLTRACE=ON), which had been previously
*  deactivated earlier on in this job.
*
*    Parm 1 is the address of the RPB.
*    Parm 2 is the address of the REQUEST-CODE and RETURN-CODE.
*********************************************************

     MOVE 0 TO REQUEST-CODE
     CALL 'IDMSIN01' USING RPB REQ-WK.
```

Appendix E. CA-IDMS Keywords

E.1 About this appendix E-3
E.2 List of keywords E-4

E.1 About this appendix

This appendix contains a list of keywords recognized by the DML precompiler, including words applicable in the online environment only. All keywords marked with an asterisk are also *reserved* words. Reserved words cannot be used for user-defined element, record, set, paragraph, or area variable names.

Note: The method of parsing used by the IDMSDMLC preprocessor is significantly different in CA-IDMS release 12.0 and later releases from that used in prior releases. The current parsing method looks at individual words in the source code. If it encounters a keyword, it assumes that the keyword should be expanded and tries to do so. Invalid use of reserved words can thus result in either coding errors or syntax errors. For example, if you use FIND as a variable, the parser will try to handle it as the DML verb FIND.

E.2 List of keywords

*ABEND
ABORT
*ACCEPT
AID
ALARM
ALL
ALPHAMERIC
ALWAYS
ANY
AREA
ASSIGN
AT
*ATTACH
ATTRIBUTES
BACKPAGE
BACKSCAN
*BIND
BLINK
BLUE
BRIGHT
BROWSE
BUFFER
BUT
BY
CALC
*CALL
CANCEL
*CHANGE
CHANGED
*CHECK
CLASS
CLEAR
CODE
*COMMIT
COMP
COMP-3
*CONNECT
CONTENTS
CONTINUE
CONTROL
COPIES
*COPY
CORRECT
CURRENCY
CURRENT
CURSOR
DARK
*DATA
DATABASE-KEY
DATASTREAM
DATE
DB
DB-KEY
DBNAME
*DC
DEBUG
INTERNAL
INTERVAL
INTO
INVOKED
IO
IS
JOURNAL
JUSTIFY
*KEEP
KEY
LAST
LEAVE
LEFT
LENGTH
LEVELS
LINE
LINK
*LINKAGE
LIST
LITERALS
*LOAD
LOCK
LOG
LONG
LONGTERM
LR
LSSC-NODN
LTERM
MANUAL
*MAP
MAP-BINDS
MAP-CONTROL
MAP-CONTROLS
MAP-RECORDS
MAPS
MAX
MDT
MEMBER
MEMBERS
MESSAGE
MODE
MODIFIED
*MODIFY
MODULE
MOVE
MRB-FLDLST
NAME
NATIVE
NEWPAGE
NEXT
NLCR
NO
NOALARM
NOBACKPAGE
NOBACKSCAN
NOBLINK
*REMARKS
REPLACE
REPLY
REPORT
REQUIRED
REREAD
RESETKBD
RESETMDT
RESUME
RETENTION
RETURNKEY
RETRIEVAL
RETRY
*RETURN
REVERSE-VIDEO
REVERSED
REWIND
RIGHT
*ROLLBACK
RUN
RUN-UNIT
*SCHEMA
SCRATCH
SCREEN
SCREENSIZE
SECONDS
*SECTION
*SELECT
SELECTIVE
*SEND
SEQUENCE
SEQUENCE-NUMBER
SESSION
*SET
SHARE
SHARED
SHORT
SKIP
SKIP1
SKIP2
SKIP3
SNAP
SOME
SPAN
STANDARD
START
STARTPAGE
STARTPRT
STATISTICS
STGID
*STOP
STORAGE
*STORE
SUBSCHEMA-AREANAMES
SUBSCHEMA-BINDS
SUBSCHEMA-CONTROL

*DECLARATIVES
*DELETE
*DEQUEUE
DEST
DESTINATION
DETAIL
DETECT
DFLD
*DISCONNECT
DISP
DISPLAY
DIVISION
DUMP
DUPLICATE
EAU
ECHO
EDIT
EJECT
EMPTY
*END
ENDPAGE
ENDRPT
*ENQUEUE
*ENTRY
*ENVIRONMENT
*ERASE
ERROR
EVENT
EXCEPT
EXCLUSIVE
EXIT
EXITS
EXTENDED
EXTERNAL
EXTRANEOUS
FIELD
FIELDS
FILE
*FIND
*FINISH
FIRST
FOR
*FREE
FROM
*GET
GREEN
HEADER
HOLD

I-O
*ID
*IDENTIFICATION
IDMS
*IDMS-CONTROL
IDMS-RECORDS
IDMS-STATISTICS
*IF
IGNORED
IN
INCREMENTED
INPUT
*INQUIRE
INTENT

NOCOLOR
NODEADLOCK
NODENAME
NODUMP
NOIO
NOKBD
NOLOCK
NOMDT
NONE
NOPAD
NOPRT
NORETURN
NORMAL
NORMAL-VIDEO
NOSPAN
NOT
*NOTE
NOTIFICATION
NOTIFY
NOUNDERSCORE
NOWAIT
NOWRITE
NULL
NUMERIC
*OBTAIN
OF
OFF
ON
ONLY
*OPEN
OPTIONAL
OUT
OUTIN
OUTPUT
OWNER
PAD
PAGE
PAGING
PARMS
PERMANENT
PINK
POSITION
*POST
PREFIX
PRINTER
PRIOR
PRIORITY
PRIVACY

*PROCEDURE
PROGRAM
*PROGRAM-ID
PROTECTED
PROTOCOL
PTERM
*PUT
QUEUE
*READ
*READY
RECORD
RED
REDISPATCH
RELEASE

SUBSCHEMA-CTRL
SUBSCHEMA-DESCRIPTION
SUBSCHEMA-DML-LR-
DESCRIPTION
SUBSCHEMA-LR-CONTROL
SUBSCHEMA-LR-CTRL
SUBSCHEMA-LR-
DESCRIPTION
SUBSCHEMA-LR-NAMES
SUBSCHEMA-LR-RECORDS
SUBSCHEMA-NAMES
SUBSCHEMA-RECNAMES
SUBSCHEMA-RECORDS
SUBSCHEMA-SETNAMES
SUBSCHEMA-SSNAME
SYSTEM
SYSVERSION
TABLE
TASK
TEMPORARY
TERMINAL
TEST
TEXT
THEN
TIME
TIMEOUT
TIMER
TITLE
TO
TRACE
TRANSACTION
*TRANSFER
TRUNCATED
TURQUOISE
TYPE
UNDERSCORE
UNFORMATTED
UNPROTECTED
UPDATE
UPGRADE
USAGE-MODE
USER
USING
VALUE
VERSION
*WAIT
WCC
WHERE

WHITE
WITH
WITHIN
*WORKING-STORAGE
*WRITE
XCTL
YELLOW
YES
40CR
64CR
80CR

Appendix F. Notes to Teleprocessing Monitor Users

F.1 About this appendix F-3
F.2 TP monitor coding guidelines F-4
F.3 TP monitor coding requirements F-5

F.1 About this appendix

This appendix describes special considerations relating to application programs running under teleprocessing (TP) monitors supported by CA-IDMS (that is, CICS, INTERCOMM, SHADOW, TASK/MASTER, UTM, and WESTI).

F.2 TP monitor coding guidelines

While there are no special coding requirements for TP monitor transactions, the following guidelines should be adhered to:

- DML statements should be coded such that all database requests (for example, BIND, READY, OBTAIN, FINISH) are executed together whenever possible to achieve maximum efficiency and ease of recovery.
- For each TP monitor, you should check with the DBA to determine the operating mode (protocol) installed. The proper mode must then be specified in the IDMS-CONTROL SECTION of the ENVIRONMENT DIVISION.
- For CICS, INTERCOMM, SHADOW, UTM, and WESTI applications, the mode as installed may require the inclusion of additional statements in the IDMS-CONTROL SECTION, WORKING-STORAGE SECTION, and LINKAGE SECTION of each program. These requirements and the applicable modes are outlined in the following table.

Note: The same rules apply to the COPY IDMS statements used to insert logical-record source code components into the program: IDMS-RECORDS MANUAL should be coded in the ENVIRONMENT DIVISION; SUBSCHEMA-LR-NAMES should be copied into the WORKING-STORAGE SECTION; and SUBSCHEMA-CTRL, SUBSCHEMA-LR-CTRL, and SUBSCHEMA-LR-RECORDS should be copied into the LINKAGE-SECTION (except under CICS-EXEC or CICS-EXEC-AUTO, when all required components should be copied into the WORKING-STORAGE SECTION).

- The DML compiler should be executed before the teleprocessing monitor preprocessor.

F.3 TP monitor coding requirements

TP MONITOR	IF MODE IS	IDMS-CONTROL SECTION	WORKING-STORAGE SECTION	LINKAGE SECTION	PROCEDURE DIVISION
CICS	CICS	IDMS-RECORDS MANUAL.	COPY IDMS SUBSCHEMA-NAMES.	*01 TWA 03 FILLER PIC S9(8) COMP SYNC. 03 COPY IDMS SUBSCHEMA-CTRL. 03 COPY IDMS SUBSCHEMA-RECORDS. OR **COPY IDMS SUBSCHEMA-CTRL. COPY IDMS SUBSCHEMA-RECORDS. (A CICS GETMAIN must be issued for the SUBSCHEMA-CTRL and for each record being copied.)	COPY IDMS IDMS-WAIT.
	CICS-EXEC CICS-EXEC-AUTO	IDMS-RECORDS MANUAL.	COPY IDMS SUBSCHEMA-CTRL. COPY IDMS SUBSCEMA-NAMES. COPY IDMS SUBSCHEMA-RECORDS.		
INTERCOMM	INTERCOMM INTERCOMM-AUTO	IDMS-RECORDS MANUAL.	COPY IDMS SUBSCHEMA-NAMES.	COPY IDMS SUBSCHEMA-CTRL. COPY IDMS SUBSCHEMA-RECORDS.	
SHADOW	SHADOW SHAD-AUTOSTATUS	IDMS-RECORDS MANUAL.	COPY IDMS SUBSCHEMA-NAMES.	COPY IDMS SUBSCHEMA-CTRL. COPY IDMS SUBSCHEMA-RECORDS.	
UTM	UTM UTM-AUTOSTATUS	IDMS-RECORDS MANUAL.	COPY IDMS SUBSCHEMA-NAMES.	COPY KCKBC. 05 X PIC S9(8) COMP SYNC. 05 COPY IDMS SUBSCHEMA-CTRL. 05 COPY IDMS SUBSCHEMA-RECORDS. COPY KCPAC.	MOVE LOW-VALUES to SUBSCHEMA-CTRL before each BIND RUN-UNIT.
WESTI	WESTI-REENT WESTI-REENT-AUTO	IDMS-RECORDS MANUAL.	COPY IDMS SUBSCHEMA-NAMES.	COPY IDMS SUBSCHEMA-CTRL. COPY IDMS SUBSCHEMA-RECORDS.	

*If SUBSCHEMA-CTRL, SUBSCHEMA-RECORDS, and additional data does not exceed 4,096 bytes.
**If SUBSCHEMA-CTRL, SUBSCHEMA-RECORDS, and additional data exceeds 4,096 bytes.

Appendix G. EMPLOYEE Database Definition

G.1 About this appendix G-3
G.2 IDMSRPTS utility report listings G-4
G.3 EMPLOYEE database structure diagram G-12

G.1 About this appendix

This appendix contains the IDMSRPTS utility and the data structure diagram for the EMPLOYEE database from which most of the examples in this manual are taken. Both of the sample programs listed earlier in this manual access this database. For more information on the IDMSRPTS utility, refer to *CA-IDMS Utilities*.

G.2 IDMSRPTS utility report listings

```
 IDMSRPTS 14.1                          — SCHEMA RECORD DESCRIPTION LISTING ——                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                          07/09/99  203155     1
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ COVERAGE                                      RLGTH=   36
RECORD VERSION..... 0100                                          DLGTH=   20
RECORD ID.......... 0400                                          KLGTH=   16
RECORD LENGTH...... FIXED                                         DSTRT=   16
LOCATION MODE...... VIA SET      EMP-COVERAGE        DISPLACEMENT 0000 PAGES
WITHIN............. INS-DEMO-REGION     OFFSET       5 PGS FOR        20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    EMP-COVERAGE     MEMBER          1     2     3
                    COVERAGE-CLAIMS  OWNER           4
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 SELECTION-DATE-0400          DISPLAY                              1     8
03 SELECTION-YEAR-0400          DISPLAY                9(4)          1     4
03 SELECTION-MONTH-0400         DISPLAY                9(2)          5     2
03 SELECTION-DAY-0400           DISPLAY                9(2)          7     2
02 TERMINATION-DATE-0400        DISPLAY                              9     8
03 TERMINATION-YEAR-0400        DISPLAY                9(4)          9     4
03 TERMINATION-MONTH-0400       DISPLAY                9(2)         13     2
03 TERMINATION-DAY-0400         DISPLAY                9(2)         15     2
02 TYPE-0400                    DISPLAY                X            17     1
88 MASTER-0400                  COND         'M'                    17
88 FAMILY-0400                  COND         'F'                    17
88 DEPENDENT-0400               COND         'D'                    17
02 INS-PLAN-CODE-0400           DISPLAY                X(3)         18     3
88 GROUP-LIFE-0400              COND         '001'                  18
88 HMO-0400                     COND         '002'                  18
88 GROUP-HEALTH-0400            COND         '003'                  18
88 GROUP-DENTAL-0400            COND         '004'                  18
```

```
 IDMSRPTS 14.1                          — SCHEMA RECORD DESCRIPTION LISTING —                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                         07/09/99  203155     3
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ DENTAL-CLAIM                                  RLGTH=  944
RECORD VERSION..... 0100                                          DLGTH=  936
RECORD ID.......... 0405                                          KLGTH=    8
RECORD LENGTH...... VARIABLE                                      DSTRT=   12
MINIMUM ROOT.......   132  CHARACTERS
MINIMUM FRAGMENT...   932  CHARACTERS
LOCATION MODE...... VIA SET      COVERAGE-CLAIMS     DISPLACEMENT 0000 PAGES
WITHIN............. INS-DEMO-REGION     OFFSE        5 PGS FOR        20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    COVERAGE-CLAIMS  MEMBER          1
                    (FRAGMENT CHAIN) INTRNL          2
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 CLAIM-DATE-0405              DISPLAY                            1     8
03 CLAIM-YEAR-0405              DISPLAY                 9(4)       1     4
03 CLAIM-MONTH-0405             DISPLAY                 9(2)       5     2
03 CLAIM-DAY-0405               DISPLAY                 9(2)       7     2
02 PATIENT-NAME-0405            DISPLAY                            9    25
03 PATIENT-FIRST-NAME-0405      DISPLAY                 X(10)      9    10
03 PATIENT-LAST-NAME-0405       DISPLAY                 X(15)     19    15
02 PATIENT-BIRTH-DATE-0405      DISPLAY                           34     8
03 PATIENT-BIRTH-YEAR-0405      DISPLAY                 9(4)      34     4
03 PATIENT-BIRTH-MONTH-0405     DISPLAY                 9(2)      38     2
03 PATIENT-BIRTH-DAY-0405       DISPLAY                 9(2)      40     2
02 PATIENT-SEX-0405             DISPLAY                 X         42     1
02 RELATION-TO-EMPLOYEE-0405    DISPLAY                 X(10)     43    10
02 DENTIST-NAME-0405            DISPLAY                           53    25
03 DENTIST-FIRST-NAME-0405      DISPLAY                 X(10)     53    10
03 DENTIST-LAST-NAME-0405       DISPLAY                 X(15)     63    15
02 DENTIST-ADDRESS-0405         DISPLAY                           78    46
03 DENTIST-STREET-0405          DISPLAY                 X(20)     78    20
03 DENTIST-CITY-0405            DISPLAY                 X(15)     98    15
03 DENTIST-STATE-0405           DISPLAY                 X(2)     113     2
03 DENTIST-ZIP-0405             DISPLAY                          115     9
04 DENTIST-ZIP-FIRST-FIVE-0405  DISPLAY                 X(5)     115     5
04 DENTIST-ZIP-LAST-FOUR-0405   DISPLAY                 X(4)     120     4
02 DENTIST-LICENSE-NUMBER-0405  DISPLAY                 9(6)     124     6
02 NUMBER-OF-PROCEDURES-0405    COMP                    9(2)     130     2
02 FILLER                       DISPLAY                 X        132     1
02 DENTIST-CHARGES-0405         DISPLAY  OCCURS 0 TO 10          133   800
   DEPENDING ON --- NUMBER-OF-PROCEDURES-0405
03 TOOTH-NUMBER-0405            DISPLAY                 9(2)       1     2
03 SERVICE-DATE-0405            DISPLAY                            3     8
04 SERVICE-YEAR-0405            DISPLAY                 9(4)       3     4
04 SERVICE-MONTH-0405           DISPLAY                 9(2)       7     2
04 SERVICE-DAY-0405             DISPLAY                 9(2)       9     2
03 PROCEDURE-CODE-0405          DISPLAY                 9(4)      11     4
03 DESCRIPTION-OF-SERVICE-0405  DISPLAY                 X(60)     15    60
03 FEE-0405                     COMP-3                  S9(7)V99  75     5
03 FILLER                       DISPLAY                 X         80     1
```

```
 IDMSRPTS 14.1                          — SCHEMA RECORD DESCRIPTION LISTING —                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                         07/09/99  203155     6
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ DEPARTMENT                                    RLGTH=   72
RECORD VERSION..... 0100                                          DLGTH=   56
RECORD ID.......... 0410                                          KLGTH=   16
RECORD LENGTH...... FIXED                                         DSTRT=   16
LOCATION MODE...... CALC USING   DEPT-ID-0410         DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC             MEMBER          1     2
                    DEPT-EMPLOYEE    INDEX OWNER     3     4
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 DEPT-ID-0410                 DISPLAY                 9(4)       1     4
02 DEPT-NAME-0410               DISPLAY                 X(45)      5    45
02 DEPT-HEAD-ID-0410            DISPLAY                 9(4)      50     4
02 FILLER                       DISPLAY                 XXX       54     3
```

```
 IDMSRPTS 14.0                          — SCHEMA RECORD DESCRIPTION LISTING —                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                        07/09/99  203155     7
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ EMPLOYEE                                     RLGTH=  192
RECORD VERSION..... 0100                                         DLGTH=  120
RECORD ID.......... 0415                                         KLGTH=   72
RECORD LENGTH...... FIXED                                        DSTRT=   72
LOCATION MODE...... CALC USING   EMP-ID-0415          DUPLICATES NOT ALLOWED
WITHIN............. EMP-DEMO-REGION     OFFSET        5 PGS FOR       45 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC             MEMBER          1     2
                    DEPT-EMPLOYEE    INDEX MEMBER    3           4
                    EMP-NAME-NDX     INDEX MEMBER    5
                    EMP-SSN-NDX      INDEX MEMBER    6
                    OFFICE-EMPLOYEE  INDEX MEMBER    7           8
                    EMP-COVERAGE     OWNER           9    10
                    EMP-EMPOSITION   OWNER          11    12
                    EMP-EXPERTISE    OWNER          13    14
                    MANAGES          OWNER          15    16
                    REPORTS-TO       OWNER          17    18
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 EMP-ID-0415                  DISPLAY                 9(4)        1     4
02 EMP-NAME-0415                DISPLAY                             5    25
03 EMP-FIRST-NAME-0415          DISPLAY                 X(10)       5    10
03 EMP-LAST-NAME-0415           DISPLAY                 X(15)      15    15
02 EMP-ADDRESS-0415             DISPLAY                            30    46
03 EMP-STREET-0415              DISPLAY                 X(20)      30    20
03 EMP-CITY-0415                DISPLAY                 X(15)      50    15
03 EMP-STATE-0415               DISPLAY                 X(2)       65     2
03 EMP-ZIP-0415                 DISPLAY                            67     9
04 EMP-ZIP-FIRST-FIVE-0415      DISPLAY                 X(5)       67     5
04 EMP-ZIP-LAST-FOUR-0415       DISPLAY                 X(4)       72     4
02 EMP-PHONE-0415               DISPLAY                 9(10)      76    10
02 STATUS-0415                  DISPLAY                 X(2)       86     2
88 ACTIVE-0415                  COND         '01'                  86
88 ST-DISABIL-0415              COND         '02'                  86
88 LT-DISABIL-0415              COND         '03'                  86
88 LEAVE-OF-ABSENCE-0415        COND         '04'                  86
88 TERMINATED-0415              COND         '05'                  86
02 SS-NUMBER-0415               DISPLAY                 9(9)       88     9
02 START-DATE-0415              DISPLAY                            97     8
03 START-YEAR-0415              DISPLAY                 9(4)       97     4
03 START-MONTH-0415             DISPLAY                 9(2)      101     2
03 START-DAY-0415               DISPLAY                 9(2)      103     2
02 TERMINATION-DATE-0415        DISPLAY                           105     8
03 TERMINATION-YEAR-0415        DISPLAY                 9(4)      105     4
03 TERMINATION-MONTH-0415       DISPLAY                 9(2)      109     2
03 TERMINATION-DAY-0415         DISPLAY                 9(2)      111     2
02 BIRTH-DATE-0415              DISPLAY                           113     8
03 BIRTH-YEAR-0415              DISPLAY                 9(4)      113     4
03 BIRTH-MONTH-0415             DISPLAY                 9(2)      117     2
03 BIRTH-DAY-0415               DISPLAY                 9(2)      119     2
```

```
 IDMSRPTS 14.1                          — SCHEMA RECORD DESCRIPTION LISTING —                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                         07/09/99  203155     8
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ EMPOSITION                                   RLGTH=   56
RECORD VERSION..... 0100                                         DLGTH=   32
RECORD ID.......... 0420                                         KLGTH=   24
RECORD LENGTH...... FIXED                                        DSTRT=   24
LOCATION MODE...... VIA SET      EMP-EMPOSITION     DISPLACEMENT 0000  PAGES
WITHIN............. EMP-DEMO-REGION     OFFSET      5 PGS FOR         45 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    EMP-EMPOSITION   MEMBER          1     2     3
                    JOB-EMPOSITION   MEMBER          4     5     6
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 START-DATE-0420              DISPLAY                            1     8
03 START-YEAR-0420              DISPLAY                 9(4)       1     4
03 START-MONTH-0420             DISPLAY                 9(2)       5     2
03 START-DAY-0420               DISPLAY                 9(2)       7     2
02 FINISH-DATE-0420             DISPLAY                            9     8
03 FINISH-YEAR-0420             DISPLAY                 9(4)       9     4
03 FINISH-MONTH-0420            DISPLAY                 9(2)      13     2
03 FINISH-DAY-0420              DISPLAY                 9(2)      15     2
02 SALARY-GRADE-0420            DISPLAY                 9(2)      17     2
02 SALARY-AMOUNT-0420           COMP-3                  S9(7)V99  19     5
02 BONUS-PERCENT-0420           COMP-3                  SV999     24     2
02 COMMISSION-PERCENT-0420      COMP-3                  SV999     26     2
02 OVERTIME-RATE-0420           COMP-3                  S9V99     28     2
02 FILLER                       DISPLAY                 XXX       30     3
```

```
 IDMSRPTS 14.1                          — SCHEMA RECORD DESCRIPTION LISTING —                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                         07/09/99  203155    10
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ EXPERTISE                                    RLGTH=   32
RECORD VERSION..... 0100                                         DLGTH=   12
RECORD ID.......... 0425                                         KLGTH=   20
RECORD LENGTH...... FIXED                                        DSTRT=   20
LOCATION MODE...... VIA SET      EMP-EXPERTIS       DISPLACEMENT 0000  PAGES
WITHIN............. EMP-DEMO-REGION     OFFSET      5 PGS FOR         45 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    EMP-EXPERTISE    MEMBER          1     2     3
                    SKILL-EXPERTISE  INDEX MEMBER    4           5
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 SKILL-LEVEL-0425             DISPLAY                 XX         1     2
88 EXPERT-0425                  COND         '04'                  1
88 PROFICIENT-0425              COND         '03'                  1
88 COMPETENT-0425               COND         '02'                  1
88 ELEMENTARY-0425              COND         '01'                  1
02 EXPERTISE-DATE-0425          DISPLAY                            3     8
03 EXPERTISE-YEAR-0425          DISPLAY                 9(4)       3     4
03 EXPERTISE-MONTH-0425         DISPLAY                 9(2)       7     2
03 EXPERTISE-DAY-0425           DISPLAY                 9(2)       9     2
02 FILLER                       DISPLAY                 XX        11     2
```

```
 IDMSRPTS 14.1                          — SCHEMA RECORD DESCRIPTION LISTING —                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                        07/09/99  203155    10
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ HOSPITAL-CLAIM                                RLGTH=  304
RECORD VERSION..... 0100                                          DLGTH=  300
RECORD ID.......... 0430                                          KLGTH=    4
RECORD LENGTH...... FIXED                                         DSTRT=    4
LOCATION MODE...... VIA SET      COVERAGE-CLAIMS    DISPLACEMENT 0000  PAGES
WITHIN............. INS-DEMO-REGION     OFFSET      5 PGS FOR         20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    COVERAGE-CLAIMS  MEMBER          1
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 CLAIM-DATE-0430               DISPLAY                            1     8
03 CLAIM-YEAR-0430               DISPLAY                 9(4)       1     4
03 CLAIM-MONTH-0430              DISPLAY                 9(2)       5     2
03 CLAIM-DAY-0430                DISPLAY                 9(2)       7     2
02 PATIENT-NAME-0430             DISPLAY                            9    25
03 PATIENT-FIRST-NAME-0430       DISPLAY                 X(10)      9    10
03 PATIENT-LAST-NAME-0430        DISPLAY                 X(15)     19    15
02 PATIENT-BIRTH-DATE-0430       DISPLAY                           34     8
03 PATIENT-BIRTH-YEAR-0430       DISPLAY                 9(4)      34     4
03 PATIENT-BIRTH-MONTH-0430      DISPLAY                 9(2)      38     2
03 PATIENT-BIRTH-DAY-0430        DISPLAY                 9(2)      40     2
02 PATIENT-SEX-0430              DISPLAY                 X         42     1
02 RELATION-TO-EMPLOYEE-0430     DISPLAY                 X(10)     43    10
02 HOSPITAL-NAME-0430            DISPLAY                 X(25)     53    25
02 HOSP-ADDRESS-0430             DISPLAY                           78    46
03 HOSP-STREET-0430              DISPLAY                 X(20)     78    20
03 HOSP-CITY-0430                DISPLAY                 X(15)     98    15
03 HOSP-STATE-0430               DISPLAY                 X(2)     113     2
03 HOSP-ZIP-0430                 DISPLAY                          115     9
04 HOSP-ZIP-FIRST-FIVE-0430      DISPLAY                 X(5)     115     5
04 HOSP-ZIP-LAST-FOUR-0430       DISPLAY                 X(4)     120     4
02 ADMIT-DATE-0430               DISPLAY                          124     8
03 ADMIT-YEAR-0430               DISPLAY                 9(4)     124     4
03 ADMIT-MONTH-0430              DISPLAY                 9(2)     128     2
03 ADMIT-DAY-0430                DISPLAY                 9(2)     130     2
02 DISCHARGE-DATE-0430           DISPLAY                          132     8
03 DISCHARGE-YEAR-0430           DISPLAY                 9(4)     132     4
03 DISCHARGE-MONTH-0430          DISPLAY                 9(2)     136     2
03 DISCHARGE-DAY-0430            DISPLAY                 9(2)     138     2
02 DIAGNOSIS-0430                DISPLAY  OCCURS 2       X(60)    140   120
02 HOSPITAL-CHARGES-0430         DISPLAY                          260    41
03 ROOM-AND-BOARD-0430           DISPLAY                          260    26
04 WARD-0430                     DISPLAY                          260    13
05 WARD-DAYS-0430                COMP-3                  S9(5)    260     3
05 WARD-RATE-0430                COMP-3                  S9(7)V99 263     5
05 WARD-TOTAL-0430               COMP-3                  S9(7)V99 268     5
04 SEMI-PRIVATE-0430             DISPLAY                          273    13
05 SEMI-DAYS-0430                COMP-3                  S9(5)    273     3
05 SEMI-RATE-0430                COMP-3                  S9(7)V99 276     5
05 SEMI-TOTAL-0430               COMP-3                  S9(7)V99 281     5
03 OTHER-CHARGES-0430            DISPLAY                          286    15
04 DELIVERY-COST-0430            COMP-3                  S9(7)V99 286     5
04 ANESTHESIA-COST-0430          COMP-3                  S9(7)V99 291     5
04 LAB-COST-0430                 COMP-3                  S9(7)V99 296     5
```

```
 IDMSRPTS 14.1                          ── SCHEMA RECORD DESCRIPTION LISTING ──                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                          07/09/99  203155    12
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ INSURANCE-PLAN                                RLGTH=  140
RECORD VERSION..... 0100                                          DLGTH=  132
RECORD ID.......... 0435                                          KLGTH=    8
RECORD LENGTH...... FIXED                                         DSTRT=    8
LOCATION MODE...... CALC USING   INS-PLAN-CODE-0435   DUPLICATES NOT ALLOWED
WITHIN............. INS-DEMO-REGION     OFFSET        1 PGS FOR        4 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC             MEMBER          1     2
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 INS-PLAN-CODE-0435            DISPLAY                X(3)        1     3
88 GROUP-LIFE-0435               COND         '001'                  1
88 HMO-0435                      COND         '002'                  1
88 GROUP-HEALTH-0435             COND         '003'                  1
88 GROUP-DENTAL-0435             COND         '004'                  1
02 INS-CO-NAME-0435              DISPLAY                X(45)       4    45
02 INS-CO-ADDRESS-0435           DISPLAY                            49    46
03 INS-CO-STREET-0435            DISPLAY                X(20)      49    20
03 INS-CO-CITY-0435              DISPLAY                X(15)      69    15
03 INS-CO-STATE-0435             DISPLAY                X(2)       84     2
03 INS-CO-ZIP-0435               DISPLAY                            86     9
04 INS-CO-ZIP-FIRST-FIVE-0435    DISPLAY                X(5)       86     5
04 INS-CO-ZIP-LAST-FOUR-0435     DISPLAY                X(4)       91     4
02 INS-CO-PHONE-0435             DISPLAY                9(10)      95    10
02 GROUP-NUMBER-0435             DISPLAY                9(6)      105     6
02 PLAN-DESCRIPTION-0435         DISPLAY                          111    20
03 DEDUCT-0435                   COMP-3                 S9(7)V99  111     5
03 MAXIMUM-LIFE-COST-0435        COMP-3                 S9(7)V99  116     5
03 FAMILY-COST-0435              COMP-3                 S9(7)V99  121     5
03 DEP-COST-0435                 COMP-3                 S9(7)V99  126     5
02 FILLER                        DISPLAY                XX        131     2
```

```
 IDMSRPTS 14.1                          ── SCHEMA RECORD DESCRIPTION LISTING ──                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                          07/09/99  203155    14
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ JOB                                           RLGTH=  324
RECORD VERSION..... 0100                                          DLGTH=  300
RECORD ID.......... 0440                                          KLGTH=   24
RECORD LENGTH...... FIXED  (INTERNALLY VARIABLE)                  DSTRT=   28
MINIMUM ROOT.......    24  CHARACTERS
MINIMUM FRAGMENT...   296  CHARACTERS
LOCATION MODE...... CALC USING   JOB-ID-0440          DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
CALL PROCEDURES.... NAME.... WHEN.. FUNCTION
                    IDMSCOMP BEFORE STORE
                    IDMSCOMP BEFORE MODIFY
                    IDMSDCOM AFTER  GET
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC             MEMBER          1     2
                    JOB-TITLE-NDX    INDEX MEMBER    3
                    JOB-EMPOSITION   OWNER           4     5
                    (FRAGMENT CHAIN) INTRNL          6
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 JOB-ID-0440                   DISPLAY                9(4)        1     4
02 TITLE-0440                    DISPLAY                X(20)       5    20
02 DESCRIPTION-0440              DISPLAY                            25   120
03 DESCRIPTION-LINE-0440         DISPLAY  OCCURS 2      X(60)      25   120
02 REQUIREMENTS-0440             DISPLAY                           145   120
03 REQUIREMENT-LINE-0440         DISPLAY  OCCURS 2      X(60)     145   120
02 MINIMUM-SALARY-0440           DISPLAY                S9(6)V99  265     8
02 MAXIMUM-SALARY-0440           DISPLAY                S9(6)V99  273     8
02 SALARY-GRADES-0440            DISPLAY  OCCURS 4      9(2)      281     8
02 NUMBER-OF-POSITIONS-0440      DISPLAY                9(3)      289     3
02 NUMBER-OPEN-0440              DISPLAY                9(3)      292     3
02 FILLER                        DISPLAY                XX        295     2
```

```
 IDMSRPTS 14.1                          — SCHEMA RECORD DESCRIPTION LISTING —                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                        07/09/99  203155    15
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ NON-HOSP-CLAIM                                RLGTH= 1064
RECORD VERSION..... 0100                                          DLGTH= 1056
RECORD ID.......... 0445                                          KLGTH=    8
RECORD LENGTH...... VARIABLE                                      DSTRT=   12
MINIMUM ROOT.......   248  CHARACTERS
MINIMUM FRAGMENT...  1052  CHARACTERS
LOCATION MODE...... VIA SET      COVERAGE-CLAIMS    DISPLACEMENT 0000  PAGES
WITHIN............. INS-DEMO-REGION     OFFSET      5 PGS FOR         20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    COVERAGE-CLAIMS  MEMBER          1
                    (FRAGMENT CHAIN) INTRNL          2
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 CLAIM-DATE-0445               DISPLAY                            1     8
03 CLAIM-YEAR-0445               DISPLAY                 9(4)       1     4
03 CLAIM-MONTH-0445              DISPLAY                 9(2)       5     2
03 CLAIM-DAY-0445                DISPLAY                 9(2)       7     2
02 PATIENT-NAME-0445             DISPLAY                            9    25
03 PATIENT-FIRST-NAME-0445       DISPLAY                 X(10)      9    10
03 PATIENT-LAST-NAME-0445        DISPLAY                 X(15)     19    15
02 PATIENT-BIRTH-DATE-0445       DISPLAY                           34     8
03 PATIENT-BIRTH-YEAR-0445       DISPLAY                 9(4)      34     4
03 PATIENT-BIRTH-MONTH-0445      DISPLAY                 9(2)      38     2
03 PATIENT-BIRTH-DAY-0445        DISPLAY                 9(2)      40     2
02 PATIENT-SEX-0445              DISPLAY                 X         42     1
02 RELATION-TO-EMPLOYEE-0445     DISPLAY                 X(10)     43    10
02 PHYSICIAN-NAME-0445           DISPLAY                           53    25
03 PHYSICIAN-FIRST-NAME-0445     DISPLAY                 X(10)     53    10
03 PHYSICIAN-LAST-NAME-0445      DISPLAY                 X(15)     63    15
02 PHYSICIAN-ADDRESS-0445        DISPLAY                           78    46
03 PHYSICIAN-STREET-0445         DISPLAY                 X(20)     78    20
03 PHYSICIAN-CITY-0445           DISPLAY                 X(15)     98    15
03 PHYSICIAN-STATE-0445          DISPLAY                 X(2)     113     2
03 PHYSICIAN-ZIP-0445            DISPLAY                          115     9
04 PHYSICIAN-ZIP-FIRST-FIVE-0445 DISPLAY                 X(5)     115     5
04 PHYSICIAN-ZIP-LAST-FOUR-0445  DISPLAY                 X(4)     120     4
02 PHYSICIAN-ID-0445             DISPLAY                 9(6)     124     6
02 DIAGNOSIS-0445                DISPLAY  OCCURS 2       X(60)    130   120
02 NUMBER-OF-PROCEDURES-0445     COMP                    9(2)     250     2
02 FILLER                        DISPLAY                 X        252     1
02 PHYSICIAN-CHARGES-0445        DISPLAY  OCCURS 0 TO 10          253   800
   DEPENDING ON -- NUMBER-OF-PROCEDURES-0445
03 SERVICE-DATE-0445             DISPLAY                            1     8
04 SERVICE-YEAR-0445             DISPLAY                 9(4)       1     4
04 SERVICE-MONTH-0445            DISPLAY                 9(2)       5     2
04 SERVICE-DAY-0445              DISPLAY                 9(2)       7     2
03 PROCEDURE-CODE-0445           DISPLAY                 9(4)       9     4
03 DESCRIPTION-OF-SERVICE-0445   DISPLAY                 X(60)     13    60
03 FEE-0445                      COMP-3                  S9(7)V99  73     5
03 FILLER                        DISPLAY                 XXX       78     3
```

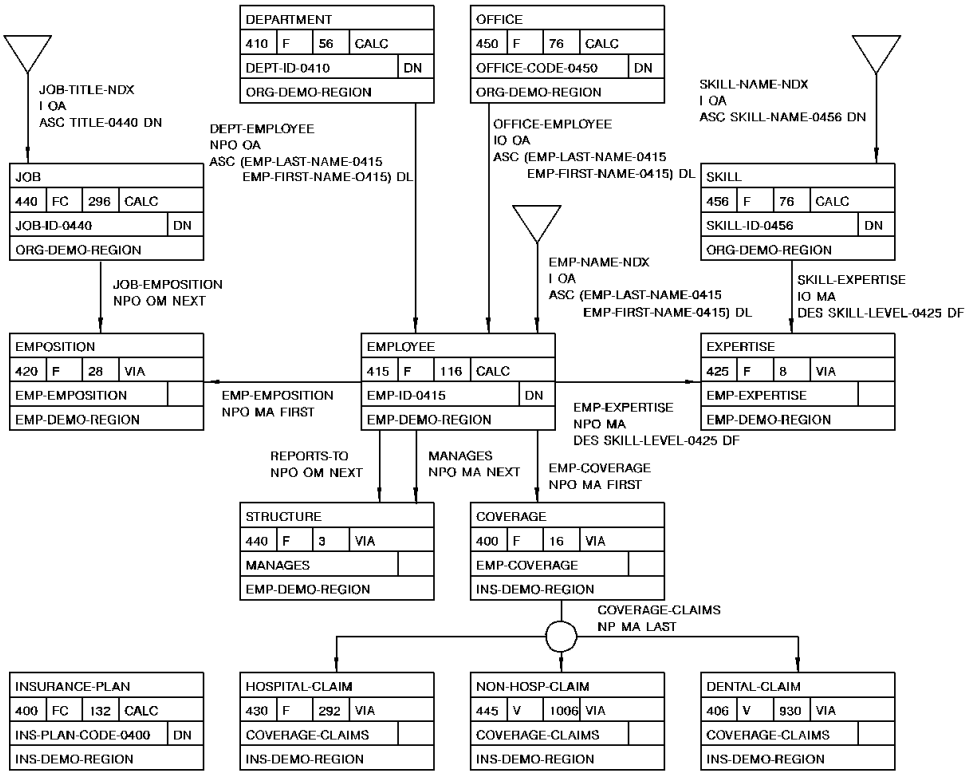
```
 IDMSRPTS 14.1                          ── SCHEMA RECORD DESCRIPTION LISTING ──                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                          07/09/99  203155    18
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ OFFICE                                      RLGTH=   92
RECORD VERSION..... 0100                                        DLGTH=   76
RECORD ID.......... 0450                                        KLGTH=   16
RECORD LENGTH...... FIXED                                       DSTRT=   16
LOCATION MODE...... CALC USING   OFFICE-CODE-0450     DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC            MEMBER          1     2
                    OFFICE-EMPLOYEE  INDEX OWNER     3     4
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 OFFICE-CODE-0450             DISPLAY                 X(3)        1     3
02 OFFICE-ADDRESS-0450          DISPLAY                             4    46
03 OFFICE-STREET-0450           DISPLAY                 X(20)       4    20
03 OFFICE-CITY-0450             DISPLAY                 X(15)      24    15
03 OFFICE-STATE-0450            DISPLAY                 X(2)       39     2
03 OFFICE-ZIP-0450              DISPLAY                            41     9
04 OFFICE-ZIP-FIRST-FIVE-0450   DISPLAY                 X(5)       41     5
04 OFFICE-ZIP-LAST-FOUR-0450    DISPLAY                 X(4)       46     4
02 OFFICE-PHONE-0450            DISPLAY  OCCURS 3       9(7)       50    21
02 OFFICE-AREA-CODE-0450        DISPLAY                 X(3)       71     3
02 SPEED-DIAL-0450              DISPLAY                 X(3)       74     3
```

```
 IDMSRPTS 14.1                          ── SCHEMA RECORD DESCRIPTION LISTING ──                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                          07/09/99  203155    20
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ SKILL                                       RLGTH=   96
RECORD VERSION..... 0100                                        DLGTH=   76
RECORD ID.......... 0455                                        KLGTH=   20
RECORD LENGTH...... FIXED                                       DSTRT=   20
LOCATION MODE...... CALC USING   SKILL-ID-0455        DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC            MEMBER          1     2
                    SKILL-NAME-NDX   INDEX MEMBER    3
                    SKILL-EXPERTISE  INDEX OWNER     4     5
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 SKILL-ID-0455                DISPLAY                 9(4)        1     4
02 SKILL-NAME-0455              DISPLAY                 X(12)       5    12
02 SKILL-DESCRIPTION-0455       DISPLAY                 X(60)      17    60
```

```
 IDMSRPTS 14.1                          ── SCHEMA RECORD DESCRIPTION LISTING ──                          DATE     TIME   PAGE
 RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                          07/09/99  203155    22
                                              SCHEMA EMPSCHM  VERSION  100


RECORD NAME........ STRUCTURE                                   RLGTH=   36
RECORD VERSION..... 0100                                        DLGTH=   12
RECORD ID.......... 0460                                        KLGTH=   24
RECORD LENGTH...... FIXED                                       DSTRT=   24
LOCATION MODE...... VIA SET      MANAGES            DISPLACEMENT 0000  PAGES
WITHIN............. EMP-DEMO-REGION     OFFSET      5 PGS FOR         45 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    MANAGES         MEMBER          1     2     3
                    REPORTS-TO      MEMBER          4     5     6
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 STRUCTURE-CODE-0460          DISPLAY                 X(2)        1     2
88 ADMIN-0460                   COND 'A'                            1
88 PROJECT-0460                 COND 'P1' THRU 'P9'                 1
02 STRUCTURE-DATE-0460          DISPLAY                             3     8
03 STRUCTURE-YEAR-0460          DISPLAY                 9(4)        3     4
03 STRUCTURE-MONTH-0460         DISPLAY                 9(2)        7     2
03 STRUCTURE-DAY-0460           DISPLAY                 9(2)        9     2
02 FILLER                       DISPLAY                 XX         11     2
```

G.3 EMPLOYEE database structure diagram

The data structure diagram illustrates record relationships in the EMPLOYEE database.



Appendix H. VS COBOL II Support

H.1 About this appendix H-3
H.2 Features supported by CA-IDMS H-4
H.3 Features not supported by CA-IDMS H-7

H.1 About this appendix

This appendix discusses CA-IDMS support for programs compiled under the VS COBOL II compiler. It is divided into two parts:

- Features of VS COBOL II that are supported by CA-IDMS
- Features of VS COBOL II that are not supported by CA-IDMS

Note: All the provisions of this Appendix also apply to programs compiled under an LE-compliant compiler, unless otherwise noted. See Appendix I, "Considerations for IBM Language Environment" on page I-1, for a further discussion of the IBM Language Environment and LE-compliant compilers.

H.2 Features supported by CA-IDMS

The following COBOL II features are supported by CA-IDMS:

- **Reentrancy** — Fully reentrant and non-reentrant programs are supported. The RENT compiler option must be specified if the program is reentrant. NORENT must be specified if the program is non-reentrant.

  Note: Quasi-reentrancy is not supported for VS COBOL II programs. It is strongly recommended that all COBOL II programs be compiled with the RENT option. A separate copy of each NORENT COBOL II program will be loaded for each concurrent task. CPU and storage utilization will be extremely high.

- **Residency** — Resident and nonresident programs are supported. The NORES compiler option causes all necessary VS COBOL II runtime support modules to be linked with the program. The program can then be executed without having to load any support modules. The RES option causes the runtime support modules to be brought in as needed during execution.

  The following combinations of RENT and RES options are supported:

  – RENT RES

  – NORENT NORES

  – NORENT RES

  Note: 31-bit programs require the RENT and RES options. This combination is recommended for most efficent processing. The RES option is not relevant to LE-compliant compilers, which always use this option.

  The RENT/NORES combination is not allowed by the VS COBOL II compiler.

  Do not confuse the RES compiler option with the CA-IDMS RESIDENT parameter (assigned at SYSGEN or by using a DCMT command). The CA-IDMS RESIDENT parameter causes the user program to be loaded into the resident pool during startup, and remains there for the duration of system execution.

- **XA support** — Full 31-bit support is provided. COBOL II programs can reside above or below the 16-meg line, and can execute in 24-bit or 31-bit mode. User data areas can reside below the 16-meg line (compiler option DATA(24)) or anywhere in the region (DATA(31)). The following table shows the default attributes assigned by the COBOL II compiler based on the combination of RES and RENT compiler options.

RMODE and AMODE attributes

Compiler options	Default RMODE/AMODE
RES/RENT	RMODE(ANY) AMODE(ANY)
RES/NORENT	RMODE(24) AMODE(ANY)

Compiler options	Default RMODE/AMODE
NORES/NORENT	RMODE(24) AMODE(24)

To run a task in 31-bit mode, it must be defined with a LOCATION of ANY (at SYSGEN or at runtime using a DCMT VARY DYNAMIC PROGRAM command).

- **Static and dynamic calls** — CA-IDMS supports the following types of calls provided by VS COBOL II:
  - CALL literal with NODYNAM, static (more storage, less CPU)
  - CALL identifier, dynamic (less storage, more CPU)

  To call a program dynamically you must use the call identifier format.

  A VS COBOL II program can use the COBOL CALL verb to invoke an assembler or COBOL II subprogram. The CA-IDMS TRANSFER CONTROL (LINK or XCTL) must be used for invoking VS COBOL subprograms. The subprogram must be defined to the system either at SYSGEN or by using a DCMT VARY DYNAMIC PROGRAM command. The correct language must be specified, and the NONOVERLAYABLE attribute must also be specified.

  A COBOL II program and all the COBOL II subprograms that it calls dynamically must be compiled with the same RES/NORES compiler option.

  A dynamic call is often a more efficient way for one online VS-COBOL II program to call another than the use of a TRANSFER CONTROL DML command. Note, however, that when a dynamic call is made, the CA-IDMS/DC/UCF system is not aware that the application is running in a new program. Therefore, error messages and program statistics will not reflect the call.

  There are also restrictions on using static or dynamic calls when invoking an assembler subprogram. If the assembler program is not fully reentrant or if the assembler program issues any operating system SVC instructions, the program must be invoked with a DC TRANSFER control statement. Note that use of SVC instructions in an online program presents security and performance concerns. Such instructions should be avoided unless they are absolutely necessary. In most cases, CA-IDMS/DC/UCF system functions can be used instead.

  Note: Also see "Performance improvements with RHDCLEFE" in Appendix I, “Considerations for IBM Language Environment” on page I-1.

- **Optimizer** — The COBOL II optimizer is fully supported. Service reloads do not have to be explicitly coded in the program, as is required for VS COBOL.

- **STRING/UNSTRING/INSPECT** — COBOL II verbs that require GETMAIN services are supported; this includes STRING, UNSTRING, and INSPECT.

  Note: Exercise caution with STRING, UNSTRING, and INSPECT, using them may increase SRB time.

- **Compiler options** — The following compiler options that affect object code execution are supported.

- DATA
- OPTIMIZE
- PFDSGN
- RENT
- RESIDENT
- SSRANGE
- TRUNC
- ZWB

- **Execution time options** — COBOL II has an options module (IGZEOPT) that can be assembled and link-edited to control options at execution time. The module needs to be linked with each online VS-COBOL II application program. Valid macro values for parameters that affect IDMS performance are shown below:

```
IGZOPT SYSTYPE=OS,
       DEBUG=NO,
       STAE=NO,
       AIXBLD=NO,
       SSRANGE=YES/NO,
       SPOUT=YES/NO
```

- **Space management tuning** — The tuning table (macro IGZTUNE) can be used to define application-specific space management tuning factors. If SPOUT=YES is specified in IGZOPT, space usage statistics are written out to the log. SPOUT should be turned on only for the duration of the space usage measurement. COBOL II programs are typically not run with this option in a production environment.

  Note: SPOUT=YES is not valid for COBOL II programs running under CA-IDMS/DC in an IBM Language Environment runtime system.

- **COBPACKs** — Frequently used COBOL II runtime support modules can be linked together into COBPACKs to improve performance. The two COBPACKs are:
  - IGZCPAC — general subroutines
  - IGZCPCO — OS-specific modules

H.3 Features not supported by CA-IDMS

The following COBOL II features are not supported by CA-IDMS:

- ENVIRONMENT and DATA DIVISION entries normally associated with file management (for example, INPUT-OUTPUT SECTION, FILE SECTION)
- I/O statements, including ACCEPT, CLOSE, DELETE, DISPLAY, OPEN, READ, REWRITE, and WRITE
- The debugging features FDUMP and TEST
- The sorting features SORT and MERGE
- The compiler options ADV, DYNAM, FASTSRT, GRAPHIC, and OUTDD

Appendix I. Considerations for IBM Language Environment

I.1 Overview I-3
I.2 Considerations about LE runtime I-4
I.3 Running LE-compliant compiler programs under CA-IDMS/DC I-6
I.4 Supported LE functions I-10
I.5 Unsupported LE functions I-11
I.6 Performance improvements with RHDCLEFE I-12

I.1 Overview

Note: This section applies only to runtime support for COBOL programs that run in an online CA-IDMS/DC/UCF region. It does not apply to batch or CICS programs that access CA-IDMS. It also does not apply to online COBOL programs compiled with the "old" VS-COBOL compiler, prior to VS-COBOL II. Online VS-COBOL programs must comply with the compile and linkage restrictions described below. If these restrictions are observed, the LE runtime environment will not be accessed by VS-COBOL programs.

What is IBM Language Environment (LE)?: LE is a runtime environment that replaces the language-specific runtime environments that existed previously. For example, COBOL had its own runtime environment; COBOL II had another. CA-IDMS can execute programs that are designed to use the LE runtime environment. It can also execute programs compiled with pre-LE compilers that use the LE runtime environment.

Language Environment has had several names for different operating systems and release levels. The term "LE" will be used in this document to refer to any of the following, unless otherwise noted:

- LE/370
- LE for MVS and VM
- LE for OS/390 and VM
- LE/VSE

Note: This section applies only to runtime support in CA-IDMS/DC. It does not apply to batch or CICS programs that access CA-IDMS.

How can you use LE with CA-IDMS/DC?: To execute online programs using the LE runtime libraries, follow these steps to bring up your CA-IDMS environment:

1. Ensure that the CA-IDMS system has been generated with a 24-bit reentrant pool (or program pool, if no reentrant pool is generated) that is large enough to contain the IBM-supplied LE application program interface module CEEPIPI. The size of this module is approximately 100K.
2. Ensure that the CA-IDMS system has been generated with an XA reentrant pool that is large enough to maintain residence for several IBM-supplied LE support modules. Allow 1 megabyte for these programs.
3. Include the LE runtime load libraries in the CDMSLIB loadlib concatenation before any other IBM language loadlibs you are using. For example, before COBOL II.

I.2 Considerations about LE runtime

Running pre-LE programs: There are restrictions that apply when you run pre-LE programs in an LE runtime environment within CA-IDMS/DC. *Pre-LE programs* are programs that were compiled with a non-LE compliant compiler, such as COBOL II V4.0.

Some of these restrictions are already documented in Chapter 1, “Introduction to CA-IDMS Data Manipulation Language” on page 1-1 and Appendix H, “VS COBOL II Support” on page H-1. Additional restrictions for LE are:

- VS COBOL II programs have to run without storage protection unless RHDCLEFE (see "Performance improvements with RHDCLEFE" below) is in use.
- VS COBOL II programs must be linked with an IGZEOPT module that specifies STAE=NO (see "Execution time options" in Appendix H, “VS COBOL II Support” on page H-1, for more information on the use of IGZEOPT). If this restriction is not observed, a program check in a COBOL program will result in immediate termination of the program with no indication of an error. Certain other abnormal abend conditions may also go unreported. This restriction does not apply if one of the following conditions is true:
  - RHDCLEFE is in use. See Performance Improvements with RHDCLEFE later in this appendix for more information.
  - A special CEEDOPT is in use as described later in this appendix under Runtime Options, and either or both of the following options is specified:

    ```
    ABTERMENC=((ABEND,...
    TRAP=((OFF,...
    ```
- The IBM LE support module CEEPIPI must be loaded once before any VS COBOL II program is run. This is most easily done by defining CEEPIPI as RESIDENT in the CA-IDMS/DC sysgen using the following syntax.

  ```
  ADD PROGRAM CEEPIPI CONCURRENT ENABLED LANGUAGE ASSEMBLER
  NONOVERLAYABLE PROGRAM PROTECT REENTRANT RESIDENT REUSABLE .
  ```
- Restrictions mentioned in the IBM documentation (for example, the *IBM COBOL/370 Migration Guide*) apply.

Note: Running pre-LE programs with LE runtime can degrade performance in some circumstances. If you notice poor performance, you should consider recompiling the programs with the newer compiler or running with RHDCLEFE (see "Performance improvements with RHDCLEFE" below). The use of RHDCLEFE also removes the necessity of forcing the load of CEEPIPI before running any VS COBOL II programs.

Running LE programs: *LE programs* are programs that were compiled with an LE-compliant compiler. CA-IDMS/DC supports these LE-compliant compilers:

- COBOL/370 V1R1
- IBM COBOL for MVS & VM V1R2

- IBM COBOL for OS/390 and VM
- COBOL/VSE

For convenience, programs compiled with an LE-compliant compiler are referred to as "LE COBOL" programs below.

I.3 Running LE-compliant compiler programs under CA-IDMS/DC

This section describes what you need to do to compile, link, and run a program compiled with an LE-compliant compiler.

Note: Also see Chapter 1, "Introduction to CA-IDMS Data Manipulation Language" on page 1-1 and Appendix H, "VS COBOL II Support" on page H-1. The restrictions on VS-COBOL and VS-COBOL II compile and runtime options also apply to programs compiled with an LE-compliant COBOL compiler unless specifically noted below.

General preparation: The following steps describe how to prepare LE-compiled programs for use with CA-IDMS/DC:

Note: Steps 1 through 4 are not needed for CA-IDMS/DC Release 14.1 and higher. There is no longer any need to link RHDCLINT (MVS) or RHDCLENT, IDMSBALI, and CEEBINT (VSE) with each application program. Previously linked load modules that include them do not need to be relinked. There are no adverse effects if they are included.

RHDCLINT/RHDCLENT is also not required when compiling under VS-COBOL II or an earlier release of COBOL. It is needed only for application programs compiled with an LE-compliant compiler that is to run on a CA-IDMS system prior to release 14.1.

Also note that when RHDCLINT/RHDCLENT is required, steps 1 and 2 can be skipped for most sites. There is seldom a need to have CEEBINT tailored for IDMS and also for some other site-specific purpose.

1. If you must make site-specific modifications to CEEBINT, you *must* replace the default version of CEEBINT supplied by IBM with the CEEBINT source module in the CA-IDMS SMP PPOPTION library and use it as the basis for any site-specific modifications.
2. Link-edit the updated CEEBINT with CA-IDMS modules IDMSBALI and RHDCLENT. Name the resulting load module RHDCLINT and put it in a load library accessible to the LE-compliant language compiler. You can use a JCLIN type USERMOD to modify the RHDCLINT module installed into the CA-IDMS SMP target library.
3. The RHDCLINT load module delivered by CA *or* the one modified in steps 1 and 2 above, must be included in the link edit of every LE-compiled program that is to run on a CA-IDMS system, including database or table procedures. The RHDCLINT module is compatible with batch programs, so the same load module can be used for database procedures in batch or under the central version.
4. Under VSE, there is no RHDCLINT module. Instead, object modules RHDCLENT, CEEBINT, and IDMSBALI must be linked with each LE-compiled program that is to run on a CA-IDMS system.

5. All programs compiled with an LE-compliant compiler must be compiled with the RENT option. The RENT option should also be specified for the link edit.
6. Include the IDMS module in the link-edit for all DC COBOL programs regardless of compiler level.

Note: You do not need to link-edit the RHDCLINT module(s) for VS-COBOL or VS-COBOL II programs that run under runtime LE. It is needed only for programs compiled with an LE-compliant compiler.

Runtime options: The IBM Language Environment provides numerous options that control how programs operate at runtime. The default values are designed to be suitable in a batch environment. Therefore, it is necessary to modify some values for applications that are to run in a CA-IDMS/DC/UCF online system.

Note: As stated in the introduction to this Appendix, this Appendix does not apply to programs that run in a CICS or other region, even if they access CA-IDMS via DML or SQL commands. It does apply to programs that run a CA-IDMS/DC/UCF online system, which are invoked from another front-end via UCF (such as an ADS/O application that is accessed via UCFCICS from a CICS front-end).

The IBM Language Environment provides a number of ways to specify runtime options. Three methods are supported for CA-IDMS/DC online programs:

1. Modify, assemble, and link the IBM-supplied CEEUOPT module. Link the resulting module with each application program. Product Documentation Change LI18624 contains a sample version of the CEEUOPT with values that are appropriate for most online CA-IDMS applications. Also consult the section "Creating an Application-Specific Runtime Options Module" in IBM's LE Installation and Customization Manual.
2. Assemble and link a CEEUOPT module as described above. Link the resulting module with RHDCLEFE. Make sure that RHDCLEFE is defined in the CA-IDMS/DC/UCF Sysgen (as described under "Performance improvements using RHDCLEFE" below). This option is available only for release 14.1 and higher of CA-IDMS.
3. Assemble and link a specialized CEEDOPT module. If this method is chosen, special copies of the IBM modules CEEBINIT and CEEPIPI must be maintained for use with online CA-IDMS/DC/UCF systems only. Due to maintainance considerations, this method is not recommended for release 14.1 and higher of CA-IDMS. Use method 2 above. However, this method is needed when running VS-COBOL II online applications under runtime LE using an earlier release of CA-IDMS. See Product Documentation Change LI23664 for further details.

Except as discussed below, the IBM-supplied default runtime options can be used with any site-specific desired modifications. Note that the MSGFILE parameter is ignored and messages are sent to the CA-IDMS log file.

Recommended settings for certain parameters are as shown below. For more details on these parameters see the IBM Language Environment Customization manual.

- ABTERMENC=(RETCODE) or ABTERMENC=(ABEND): This parameter affects the action taken when an LE enclave ends with an unhandled condition of severity 2 or higher. If RETCODE code is specified, the DC task will abend with message DC128004. If ABEND is specified, the DC task will abend with a Uxxx where xxx corresponds to the hexidecimal value of the user abend code set by LE. For example, an LE user abend 4093 would result in a DC task abend with code UFFD.
- ALL31=(ON): This parameter will minimize the amount of below-the-line storage that will be allocated by LE. This parameter requires that all COBOL programs are linked with AMODE(31). It is strongly recommended that any non-conforming programs be relinked so that ALL31=(ON) can be specified.
- INTERRUPT=(OFF): Attention interrupts are handled by the CA-IDMS/DC system and not by LE runtime support. Application COBOL programs can test for attention interrupts using the DC-ATTN-INT condition name under LE just as with earlier COBOL runtime environments.
- POSIX=(OFF): POSIX is not supported under CA-IDMS/DC/UCF.
- RPTSTG=(OFF) or RPTSTG=(ON): Normally OFF should be specified. OFF must be specified for systems prior to release 14.1.

  The purpose of RPTSTG is to determine the storage utilization for a particular application. The report is produced at the end of a COBOL task thread and is written to the CA-IDMS log file. For efficiency reasons, the termination phase of COBOL processing is normally not executed in an online DC environment. If it is necessary to obtain storage information for a particular application, optional bit 196 can be set (See "COBOL II and LE COBOL task management in Appendix M). Note that this option adversely affects performance. Storage reports are therefore normally produced only in a test or development system.
- TERMTHDACT=(QUIET) or TERMTHDACT=(TRACE): This option controls the extent of LE runtime information that will be supplied when an application terminates. All messages will be written to the DC log file.
- TRAP=(ON) or TRAP=(OFF): If ON is specified, program checks in an LE application will result in IBM LE error-handling being put into effect. COBOL-specific and LE messages will be written to the log. After these messages are written and the COBOL thread ends abnormally, the DC task will abend with message DC128004 and a task snap will be taken.

  If OFF is specified, program checks in an LE application will result in an immediate task snap. This is similar to the result in a VS-COBOL or VS-COBOL II runtime environment. No LE messages related to the program check will be written. Furthermore, if any PL/I applications are included in the online system, any ON ERROR clauses will not be handled properly.

In addition to the parameters above, we strongly recommend that you use smaller values than the default ones for the various heap (e.g., ANYHEAP, BELOWHEAP, HEAP) and stack (e.g., LIBSTACK, STACK) parameters since these are allocated on a task thread basis. Storage allocation is most efficient if relatively large values are specified as sixteen bytes less than a multiple of 4096. Smaller values than 4096

should be set for some parameters to avoid wasting storage. The following values have been found to be suitable for most CA-IDMS/DC/UCF systems:

```
ANYHEAP=(2032,4080,ANYWHERE,FREE)
BELOWHEAP=(496,496,FREE)
HEAP=(2032,4080,ANYWHERE,KEEP,4080,4080)
LIBSTACK=(100,2032,FREE)
STACK=(4080,8176,ANY,KEEP)
STORAGE=(NONE,NONE,NONE,4080)
THREADHEAP=(2032,4080,,ANYWHERE,KEEP)
```

I.4 Supported LE functions

CA-IDMS/DC supports these LE functions:

- Math services
- National language support services

CA-IDMS/DC also supports storage management services, but for performance reasons, they are not recommended. The storage management services are:

- CEECRHP: Create heap segment
- CEECZST: Re-allocate (change size of) heap storage
- CEEDSHP: Discard heap segment
- CEEFRST: Free heap storage
- CEEGTST: Get heap storage

I.5 Unsupported LE functions

CA-IDMS/DC does not support the following LE functions:

- CEE3PRM: Get exec parms
- CEETDLI: Call IMS
- CEETEST: Invoke debugging environment
- Date and time services: Use the DML GET TIME command instead

I.6 Performance improvements with RHDCLEFE

Beginning with Release 14.1, CA-IDMS supports a more effcient method of running online VS-COBOL II and LE COBOL programs under LE runtime. In order to realize this performance improvement, link RHDCLEFE and define it in the CA-IDMS sysgen with the following values:

```
ADD PROGRAM RHDCLEFE
  CONCURRENT
  DYNAMIC
  ENABLED
  LANGUAGE IS ASSEMBLER
  NEW COPY IS ENABLED
  NONOVERLAYABLE
  PROGRAM
  NOPROTECT
  REENTRANT
  RESIDENT
  REUSABLE.
```

The advantages of using defining RHDCLEFE in an LE runtime environment are as follows.

- COBOL II programs can run with Storage Protect.
- If RHDCLEFE is in use, it is not necessary to link CEEUOPT with each application program.
- If a VS-COBOL II or an LE COBOL program is invoked multiple times in the same task using a CA-IDMS DML call (#LINK from Assembler, DC TRANSFER from COBOL or PL/I, or LINK from ADS/O), then only one LE enclave and one LE environment will be established.

  The use of RHDCLEFE can reduce the CPU usage for TRANSFER CONTROL to another COBOL program, particularly a VS-COBOL II program. Without RHDCLEFE, each such invocation of a VS-COBOL II program will result in the establishment and termination of both the environment and the enclave. Each such invocation of a LE COBOL program will result in the establishment and termination of the enclave.

Note: RHDCLEFE is linked with a CEEUOPT with ALL31=(ON). As a consequence, all LE COBOL and VS COBOL II programs must be linked with AMODE(31) or AMODE(any).

Appendix J. 18-Byte Communications Blocks

J.1 About this appendix J-3
J.2 18-byte IDMS block J-4
J.3 18-byte IDMS-DC block J-5

J.1 About this appendix

As an alternative to using the 16-byte IDMS and IDMS-DC communications blocks, you can specify 18-byte blocks. This appendix describes where to specify an 18-byte communications block and contains figures showing these blocks. The difference between 16-byte blocks and 18-byte blocks is that an 18-byte block contains an additional 18-byte filler field, and the following fields are 18 bytes instead of 16 bytes:

- RECORD-NAME
- AREA-NAME
- ERROR-SET
- ERROR-RECORD
- ERROR-AREA

►► For descriptions of the fields in IDMS and IDMS-DC, refer to Chapter 3, “Communications Blocks and Error Detection” on page 3-1.

For COBOL, you specify an 18-byte communications block in the SUBSCHEMA-NAMES LENGTH IS clause found in the SCHEMA SECTION of the DATA DIVISION (see "SCHEMA SECTION" in Chapter 4).

J.2 18-byte IDMS block

The following figure shows the 18-byte IDMS communications block.

		Field	Data Type	(bytes)	Initial Value
*	1 8	PROGRAM-NAME	Alphanumeric	8	Program Name
	9 12	ERROR-STATUS	Alphanumeric	4	'1400'
	13 16	DBKEY	Binary	4 (Fullword)	0000
	17 34	RECORD-NAME	Alphanumeric	18	Spaces
	35 52	AREA-NAME	Alphanumeric	18	Spaces
	53 70	FILLER	Alphanumeric	18	Spaces
	71 88	ERROR-SET	Alphanumeric	18	Spaces
	89 106	ERROR-RECORD	Alphanumeric	18	Spaces
	107 124	ERROR-AREA	Alphanumeric	18	Spaces
**	97 100	PAGE-INFO	Binary	4 (Fullword)	0000
	125 ... 224	IDMSSCOM-AREA	Alphanumeric	100	Low Values
	225 228	DIRECT-DBKEY	Binary	4 (Fullword)	0000
NON-CICS	229 235	DATABASE-STATUS	Alphanumeric	7	Spaces
	236	FILLER	...	1	...
	237 240	RECORD-OCCUR	Binary	4 (Fullword)	0000
	241 244	DML-SEQUENCE	Binary	4 (Fullword)	0000
CICS	229 244	FILLER	...	16	Spaces
	245 251	DATABASE-STATIS	Alphanumeric	7	Spaces
	252	FILLER	...	1	...
	253 256	RECORD-OCCUR	Binary	4 (Fullword)	0000
	257 260	DML-SEQUENCE	Binary	4 (Fullword)	0000

* word aligned
** PAGE-INFO-GROUP overlays bytes 97 and 98 and PAGE-INFO-DBK-FORMAT overlays bytes 99 and 100. Both of these fields are binary datatype, each with a length of two bytes. Suggested initial values for both are 00. Together these two fields represent PAGE-INFO.

J.3 18-byte IDMS-DC block

The following figure shows the 18-byte IDMS-DC communications block.

		Field	Data Type	(bytes)	Initial Value
*	1 8	PROGRAM	Alphanumeric	8	Program Name
	9 12	ERROR-STATUS	Alphanumeric	4	'1400'
	13 16	DBKEY	Binary	4 (Fullword)	0000
	17 34	RECORD-NAME	Alphanumeric	18	Spaces
	35 52	AREA-NAME	Alphanumeric	18	Spaces
	53 70	FILLER	Alphanumeric	18	Spaces
	71 88	ERROR-SET	Alphanumeric	18	Spaces
	89 106	ERROR-RECORD	Alphanumeric	18	Spaces
	107 124	ERROR-AREA	Alphanumeric	18	Spaces
**	97 100	PAGE-INFO	Binary	4 (Fullword)	0000
	125 ... 224	IDMSSCOM-AREA	Alphanumeric	100	Low Values
	225 228	DIRECT-DBKEY	Binary	4	0000
	229 ... 328	DCBMSCOM-AREA	Alphanumeric	100	Low Values
	329 332	SSC-ERRSTAT-SAVE	Alphanumeric	4	Spaces
	333 336	SSC-DMLSEQ-SAVE	Binary	4 (Fullword)	0000
	337 340	DML-SEQUENCE	Binary	4 (Fullword)	0000
	341 344	RECORD-OCCUR	Binary	4 (Fullword)	0000
	345 348	SUBSCHEMA-CTRL-END	Alphanumeric	4	Spaces

* word aligned
** PAGE-INFO-GROUP overlays bytes 97 and 98 and PAGE-INFO-DBK-FORMAT overlays bytes 99 and 100. Both of these fields are binary datatype, each with a length of two bytes. Suggested initial values for both are 00. Together these two fields represent PAGE-INFO.

Appendix K. FORTRAN DML Precompiler

K.1 About this appendix K-3
K.2 Compiling a FORTRAN program with DML statements K-4
K.3 IDMS communications block K-5
K.3.1 Record and element synonyms K-5
K.4 Compiler options statements K-7
K.5 Compiler-directive statements K-8
K.5.1 INVOKE statement K-8
K.5.1.1 Syntax K-9
K.5.1.2 Parameters K-9
K.5.1.3 Usage K-10
K.5.2 INCLUDE IDMS K-10
K.5.2.1 Syntax K-10
K.5.2.2 Parameters K-11
K.5.2.3 Usage K-12
K.5.3 INCLUDE IDMS SUBSCHEMA-BINDS K-13
K.5.3.1 Syntax K-14
K.5.3.2 Parameters K-14
K.5.3.3 Usage K-14
K.5.4 INCLUDE IDMS MODULE K-14
K.5.4.1 Syntax K-15
K.5.4.2 Parameters K-15
K.6 Control statements K-16
K.7 Retrieval statements K-17
K.8 Modification statements K-18
K.9 Save statements K-19
K.10 ACCEPT IDMS STATISTICS K-20
K.10.1.1 Syntax K-20
K.10.1.2 Usage K-20
K.11 Logical Record Facility statements K-23
K.12 Program compilation K-24
K.13 Link-edit considerations K-29
K.14 Sample CA-IDMS FORTRAN program K-30

K.1 About this appendix

FORTRAN programmers embed CA-IDMS FORTRAN Data Manipulation Statements (DML) in CA-IDMS FORTRAN applications to request database services. Once coded, the FORTRAN program is submitted to the CA-IDMS FORTRAN precompiler (IDMSDMLF) for preprocessing.

The steps involved in precompiling, compiling, and executing a CA-IDMS program are the same for all programming languages that CA-IDMS supports. However, program preparation, the precompiler and DML syntax will vary based on the host language you are using.

This appendix presents host language differences and describes how to handle them in a CA-IDMS FORTRAN application.

Use this appendix in conjunction with Parts I and II of this document when coding CA-IDMS FORTRAN programs.

Syntax for all FORTRAN DML statements is presented in this appendix. Parameter descriptions are only provided for those statements that are different from the COBOL DML statements.

Sample compile and link-edit JCL is also provided.

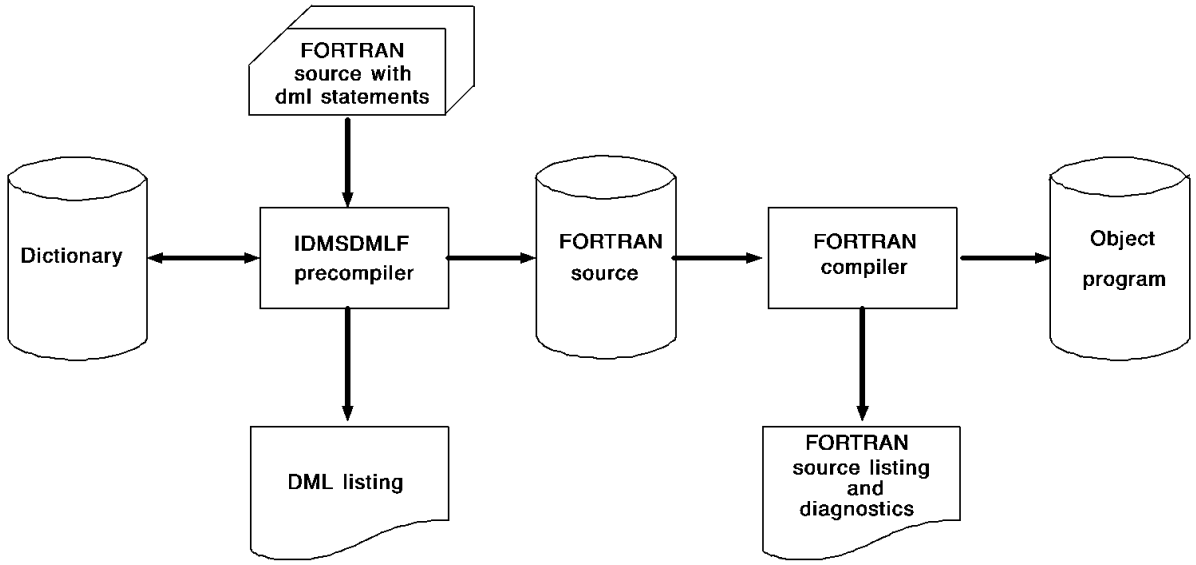
Note: The CA-IDMS FORTRAN precompiler only supports FORTRAN programs executing in batch mode.

K.2 Compiling a FORTRAN program with DML statements

The DMLF precompiler (IDMSDMLF) executes as a step preceding execution of the FORTRAN compiler. Input to the DMLF precompiler consists of a source statement file of FORTRAN and DML statements. Output from the DMLF precompiler is a source (card-image) file that serves as the input to the compiler.

After the source program is processed by the DMLF precompiler, generated FORTRAN source statements must be compiled and link edited.

The following diagram illustrates the steps involved in compiling a CA-IDMS FORTRAN program.



K.3 IDMS communications block

All application programs that request access to a CA-IDMS database require the IDMS communications block, which establishes communication between CA-IDMS and the application program. Whenever a run unit executes a call to CA-IDMS for a database operation, the system posts information concerning the outcome of the requested service in the IDMS communications block established for the program. The error-status indicator (ERSTAT field) contains the return code, which indicates the status of the requested service.

Checking the ERSTAT field: After every call to the DBMS, the application program should examine the ERSTAT field, which occupies positions 9 through 12 of the IDMS communications block. The program can then perform whatever error processing routines it requires, depending on the value contained in ERSTAT. Such error processing may include examining other fields in the IDMS communications block that have been updated following the call to CA-IDMS; the specific fields updated are determined by the DML statement issued and its outcome.

Defining the IDMS communications block The IDMS communications block must be defined in program variable storage. It is copied from the dictionary into the program by the INCLUDE IDMS statement, and is identified as the group field SSCTRL.**:** The layout of the IDMS communications block is shown in Chapter 3, “Communications Blocks and Error Detection” on page 3-1.

K.3.1 Record and element synonyms

The DMLF precompiler supports 31 character record and element names. However, if you are using a FORTRAN compiler that does not support 31 character record and element names, you can define alternative and unique record and element names in the dictionary as synonyms and use them explicitly in FORTRAN programs.

For example, in some versions of FORTRAN, data names are restricted to six characters.

Use of synonyms is recommended if INCLUDE IDMS and INVOKE statements are included in FORTRAN programs.

For a complete description of synonyms, see *IDD User Guide*.

Note: Synonyms cannot be defined for logical record names. FORTRAN programs that access logical records must use a separate subschema in which logical records are defined with names that adhere to FORTRAN conventions.

For a complete description of programming with logical records, refer to *CA-IDMS Logical Record Facility*.

Language-dependent synonyms: The DMLF precompiler automatically copies these language-dependent synonyms in place of the primary names whenever an INCLUDE IDMS statement appears in the application program. When the DMLF

precompiler copies record descriptions from the dictionary into variable storage, it copies synonyms according to the following rules:

- If a record synonym is defined for the program's language (i.e., FORTRAN) but the primary record name is not, the synonym is copied into the program.
- If more than one synonym for a given record is defined for FORTRAN, the first one found in the dictionary is copied.
- If the primary record name is defined for FORTRAN, the primary name is copied into the program.

Example: For example, assume that a record is defined in the dictionary with three synonyms, as follows:

```
RECORD  EMP-REC-XYZ
RECORD  NAME  SYNONYM  XYZSN1  LANGUAGE  FORTRAN
RECORD  NAME  SYNONYM  XYZ-SYN2
RECORD  NAME  SYNONYM  XYZSN3  LANGUAGE  FORTRAN
```

Since the dictionary includes XYZSN1 as the first synonym for FORTRAN, the DMLF precompiler copies it into the program. The DMLF processor would copy the primary record name (e.g., EMP-REC-XYZ) if it were defined for FORTRAN.

These rules apply regardless of the record name or synonym that appears in the schema and subschema invoked by the program.

The DMLF precompiler treats a synonym as if it were the primary record. For example, the expansion of a DML statement will include the record id (SR*nnn*) of the primary record name, even when synonyms are used to name the record and its elements in program variable storage.

►► Refer to *IDD User Guide* for further details on synonym facilities.

K.4 Compiler options statements

FORTRAN syntax for compiler options is presented below. For a description of the compiler options, see Chapter 2, “Precompiler Options” on page 2-1.

```
      Col
      1
      ↓
►►──┬─ CRETRIEVAL ──────────┬──────────────────────────────────────►◄
    └─ CPROTECTED-UPDATE ───┘

      Col
      1
      ↓
►►──┬─ CDMLIST ─────────┬──────────────────────────────────────────►◄
    └─ CNODMLIST ◄ ─────┘

      Col
      1
      ↓
►►──┬─ CSCHEMA-COMMENTS ──┬──────────────────────────────────────────►◄
    └─ CNO-ACTIVITY-LOG ──┘

      Col
      1
      ↓
►►──┬─ CNO-ACTIVITY-LOG ──┬──────────────────────────────────────────►◄
    └─ CNO-ACTIVITY-LOG ──┘
```

K.5 Compiler-directive statements

This section presents the compiler-directive statements that are specific to FORTRAN.

These are the only valid compiler-directives for FORTRAN programs.

Compiler-directive statements instruct the DMLF precompiler to copy source code from the dictionary into the FORTRAN application program. These statements do not produce any executable commands. Four compiler-directive statements are available as follows:

- **INVOKE** identifies a subschema and the operating environment in which the program will execute.
- **INCLUDE IDMS** copies control blocks and record descriptions from the dictionary into the program and optionally allows records to be defined in FORTRAN common blocks.
- **INCLUDE IDMS SUBSCHEMA-BINDS** generates a standard BIND RUN-UNIT statement and the appropriate standard BIND RECORD statements.
- **INCLUDE IDMS MODULE** copies source modules from the dictionary into the program.

The programmer can code compiler-directive statements anywhere in positions 7 through 72. Each statement is discussed separately below.

K.5.1 INVOKE statement

The INVOKE statement provides the following information to the DML processor:

- The subschema view of the database. The named subschema determines which CA-IDMS record descriptions the processor copies into the program. Also, CA-IDMS checks every DML statement issued by the program against the record, set, and area access restrictions specified in this subschema.
- The name of the program.
- The operating environment (e.g., batch) in which the program executes. This is used by CA-IDMS when generating CALL statements.
- Whether the DMLF precompiler enters the sequence number for each DML statement in the IDMS communications block at run time to aid in program debugging.

Any program that accesses the database (i.e., issues DML statements) must include an INVOKE statement identifying the schema. This statement must be specified before any other compiler-directive or DML statement, with the exception of the INCLUDE IDMS MODULE statement. The INCLUDE IDMS MODULE statement can precede the INVOKE statement, as long as the module it copies does not contain DML statements. If the program does not access the database, the program can omit the INVOKE statement as well; the DMLF precompiler can still be used to copy source

code from the dictionary by using the INCLUDE IDMS MODULE statement, discussed later in this section.

K.5.1.1 Syntax

```
►►── INVOKE(┬──────────────────────────────────────────────────────────►
           └─ PROGRAM=program-name ┬──────────────────────────┬
                                   └─ ,VERSION=version-number ─┘

►── ,SUBSCHEMA=subschema-name ──────────────────────────────────────────►

►── ,SCHEMA=schema-name ┬──────────────────────────────────────────────►
                        └─ ,VERSION=version-number ─┘

►┬──────────────────┬┬──────────────┬─ ) ──────────────────────────────►◄
 └─ ,MODE=mode ─────┘└─ ,DEBUG ─────┘
```

K.5.1.2 Parameters

PROGRAM=program-name

Specifies the name of the program.

If program registration is in effect, this clause is required and must be the first clause in the INVOKE statement.

If the program is defined in the dictionary through IDD facilities, *program-name* must match the dictionary-assigned name in order for the DMLF precompiler to recognize it as the same program.

You must specify this parameter for program activity statistics to be logged in the dictionary.

►► For more information on logging program activity statistics, see Chapter 2, “Precompiler Options” on page 2-1.

,VERSION=version-number

Optionally distinguishes one version of a source program from another (e.g., for purposes of testing or development). *Version-number* is the number of the desired program version and must be in the range 1 through 9999. If not specified, *version-number* defaults to the highest number defined in the dictionary for the named program, or to 1 if the program does not already exist in the dictionary.

,SUBSCHEMA=subschema-name

Specifies a previously compiled subschema. If program registration is in effect, the program named must be predefined in the dictionary and associated with *subschema-name*.

,SCHEMA=schema-name

Specifies the schema under which *subschema-name* is compiled.

,VERSION=version-number

Optionally qualifies *schema-name* with a version number. If omitted, *version-number* defaults to the highest version number present in the dictionary for that schema.

,MODE=mode

Identifies the mode the DML processor uses to generate CALL statements for the program's DML statements. You can specify either one of these options:

- MODE=BATCH — Specifies that the program executes in batch mode
- MODE=FORT77 — Specifies that the program will use the FORTRAN 77 (FORTRAN/VS) compiler

,DEBUG

Specifies that, for each DML call, the DMLF precompiler places a unique sequence number in the IDMS communications block at run time. These numbers appear in positions 73 through 80 of the FORTRAN compiler output listing, in the form DMLF*nnnn*.

K.5.1.3 Usage

Using sequence numbers

The DMLF precompiler generates unique sequence numbers to identify the sequence in which DML statements appear in the program. Depending on the error routine(s) defined, the program can report DML sequence numbers when errors occur and then the programmer can use the numbers to assist in debugging the FORTRAN application.

K.5.2 INCLUDE IDMS

The INCLUDE IDMS statement copies source data descriptions and source modules from the dictionary into the program at the location of the INCLUDE IDMS statement. The source code copied from the dictionary usually consists of CA-IDMS database record descriptions, logical record descriptions, and the IDMS communications block. However, any source module or record description stored in the dictionary can be copied into the program by the DMLF precompiler.

K.5.2.1 Syntax

```
►►─┬──────────────────────────────────────────────────────────────►─
   │ COMMON ─┬──────────────────────┬─
   │         └─ /block-identifier/ ─┘
   │
   │
►──┴─ INCLUDE IDMS(─┬─ SUBSCHEMA-DML-LR-DESCRIPTION ──────────────┬─)─►◄
                    ├─ SUBSCHEMA-DESCRIPTION ─────────────────────┤
                    ├─ SUBSCHEMA-CTRL ────────────────────────────┤
                    ├─ SUBSCHEMA-RECORDS ─────────────────────────┤
                    ├─ record-name ─┬──────────────────────────┬──┤
                    │               └─,VERSION=version-number ─┘  │
                    ├─ SUBSCHEMA-LR-DESCRIPTION ──────────────────┤
                    ├─ SUBSCHEMA-LR-CTRL ─┬──────────────────────┬┤
                    │                     └─ ,SIZE=lrc-block-size ─┘│
                    ├─ SUBSCHEMA-LR-RECORDS ──────────────────────┤
                    └─ LR=logical-record-name ────────────────────┘
```

K.5.2.2 Parameters

COMMON/<u>block-identifier</u>/INCLUDE IDMS

Copies the specified component into the program. Optionally identifies an area in COMMON into which the record(s) will be defined for use by more than one program unit.

Block-identifier specifies the one- to six-character name given to the COMMON area. COMMON elements can only be assigned user-specified initial data values (as defined using the schema compiler or IDD compiler) within a BLOCK-DATA subprogram. If a record is not included in COMMON no restrictions exist on initial values.

SUBSCHEMA-DML-LR-DESCRIPTION

Copies all components required for access to both database and logical records (i.e., SUBSCHEMA-CTRL, SUBSCHEMA-RECORDS, SUBSCHEMA-LR-CTRL, SUBSCHEMA-LR-RECORDS). Do not specify SUBSCHEMA-DML-LR-DESCRIPTION if the usage in the program's subschema is DML or LR; it should be specified only when the usage is MIXED.

SUBSCHEMA-DESCRIPTION

Copies all components required for access to database records (SUBSCHEMA-CTRL and SUBSCHEMA-RECORDS). Do not specify SUBSCHEMA-DESCRIPTION if the subschema's usage is LR.

SUBSCHEMA-CTRL

Copies the IDMS communications block data description into the program.

SUBSCHEMA-RECORDS

Copies the descriptions of all records contained in the subschema. According to the rules of synonym usage, the DMLF precompiler copies FORTRAN synonyms defined for the subschema records from the dictionary into the program. Do not specify SUBSCHEMA-RECORDS if the subschema's usage is LR.

<u>record-name</u>

Copies the description of an individual record from the dictionary. *Record-name* can be the primary name or a synonym for a record or module stored in the dictionary.

A record that is copied into a schema (i.e., a schema-owned record) can only be copied into a program that uses a subschema associated with that schema and cannot be copied into a non-IDMS program (i.e, a program that doesn't access a CA-IDMS database). However, a synonym defined for the schema-owned record can be copied into a non-IDMS program (use the VERSION clause to identify the synonym).

,VERSION=<u>version-number</u>

Optionally qualifies the named record with a version number. *Version-number* specifies an explicit version number, corresponding to a record to be copied by the DMLF precompiler.

If no version number is specified, CA-IDMS assumes that *record-name* identifies a record that is included in the subschema named in the INVOKE statement. If the record is not associated with the subschema, or if it is a module,

version-number defaults to the highest version number of the record or module defined in the dictionary for the operating mode under which the program is being compiled.

Note: A record description should be copied into variable storage only once.

Do not specify the *record-name* parameter if the subschema's usage is LR.

K.5.2.3 Usage

Finding record name

If the DMLF precompiler cannot find a record named *record-name* in the dictionary, it searches for a module by that name. The module, which may have been stored using the IDD DDDL compiler, presumably contains a definition of records not included in the subschema. If an operating mode is associated with the named record or module in the dictionary, it must agree with the mode in effect for this program (see "INVOKE" above).

Using the VERSION clause

You can use the VERSION clause with *record-name* to qualify IDD records or modules with a version number. *Version-number* specifies an explicit version number, corresponding to a record to be copied by the DMLF precompiler.

You cannot use the VERSION clause to qualify schema-owned records.

FORTRAN data types

When an INCLUDE IDMS statement is used to copy records and record elements into a FORTRAN program, the DMLF precompiler assigns appropriate FORTRAN data types.

Records are given a LOGICAL*1 type and a dimension corresponding to the number of bytes displaced by the record. For elements, the following data types defined in the schema are generated as comments in the subschema record description:

- COMP-3
- BIT
- POINTER

CONDITION-NAME and FILLER items are also generated as comments. Other element data types are assigned according to the following correspondences:

DDL or DDDL Usage	FORTRAN Type
DISPLAY	LOGICAL*1
COMP (halfword)	INTEGER*2

DDL or DDDL Usage	FORTRAN Type
COMP (fullword)	INTEGER*4
COMP-1	REAL*4
COMP-2	REAL*8

The DMLF precompiler automatically generates EQUIVALENCE statements that position elements within records.

Record elements with display usage

A record element with display usage can be assigned a FORTRAN type other than LOGICAL*1 (to use the element in a comparison, for example).

To do this, the database administrator must add a comment entry for the element through the schema compiler or the IDD DDDL compiler.

The comment entry should be the first comment entry for the element and in this form:

```
COMMENTS 'FORTRAN TYPE IS type-a'
```

The DMLF precompiler will assign the FORTRAN type defined by *type-a* to the record element and will give the record element an appropriate dimension, calculated by dividing the display record element length by the length specified within *type-a*.

Valid *type-a* qualifiers are:

- INTEGER*2
- INTEGER*4
- REAL*4,
- REAL*8

K.5.3 INCLUDE IDMS SUBSCHEMA-BINDS

The INCLUDE IDMS SUBSCHEMA-BINDS statement instructs the DMLF processor to generate in the source program a standard BIND RUN-UNIT statement and appropriate standard BIND RECORD statements for each IDMS record in the program's variable storage. The DMLF precompiler converts the INCLUDE IDMS statement itself to a comment.

K.5.3.1 Syntax

```
►►──┬──────────────────────────────────────────────────────────────►◄
    └─ INCLUDE IDMS(SUBSCHEMA-BINDS) ─┘
```

K.5.3.2 Parameters

INCLUDE IDMS(SUBSCHEMA-BINDS)

The INCLUDE IDMS SUBSCHEMA-BINDS statement generates BINDS only for subschema records explicitly copied into the program by means of INCLUDE IDMS statements. It does not automatically generate BINDS for all subschema records.

Note that the INCLUDE IDMS SUBSCHEMA-BINDS statement does not generate BINDS for logical records, since BINDS are not allowed for logical records.

All INCLUDE IDMS RECORD statements must precede any INCLUDE IDMS SUBSCHEMA-BINDS statement, since the DMLF precompiler is a one-pass processor. The DMLF precompiler will not generate BINDS for any record types whose descriptions it copies into the program after the INCLUDE IDMS SUBSCHEMA-BINDS statement.

K.5.3.3 Usage

Issuing separate BIND statements

Instead of issuing an INCLUDE IDMS SUBSCHEMA-BINDS statement, the programmer can issue BIND RUN-UNIT and BIND RECORD statements separately, allowing the program to:

- Call a subroutine to check the ERSTAT field of the IDMS communications block after each BIND RECORD statement.
- Bind several records to the same location by including a DML BIND statement for each record.

K.5.4 INCLUDE IDMS MODULE

The INCLUDE IDMS MODULE statement instructs the processor to copy a source module into a program from the dictionary. Modules must have been previously placed in the dictionary by means of the IDD DDDL compiler.

The DMLF precompiler places the module into the program at the location of the request without modification. The module can, but need not, contain CA-IDMS DML statements. If DML statements are present, the DMLF precompiler examines and expands them within the context of the program's subschema view and compile mode as if the programmer had coded them directly.

You can nest INCLUDE IDMS MODULE statements; code invoked by an INCLUDE IDMS MODULE entry can, itself, contain an INCLUDE IDMS MODULE statement. However, you should ensure that a copied module does not, in turn, copy itself.

K.5.4.1 Syntax

```
►►──┬─ INCLUDE IDMS(module-name ──┬──────────────────────────┬──)──┬──►◄
    │                             │ ,VERSION=version-number  │     │
    └─────────────────────────────┴──────────────────────────┴─────┘
```

K.5.4.2 Parameters

module-name

Specifies the name of the module to be copied. It must be the name of a module defined in the dictionary.

,VERSION=version-number

Optionally qualifies *module-name* with a version number. If not specified, *version-number* defaults to the highest value associated in the dictionary with the named module, for the mode under which the program is being compiled.

If no mode-specific version exists for *module-name*, the DMLF precompiler copies the non-mode-specific version, if present. If the dictionary contains neither a non-mode-specific entry for *module-name* nor mode-specific entry applicable to this program's compile mode, an error results. The same holds true for language (i.e., *version-number* defaults to the highest value defined for the module for the program's language, etc).

K.6 Control statements

```
►►── BIND(PROCEDURE=procedure-name,procedure-control-location) ──────────►◄

►►── BIND(RECORD=record-name ┬──────────────────────┐ ) ──────────────►◄
                             └ ,record-location ────┘

►►── BIND(RUN-UNIT) ──────────────────────────────────────────────────►─

─►┬──────────────────────────────────────────────────────────────────┐►◄
  ├─ ,SUBSCHEMA=subschema-name ┬──────────────────────────────────┐
  │                            └,SUBSCHEMA-CTRL=subschema-cntrl-loc┘
  ├─ ,DBNODE=nodename ──────────────────────────────────────────────┤
  ├─ ,DBNAME=database-name ─────────────────────────────────────────┤
  ├─ ,DICTNODE=nodename ────────────────────────────────────────────┤
  └─ ,DICTNAME=dictionary-name ─────────────────────────────────────┘

►►── IF(┬────────┐ EMPTY(set-name)) imperative-statement ──────────►◄
      └.NOT.────┘

►►── IF(┬────────┐ MEMBER(set-name)) imperative-statement ──────────►◄
      └.NOT.────┘

►►── KEEP(┬──────────────┐ CURRENT ┬──────────────────────────┐)──►◄
        └ EXCLUSIVE ────┘         ├─,RECORD=record-name ──────┤
                                  ├─,SET=set-name ────────────┤
                                  └─,AREA=area-name ──────────┘

►►── COMMIT ┬──────────┐──────────────────────────────────────────►◄
           └ (ALL) ───┘

►►── FINISH ┬──────────┐──────────────────────────────────────────►◄
           └ (ALL) ───┘

►►── READY(┬─ ALL ◄ ──────────┐┬──────────────────┐┬─ RETRIEVAL ◄ ─┬─ ) ──►◄
          └─ AREA=area-name ──┘├─ PROTECTED ──────┤└─ UPDATE ───────┘
                               └─ EXCLUSIVE ───────┘

►►── ROLLBACK ┬──────────────┐──────────────────────────────────►◄
             └ (CONTINUE) ──┘
```

K.7 Retrieval statements

FORTRAN syntax for DML retrieval statements is presented below. For a description of DML statement parameters, see Chapter 5, “Data Manipulation Language Statements” on page 5-1.

```
►►─┬─ FIND ──┬─(─┬──────────────────────┬─┬─ CALC(ANY) ─┬─,RECORD=record-name)─►◄
   └─ OBTAIN ─┘   └─ KEEP ─┬──────────┬─┘ └─ DUPLICATE ─┘
                           └─ EXCLUSIVE ─┘

►►─┬─ FIND ──┬─(─┬──────────────────────┬──────────────────────────────────────►
   └─ OBTAIN ─┘   └─ KEEP ─┬──────────┬─┘
                           └─ EXCLUSIVE ─┘

 ►── CURRENT ─┬──────────────────────────┬── ) ──────────────────────────────►◄
              ├─ ,RECORD=record-name ────┤
              ├─ ,SET=set-name ──────────┤
              └─ ,AREA=area-name ────────┘

►►─┬─ FIND ──┬─(─┬──────────────────────┬──────────────────────────────────────►
   └─ OBTAIN ─┘   └─ KEEP ─┬──────────┬─┘
                           └─ EXCLUSIVE ─┘

 ►── KEY=db-key ─┬──────────────────────┬── ) ──────────────────────────────►◄
                 └─,RECORD=record-name ─┘

►►─┬─ FIND ──┬─(─┬──────────────────────┬─ OWNER,SET=set-name)──────────────►◄
   └─ OBTAIN ─┘   └─ KEEP ─┬──────────┬─┘
                           └─ EXCLUSIVE ─┘

►►─┬─ FIND ──┬─(─┬──────────────────────┬──────────────────────────────────────►
   └─ OBTAIN ─┘   └─ KEEP ─┬──────────┬─┘
                           └─ EXCLUSIVE ─┘
 ►── RECORD=record-name,─┬──────────┬─ SET=set-name ──────────────────────►
                         └─ CURRENT ─┘

 ►──,USING=sort-field-name) ──────────────────────────────────────────────►◄

►►─┬─ FIND ──┬─(─┬──────────────────────┬─┬─ NEXT ──────────────────────┬──►
   └─ OBTAIN ─┘   └─ KEEP ─┬──────────┬─┘ ├─ PRIOR ─────────────────────┤
                           └─ EXCLUSIVE ─┘ ├─ FIRST ─────────────────────┤
                                           ├─ LAST ──────────────────────┤
                                           └─ OFFSET=sequence-number ────┘

 ►─┬──────────────────────────┬─┬─,SET=set-name ─┬── ) ──────────────────►◄
   └─ RECORD=record-name ─────┘ └─,AREA=area-name ─┘

►►── GET(RECORD=record-name) ──────────────────────────────────────────────►◄
```

K.8 Modification statements

FORTRAN syntax for DML modification statements is presented below. For a complete description of DML statement parameters, see Chapter 5, “Data Manipulation Language Statements” on page 5-1.

```
►►── CONNECT(RECORD=record-name,SET=set-name) ──────────────────────────►◄

►►── DISCONNECT(RECORD=record-name,SET=set-name) ───────────────────────►◄

►►── MODIFY(RECORD=record-name)──────────────────────────────────────────►◄

►►── ERASE(RECORD=record-name ┬──────────────────┬ ) ──────────────────►◄
                              ├── ,PERMANENT ────┤
                              ├── ,SELECTIVE ────┤
                              └── ,ALL ──────────┘

►►── STORE(RECORD=record-name) ──────────────────────────────────────────►◄
```

K.9 Save statements

FORTRAN syntax for DML save statements is presented below.

The IDMS Statistics block is different for FORTRAN programs and is therefore described below. For a description of the other DML save statement parameters, see Chapter 5, “Data Manipulation Language Statements” on page 5-1.

```
►►── ACCEPT(BIND=bind-address,RECORD=record-name) ──────────────────────►◄

►►── ACCEPT(IDMS-STATISTICS=db-statistics) ──────────────────────────────►◄

►►── ACCEPT(CURRENCY=db-key ──────────────────────────────────────────►
 ►─┬──────────────────────────┬──── ) ──────────────────────────────────►◄
   ├─,RECORD=record-name ─────┤
   ├─,SET=set-name ───────────┤
   └─,AREA=area-name ─────────┘

►►── ACCEPT(CURRENCY=db-key,SET=set-name ─┬─,NEXT ──┬─ ) ──────────────►◄
                                           ├─,PRIOR ─┤
                                           └─,OWNER ─┘

►►── ACCEPT(PROCEDURE=procedure-name,procedure-control-location) ──────►◄

►►── RETURN(CURRENCY=db-key,SET=index-set-name ─┬─,FIRST ─┬─ ) ──────►◄
                                                ├─,LAST ──┤
                                                ├─,NEXT ──┤
                                                └─,PRIOR ─┘

►►── RETURN(CURRENCY=db-key,SET=index-set-name,USING=index-key-value) ──►◄
```

K.10 ACCEPT IDMS STATISTICS

The ACCEPT IDMS STATISTICS statement is used to move system run-time statistics located in the program's IDMS Statistics block to program variable storage. You can issue this statement any number of times during the execution of a run unit.

For example, the user might request database statistics after storing a variable length record to determine whether the entire record was stored in one place or if fragments were placed in an overflow area. The ACCEPT IDMS STATISTICS statement does not reset any of the statistics fields; resetting of IDMS Statistics block fields occurs only upon issuing a FINISH.

After completion of the ACCEPT IDMS STATISTICS function, the ERSTAT field in the IDMS communications block indicates the outcome of the operation as follows:

Error Status	Meaning
1518	The database statistics location has not been bound properly.

The only other acceptable error-status code returned for an ACCEPT IDMS STATISTICS statement is 0000.

K.10.1.1 Syntax

The syntax of the ACCEPT IDMS STATISTICS command is:

```
ACCEPT(IDMS STATISTICS=db-statistics)
```

This command specifies that the system run-time statistics contained in IDMS STATISTICS are to be moved to a location in program variable storage identified by *db-statistics*, which is an aligned, 100-byte field. Its description can be stored in the dictionary as a record or a module and copied into the program, or can be defined by the FORTRAN programmer.

K.10.1.2 Usage

IDMS Statistics

The following is the suggested definition of the field that will contain statistics from the IDMS Statistics block.

```
      LOGICAL*1  DBSTAT   (100)
      REAL*8     DATETD   (8)
                          EQUIVALENCE  (DBSTAT (1),DATETD )
      REAL*8     TIMETD   (8)
                          EQUIVALENCE  (DBSTAT (9),TIMETD )
      INTEGER*4  PGSRED
                          EQUIVALENCE  (DBSTAT (17),PGSRED )
      INTEGER*4  PGSRIT
                          EQUIVALENCE  (DBSTAT (21),PGSRIT )
      INTEGER*4  PGSREQ
                          EQUIVALENCE  (DBSTAT (25),PGSREQ )
      INTEGER*4  CALCTR
                          EQUIVALENCE (DBSTAT (29),CALCCTR )
      INTEGER*4  CALCOV
                          EQUIVALENCE (DBSTAT (33),CALCOV )
      INTEGER*4  VIATRG
                          EQUIVALENCE (DBSTAT (37),VIATRG )
      INTEGER*4  VIAOVR
                          EQUIVALENCE (DBSTAT (41),VIAOVR )
      INTEGER*4  LNSREQ
                          EQUIVALENCE (DBSTAT (45),LNSREQ )
      INTEGER*4  RECSCR
                          EQUIVALENCE (DBSTAT (49),RECSCR )
      INTEGER*4  CALLDB
                          EQUIVALENCE (DBSTAT (53),CALLDB )
      INTEGER*4  FRGSTR
                          EQUIVALENCE (DBSTAT (57),FRGSTR )
      INTEGER*4  RECSRL
                          EQUIVALENCE (DBSTAT (61),RECSRL )
      INTEGER*4  LOKREQ #
                          EQUIVALENCE (DBSTAT (65),LOKREQ )
      INTEGER*4  SLOKHL #
                          EQUIVALENCE (DBSTAT (69),SLOKHL )
      INTEGER*4  ULOKHL #
                          EQUIVALENCE (DBSTAT (73),ULOKHL )
      INTEGER*4  RUNUNT #
                          EQUIVALENCE (DBSTAT (77),RUNUNT )
      INTEGER*4  TASKID #
                          EQUIVALENCE (DBSTAT (81),TASKID )
      REAL*8     LOCLID  (8)
                          EQUIVALENCE (DBSTAT (85),LOCLID )
C     LOGICAL*1  FILLER  (8)

  #Applies to central version only
```

Meaning of fields

The meaning of these IDMS Statistics block fields are as follows:

```
DBSTAT-- IDMS statistics         RECSCR-- Records current
DATETD-- Date today              CALLDB-- Calls to IDMS
TIMETD-- Time today              FRGSTR-- Fragments stored
PGSRED-- Pages read              RECSRL-- Records relocated
PGSRIT-- Pages written           LOCREQ-- Locks requested
PGSREQ-- Pages requested         SLOKHL-- Select locks held
CALCTR-- CALC target             ULOKHL-- Update locks held
CALCOV-- CALC overflow           RUNUNT-- Run unit ID
VIATRG-- VIA target              TASKID-- Task ID
VIAOVR-- VIA overflow            LOCLID-- Local ID
LNSREQ-- Lines requested
```

Note: The LOCLID field consists of the 4-byte, LOGICAL*1 identifier of the interface in which the run unit originated (e.g., BATC) and a unique identifier

(an INTEGER*4 value) assigned to the run unit by that interface. For batch run units, this INTEGER*4 identifier specifies the internal machine time. The DBSTAT field can be modified to define two subordinate fields for the LOCLID field.

Example

The following statements establish currency for the sets in which a new ITEM record will participate as a member, store the ITEM record, and move statistics regarding the stored ITEM record to the DBSTAT location in main storage:

```
PRODNO=IPRDNO
FIND(CALC,RECORD=PRODCT)
ORDRNO=IORDNO
FIND(CALC,RECORD=ORDOR)
STORE(RECORD=ITEM)
ACCEPT(IDMS-STATISTICS=DBSTAT)
```

K.11 Logical Record Facility statements

The DML statements available to access logical records are OBTAIN, MODIFY, STORE, and ERASE. In contrast to the DML statements used to access database records, each of these statements can include a WHERE clause to select the logical record to be accessed.

Logical record DML statements are presented below. For a complete description of LRF DML statement parameters, see Chapter 5, “Data Manipulation Language Statements” on page 5-1.

►► For a complete discussion of programming with logical records, see *CA-IDMS Logical Record Facility*.

```
►►── OBTAIN ┬──────────┬── (RECORD=logical-record-name ──────────────►
          ├─ FIRST ─┤
          └─ NEXT ◄ ─┘

 ►┬──────────────────────────────────┬──────────────────────────────►
  └,INTO=alt-logical-record-location ┘

 ►┬──────────────────────────────┬── ) ──────────────────────────►◄
  └,WHERE=(boolean-expression) ─┘

►►── MODIFY(RECORD=logical-record-name ──────────────────────────►◄

 ►┬──────────────────────────────────┬──────────────────────────────►
  └,FROM=alt-logical-record-location ┘

 ►┬──────────────────────────────┬── ) ──────────────────────────►◄
  └,WHERE=(boolean-expression)─┘

►►── STORE(RECORD=logical-record-name ──────────────────────────►◄

 ►┬──────────────────────────────────┬──────────────────────────────►
  └,FROM=alt-logical-record-location ┘

 ►┬──────────────────────────────┬── ) ──────────────────────────►◄
  └,WHERE=(boolean-expression)─┘

►►── ERASE(RECORD=logical-record-name ──────────────────────────►◄

 ►┬──────────────────────────────────┬──────────────────────────────►
  └,FROM=alt-logical-record-location ┘

 ►┬──────────────────────────────┬── ) ──────────────────────────►◄
  └,WHERE=(boolean-expression)─┘
```

K.12 Program compilation

To compile a FORTRAN program under the DMLF precompiler, execute the program IDMSDMLF followed by the FORTRAN compiler and the linkage editor.

IDMSDMLF: Input to IDMSDMLF consists of:

- A source program written in FORTRAN DML
- Protocol/control information
- Dictionary record descriptions

Output from IDMSDMLF is:

- A source FORTRAN program
- A DML source listing and diagnostics

FORTRAN compiler: Input to the FORTRAN compiler consists of the source program produced by IDMSDMLF.

Output is:

- An object program
- FORTRAN listings

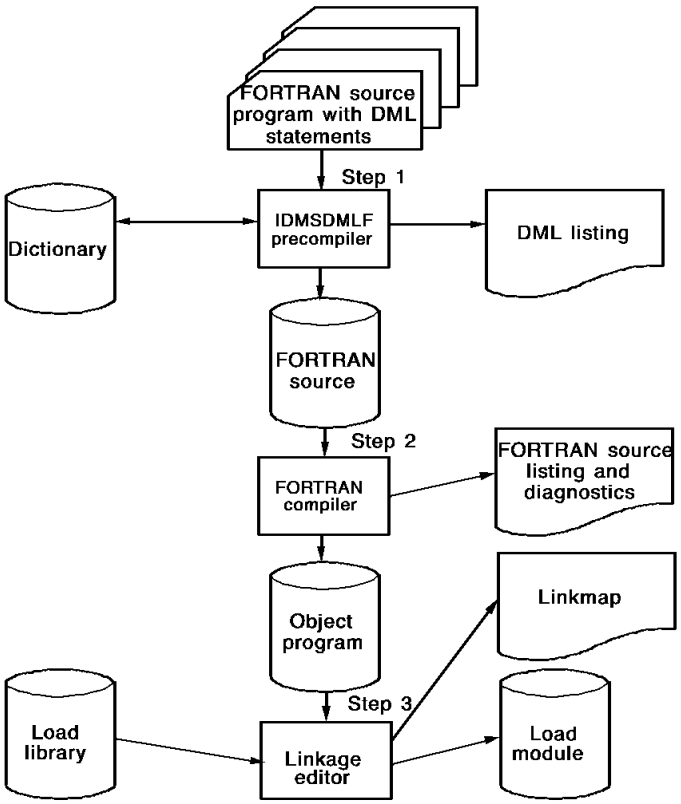
Linkage editor: Input to the linkage editor consists of the object program produced by the FORTRAN compiler.

Output is:

- A load module
- A link-edit map

Compiling a FORTRAN program

The following diagram shows the steps required to compile and link edit a CA-IDMS FORTRAN program.



Central version JCL: The MVS JCL used to compile and link edit DMLF source statements under the central version is shown below.

IDMSDMLF (MVS)

```
//* ******************************************************************
//* **                      PRECOMPILE FORTRAN PROGRAM                **
//* ******************************************************************
//precomp  EXEC PGM=IDMSDMLF,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYS001   DD   UNIT=disk,SPACE=(TRK,(10,10))
//SYS002   DD   UNIT=disk,SPACE=(TRK,(10,10))
//SYS003   DD   UNIT=disk,SPACE=(TRK,(10,10))
//SYSPCH   DD   DSN=&&source,DISP=(NEW,PASS,DELETE),
//              UNIT=disk,SPACE=(TRK,(10,5),RLSE),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSLST   DD   SYSOUT=A
//SYSIDMS  DD   *

DMCL=dmcl-name
DICTNAME=dictionary-name
Other SYSIDMS parameters, as appropriate
/*

//SYSIPT     DD  *


FORTRAN DML source statements
//*******************************************************************
//**                      COMPILE FORTRAN PROGRAM                  **
//*******************************************************************
//forcmp   EXEC PGM=FORTVS,REGION=512K,
//             PARM='DECK,NOLOAD,NOLIB,BUF=50000,SIZE=150K'
//SYSLIB   DD   DSN=sys1.fortlib,DISP=SHR
//SYSUT1   DD   UNIT=disk,SPACE=(TRK,(10,5))
//SYSUT2   DD   UNIT=disk,SPACE=(TRK,(10,5))
//SYSUT3   DD   UNIT=disk,SPACE=(TRK,(10,5))
//SYSPUNCH DD   DSN=&&object,UNIT=disk,DISP=(NEW,PASS),
//             SPACE=(80,(400,40))
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSN=&&source,DISP=(OLD,DELETE)
//*******************************************************************
//**                      LINK PROGRAM MODULE                      **
//*******************************************************************
//link     EXEC PGM=IEWL,REGION=96K,
//             PARM='XREF,LIST,LET,SIZE=(96K,12K)'
//SYSLIN   DD   DSN=&&OBJECT,DISP=(OLD,DELETE)
//         DD   DDNAME=SYSIN
//SYSUT1   DD   UNIT=disk,SPACE=(TRK,(20,5))
//SYSLIB   DD   DSN=sys1.fortlib,DISP=SHR
//loadlib  DD   DSN=idms.loadlib,DISP=SHR
//SYSLMOD  DD   DSN=user.loadlib,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
 INCLUDE loadlib(IDMS)      required
 INCLUDE loadlib(IDMSCANC)  required for BATCH and DC-BATCH
                                  if using IDMS-STATUS module
 INCLUDE loadlib(IDMSOPTI)  optional;
                                  BATCH and DC-BATCH only
 ENTRY   userentry
 NAME    userprog(R)
/*
```

1024K	size of region to run IDMSDMLC

`idms.dba.loadib`	data set name of the load library containing the DMCL and database name table load modules
`idms.loadlib`	data set name of the load library containing the CA-IDMS executable modules
`sysctl`	ddname of SYSCTL file
`idms.sysctl`	data set name of SYSCTL file
`disk`	symbolic device name for work files
`&&source`	name of temporary data set output from DMLF precompiler
`dcmsg`	ddname of the system message (DDLDCMSG) area
`idms.sysmsg.ddldcmsg`	data set name of the system message (DDLDCMSG) area
`SYSIDMS`	name of the parameter file where you specify physical requirements of the environment, runtime directives, or operating system-dependent file information
`dmcl-name`	specifies the name of the dictionary the DMLF precompiler should access
`dictionary-name`	identifies the CA-IDMS system to bind to at runtime
`sys1.fortlib`	data set name of the FORTRAN library
`&&object`	name of temporary data set output from FORTRAN compiler
`user.loadlib`	user application load library
`userentry`	name of program entry point
`userprog`	name of user application program

Note: Depending on the CV operating environment, an IDMSOPTI module link edited with IDMSDMLF can be used in place of or in addition to the SYSCTL file.

Parameter option available to code in the EXEC statement of the precompile step is:

- DICTNAME=*dictionary-name*

▸▸ For a complete description of all the parameter options, see *CA-IDMS SQL Programming*.

▸▸ For a complete description of SYSIDMS parameters and syntax, see *CA-IDMS Navigational DML Programming*.

Local mode JCL: To execute the IDMSDMLF precompiler in local mode remove the SYSCTL statement from the precompile step and replace it with the following:

IDMSDMLF (Local)

```
//dictdb   DD DSN=idms.appldict.ddldml,DISP=SHR
//sysjrnl  DD DSN=idms.tapejrnl,DISP=(NEW,KEEP),UNIT=
tape
```

dictdb	ddname of the application dictionary DDLDML area
idms.appldict.ddldml	data set name of the application dictionary DDLDML area
sysjrnl	ddname of the tape journal file
idms.tapejrnl	data set name of the tape journal file

In local mode you will also need to specify the name of the DMCL in the SYSIDMS parameter file unless you're using the default name of IDMSDMCL. In addition, you must include the load library containing the DMCL in the STEPLIB concatenation:

```
//STEPLIB  DD  DSN=idms.dbalib,DISP=SHR
//         DD  DSN=idms.loadlib,DISP=SHR
```

K.13 Link-edit considerations

The modules involved in the link edit of an application program contain three external references. Some must be resolved, others can be left unresolved depending on the mode of operation. The table below lists and explains the external references.

Reference	Referenced by	Resolved by	Comments
ABORT	Application program	IDMSCANC	Should be resolved ONLY in a batch environment; should NOT be included in a tp environment.
IDMS	Application program	IDMS	Must be resolved
IDMSOPTI (1)	IDMS	IDMSOPTI module	Must be resolved if using the central version without a SYSCTL file

(1) IDMSOPTI is a weak external reference (WXTRN).

K.14 Sample CA-IDMS FORTRAN program

This section contains listings that document a sample FORTRAN program that accesses a CA-IDMS database. The program is shown as it appears in various stages of the compilation process.

- The user creates a program using FORTRAN and FORTRAN DML source statements.
- This program is input to the DMLF precompiler, which produces a listing containing diagnostics and, optionally, DML source statements.
- The DMLF-expanded code is then input to the FORTRAN compiler, which generates a listing of the fully expanded code and diagnostics.

FORTRAN program as input to DMLF precompiler

```
CRETREVIAL
CDMLIST
       INVOKE (PROGRAM=SAMPFORT,SUBSCHEMA=EMPSS01,SCHEMA=EMPSCHM,
       VERSION=100,MODE=BATCH)
       REAL*8        DICTNM /'APPLDICT'/
       INTEGER*4     ER0326 /'0326'/
       INTEGER*4     ER0307 /'0307'/
       INTEGER*4     ER00   /'0000'/
       INTEGER*2     OK
       INTEGER*2     LINES  /50/
       INTEGER*2     PAGES  /0/
       COMMON /SSC/ INCLUDE IDMS(SUBSCHEMA-CTRL)
       INCLUDE IDMS(EMPLOYEE)
       INCLUDE IDMS(OFFICE)
       INCLUDE IDMS(DEPARTMENT)
       PRNAME = IDMSPP
       BIND(RUN-UNIT,SUBSCHEMA=IDMSSS,DICTNAME=DICTNM)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       WRITE(6,3)
   3   FORMAT(1H1,20X,'BIND RUN-UNIT DONE')
       BIND(RECORD=OFFCE)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       BIND(RECORD=EMPLOY)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       WRITE(6,7)
   7   FORMAT(1H1,20X,'BIND RECORD EMPLOYEE DONE')
       BIND(RECORD=DEPT)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       WRITE(6,9)
   9   FORMAT(1H1,20X,'BIND RECORD DEPT DONE')
       READY(AREA=EMP-DEMO-REGION,RETRIEVAL)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       WRITE(6,12)
```

```
   12  FORMAT(1H1,20X,'READY EMP-DEMO DONE')
       READY(AREA=ORG-DEMO-REGION,RETRIEVAL)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       WRITE(6,15)
   15  FORMAT(1H1,20X,'READY ORG-DEMO DONE')
C
       CALL PAGE(LINES,PAGES)
       OBTAIN(FIRST,RECORD=DEPARTMENT,AREA=ORG-DEMO-REGION)
       IF(ERSTAT.EQ.ER0326) GO TO 99
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
   37  FIND(NEXT RECORD=EMPLOYEE,SET=DEPT-EMPLOYEE)
       IF(ERSTAT.EQ.ER0307) GO TO 99
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       GET(RECORD=EMPLOYEE)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       WRITE(6,40) EMPID,EMPNAME, EMPSTE,EMPZF5
   40  FORMAT(1H ,3X,4A1,6X,25A1,2X,2A1,6X,5A1)
       OBTAIN(OWNER,SET=OFFICE-EMPLOYEE)
       CALL STATUS(OK)
       IF(OK.NE.1) GO TO 99
       IF(OK.EQ.1) GO TO 37
   99  CONTINUE
       FINISH
       STOP
       END

C
C    SUBROUTINE TO CHECK IDMS STATUS:
       SUBROUTINE STATUS(STAT)
       INTEGER*2     STAT
       INTEGER*4     ER00   /'0000'/
       COMMON /SSC/ INCLUDE IDMS(SUBSCHEMA-CTRL)
       STAT=1
       IF(ERSTAT.EQ.ER00) GO TO 20
       STAT=0
       WRITE(6,10) ERSTAT,ERSET,ERREC,ERAREA
   10  FORMAT(1H0,A4,2X,2A8,2X,2A8,2X,2A8)
   20  RETURN
       END
C
C    SUBROUTINE TO SKIP TO A NEW PAGE AND WRITE HEADER
       SUBROUTINE PAGE(LLL,PPP)
       INTEGER*2     LLL
       INTEGER*2     PPP
       LLL=LLL+1
       IF(LLL.LT.50) GO TO 40
       LLL=0
       PPP=PPP+1
       WRITE(6,5) PPP
   5   FORMAT(1H1,8X,'     EMPLOYEE LISTING     ',16X,'PAGE ',I2)
       WRITE(6,15)
   15  FORMAT(1H0,' EMPLOYEE',8X,'EMPLOYEE',13X,'HOME ',5X,'ZIP')
       WRITE(6,25)
   25  FORMAT(1H ,'    ID   ',8X,'  NAME  ',13X,'STATE',5X,'CODE')
   40  RETURN
       END
/*
```

Expanded output from DMLF precompiler: Since the CDMLIST option is specified in the sample program, printed output consists of DMLF-expanded source code as well as diagnostics. This output appears in the following format:

Item	Description
Heading	The top of each page of the listing contains: ■ The name and the release number of the DMLF precompiler. For example, IDMSDMLF 14.1. ■ The name of the listing. For example, LISTING OF MESSAGES. ■ Date ■ Time ■ Page number
Input listing and DMLF-generated code	The body of the printout contains the sample program input listing along with the DMLF-generated code, formatted as follows: ■ Column 1 — Sequence numbers generated by the DMLF precompiler ■ Column 13 — Text of the FORTRAN source code generated by the user and the DMLF precompiler
Warning and error messages	Diagnostics are embedded in the input listing and the DMLF-generated code following any line of source code with errors. See *CA-IDMS Messages and Codes* for a complete description of the DMLF precompiler messages.

```
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                              DML-FORTRAN LISTING
            CDMLIST
DMLF                INVOKE (PROGRAM=SAMPFORT,SUBSCHEMA=EMPSS01,SCHEMA=EMPSCHM,
DMLF                VERSION=100,MODE=BATCH)
                   REAL*8        IDMSPP  /'SAMPFORT'/
                   REAL*8        IDMSSS  /'EMPSS01 '/
                    REAL*8        DICTNM /'APPLDICT'/
                    INTEGER*4     ER0326 /'0326'/
                    INTEGER*4     ER0307 /'0307'/
                    INTEGER*4     ER00   /'0000'/
                    INTEGER*2     OK
                    INTEGER*2     LINES  /50/
                    INTEGER*2     PAGES  /0/
DMLF                COMMON /SSC/ INCLUDE IDMS(SUBSCHEMA-CTRL)
                   COMMON        /SSC/   SSCTRL (224)
                   LOGICAL*1     SSCTRL
                   REAL*8        PRNAME
                                       EQUIVALENCE (SSCTRL (1),
                  +                    PRNAME )
                    INTEGER*4     ERSTAT
                                       EQUIVALENCE (SSCTRL (9),
                  +                    ERSTAT )
                    INTEGER*4     DBKEY
                                       EQUIVALENCE (SSCTRL (13),
                  +                    DBKEY )
                    REAL*8        RCNAME (2)
                                       EQUIVALENCE (SSCTRL (17),
                  +                    RCNAME (1))
                    LOGICAL*1     RRNAME (16)
                                       EQUIVALENCE (SSCTRL (17),
                  +                    RRNAME (1))
                    REAL*8        SSCNOD
                                       EQUIVALENCE (SSCTRL (17),
                  +                    SSCNOD )
                    REAL*8        SSCDBN
                                       EQUIVALENCE (SSCTRL (25),
                  +                    SSCDBN )
                    REAL*8        ARNAME (2)
                                       EQUIVALENCE (SSCTRL (33),
                  +                    ARNAME (1))
```

```
                    LOGICAL*1     AREAR (16)
                                        EQUIVALENCE (SSCTRL (33),
                   +                    AREAR (1))
                    REAL*8        SSCDNO
                                        EQUIVALENCE (SSCTRL (33),
                   +                    SSCDNO )
                    REAL*8        SSCDNA
                                        EQUIVALENCE (SSCTRL (41),
                   +                    SSCDNA )
                    REAL*8        ERSET (2)
                                        EQUIVALENCE (SSCTRL (49),
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                                DML-FORTRAN LISTING
                   +                    ERSET (1))
                    REAL*8        ERREC (2)
                                        EQUIVALENCE (SSCTRL (65),
                   +                    ERREC (1))
                    REAL*8        ERAREA (2)
                                        EQUIVALENCE (SSCTRL (81),
                   +                    ERAREA (1))
                    LOGICAL*1     IDBCM (100)
                                        EQUIVALENCE (SSCTRL (97),
                   +                    IDBCM (1))
                    INTEGER*4     DIRDBK
                                        EQUIVALENCE (SSCTRL (197),
                   +                    DIRDBK )
                    LOGICAL*1     DBSTAT (7)
                                        EQUIVALENCE (SSCTRL (201),
                   +                    DBSTAT (1))
                    LOGICAL*1     DBSTCD (2)
                                        EQUIVALENCE (SSCTRL (201),
                   +                    DBSTCD (1))
                    LOGICAL*1     DBSSCD (5)
                                        EQUIVALENCE (SSCTRL (203),
                   +                    DBSSCD (1))
              CDMLF LOGICAL*1     FILLER (1)
                    INTEGER*4     RECOCC
```

```
                                        EQUIVALENCE (SSCTRL (209),
                     +                  RECOCC )
                      INTEGER*4     DMLSEQ
                                        EQUIVALENCE (SSCTRL (213),
                     +                  DMLSEQ )
                      INTEGER*4     ER0000
                                        EQUIVALENCE (SSCTRL (217),
                     +                  ER0000 )
                      INTEGER*4     ER1601
                                        EQUIVALENCE (SSCTRL (221),
                     +                  ER1601 )
DMLF                   INCLUDE IDMS(EMPLOYEE)
                      LOGICAL*1     EMPLOY (120)
                      LOGICAL*1     EMPID (4)
                                        EQUIVALENCE (EMPLOY (1),
                     +                  EMPID (1))
                      LOGICAL*1     EMPNAM (25)
                                        EQUIVALENCE (EMPLOY (5),
                     +                  EMPNAM (1))
                      LOGICAL*1     EMPFNM (10)
                                        EQUIVALENCE (EMPLOY (5),
                     +                  EMPFNM (1))
                      LOGICAL*1     EMPLNM (15)
                                        EQUIVALENCE (EMPLOY (15),
                     +                  EMPLNM (1))
                      LOGICAL*1     EMPADR (46)
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                                 DML-FORTRAN LISTING
                                        EQUIVALENCE (EMPLOY (30),
                     +                  EMPADR (1))
                      LOGICAL*1     EMPST (20)
                                        EQUIVALENCE (EMPLOY (30),
                     +                  EMPST (1))
                      LOGICAL*1     EMPCTY (15)
                                        EQUIVALENCE (EMPLOY (50),
                     +                  EMPCTY (1))
```

```
      LOGICAL*1     EMPSTE (2)
                          EQUIVALENCE (EMPLOY (65),
     +                    EMPSTE (1))
      LOGICAL*1     EMPZIP (9)
                          EQUIVALENCE (EMPLOY (67),
     +                    EMPZIP (1))
      LOGICAL*1     EMPZF5 (5)
                          EQUIVALENCE (EMPLOY (67),
     +                    EMPZF5 (1))
      LOGICAL*1     EMPZL4 (4)
                          EQUIVALENCE (EMPLOY (72),
     +                    EMPZL4 (1))
      LOGICAL*1     EMPPHN (10)
                          EQUIVALENCE (EMPLOY (76),
     +                    EMPPHN (1))
      LOGICAL*1     EMPSTA (2)
                          EQUIVALENCE (EMPLOY (86),
     +                    EMPSTA (1))
CDMLF               ACTIVE
CDMLF               STDSBL
CDMLF               LTDSBL
CDMLF               LVOFAB
CDMLF               TRMNTD
      LOGICAL*1     EMPSS (9)
                          EQUIVALENCE (EMPLOY (88),
     +                    EMPSS (1))
      LOGICAL*1     EMPSDT (6)
                          EQUIVALENCE (EMPLOY (97),
     +                    EMPSDT (1))
      LOGICAL*1     EMPSYR (2)
                          EQUIVALENCE (EMPLOY (97),
     +                    EMPSYR (1))
      LOGICAL*1     EMPSMO (2)
                          EQUIVALENCE (EMPLOY (99),
     +                    EMPSMO (1))
      LOGICAL*1     EMPSDA (2)
                          EQUIVALENCE (EMPLOY (101),
     +                    EMPSDA (1))
      LOGICAL*1     EMPTDT (6)
                          EQUIVALENCE (EMPLOY (103),
     +                    EMPTDT (1))
      LOGICAL*1     EMPTYR (2)
```

```
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                                DML-FORTRAN LISTING
                                        EQUIVALENCE (EMPLOY (103),
                 +                      EMPTYR (1))
                  LOGICAL*1     EMPTMO (2)
                                        EQUIVALENCE (EMPLOY (105),
                 +                      EMPTMO (1))
                  LOGICAL*1     EMPTDA (2)
                                        EQUIVALENCE (EMPLOY (107),
                 +                      EMPTDA (1))
                  LOGICAL*1     EMPBDT (6)
                                        EQUIVALENCE (EMPLOY (109),
                 +                      EMPBDT (1))
                  LOGICAL*1     EMPBYR (2)
                                        EQUIVALENCE (EMPLOY (109),
                 +                      EMPBYR (1))
                  LOGICAL*1     EMPBMO (2)
                                        EQUIVALENCE (EMPLOY (111),
                 +                      EMPBMO (1))
                  LOGICAL*1     EMPBDA (2)
                                        EQUIVALENCE (EMPLOY (113),
                 +                      EMPBDA (1))
             CDMLF LOGICAL*1     FILLER (2)
DMLF               INCLUDE IDMS(OFFICE)
                  LOGICAL*1     OFFCE (80)
                  LOGICAL*1     OFCODE (3)
                                        EQUIVALENCE (OFFCE (1),
                 +                      OFCODE (1))
                  LOGICAL*1     OFADDR (46)
                                        EQUIVALENCE (OFFCE (4),
                 +                      OFADDR (1))
                  LOGICAL*1     OFSTRT (20)

                                        EQUIVALENCE (OFFCE (4),
                 +                      OFSTRT (1))
                  LOGICAL*1     OFCITY (15)
                                        EQUIVALENCE (OFFCE (24),
                 +                      OFCITY (1))
                  LOGICAL*1     OFSTAT (2)
                                        EQUIVALENCE (OFFCE (39),
                 +                      OFSTAT (1))
                  LOGICAL*1     OFZIP (9)
                                        EQUIVALENCE (OFFCE (41),
                 +                      OFZIP (1))
                  LOGICAL*1     OFZPF5 (5)
                                        EQUIVALENCE (OFFCE (41),
                 +                      OFZPF5 (1))
                  LOGICAL*1     OFZPL4 (4)
                                        EQUIVALENCE (OFFCE (46),
                 +                      OFZPL4 (1))
                  LOGICAL*1     OFPHON (7,3)
                                        EQUIVALENCE (OFFCE (50),
                 +                      OFPHON (1,1))
```

```
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                                DML-FORTRAN LISTING
                  LOGICAL*1     OFAREA (3)
                                      EQUIVALENCE (OFFCE (71),
                 +                    OFAREA (1))
                  LOGICAL*1     OFSPED (3)
                                      EQUIVALENCE (OFFCE (74),
                 +                    OFSPED (1))
DMLF                INCLUDE IDMS(DEPARTMENT)
                  LOGICAL*1     DEPT (56)
                  LOGICAL*1     DPID (4)
                                      EQUIVALENCE (DEPT (1),
                 +                    DPID (1))
                  LOGICAL*1     DPNAME (45)
                                      EQUIVALENCE (DEPT (5),
                 +                    DPNAME (1))
                  LOGICAL*1     DPHDID (4)
                                      EQUIVALENCE (DEPT (50),
                 +                    DPHDID (1))
            CDMLF LOGICAL*1     FILLER (3)
                   PRNAME = IDMSPP
DMLF0001           BIND(RUN-UNIT,SUBSCHEMA=IDMSSS,DICTNAME=DICTNM)
                   CALL STATUS(OK)
                   IF(OK.NE.1) GO TO 99
                   WRITE(6,3)
               3   FORMAT(1H1,20X,'BIND RUN-UNIT DONE')
DMLF0002           BIND(RECORD=OFFCE)
                   CALL STATUS(OK)
                   IF(OK.NE.1) GO TO 99
DMLF0003           BIND(RECORD=EMPLOY)
                   CALL STATUS(OK)
                   IF(OK.NE.1) GO TO 99
                   WRITE(6,7)
```

```
                7   FORMAT(1H1,20X,'BIND RECORD EMPLOYEE DONE')
DMLF0004            BIND(RECORD=DEPT)
                    CALL STATUS(OK)
                    IF(OK.NE.1) GO TO 99
                    WRITE(6,9)
                9   FORMAT(1H1,20X,'BIND RECORD DEPT DONE')
DMLF0005            READY(AREA=EMP-DEMO-REGION,RETRIEVAL)
                    CALL STATUS(OK)
                    IF(OK.NE.1) GO TO 99
                    WRITE(6,12)
                12  FORMAT(1H1,20X,'READY EMP-DEMO DONE')
DMLF0006            READY(AREA=ORG-DEMO-REGION,RETRIEVAL)
                    CALL STATUS(OK)
                    IF(OK.NE.1) GO TO 99
                    WRITE(6,15)
                15  FORMAT(1H1,20X,'READY ORG-DEMO DONE')
            C
                    CALL PAGE(LINES,PAGES)
DMLF0007            OBTAIN(FIRST,RECORD=DEPARTMENT,AREA=ORG-DEMO-REGION)
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                                  DML-FORTRAN LISTING
                    IF(ERSTAT.EQ.ER0326) GO TO 99
                    CALL STATUS(OK)
                    IF(OK.NE.1) GO TO 99
DMLF0008        37  FIND(NEXT RECORD=EMPLOYEE,SET=DEPT-EMPLOYEE)
                    IF(ERSTAT.EQ.ER0307) GO TO 99
                    CALL STATUS(OK)
                    IF(OK.NE.1) GO TO 99
DMLF0009            GET(RECORD=EMPLOYEE)
                    CALL STATUS(OK)
                    IF(OK.NE.1) GO TO 99
                    WRITE(6,40) EMPID,EMPNAME, EMPSTE,EMPZF5

                40  FORMAT(1H ,3X,4A1,6X,25A1,2X,2A1,6X,5A1)
DMLF0010            OBTAIN(OWNER,SET=OFFICE-EMPLOYEE)
                    CALL STATUS(OK)
                    IF(OK.NE.1) GO TO 99
                    IF(OK.EQ.1) GO TO 37
                99  CONTINUE
DMLF0011            FINISH
                    STOP
                    END
            C
            C    SUBROUTINE TO CHECK IDMS STATUS:
                    SUBROUTINE STATUS(STAT)
                    INTEGER*2     STAT
                    INTEGER*4     ER00   /'0000'/
DMLF                COMMON /SSC/ INCLUDE IDMS(SUBSCHEMA-CTRL)
                   COMMON        /SSC/   SSCTRL (224)
                   LOGICAL*1     SSCTRL
                   REAL*8        PRNAME
                                       EQUIVALENCE (SSCTRL (1),
                  +                    PRNAME )
                   INTEGER*4     ERSTAT
                                       EQUIVALENCE (SSCTRL (9),
                  +                    ERSTAT )
                   INTEGER*4     DBKEY
                                       EQUIVALENCE (SSCTRL (13),
                  +                    DBKEY )
                   REAL*8        RCNAME (2)
                                       EQUIVALENCE (SSCTRL (17),
                  +                    RCNAME (1))
                   LOGICAL*1     RRNAME (16)
                                       EQUIVALENCE (SSCTRL (17),
                  +                    RRNAME (1))
```

```
                    REAL*8        SSCNOD
                                        EQUIVALENCE (SSCTRL (17),
                   +                    SSCNOD )
                    REAL*8        SSCDBN
                                        EQUIVALENCE (SSCTRL (25),
                   +                    SSCDBN )
                    REAL*8        ARNAME (2)
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                                  DML-FORTRAN LISTING
                                        EQUIVALENCE (SSCTRL (33),
                   +                    ARNAME (1))
                    LOGICAL*1     AREAR (16)
                                        EQUIVALENCE (SSCTRL (33),
                   +                    AREAR (1))
                    REAL*8        SSCDNO
                                        EQUIVALENCE (SSCTRL (33),
                   +                    SSCDNO )
                    REAL*8        SSCDNA
                                        EQUIVALENCE (SSCTRL (41),
                   +                    SSCDNA )
                    REAL*8        ERSET (2)
                                        EQUIVALENCE (SSCTRL (49),
                   +                    ERSET (1))
                    REAL*8        ERREC (2)
                                        EQUIVALENCE (SSCTRL (65),
                   +                    ERREC (1))
                    REAL*8        ERAREA (2)
                                        EQUIVALENCE (SSCTRL (81),
                   +                    ERAREA (1))
                    LOGICAL*1     IDBCM (100)
                                        EQUIVALENCE (SSCTRL (97),
                   +                    IDBCM (1))
                    INTEGER*4     DIRDBK
                                        EQUIVALENCE (SSCTRL (197),
                   +                    DIRDBK )
                    LOGICAL*1     DBSTAT (7)
                                        EQUIVALENCE (SSCTRL (201),
                   +                    DBSTAT (1))
                    LOGICAL*1     DBSTCD (2)
                                        EQUIVALENCE (SSCTRL (201),
                   +                    DBSTCD (1))
                    LOGICAL*1     DBSSCD (5)
                                        EQUIVALENCE (SSCTRL (203),
                   +                    DBSSCD (1))
```

```
            CDMLF LOGICAL*1     FILLER (1)
                  INTEGER*4     RECOCC
                                      EQUIVALENCE (SSCTRL (209),
                 +                    RECOCC )
                  INTEGER*4     DMLSEQ
                                      EQUIVALENCE (SSCTRL (213),
                 +                    DMLSEQ )
                  INTEGER*4     ER0000
                                      EQUIVALENCE (SSCTRL (217),
                 +                    ER0000 )
                  INTEGER*4     ER1601
                                      EQUIVALENCE (SSCTRL (221),
                 +                    ER1601 )
                    STAT=1
                    IF(ERSTAT.EQ.ER00) GO TO 20
IDMSDMLF  14.1   CAGJC0   COMPUTER ASSOCIATES INTL.      DML PROCESSOR FOR FORTR
             *                              DML-FORTRAN LISTING
                    STAT=0
                    WRITE(6,10) ERSTAT,ERSET,ERREC,ERAREA
                 10 FORMAT(1H0,A4,2X,2A8,2X,2A8,2X,2A8)
                 20 RETURN
                    END
            C
            C    SUBROUTINE TO SKIP TO A NEW PAGE AND WRITE HEADER
                    SUBROUTINE PAGE(LLL,PPP)
                    INTEGER*2     LLL
                    INTEGER*2     PPP
                    LLL=LLL+1
                    IF(LLL.LT.50) GO TO 40
                    LLL=0
                    PPP=PPP+1
                    WRITE(6,5) PPP
                 5  FORMAT(1H1,8X,'     EMPLOYEE LISTING     ',16X,'PAGE ',I2)
                    WRITE(6,15)
                 15 FORMAT(1H0,' EMPLOYEE',8X,'EMPLOYEE',13X,'HOME ',5X,'ZIP')
                    WRITE(6,25)
                 25 FORMAT(1H ,'    ID   ',8X,'  NAME  ',13X,'STATE',5X,'CODE')
                 40 RETURN
                    END

0000 MESSAGES FOR PROGRAM SAMPFORT
```

Program after processing by the FORTRAN compiler: The sample program after it has been processed by the FORTRAN compiler is shown below. The original code has been further expanded and includes:

- Line numbers generated by the FORTRAN compiler
- IDMS call statements for the requested DML functions
- Diagnostic messages
- DMLF statement sequence numbers listed in FORTRAN columns 73-80

```
LEVEL 21.8 ( JUN 74 )                          OS/360  FORTRAN H
          COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=58,SIZE=0000K,
                             SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF
               CRETREVIAL
               CDMLIST
               CDMLF  INVOKE (PROGRAM=SAMPFORT,SUBSCHEMA=EMPSS01,SCHEMA=EMPSCHM
               CDMLF  VERSION=100,MODE=BATCH)
  ISN 0002           REAL*8        IDMSPP  /'SAMPFORT'/
  ISN 0003           REAL*8        IDMSSS  /'EMPSS01 '/
  ISN 0004            REAL*8        DICTNM /'APPLDICT'/
  ISN 0005            INTEGER*4     ER0326 /'0326'/
  ISN 0006            INTEGER*4     ER0307 /'0307'/
  ISN 0007            INTEGER*4     ER00   /'0000'/
  ISN 0008            INTEGER*2     OK
  ISN 0009            INTEGER*2     LINES  /50/
  ISN 0010            INTEGER*2     PAGES  /0/
               CDMLF  COMMON /SSC/ INCLUDE IDMS(SUBSCHEMA-CTRL)
  ISN 0011           COMMON        /SSC/   SSCTRL (224)
  ISN 0012           LOGICAL*1     SSCTRL
  ISN 0013           REAL*8        PRNAME
  ISN 0014                                 EQUIVALENCE (SSCTRL (1),
                    +                       PRNAME )
  ISN 0015            INTEGER*4     ERSTAT
  ISN 0016                                 EQUIVALENCE (SSCTRL (9),
                    +                       ERSTAT )
  ISN 0017            INTEGER*4     DBKEY
  ISN 0018                                 EQUIVALENCE (SSCTRL (13),
                    +                       DBKEY )
  ISN 0019            REAL*8        RCNAME (2)
  ISN 0020                                 EQUIVALENCE (SSCTRL (17),
                    +                       RCNAME (1))

  ISN 0021            LOGICAL*1     RRNAME (16)
  ISN 0022                                 EQUIVALENCE (SSCTRL (17),
                    +                       RRNAME (1))
  ISN 0023            REAL*8        SSCNOD
  ISN 0024                                 EQUIVALENCE (SSCTRL (17),
                    +                       SSCNOD )
  ISN 0025            REAL*8        SSCDBN
  ISN 0026                                 EQUIVALENCE (SSCTRL (25),
                    +                       SSCDBN )
  ISN 0027            REAL*8        ARNAME (2)
  ISN 0028                                 EQUIVALENCE (SSCTRL (33),
                    +                       ARNAME (1))
  ISN 0029            LOGICAL*1     AREAR (16)
  ISN 0030                                 EQUIVALENCE (SSCTRL (33),
                    +                       AREAR (1))
  ISN 0031            REAL*8        SSCDNO
  ISN 0032                                 EQUIVALENCE (SSCTRL (33),
                    +                       SSCDNO )
  ISN 0033            REAL*8        SSCDNA
  ISN 0034                                 EQUIVALENCE (SSCTRL (41),
                    +                       SSCDNA )
  ISN 0035            REAL*8        ERSET (2)
  ISN 0036                                 EQUIVALENCE (SSCTRL (49),
                    +                       ERSET (1))
  ISN 0037            REAL*8        ERREC (2)
  ISN 0038                                 EQUIVALENCE (SSCTRL (65),

                    +                       ERREC (1))
```

```
ISN 0039          REAL*8        ERAREA (2)
ISN 0040                              EQUIVALENCE (SSCTRL (81),
                +                     ERAREA (1))
ISN 0041          LOGICAL*1     IDBCM (100)
ISN 0042                              EQUIVALENCE (SSCTRL (97),
                +                     IDBCM (1))
ISN 0043          INTEGER*4     DIRDBK
ISN 0044                              EQUIVALENCE (SSCTRL (197),
                +                     DIRDBK )
ISN 0045          LOGICAL*1     DBSTAT (7)
ISN 0046                              EQUIVALENCE (SSCTRL (201),
                +                     DBSTAT (1))
ISN 0047          LOGICAL*1     DBSTCD (2)
ISN 0048                              EQUIVALENCE (SSCTRL (201),
                +                     DBSTCD (1))
ISN 0049          LOGICAL*1     DBSSCD (5)
ISN 0050                              EQUIVALENCE (SSCTRL (203),
                +                     DBSSCD (1))
              CDMLF LOGICAL*1     FILLER (1)
ISN 0051          INTEGER*4     RECOCC
ISN 0052                              EQUIVALENCE (SSCTRL (209),
                +                     RECOCC )
ISN 0053          INTEGER*4     DMLSEQ
ISN 0054                              EQUIVALENCE (SSCTRL (213),
                +                     DMLSEQ )
ISN 0055          INTEGER*4     ER0000
ISN 0056                              EQUIVALENCE (SSCTRL (217),
                +                     ER0000 )
ISN 0057          INTEGER*4     ER1601
ISN 0058                              EQUIVALENCE (SSCTRL (221),
                +                     ER1601 )
              CDMLF  INCLUDE IDMS(EMPLOYEE)
ISN 0059          LOGICAL*1     EMPLOY (120)
ISN 0060          LOGICAL*1     EMPID (4)
ISN 0061                              EQUIVALENCE (EMPLOY (1),
                +                     EMPID (1))

ISN 0062          LOGICAL*1     EMPNAM (25)
ISN 0063                              EQUIVALENCE (EMPLOY (5),
                +                     EMPNAM (1))
ISN 0064          LOGICAL*1     EMPFNM (10)
ISN 0065                              EQUIVALENCE (EMPLOY (5),
                +                     EMPFNM (1))
ISN 0066          LOGICAL*1     EMPLNM (15)
ISN 0067                              EQUIVALENCE (EMPLOY (15),
                +                     EMPLNM (1))
ISN 0068          LOGICAL*1     EMPADR (46)
ISN 0069                              EQUIVALENCE (EMPLOY (30),
                +                     EMPADR (1))
ISN 0070          LOGICAL*1     EMPST (20)
ISN 0071                              EQUIVALENCE (EMPLOY (30),
                +                     EMPST (1))
ISN 0072          LOGICAL*1     EMPCTY (15)
ISN 0073                              EQUIVALENCE (EMPLOY (50),
                +                     EMPCTY (1))
ISN 0074          LOGICAL*1     EMPSTE (2)

ISN 0075                              EQUIVALENCE (EMPLOY (65),
                +                     EMPSTE (1))
ISN 0076          LOGICAL*1     EMPZIP (9)
ISN 0077                              EQUIVALENCE (EMPLOY (67),
                +                     EMPZIP (1))
ISN 0078          LOGICAL*1     EMPZF5 (5)
```

```
ISN 0079                                EQUIVALENCE (EMPLOY (67),
                 +                      EMPZF5 (1))
ISN 0080          LOGICAL*1     EMPZL4 (4)
ISN 0081                                EQUIVALENCE (EMPLOY (72),
                 +                      EMPZL4 (1))
ISN 0082          LOGICAL*1     EMPPHN (10)
ISN 0083                                EQUIVALENCE (EMPLOY (76),
                 +                      EMPPHN (1))
ISN 0084          LOGICAL*1     EMPSTA (2)
ISN 0085                                EQUIVALENCE (EMPLOY (86),
                 +                      EMPSTA (1))
             CDMLF               ACTIVE
             CDMLF               STDSBL
             CDMLF               LTDSBL
             CDMLF               LVOFAB
             CDMLF               TRMNTD
ISN 0086          LOGICAL*1     EMPSS (9)
ISN 0087                                EQUIVALENCE (EMPLOY (88),
                 +                      EMPSS (1))
ISN 0088          LOGICAL*1     EMPSDT (6)
ISN 0089                                EQUIVALENCE (EMPLOY (97),
                 +                      EMPSDT (1))
ISN 0090          LOGICAL*1     EMPSYR (2)
ISN 0091                                EQUIVALENCE (EMPLOY (97),
                 +                      EMPSYR (1))
ISN 0092          LOGICAL*1     EMPSMO (2)
ISN 0093                                EQUIVALENCE (EMPLOY (99),
                 +                      EMPSMO (1))
ISN 0094          LOGICAL*1     EMPSDA (2)
ISN 0095                                EQUIVALENCE (EMPLOY (101),
                 +                      EMPSDA (1))
ISN 0096          LOGICAL*1     EMPTDT (6)
ISN 0097                                EQUIVALENCE (EMPLOY (103),
                 +                      EMPTDT (1))
ISN 0098          LOGICAL*1     EMPTYR (2)
ISN 0099                                EQUIVALENCE (EMPLOY (103),
                 +                      EMPTYR (1))
ISN 0100          LOGICAL*1     EMPTMO (2)
ISN 0101                                EQUIVALENCE (EMPLOY (105),
                 +                      EMPTMO (1))
```

```
ISN 0102          LOGICAL*1     EMPTDA (2)
ISN 0103                              EQUIVALENCE (EMPLOY (107),
                +                     EMPTDA (1))
ISN 0104          LOGICAL*1     EMPBDT (6)
ISN 0105                              EQUIVALENCE (EMPLOY (109),
                +                     EMPBDT (1))
ISN 0106          LOGICAL*1     EMPBYR (2)
ISN 0107                              EQUIVALENCE (EMPLOY (109),
                +                     EMPBYR (1))
ISN 0108          LOGICAL*1     EMPBMO (2)

ISN 0109                              EQUIVALENCE (EMPLOY (111),
                +                     EMPBMO (1))
ISN 0110          LOGICAL*1     EMPBDA (2)
ISN 0111                              EQUIVALENCE (EMPLOY (113),
                +                     EMPBDA (1))
            CDMLF LOGICAL*1     FILLER (2)
            CDMLF  INCLUDE IDMS(OFFICE)
ISN 0112          LOGICAL*1     OFFCE (80)
ISN 0113          LOGICAL*1     OFCODE (3)
ISN 0114                              EQUIVALENCE (OFFCE (1),
                +                     OFCODE (1))
ISN 0115          LOGICAL*1     OFADDR (46)
ISN 0116                              EQUIVALENCE (OFFCE (4),
                +                     OFADDR (1))
ISN 0117          LOGICAL*1     OFSTRT (20)
ISN 0118                              EQUIVALENCE (OFFCE (4),
                +                     OFSTRT (1))
ISN 0119          LOGICAL*1     OFCITY (15)
ISN 0120                              EQUIVALENCE (OFFCE (24),
                +                     OFCITY (1))
ISN 0121          LOGICAL*1     OFSTAT (2)

ISN 0122                              EQUIVALENCE (OFFCE (39),
                +                     OFSTAT (1))
ISN 0123          LOGICAL*1     OFZIP (9)
ISN 0124                              EQUIVALENCE (OFFCE (41),
                +                     OFZIP (1))
ISN 0125          LOGICAL*1     OFZPF5 (5)
ISN 0126                              EQUIVALENCE (OFFCE (41),
                +                     OFZPF5 (1))
ISN 0127          LOGICAL*1     OFZPL4 (4)
ISN 0128                              EQUIVALENCE (OFFCE (46),
                +                     OFZPL4 (1))
ISN 0129          LOGICAL*1     OFPHON (7,3)
ISN 0130                              EQUIVALENCE (OFFCE (50),
                +                     OFPHON (1,1))
ISN 0131          LOGICAL*1     OFAREA (3)
ISN 0132                              EQUIVALENCE (OFFCE (71),
                +                     OFAREA (1))
ISN 0133          LOGICAL*1     OFSPED (3)
ISN 0134                              EQUIVALENCE (OFFCE (74),
                +                     OFSPED (1))
            CDMLF  INCLUDE IDMS(DEPARTMENT)
ISN 0135          LOGICAL*1     DEPT (56)
ISN 0136          LOGICAL*1     DPID (4)
ISN 0137                              EQUIVALENCE (DEPT (1),
                +                     DPID (1))
ISN 0138          LOGICAL*1     DPNAME (45)
ISN 0139                              EQUIVALENCE (DEPT (5),
                +                     DPNAME (1))
ISN 0140          LOGICAL*1     DPHDID (4)
```

```
ISN 0141                              EQUIVALENCE (DEPT (50),
                 +                    DPHDID (1))
             CDMLF LOGICAL*1     FILLER (3)
ISN 0142            PRNAME = IDMSPP
             CDMLF  BIND(RUN-UNIT,SUBSCHEMA=IDMSSS,DICTNAME=DICTNM)
ISN 0143               CALL IDMSMV(SSCTRL,'        ',8,33)

ISN 0144               CALL IDMSMV(SSCTRL,DICTNM,8,41)
ISN 0145               CALL IDMS(SSCTRL
                 *             ,IDBCM (59)
                 *             ,SSCTRL
                 *             ,IDMSSS
                 *             ,SSCTRL
                 *             ,SSCTRL
                 *             ,SSCDNO
                 *                            )
ISN 0146            CALL STATUS(OK)
ISN 0147            IF(OK.NE.1) GO TO 99
ISN 0149            WRITE(6,3)
ISN 0150        3   FORMAT(1H1,20X,'BIND RUN-UNIT DONE')
             CDMLF  BIND(RECORD=OFFCE)
ISN 0151               CALL IDMS(SSCTRL
                 *             ,IDBCM (48)
                 *             ,'OFFICE          '
                 *             ,OFFCE
                 *                            )
ISN 0152            CALL STATUS(OK)
ISN 0153            IF(OK.NE.1) GO TO 99
             CDMLF  BIND(RECORD=EMPLOY)
```

```
ISN 0155              CALL IDMS(SSCTRL
                 *             ,IDBCM (48)
                 *             ,'EMPLOYEE        '
                 *             ,EMPLOY
                 *                            )
ISN 0156           CALL STATUS(OK)
ISN 0157           IF(OK.NE.1) GO TO 99
ISN 0159           WRITE(6,7)
ISN 0160       7   FORMAT(1H1,20X,'BIND RECORD EMPLOYEE DONE')
            CDMLF  BIND(RECORD=DEPT)
ISN 0161              CALL IDMS(SSCTRL
                 *             ,IDBCM (48)
                 *             ,'DEPARTMENT      '
                 *             ,DEPT
                 *                            )
ISN 0162           CALL STATUS(OK)
ISN 0163           IF(OK.NE.1) GO TO 99
ISN 0165           WRITE(6,9)
ISN 0166       9   FORMAT(1H1,20X,'BIND RECORD DEPT DONE')
            CDMLF  READY(AREA=EMP-DEMO-REGION,RETRIEVAL)
ISN 0167              CALL IDMS(SSCTRL
                 *             ,IDBCM (37)
                 *             ,'EMP-DEMO-REGION '
                 *                            )
ISN 0168           CALL STATUS(OK)
ISN 0169           IF(OK.NE.1) GO TO 99
ISN 0171           WRITE(6,12)
ISN 0172      12   FORMAT(1H1,20X,'READY EMP-DEMO DONE')
            CDMLF  READY(AREA=ORG-DEMO-REGION,RETRIEVAL)
ISN 0173              CALL IDMS(SSCTRL
                 *             ,IDBCM (37)
                 *             ,'ORG-DEMO-REGION '
                 *                            )
ISN 0174           CALL STATUS(OK)

ISN 0175           IF(OK.NE.1) GO TO 99
ISN 0177           WRITE(6,15)
ISN 0178      15   FORMAT(1H1,20X,'READY ORG-DEMO DONE')
            C
```

```
ISN 0179            CALL PAGE(LINES,PAGES)
             CDMLF  OBTAIN(FIRST,RECORD=DEPARTMENT,AREA=ORG-DEMO-REGION)
ISN 0180               CALL IDMS(SSCTRL
                  *             ,IDBCM (19)
                  *             ,'DEPARTMENT      '
                  *             ,'ORG-DEMO-REGION '
                  *             ,IDBCM (43)
                  *                            )
ISN 0181            IF(ERSTAT.EQ.ER0326) GO TO 99
ISN 0183            CALL STATUS(OK)
ISN 0184            IF(OK.NE.1) GO TO 99
             CDMLF  FIND(NEXT RECORD=EMPLOYEE,SET=DEPT-EMPLOYEE)
ISN 0186        37     CALL IDMS(SSCTRL
                  *             ,IDBCM (10)
                  *             ,'EMPLOYEE        '
                  *             ,'DEPT-EMPLOYEE   '
                  *                            )
ISN 0187            IF(ERSTAT.EQ.ER0307) GO TO 99
ISN 0189            CALL STATUS(OK)
ISN 0190            IF(OK.NE.1) GO TO 99
             CDMLF  GET(RECORD=EMPLOYEE)
ISN 0192               CALL IDMS(SSCTRL
                  *             ,IDBCM (34)
                  *             ,'EMPLOYEE        '
                  *                            )
ISN 0193            CALL STATUS(OK)
ISN 0194            IF(OK.NE.1) GO TO 99
ISN 0196            WRITE(6,40) EMPID,EMPNAME, EMPSTE,EMPZF5
ISN 0197        40  FORMAT(1H ,3X,4A1,6X,25A1,2X,2A1,6X,5A1)
             CDMLF  OBTAIN(OWNER,SET=OFFICE-EMPLOYEE)
ISN 0198               CALL IDMS(SSCTRL
                  *             ,IDBCM (31)
                  *             ,'OFFICE-EMPLOYEE '
                  *             ,IDBCM (43)
                  *                            )
ISN 0199            CALL STATUS(OK)
ISN 0200            IF(OK.NE.1) GO TO 99
ISN 0202            IF(OK.EQ.1) GO TO 37

ISN 0204        99  CONTINUE
             CDMLF  FINISH
ISN 0205               CALL IDMS(SSCTRL
                  *             ,IDBCM (02)
                  *                            )
ISN 0206            STOP
ISN 0207            END
```

Report listing from sample program:

```
            EMPLOYEE LISTING                      PAGE  1
EMPLOYEE        EMPLOYEE             HOME      ZIP
   ID             NAME              STATE     CODE
  0069      JUNE      BLOOMER         MA      01675
  0100      EDWARD    HUTTON          MA      02176
  0011      RUPERT    JENSON          MA      02176
  0067      MARIANNE  KIMBALL         MA      01239
  0106      DORIS     KING            MA      02176
  0101      BRIAN     NICEMAN         MA      02176
```

Appendix L. RPG II DML Precompiler

L.1 About this appendix L-3
L.2 The DMLR precompiler (IDMSDMLR) L-4
L.2.1 Standard and non-standard data transfer L-5
L.3 Programming considerations L-7
L.3.1 IDMS communications block L-7
L.3.2 Synonym processing L-9
L.3.3 Copying non-IDMS record descriptions L-10
L.3.4 Logical Record Facility programming considerations L-10
L.3.4.1 LRC block L-11
L.3.4.2 Path status L-11
L.3.4.3 DML statement restrictions L-12
L.3.5 Precompiler options L-13
L.4 Database access under standard RPG II data transfer L-15
L.4.1 Compiler-directive statements L-16
L.4.1.1 Subschema/Schema identification L-17
L.4.1.2 Defining records/logical records L-20
L.4.1.3 Copying input records/logical records L-22
L.4.1.4 Copying output records/logical records L-25
L.4.1.5 Copying modules L-27
L.5 Database access under non-standard RPG II data transfer L-29
L.5.1 Compiler-directive statements L-30
L.5.1.1 Subschema/Schema identification L-30
L.5.1.2 Copying subschema records L-33
L.5.1.3 Copying individual record descriptions L-34
L.5.1.4 Copying modules L-35
L.5.1.5 Binding records L-36
L.5.2 Functional DML statements L-37
L.5.3 Control Statements L-38
L.5.4 Logical record DML statements L-39
L.6 Database access under combined RPG II data transfer L-40
L.7 Compiling and link-editing L-42
L.7.1 Compiling an RPG II Program L-42
L.7.2 JCL L-43
L.8 Link-edit considerations L-47
L.9 Standard RPG II data transfer return codes L-48

L.1 About this appendix

RPG II application programs use CA-IDMS RPG II Data Manipulation Language (DML) statements to access a CA-IDMS database. The RPG II programmer embeds DML statements within the RPG II program as if they were a part of the host language itself. The RPG II program is first submitted to the DML RPG II precompiler for preprocessing and then to the RPG II compiler for compilation.

For the most part, RPG II programs require the same process and rules as COBOL programs that access a CA-IDMS database. There are some special considerations for RPG II programs both standard and non-standard RPG II data transfer.

This appendix describes these programming considerations. You should use the information in this appendix along with Parts I and II of this document when coding CA-IDMS RPG II applications. Note that the DMLR precompiler supports only batch RPG II programs and therefore DML statements specific to the online environment are not used with RPG II programs.

L.2 The DMLR precompiler (IDMSDMLR)

The DMLR precompiler (IDMSDMLR) executes as a step preceding execution of the RPG II compiler. Input to the DMLR precompiler consists of a source statement file of RPG II and DML statements. Output from the DMLR precompiler is a source (card-image) file that serves as the input to the RPG II compiler.

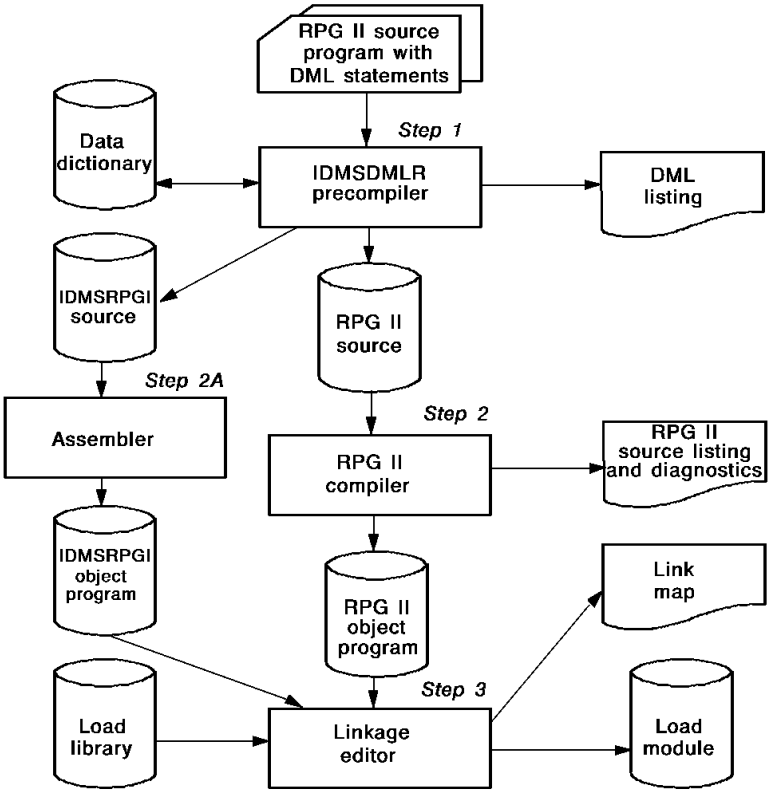
Functions of the DMLR precompiler The DMLR precompiler performs the following functions:

- Adds source code (both descriptive and procedural) to the program from the dictionary, as requested by the user.
- Changes CA-IDMS DML statements to RPG II comment entries.
- Generates appropriate CALL statements for DML requests involving transfer of runtime control to CA-IDMS.
- Generates appropriate File Specification entries for RPG II special routines used to access CA-IDMS database records.
- Ensures that all statements issued by the program are consistent with the logical structure of the database, the subschema view of the program, and the access restrictions defined in the subschema.
- Updates the dictionary with compile-time statistics used to monitor database activities for a given application program.
- Supports the use of native VSAM files in conjunction with database access functions.
- Recognizes record and element synonyms that are defined in the dictionary.
- Copies source modules and nondatabase record descriptions that are stored in the dictionary into application programs.

Optional source statement listing: The DMLR precompiler also generates an optional source statement listing that informs the user of incorrect DML entries, statements inconsistent with the program's declared subschema view, and other error conditions detected during DMLR processing. The listing includes warning messages issued by the DMLR precompiler for source code conditions that might adversely affect run units using the code.

After the source program has been processed by the DMLR precompiler, the generated RPG II source statements must be compiled and link-edited, as illustrated in the following diagram.

Compiling an RPG II program with DML statements



IDMSRPGI interface routine: The DMLR precompiler also generates an interface routine called IDMSRPGI, which serves as a runtime interface between the program and CA-IDMS. Application programs that use standard RPG II data transfer, which is introduced below, require this interface in order to request database services. IDMSRGPI must be assembled and then link edited with the object program. For non-standard data transfer, the DMLR precompiler generates a non-standard RPG II data transfer.

Standard and non-standard data transfer are introduced below.

L.2.1 Standard and non-standard data transfer

Any program that accesses the database must issue requests to CA-IDMS for database services. RPG II applications can issue these requests in two different ways, according to two types of RPG II data transfer supported by the DMLR precompiler. These types of data transfer are characterized as follows:

- **Standard data transfer** — Follows the typical RPG II cycle of operation, requiring certain entries in the File, Input, and Output Specifications to define and process database records and logical records. Each database record and logical record is represented as an RPG II file defined in the File Specifications.
- **Non-standard data transfer** — Involves use of logic outside of the typical RPG II cycle to initiate, perform, and terminate CA-IDMS database functions. Under

non-standard data transfer, no files are defined in the File Specifications for database records or logical records; instead, these records are defined in the program variable storage as data structures. Functional DML statements, coded in the Calculation Specifications, request database services for records and logical records.

Any database record or logical record can be processed by using either the standard approach or the non-standard approach. The type of data transfer selected determines the logic and code required by the application to manipulate specific record occurrences.

Standard data transfer: Standard and non-standard data transfer also differ in their capabilities, such as in the database access techniques for each. RPG II applications using standard data transfer are restricted to the following database access techniques:

- Access to database record occurrences through area sweeps
- Access to native VSAM data sets through native VSAM indexes
- Access to logical records through the Logical Record Facility (LRF)
- Access to native VSAM data sets through the Logical Record Facility

Non-standard data transfer: Under non-standard data transfer, RPG II programs can perform all of the above database access techniques, as well as the following:

- Access to database record occurrences through CALC, database, and sort keys
- Access to database record occurrences through set relationships

Combined standard and non-standard data transfer: An individual RPG II program can perform both standard and non-standard data transfer. In this case, certain database records and/or logical records are accessed through standard techniques. Special considerations for combined standard and non-standard data transfer are discussed in L.6, “Database access under combined RPG II data transfer” on page L-40 later in this appendix.

Compiler-directive statements: RPG II programs that use either standard or non-standard data transfer, or both, must include certain DMLR precompiler compiler-directive statements. For standard data transfer, these statements also do the following:

- Define database records and logical records as files
- Determine how these files are to be processed within the RPG II application
- Determine the techniques to be used to access the database records that comprise each file

General programming considerations that apply to both standard and non-standard data transfer are presented next followed by programming considerations specific to standard and non-standard data transfer.

L.3 Programming considerations

Programming in a CA-IDMS environment requires that considerations be given to:

- Establishing communication between the program and CA-IDMS
- Preparing the database for processing
- Performing database access activities in proper sequence

In most cases, these considerations are the same for both RPG II and COBOL programs. Therefore, you can refer to Chapter 5, “Data Manipulation Language Statements” on page 5-1 for complete DML syntax and parameter descriptions and *CA-IDMS Navigational DML Programming* for a complete description of CA-IDMS programming requirements.

Usage modes: Additionally, considerations differ according to the usage defined in the subschema that the program will access. The programmer must determine the subschema's usage before coding any statements. The three subschema usage modes are as follows:

- **DML** — The program can access database records only.
- **LR** — The program can access logical records only.
- **MIXED** — The program can access logical records and database records.

Note that programs that access logical records have special programming requirements, as discussed in L.3.4, “Logical Record Facility programming considerations” on page L-10 later in this section.

▸▸ For a complete discussion of programming with logical records, see *CA-IDMS Logical Record Facility*

This section presents general programming considerations for RPG II programs.

L.3.1 IDMS communications block

All application programs that request access to a CA-IDMS database require the IDMS communications block, through which communications between CA-IDMS and the application program takes place. Whenever a run unit executes a call to CA-IDMS for database operation, the system posts information concerning the outcome of the requested service in the IDMS communications block established for the program. The error-status indicator (ERST field) contains the return code, which indicates the status of the requested service.

Checking the status of the ERST field: After every call to CA-IDMS, programs that use non-standard RPG II data transfer should examine the ERST field, which occupies positions 9 through 12 of the IDMS communications block. The program can then perform whatever error processing routine it requires, depending on the value contained in ERST. Such error processing may include examining other fields in the

IDMS communications block that have been updated following the call to CA-IDMS: the specific fields updated are determined by the DML statement issued and its outcome.

Error processing for programs that use standard RPG II data transfer involves special considerations; these considerations are discussed in L.4, “Database access under standard RPG II data transfer” on page L-15.

▸▸Refer to *CA-IDMS Messages and Codes* for further information on CA-IDMS error codes.

The IDMS communications block must be defined in program variable storage. It is copied from the dictionary into the program automatically by the DMLR precompiler, and is identified as a data structure with the group name SSCT.

The general layout of the IDMS communications block and a description of the indicators are presented in Chapter 3, “Communications Blocks and Error Detection” on page 3-1. The SSCT data structure as copied into the program from the dictionary is shown in the following figure.

SSCT data structure

```
ISSCT        DS
I                                    00010008 PGNM
I                                    00090012 ERST
I*    ++++++++++++++++++++++++++++    00090008 DBSOK
I*    +   Note that these five   +    00090008 ANYS
I*    +   fields are not used    +    00090008 ANYES
I*    +   by RPG II programs     +    00090008 DEOS
I*    ++++++++++++++++++++++++++++    00090008 DRNF
I                                   B001300160DBKEY
I                                    00170032 RECN
I                                    00170032 RRECN
I                                    00170024 SSNN
I                                    00250032 SSDN
I                                    00330048 ARNM
I                                    00490064 ESET
I                                    00650080 EREC
I                                    00810096 EARE
I                                    00970196 IDBAR
I                                    00970196 IDB
I                                    00970196 FILL
I                                    00970100 DBSA
I                                    01010196 FILL
I                                   B019702000DDBK
I                                   B019702000DBK
I                                    02010207 DBST
I                                    02010202 DBSC
I                                    02030207 DBSSC
I                                    02080208 FILL
I                                   B020902120RECO
I                                   B021302160DMLS
```

L.3.2 Synonym processing

CA-IDMS allows alternative identification (i.e., synonyms) of records and elements in the dictionary. Because data names are restricted to six characters in RPG II, the DMLR precompiler truncates longer record and element names to six characters. Since records and elements often are identified in the dictionary by primary names of up to 32 characters, alternative and unique six-character names should be established in the dictionary explicitly for use by RPG II programs. The use of synonyms is recommended if compiler-directive statements are to be included in RPG II programs.

Note: Synonyms cannot be established for logical record names. RPG II programs that access logical records must use a separate subschema in which logical records are defined with names that adhere to RPG II conventions.

Language-dependent synonyms: The DMLR precompiler automatically copies these language-dependent synonyms in place of the primary names whenever a COPY statement appears in the application program. When the DMLR precompiler copies record descriptions from the dictionary into program variable storage, it copies synonyms according to the following rules:

- If a record synonym exists for the program's language (i.e., RPG II) and the primary record name is not associated with the program's language, the synonym is copied into the program.
- If more than one synonym for a given record exists for RPG II, the first one found in the dictionary is copied.
- If the primary record name is associated with RPG II, the primary name is copied into the program regardless of whether synonyms exist for RPG II.

Example: For example, assume that a record is defined in the dictionary with three synonyms, as follows:

```
RECORD EMP-REC-XYZ
RECORD NAME SYNONYM XYZSN1 LANGUAGE RPG/II
REOCRD NAME SYNONYM XYZ-SN2
RECORD NAME SYNONYM XYZSN3 LANGUAGE RPG/II
```

Since the dictionary includes XYZSN1 as the first synonym for RPG II, the DMLR precompiler copies it into the program. The DMLR precompiler would copy the primary record name (e.g., EMP-REC-XYZ) if it were defined for RPG II.

These rules apply regardless of the record name or synonym that appears in the schema and subschema invoked by the program.

Refer to the *IDD User Guide* for further details on synonym facilities.

L.3.3 Copying non-IDMS record descriptions

Record descriptions for non-IDMS records are often stored in the dictionary by using the IDD DDDL compiler. Once stored, these descriptions can be copied into *any* RPG II program as either Input Specifications, data structures, or Output Specifications. The COPY INPUT RECORD and COPY OUTPUT RECORD statements copy in these descriptions and permit various RPG II options to be specified for each record and its elements.

L.3.4 Logical Record Facility programming considerations

A program that accesses logical records differs from a program that accesses database records. A program that accesses logical records does the following:

- Copies different data descriptions into program variable storage than when accessing database records
- Issues special logical-record functional DML statements (under non-standard data transfer)
- Cannot issue BIND statements for the logical records it will access
- Does not need to keep track of currency
- Contains code to detect errors in path processing rather than code to detect errors in database-access operations directly
- Accesses only those logical records for which paths have been defined in the program's subschema
- Cannot use synonyms for logical records

However, programs that reference logical records are compiled and executed exactly as are those that access the database directly.

When referencing logical records, program requests are serviced by the Logical Record Facility (LRF). LRF in turn communicates with CA-IDMS.

The remainder of this section discusses the following topics:

- The LRC block
- Path status
- Logical-record error-status codes
- Restrictions on the use of the compiler-directive and functional DML statements with LRF

L.3.4.1 LRC block

The Logical-Record Request Control (LRC) block provides an interface between the program, and LRF. It is used to pass information to LRF about a logical-record request and to return path-status information to the program about the processing of the request. The LRC block is copied from the dictionary into program variable storage automatically by the DMLR processor and is identified as a data structure with the group name SSLRCT.

The LRC block is presented in Chapter 3, “Communications Blocks and Error Detection” on page 3-1. The data structure SSLRCT, as copied into the program from the dictionary, is presented below.

SSLRCT data structure

```
ISSLRCT     DS                      B000100020LRLNG
I                                   B000300040MAXVXP
I                                    00050008 LRIDC
I                                    00090016 LRVERB
I                                    00170032 LRNAME
I                                    00170032 LRNAM
I                                    00170024 LRNM1
I                                    00250032 LRNM2
I                                    00330048 LRSTAT
I                                    00330048 LRSTT
I                                    00330040 LRST1
I                                    00410048 LRST2
I                                    00490064 FILL
```

L.3.4.2 Path status

Path statuses are issued during execution of the path selected to service a logical-record request. LRF returns a specific path status to the LRSTAT field of the program's Logical-Record Request Control (LRC) block to indicate the result of each logical-record request made by the program. The RPG II program can examine this information and use it to determine further processing.

Path statuses are 1- to 16-character strings. Refer to *CA-IDMS Logical Record Facility* for a list of these statuses.

CA-IDMS error status: An error can also occur in the logical-record path. If this happens, the ERST field in the IDMS communications block will contain an error-status code issued by the DBMS (with a major code of 00 through 19) indicating an error code. Refer to *CA-IDMS Messages and Codes* for a list of the CA-IDMS-issued error codes.

After issuing an LR request, the program can use RPG II language COMP statements to examine the path status in the LRSTAT field. Since the COMP operation can examine only eight characters at a time, two subordinate 8-byte fields, LRST1 and LRST2, are defined for LRSTAT. If LR-ERROR is returned to LRSTAT, it may be useful to examine the ERST field of the IDMS communications block to determine the LRF or DBMS error code produced. The presence of an LR-ERROR condition can be

determined without considering the LRSTAT field; a value of other than 0000 in ERST indicates an LR-ERROR.

Once the status of the LR request is determined, error processing can be coded, depending on program requirements and DBA specifications.

L.3.4.3 DML statement restrictions

If a subschema specifies a usage of LR, a program using that subschema can access only logical records. As a result, RPG II applications that include either standard or non-standard data transfer are restricted in their use of compiler-directive and functional DML statements. The requirements for both types of data transfer are discussed separately below.

Standard data transfer requirements: The following requirements apply to programs that include standard data transfer and access a subschema with a usage of LR.

- The program must specify the logical-record-name parameter in any IDMS FILE ID, COPY INPUT RECORD/LOGICAL RECORD, or COPY OUTPUT RECORD/LOGICAL RECORD statements. Record-name cannot be specified for these statements in this case.
- The program must include an L in column 20 of any CA-IDMS FILE ID statements.

Refer to L.4, “Database access under standard RPG II data transfer” on page L-15for the syntax of the compiler-directive statements to which these requirements apply.

Non-standard data transfer requirements: The following considerations apply to the use of compiler-directive statements by programs that include non-standard data transfer and access a subschema with a usage of LR:

- The program can include the COPY SUBSCHEMA LOGICAL RECORDS statement, but not the COPY SUBSCHEMA RECORDS statement.
- The program cannot include the COPY SUBSCHEMA BINDS statement.
- If the program includes a COPY DS RECORD/LOGICAL RECORD statement, the logical-record-name parameter must be specified in this statement.

Refer to L.5, “Database access under non-standard RPG II data transfer” on page L-29 for the compiler-directive statements to which these considerations apply.

DML statement restrictions: The following functional DML statements cannot be included in these non-standard programs:

```
FIND/OBTAIN                    CONNECT
GET                            DISCONNECT
MODIFY                         IF
STORE                          KEEP
ERASE                          BIND RECORD
```

Valid DML statements: However, these non-standard programs are permitted to issue the four logical-record DML statements, in addition to the following functional DML control and save statements:

```
BIND RUN-UNIT                    ROLLBACK
BIND PROCEDURE                   FINISH
READY                            ACCEPT IDMS STATISTICS
COMMIT
```

For details on these statements, refer to Chapter 5, “Data Manipulation Language Statements” on page 5-1. The remaining functional DML save statements (ACCEPT DBKEY OF CURRENCY, ACCEPT DBKEY TO CURRENCY, ACCEPT BIND RECORD, and ACCEPT PROCEDURE) should not be used without consulting the DBA.

L.3.5 Precompiler options

Precompiler options control dictionary preparation, list generation, comment generation, and logging of program activity statistics.

Precompiler option statements can be included in RPG II source programs (standard or non-standard) that are input to the DMLR precompiler. Code precompiler options in the Header Specifications before any DML or RPG II statements.

Syntax for RPG II precompiler option statements is presented below. Complete parameter descriptions are not provided here. Only RPG II-specific notes are provided for those parameters that require them. For a complete description of precompiler options, see Chapter 2, “Precompiler Options” on page 2-1.

Syntax

```
      Col
      6
      ↓
►►──┬─ H*RETRIEVAL ──────┬──────────────────────────────────────────►◄
    └─ H*PROTECTED-UPDATE ─┘

      Col
      7
      ↓
►►──┬─ *DMLIST ────────┬──────────────────────────────────────────►◄
    └─ *NODMLIST ◄ ────┘

      Col
      6
      ↓
►──┬─ H*SCHEMA-COMMENTS ─┬──────────────────────────────────────────►◄
   └─ H*NO-ACTIVITY-LOG ─┘

      Col
      6
      ↓
►──┬─ H*NO-ACTIVITY-LOG ─┬──────────────────────────────────────────►◄
   └─ H*SCHEMA-COMMENTS ─┘
```

Parameters

H*RETRIEVAL/H*PROTECTED-UPDATE

This statement, if used, must be the first source input statement coded in the Header Specifications.

***DMLIST/*NODMLIST**

Generation of the DML listing can be turned on or off any number of times within one source program by inserting appropriate DMLIST/NODMLIST entries in the code.

H*NO-ACTIVITY-LOG

It is generally recommended that the suppress-log option not be used, so that the DMLR precompiler generates and updates the dictionary with program activity statistics.

Program activity statistics cannot be logged if the DDLDML area is readied for retrieval only.

L.4 Database access under standard RPG II data transfer

Standard data transfer enables application programs to access CA-IDMS databases according to the typical RPG II cycle of operation (i.e., input, processing, and output). Programs request standard data transfer through compiler-directive statements. These statements are coded mainly as File, Input, and Output Specifications, and define CA-IDMS database records and logical records as RPG II files. At run time, programs access database records and logical records based upon the specifications associated with these files. In addition, records can be retrieved using the RPG II READ statement.

Under standard data transfer, the programmer typically does not need to code any functional DML statements to request database services. All necessary requests are issued automatically when the proper compiler directives are included and the necessary RPG II options are included and assigned.

Supported functions: Standard data transfer supports these three basic functions that allow programs to manipulate both database and logical records in the database:

- Retrieval
- Modification
- Storage

The erase function, which is supported under non-standard data transfer, is ***not*** supported for database records and logical records accessed under standard RPG II data transfer.

Example: In order to specify the retrieval function, for example, the program might define a file, which is associated with a database record, as an RPG II input file. The program might define another file as an update file, so that occurrences of a record can be modified. An output file could be defined in order to store new record occurrences.

Special routines: Application programs that use standard data transfer communicate with CA-IDMS through RPG II special routines; the DMLR precompiler generat es the appropriate File Specification entries for these routines. Additionally, the DMLR precompiler automatically performs the following services for these applications:

- Copies the IDMS communications block data description into the program as a data structure.
- Copies a data structure labeled IVAR into the program. IVAR contains variable fields used to pass information about database requests to CA-IDMS.
- Generates File and Output Specifications for a FINISH file. The FINISH file is used to release database areas from program control when database processing terminates.

Functions of IDMSRPGI: Database housekeeping functions are automatically performed for the program at run time by the CA-IDMS runtime interface routine, IDMSRPGI. This interface routine performs the following functions:

- Signs on the application program to CA-IDMS
- Establishes addressability in variable storage for records to be accessed by the program
- Prepares the appropriate database areas for processing (unless overridden by the program)

Error handling: Error handling under standard data transfer follows RPG II rules. When an error occurs in processing a file defined for a CA-IDMS database record or logical record (i.e., a special file error with an RPG II status value of 1299), the program is terminated immediately. For programs operating under the central version, CA-IDMS performs the necessary database recovery functions; recovery functions must be initiated manually for programs running in local mode.

Remember that standard RPG II programs that access logical records can examine the LRSTAT field of the LRC block after issuing logical record requests. Path statuses returned to this field indicate the status of the request (e.g., LR-ERROR).

Refer to L.9, “Standard RPG II data transfer return codes” on page L-48 later in this appendix for further details on error handling under standard data transfer.

L.4.1 Compiler-directive statements

Five different compiler directives are available for standard data transfer. Each of these RPG-like statements is coded on one or more of the specifications forms and fulfills specific database access requirements:

- **SUBSCHEMA/SCHEMA** — Included in the Header Specifications. This statement identifies a subschema, determines the CV under which the program will execute, and/or determines the database to be accessed.
- **IDMS FILE ID** — Included in the File Specifications. This statement defines database records and logical records as files, and specifies the techniques used to access record occurrences. It also generates a special option through which the program requests CA-IDMS services for database records and logical records.
- **COPY INPUT RECORD/LOGICAL RECORD** — Included in the Input Specifications. This statement copies the description of an individual database record, logical record, or non-IDMS record defined in the dictionary.
- **COPY OUTPUT RECORD/LOGICAL RECORD** — Included in the Output Specifications. This statement copies the description of an individual database record, logical record, or non-IDMS record defined in the dictionary.
- **COPY MODULE** — Included on any of the RPG II Specifications forms (i.e., H, F, E, L, I, C, and O Specifications). This statement copies source modules, previously stored in the dictionary, into the program.

Syntax and parameter descriptions for each of the compiler-directive statements is discussed in detail below.

L.4.1.1 Subschema/Schema identification

Any program that accesses a CA-IDMS database must include a SUBSCHEMA/SCHEMA statement in the Header Specifications. The SUBSCHEMA/SCHEMA statement provides the following information to the DMLR precompiler:

- The subschema view of the database. The named subschema determines which CA-IDMS record descriptions the precompiler copies into the program. Also, CA-IDMS checks every database request issued by the RPG II application against access restrictions in this subschema.
- The name of the node (i.e., CA-IDMS system) where the database resides.
- The name of the database that the program will access.
- The device type of an output file required by the DMLR precompiler to generate source code for the CA-IDMS runtime interface, IDMSRPGI. The source code written to this output file must be assembled and link-edited with the object program. This option applies only to programs executing under VSE.
- The name of the program.

In addition, programs that use combined RPG II data transfer can include a debugging option in the SUBSCHEMA/SCHEMA statement. The debugging option instructs the DMLR precompiler to enter sequence numbers for each functional DML statement in the IDMS Communications block at run time. The SUBSCHEMA/SCHEMA statement also allows programs using standard or combined data transfer to prevent the IDMSRPGI interface routine from automatically readying database areas for processing. Instead, all areas accessed by the program are readied according to the default usage modes defined in the subschema.

Syntax

```
      Cols            Cols
      6-14            15-22
      ↓               ↓
►►── Hsubschema-name schema-name ──┬──────────────────┬──┬──────────────────┬──►
                                   │ Cols             │  │ Cols             │
                                   │ 23-27            │  │ 28-35            │
                                   │ ↓                │  │ ↓                │
                                   └─ version-number ─┘  └─ db-nodename ────┘

►──┬──────────────────┬──┬──────────────────┬──┬──────────────────┬──►
   │ Cols             │  │ Cols             │  │ Cols             │
   │ 36-43            │  │ 44-51            │  │ 52-29            │
   │ ↓                │  │ ↓                │  │ ↓                │
   └─ database-name ──┘  └─ dict-nodename ──┘  └─ dictionary-name ─┘

►──┬──────────────────┬──┬──────────┬──┬──────────┬──┬──────────────────┬──►◄
   │ Cols             │  │ Cols     │  │ Cols     │  │ Cols             │
   │ 60-64            │  │ 65-69    │  │ 70-71    │  │ 75-80            │
   │ ↓                │  │ ↓        │  │ ↓        │  │ ↓                │
   ├─ device-type ────┤  └─ DEBUG ──┘  └─ SR ─────┘  └─ program-name ────┘
   └─ FBA ────────────┘
```

Parameters

Hsubschema-name

Specifies a previously compiled subschema. If allowable program names for this subschema are defined in the dictionary, the named program must be associated with *subschema-name* in the dictionary. The RPG II application can be named in a Header Specifications line included before the SUBSCHEMA/SCHEMA statement; specify the appropriate program name in columns 75 through 80 of that line. The program can also be named in the *program-name* entry of the SUBSCHEMA/SCHEMA statement, as discussed below.

schema-name

Specifies the schema under which *subschema-name* is compiled.

version-number

Optionally qualifies *schema-name* with a number from 1 to 9999. If omitted, *version-number* defaults to the highest version number present in the dictionary for the named schema.

db-nodename

The one- to eight- alphanumeric character name that optionally identifies the CA-IDMS system (nodename) where the database resides. If specified, *db-nodename* may be overridden by IDMSOPTI module or SYSCTL file specifications.

database-name

The one- to eight- alphanumeric character name that optionally names the database to be accessed by the program. If specified, *database-name* may or may not be overridden by IDMSOPTI module or SYSCTL file specifications

dict-nodename

Identifies the one- to eight- alphanumeric character name of the node that controls the dictionary where the subschema resides. *Dict-nodename* must be the name of a field in variable storage.

dictionary-name

Identifies the one- to eight- alphanumeric character name of the dictionary load area where the subschema resides.

Dictionary-name must be the name of a field in variable storage.

device-type

For VSE users only, identifies the device type of an output file required by the DMLR precompiler. The DMLR precompiler uses this file to store source code that it has generated for the CA-IDMS runtime interface, IDMSRPGI. *Device-type* allows any of the following device types to be specified for the IDMSRPGI source file:

```
2311        3340
2314        3350
2321        3375
3330        3380
3330B       3390 (VSE/ESA only)
```

FBA

Specifies that the IDMSRPGI source file will be written to an FBA data set.

DEBUG

For combined RPG II data transfer only, specifies that, for each functional DML call, the DMLR precompiler places a unique sequence number in the IDMS communications block at run time. These numbers appear in positions 75 through 80 of the RPG II compiler output listing, in the form DR*nnnn*.

SR

Suppresses all readying of database areas performed for the program by the interface routine IDMSRPGI.

When the SR option is included, all areas accessed by the program are readied according to the default usage modes defined in the subschema.

Note: Normally, areas accessed by the program and associated with standard data transfer are readied automatically by IDMSRPGI. IDMSRPGI readies an area in exclusive update usage mode if records in the area are stored or modified by the program; otherwise, the area is readied in shared retrieval usage mode.

program-name

Optionally identifies the name of the application program. If allowable program names are defined in the dictionary, *program-name* must be associated with *subschema-name* in the dictionary. If the program is named in a Header Specification line appearing before the SUBSCHEMA/SCHEMA statement, the DMLR precompiler will ignore the *program-name* entry.

Usage: Using the DEBUG option

The DMLR precompiler generates the numbers to identify the sequence in which functional DML statements appear in the program. Depending on the error routine(s) defined, the program can report the sequence numbers when errors occur, and the programmer can use the numbers to assist in debugging the RPG II application.

L.4.1.2 Defining records/logical records

Standard RPG II programs require that CA-IDMS database records and logical records be defined as files before they are accessed. The IDMS FILE ID statement is used to define these files.

An IDMS FILE ID statement must be included in the File Specifications for each database record or logical record to be accessed by the application. This statement:

- Assigns a unique file name to the record
- Allows certain RPG II options to be specified for the named file
- Identifies the type of database access to be used in processing the file.

The DMLR precompiler uses this information to generate an RPG II special option, through which the application requests CA-IDMS services.

Entries in the IDMS FILE ID statement determine the method used to access occurrences of the database record or logical record named in the statement.

Syntax

```
      Cols       Cols                Cols    Cols
      6-14       15-19               20      21-26
      ↓          ↓                   ↓       ↓
►►── Ffile-name RPG II-options ─┬─ A ─┬─ record-name ──────────────────►
                               ├─ V ─┤  └─ logical-record-name ─┘
                               └─ L ─┘

      Cols
      37-52
      ↓
►─┬─ area-name ──────────────────────────────────────────────────────►◄
  └─ native-vsam-data-set-name ─┘
```

Parameters

file-name

Identifies a file corresponding to the database record or logical record named in the same IDMS FILE ID statement. *File-name* follow RPG II naming conventions.

RPG II-options

Allows certain RPG II entries normally coded in columns 15 through 19 to be specified for *file-name*, as appropriate. These entries are as follows:

Column	Entry
15	I, O, U
16	P, S, D
17	E, blank
18	A, blank, D
19	F

A

Specifies that area sweeps are used to access database record occurrences. If this access technique is selected, you must also specify *record-name*, beginning in column 21, and *area-name* beginning in column 37.

V

Specifies that native VSAM indexes are used to access native VSAM data sets defined in the schema. If this access technique is selected, you must also specify *record-name*, beginning in column 21, and *index-set-name*, beginning in column 37.

L

Specifies that the Logical Record Facility is used to access logical records defined in the subschema. If this access technique is selected, you must also specify *logical-record-name*, beginning in column 21 and leave columns 37-52 blank.

record-name/logical-record-name

Identifies the database record or logical record to be processed as a file. You can only specify one database or logical record for the file named in the *file-name* entry.

The *record-name* or *logical-record-name* specified in the IDMS FILE ID statement must also be specified in the COPY INPUT RECORD/LOGICAL RECORD or COPY OUTPUT RECORD/LOGICAL RECORD statement used to copy the description of the record into the program. The copied description is automatically associated with the file defined in the IDMS FILE ID statement for the same record.

In order for the program to access *record-name* or *logical-record-name*, the database area in which the record participates must first be readied for processing. Normally, the interface routine IDMSRPGI performs this readying automatically for the program.

The program can ready these additional areas by using the SR option of the SUBSCHEMA/SCHEMA statement (as discussed earlier in this section). This option suppresses any readying by IDMSRPGI, and readies all areas accessed by the program according to the default usage modes defined in the subschema.

area-name

Identifies a database area to be swept for occurrences of a specified database record.

index-set-name

native-vsam-data-set-name

Identifies a native VSAM data set defined in the schema. CA-IDMS accesses existing VSAM structures from this data set as if they were database record occurrences.

Note: A native VSAM KSDS must be defined as VSAM CALC on the primary key.

L.4.1.3 Copying input records/logical records

Each file input to a standard program requires Input Specifications to define the database record or logical record that makes up the file. The COPY INPUT RECORD/LOGICAL RECORD statement can be included in the Input Specifications to copy these definitions into the program from the dictionary. By using this statement, the programmer is saved the effort of coding these record definitions manually.

The COPY INPUT statement consists of two levels of entries, as follows:

- **Record Identification** — Identifies the database record or logical record that is associated with an input or update file, and whose description is to be copied in to the Input Specifications.
- **Field Description** — Optionally allows RPG II specifications to be entered for selected fields within the database record or logical record.

A database record or logical record named in a COPY INPUT statement is automatically associated with the file defined for that record in an IDMS FILE ID statement.

The COPY INPUT statement can also be used to copy in descriptions of non-IDMS records that have been stored in the dictionary. For this purpose, the COPY INPUT statement can be included in an RPG II program, regardless of the type of database processing (i.e., standard, non-standard, and combined, or none) performed in the program. Code the Record Identification and Field Description lines of the COPY INPUT statement, as appropriate to the record being copied.

Records specified in a COPY INPUT statement can include fields that contain one-dimensional arrays. However, the programmer must supply the necessary Extension Specifications for such fields.

Syntax: Record identification

```
      Cols  Cols                Cols                        Cols
      6-10  15-42               43-48                       59-63
      ↓     ↓                   ↓                           ↓
►►── ICOPY RPG II-options ──┬── record-name ──────────────┬──────────────►
                            └── logical-record-name ──┘   └── version-number ─┘
```

Field description

```
     ┌──────────────────────────────┐
      Col Cols       Cols
      6   53-58      59-70
      ↓   ↓          ↓
►►─v─┬──────────────────────────────┬──────────────────────────────────►◄
    └─ I   field-name RPG II-options ─┘
```

Parameters

RPG II-options

Allows the RPG II entries usually coded in columns 15 through 42 to be specified for the record name in the statement. The appropriate RPG II entries to be included are determined by program requirements and restrictions.

record-name

Identifies the name of the record whose description will be copied into the Input Specifications from the dictionary. *Record-name* can be the primary name or a synonym for a database record or non-IDMS record stored in the dictionary.

version-number

Optionally qualifies the named record with a version number from 1 to 9999. If no version number is specified, the system assumes that *record-name* identifies a record that is included in the subschema named in the SUBSCHEMA/SCHEMA statement, and looks for it there. If the record is not there (i.e., it was entered in the dictionary through the IDD DDDL compiler), *version-number* defaults to the highest version number of the record defined in the dictionary for the operating mode under which the program is being compiled.

Note: Do not specify a database record for *record-name* if the subschema's usage is LR.

logical-record-name

Identifies a logical record defined in the subschema Do not specify *logical-record-name* if the subschema's usage is DML.

Note: Synonyms are not allowed for logical-record-names.

field-name

Identifies the field (element) to be assigned RPG II specifications.

Each field for which this option is used must be described on a separate line, following the Record Identification entry. In addition, these fields must be listed in the same order in which they will be copied into the Input Specifications from the dictionary. Only those fields to be given RPG II specifications should be named in the COPY INPUT statement.

RPG II-options

Allows the RPG II entries usually coded in columns 59 through 70 to be specified for the named field. The appropriate RPG II entries to be included are determined by program requirements and restrictions.

L.4.1.4 Copying output records/logical records

Each file output from a standard program requires Output Specifications to define the database record or logical record that makes up the file. The COPY OUTPUT RECORD/LOGICAL RECORD statement can be included in the Output Specifications to copy these definitions into the program from the dictionary. By using this statement, the programmer is saved the effort of coding these record definitions manually.

The COPY OUTPUT statement consists of two levels of entries:

- **Record Identification** — Identifies the database record or logical record that is associated with an output or update file, and whose description is to be copied into the Output Specifications.
- **Field Description** — Allows RPG II specifications to be entered for selected fields within the database record or logical record.

A database record or logical record named in a COPY OUTPUT statement is automatically associated with the fields within the database record in an IDMS FILE ID statement.

The COPY OUTPUT statement can also be used to copy in descriptions of non-IDMS record that have been stored in the dictionary. For this purpose, the COPY OUTPUT statement can be included in any RPG II program, regardless of the type of database processing (i.e., standard, non-standard, combined, or none) performed in the program. Code the Record Identification and Field Description lines of the COPY OUTPUT statement as appropriate to the record being copied.

Records specified in a COPY OUTPUT statement can include fields that contain one-dimensional arrays. However, the programmer must supply the necessary Extension Specifications for such fields.

Syntax: Record identification

```
     Cols  Cols                Cols                        Cols
     6-10  15-31               32-37                       48-52
     ↓     ↓                   ↓                           ↓
►►── OCOPY RPG II-options ─┬─ record-name ──────────────┬─┬─────────────────────┬─►
                           └─ logical-record-name ──┘  └─ version-number ──┘
```

Field description

```
   ┌──────────────────────────────────────────────────┐
   │ Col Cols          Cols      Cols                 │
   │ 6   23-31         32-37     38-70                │
   │ ↓   ↓             ↓         ↓                    │
►─┼─↓─┬─ 0   RPG II-options field-name RPG II-options ─┘──────────────────────►◄
   │   └──────────────────────────────────────────────┘
```

Parameters

RPG II-options

Allows the RPG II entries usually coded in columns 15 through 31 to be specified for the record named in the statement. The appropriate RPG II entries to be included are determined by program requirements and restrictions.

record-name

Identifies the record whose description is to be copied into the Output Specifications. *Record-name* can be the primary name or a synonym for a database or non-IDMS record stored in the dictionary.

Do not specify a database record for *record-name* if the subschema's usage is LR.

version-number

Optionally qualifies the named record with a version number from 1 to 9999. If no version number is specified, the system assumes that *record-name* identifies a record that is included in the subschema named in the SUBSCHEMA/SCHEMA statement, and looks for it there. If the record is not there (i.e., it was entered in the dictionary through the IDD DDDL compiler), *version-number* defaults to the highest version number of the record defined in the dictionary for the operating mode under which the program is being compiled.

logical-record-name

Identifies the logical-record, defined in the subschema, whose description is to be copied into the Output Specifications. Do not specify *logical-record-name* if the subschema's usage is DML.

Note: Synonyms are not allowed for logical-record names.

field-name

Identifies the field (element) to be assigned RPG II specifications.

Each field for which this option is used must be described on a separate line, following the Record Identification entry. In addition, these fields must be listed in the same order in which they will be copied into the Output Specifications from the dictionary. Only those fields to be given RPG II specifications should be named in the COPY OUTPUT statement.

RPG II-options

Allows the RPG II entries usually coded in columns 38 through 70 to be specified for the named field. The appropriate RPG II entries to be included are determined by program requirements and restrictions.

L.4.1.5 Copying modules

The COPY MODULE statement instructs the precompiler to copy a source module into the program from the dictionary. The module must be defined in the dictionary. Use of the COPY MODULE statement reduces programming effort by providing source code to the program automatically (e.g., Extension Specifications for record elements that contain arrays).

The DMLR precompiler places the module into the program at the location of the request, without modification. The module can, but need not, contain CA-IDMS compiler-directive statements. If compiler-directive statements are present, the DMLR precompiler examines and expands them within the context of the program's subschema view and compile mode as if the programmer had coded them directly. The program ***can*** nest COPY MODULE statements (i.e., code invoked by a COPY MODULE entry can, itself, contain a COPY MODULE statement). However, the programmer should ensure that a copied module does not, in turn, copy itself.

Syntax: For H, F, E, L, I, or O specifications

```
      Cols  Cols            Cols
      7-11  15-46           47-51
      ↓     ↓               ↓
►►── COPYM module-name ──┬──────────────────┬──────────────────────────────►◄
                         └─ version-number ─┘
```

For C specifications

```
       Cols   Cols              Cols
       6-11   18-49             50-54
       ↓      ↓                 ↓
►►── CCOPYM module-name ──┬──────────────────┬──────────────────────────────►◄
                          └─ version-number ─┘
```

Parameters

module-name

Specifies the name of the module to be copied from the dictionary.

version-number

Optionally qualifies *module-name* with a version number from 1 to 9999. If not specified, *version-number* defaults to the highest value associated with the named module in the dictionary, for the mode under which the program is being compiled. If no mode-specific version exists for *module-name*, the DMLR precompiler copies the non-mode-specific version, if present. If the dictionary contains neither a non-mode-specific entry for *module-name* nor a mode-specific entry applicable to this program's compile mode, an error results.

►►See *CA-IDMS Messages and Codes* for a description of CA-IDMS error messages. *Version-number* also defaults to the highest value defined for the module for the program's language.

L.5 Database access under non-standard RPG II data transfer

This section contains an overview of CA-IDMS database access under nonstandard RPG II data transfer, followed by discussions of the compiler-directive and functional DML statements used by non-standard programs.

Overview of non-standard data transfer: Non-standard data transfer enables application programs to access CA-IDMS database records in a wide variety of ways, without defining RPG II files for these records. Programs request non-standard data transfer through compiler-directive statements and functional DML statements included in the source code that is input to the DMLR precompiler.

Compiler directives:

- Establish communication between the program and CA-IDMS
- Identify the view of the database available to the program
- Copy descriptions of database records into variable storage as data structures.

These statements do not produce any executable commands.

Functional DML statements: Functional DML statements issue requests to CA-IDMS for database services that allow the program to manipulate record occurrences. The DMLR precompiler expands these functional DML statements into appropriate RPG II CALL statements in the Calculation Specifications. At run time, these CALL statements transfer control to CA-IDMS, which in turn provides the requested database services.

In addition to the services available to the program through compiler-directive and functional DML statements, the DMLR precompiler automatically performs the following functions for RPG II applications:

- Copies the IDMS communications block data description into the program as a data structure.
- Copies a data structure labeled IVAR into the program. IVAR contains variable fields used to pass information about database requests to CA-IDMS.

See L.3.1, “IDMS communications block” on page L-7 for descriptions of the IDMS communications block and the LRC block.

The remainder of this section includes the following:

- Syntax and syntax rules for compiler-directive statements
- A discussion of functional DML statements

For complete syntax and parameter descriptions for all DML statements, refer to Chapter 5, “Data Manipulation Language Statements” on page 5-1.

L.5.1 Compiler-directive statements

Six different compiler directives are available for non-standard data transfer. Each of these RPG-like statements is coded within one or more of the specifications forms, and fulfills specific database access requirements, as follows:

- **SUBSCHEMA/SCHEMA** — Included in the Header Specifications. This statement identifies the subschema that the program will access.
- **COPY SUBSCHEMA RECORDS** — Included in the Input Specifications. This statement copies the descriptions of all database records contained in the subschema into program variable storage. Copied records appear as data structures.
- **COPY DS RECORD** — Included in the Input Specifications. This statement copies the description of an individual database record or logical record defined in the dictionary into program variable storage. Copied descriptions appear as data structures.
- **COPY MODULE** — Included on any of the RPG II Specifications forms (i.e., H, F, E, L, I, C, and O Specifications. This statement copies source modules, previously defined in the dictionary, into the program.
- **COPY IDMS SUBSCHEMA-BINDS** — Included in the Calculation Specifications. This statement generates appropriate standard BIND RECORD statements for database records accessed by the program.

In addition to the compiler-directive statements listed above, programs that use non-standard data transfer can include the COPY INPUT RECORD/LOGICAL RECORD and COPY OUTPUT RECORD/LOGICAL RECORD statements. Non-standard RPG II programs are permitted to use these statements in order to copy in descriptions of non-IDMS records that have been stored in the dictionary.

Each of the compiler-directive statements is discussed separately below.

L.5.1.1 Subschema/Schema identification

Any program that access the database must include a SUBSCHEMA/SCHEMA statement in the Header Specifications. The SUBSCHEMA/SCHEMA statement provides the following information to the DMLR precompiler:

- The subschema view of the database. The named subschema determines which CA-IDMS record descriptions the precompiler copies into the program. Also, CA-IDMS checks every database request issued by the RPG II application against access restrictions in this subschema.
- The device type of an output file required by the DMLR precompiler to generate source code for the CA-IDMS runtime interface, IDMSRPGI. The source code written to this output file must be assembled and link edited with the object program. This option applies only to programs executing under DOS/VS(E).

- Whether the DMLR precompiler enters the sequence number for each functional DML statement in the IDMS communications block at run time to aid in program debugging.
- The name of the program.

Syntax

```
      Cols          Cols          Cols                Cols
      6-14          15-22         23-27               44-48
      ↓             ↓             ↓                   ↓
►►─── Hsubschema-name schema-name ─┬──────────────┬─┬─ device-type ─┬──►
                                   └─ version-number ─┘ └─ FBA ──────┘

     Cols       Cols
     49-53      75-80
     ↓          ↓
►─┬─ DEBUG ─┘ └─ program-name ─┘──────────────────────────────────────►◄
```

Parameters

subschema-name

Specifies a previously compiled subschema. If allowable program names for this subschema are defined in the dictionary, the named program must be associated with *subschema-name* in the dictionary. The RPG II application can be named in a Header Specifications line included before the SUBSCHEMA/SCHEMA statement; specify the appropriate program name in columns 75 through 80 of that line. The program can also be named in the *program-name* entry of the SUBSCHEMA/SCHEMA statement, as discussed below.

schema-name

Specifies the schema under which *subschema-name* is compiled.

version-number

Optionally defaults to the highest version number from 1 to 9999. If omitted, *version-number* defaults to the highest version number present in the dictionary for the named schema.

device-type

For VSE sites only, identifies the device type of an output file required by the DMLR precompiler. The DMLR precompiler uses this file to store source code that it has generated for the CA-IDMS runtime interface, IDMSRPGI. Valid values for *device-type* are:

```
2311        3340
2314        3350
2321        3375
3330        3380
3330B       3390 (VSE/ESA only)
```

FBA

Specifies that the IDMSRPGI source file will be written to an FBA data set.

Refer to L.7, “Compiling and link-editing” on page L-42 for further considerations associated with the IDMSRPGI interface routine.

DEBUG

Specifies that, for each functional DML call, the DMLR precompiler places a unique sequence number in the IDMS communications block at run time. These numbers appear in positions 75 through 80 of the RPG II compiler output listing, in the form DR*nnnn*.

program-name

Optionally identifies the name of the application program. If allowable program names are defined in the dictionary,*program-name* must be associated with *subschema-name* in the dictionary. If the program is named in a Header Specification line appearing before the SUBSCHEMA/SCHEMA statement, the DMLR precompiler will ignore the *program-name* entry.

Usage: Using the DEBUG option

The DMLR precompiler generates the numbers to identify the sequence in which functional DML statements appear in the program. Depending on the error routine(s) defined by the DBA, the program can report the sequence numbers when errors occur, and the programmer can use the numbers to assist in debugging the RPG II application.

L.5.1.2 Copying subschema records

The COPY SUBSCHEMA RECORDS statement, coded in the Input Specifications, copies the descriptions of all database record contained in the subschema. These descriptions are copied into program variable storage as data structures. By using this statement, the programmer is saved the effort of coding these record descriptions manually.

According to the rules of synonym usage, the DMLR precompiler copies RPG II synonyms defined for the subschema records in the dictionary into the program. Refer to L.3.2, “Synonym processing” on page L-9for details on synonym processing.

Database records whose descriptions are to be copied into the program using the COPY SUBSCHEMA RECORDS statement can include fields that contain one-dimensional arrays. However, the programmer must supply the necessary Extension Specifications for all such fields.

Syntax

```
     Cols     Cols    Cols
     6-10     19-20   43-38
     ↓        ↓       ↓
►►── ICOPY ── DS ─── SSRECS ──────────────────────────────────────►◄
```

L.5.1.3 Copying individual record descriptions

The COPY RECORD/LOGICAL RECORD statement can be included in the Input Specifications of non-standard programs to copy descriptions of individual database records into variable storage. These descriptions, copied from the dictionary, appear in the program as data structures. By using this COPY statement, the programmer is saved the effort of coding these record definitions manually.

Database records and logical records specified in a COPY DS statement can include fields that contain one-dimensional arrays. However, the programmer must supply the necessary Extension Specifications for all such fields.

Syntax

```
     Cols     Cols    Cols
     6-10     19-20   43-48
     ↓        ↓       ↓
►►── ICOPY ── DS ──┬── record-name ──────────────────────────────────►◄
                  └── logical-record-name ──┘
```

Parameters

record-name

Identifies the database record whose description is to be copied into the Input Specifications as a data structure. *Record-name* can be the primary name or a synonym for a database record.

Note: Synonyms are not allowed for logical-record names. Refer to L.3.2, “Synonym processing” on page L-9 for rules governing the use of record synonyms in applications programs.

L.5.1.4 Copying modules

The COPY MODULE statement instructs the precompiler to copy a source module into the program from the dictionary. The module must exist in the dictionary. Use of the COPY MODULE statement reduces programming effort by providing source to the program automatically (e.g., Extension Specifications for record elements that contain arrays).

The DMLR precompiler places the module into the program at the location of the request, without modification. The module can, but need not, contain CA-IDMS compiler-directive or functional DML statements. If any of these statements are present, the DMLR precompiler examines and expands them within the context of the program's subschema view and compile mode as if the programmer had coded them directly. The program *can* nest COPY MODULE statements (i.e., code invoked by a COPY MODULE entry can, itself, contain a COPY MODULE statement). However, the programmer should ensure that a copied module does not, in turn, copy itself.

Syntax

```
      Cols  Cols              Cols
      7-11  15-46             47-51
      ↓     ↓                 ↓
►►── COPYM module-name ──┬──────────────────┬──────────────────────────────►◄
                         └─ version-number ─┘

       Cols   Cols               Cols
       6-11   18-49              50-54
       ↓      ↓                  ↓
►►── CCOPYM module-name ──┬──────────────────┬──────────────────────────────►◄
                          └─ version-number ─┘
```

module-name

Specifies the name of the module to be copied. It must be the name of a module defined in the dictionary.

version-number

Optionally qualifies *module-name* with a version number from 1 to 9999. If not specified, *version-number* defaults to the highest value associated in the dictionary with the named module, for the mode under which the program is being compiled. If no mode-specific version exists for *module-name*, the DMLR precompiler copies the non-mode-specific version, if present. If the dictionary contains neither a non-mode-specific entry for *module-name* nor a mode-specific entry applicable to this program's compile mode, an error results (refer to the *CA-IDMS Messages and Codes*). The same holds true for language (i.e., *version-number* defaults to the highest value defined for the module for the program's language, etc.).

L.5.1.5 Binding records

The COPY SUBSCHEMA BINDS statement, coded in the Calculation Specifications, instructs the DMLR precompiler to generate in the source program the appropriate standard BIND RECORD statements for each database record in the program's variable storage. For a discussion of the BIND RECORD statement, see Chapter 5, “Data Manipulation Language Statements” on page 5-1. The DMLR precompiler converts the COPY SUBSCHEMA BINDS statement itself to a comment.

Syntax

```
     Cols     Cols
     6-10     18-24
     ↓        ↓
►►── CCOPY ── SSBINDS ──────────────────────────────────────────────►◄
```

Parameters

CCOPY SSBINDS

Generates BIND RECORD statements only for subschema records explicitly copied into the program as data structures by means of a COPY SUBSCHEMA RECORDS or a COPY DS RECORD statement. It does not automatically generate BIND statements for all subschema records.

Note that the COPY SUBSCHEMA BINDS statement does not generate BIND statement for logical records, since logical records cannot be bound.

Usage: Issuing individual BIND RECORD statements

Instead of issuing a COPY SUBSCHEMA BINDS statement, the programmer can issue BIND RECORD statements separately, allowing the program to do the following:

- Call a subroutine to check the ERST field of the IDMS communications block after each BIND RECORD statement
- Bind several records to the same location

L.5.2 Functional DML statements

Under non-standard data transfer, functional DML statements are inserted into the following format within the Calculation Specifications:

Syntax

```
     Cols            Cols
     6-17            18-72
     ↓               ↓
►►── CRPG II-options DML statement ──────────────────────────────►

     Cols            Cols
     6-9             18-72
     ↓               ↓
►──┬──────────────────────────────────────────────────────────────►
   └─ CAND           continuation of DML statement ─┘
```

Parameters

RPG II-options

Allows RPG II entries normally coded in columns 7 through 17 to be specified for the functional DML statement that has been included. The appropriate RPG II entries to be included are determined by program requirements and restrictions.

Note that the AN/OR option for columns 7 and 8 is not allowed.

DML statement

Specifies a functional DML statement, which can be coded beginning in column 18. This functional DML statement can be continued on the next line in the Calculation Specifications if it would extend beyond column 72 of the first line. If the continuation line is used, the remainder of the functional DML statement should begin in column 18 of the continuation line.

Usage: Supported statements

All functional DML statements are available under non-standard RPG II data transfer. These statements are the same as those available to COBOL language programs; details on the use of these statements, syntax, and syntax rules are contained in Chapter 5, “Data Manipulation Language Statements” on page 5-1.

Special RPG II considerations apply to the IF *set-name* control DML statements. These considerations are described below and involve differences that exist between the RPG II DML statement and its corresponding COBOL DML statement.

L.5.3 Control Statements

```
►►── IF set-name is ──┬──────┬── EMPTY SETON indicator ──────────►◄
                     └─ NOT ─┘

►►── IF ──┬──────┬── set-name MEMBER SETON indicator ──────────►◄
         └─ NOT ─┘
```

IF

Evaluates the presence of records in a set and takes action based on the outcome of the evaluation.

The DMLR IF statement does not include the *imperative-statement* phrase. Instead, this statement uses a SETON phrase to turn on an RPG II indicator following the evaluation of a condition.

indicator

Identifies one of the resulting indicators allowed in RPG II Calculation Specifications. *Indicator* is either turned on or left off, depending on the outcome of the test performed by the IF statement. Note that only one indicator can be included in the SETON phrase.

L.5.4 Logical record DML statements

Here are the logical record DML statements you can include in an RPG II program that accesses logical records:

- **OBTAIN logical record** requests retrieval of a logical record
- **MODIFY logical record** requests modification of a logical record
- **STORE logical record** requests storage of a new logical record
- **ERASE logical record** requests deletion of a logical record

The DMLR logical record statements ***do not*** include WHERE clauses. As a result, selection of logical record occurrences is performed for the program exclusively at the subschema level.

The ON clause is also not allowed for these statements. Applications that access logical records should test for path statuses using the RPG II COMP statement, as discussed earlier in this appendix under, "Programming considerations".

L.6 Database access under combined RPG II data transfer

Combined data transfer refers to RPG II programs that perform both standard and non-standard data transfer. Under the combined approach, the program defines certain database records for standard data transfer and defines other database records for non-standard data transfer. As a result, either standard or non-standard rules apply to each record accessed by the program; it is recommended that only one type of data transfer be associated with a given database record within a program.

Special considerations: In order to perform both types of data transfer, RPG II programs require that special consideration be given to several aspects of database access. These considerations are presented below:

- Naming the subschema
  - Standard data transfer — You must include a SUBSCHEMA/SCHEMA compiler-directive statement in order to name the subschema to be accessed. Include only one SUBSCHEMA/SCHEMA statement.

    You must include the version number of the named schema.
  - Non-standard data transfer — You must include a SUBSCHEMA/SCHEMA compiler-directive statement in order to name the subschema to be accessed. Include only one SUBSCHEMA/SCHEMA statement.
- Copying record descriptions
  - Standard data transfer — Use the COPY INPUT RECORD/LOGICAL RECORD and COPY OUTPUT RECORD/LOGICAL RECORD statements. These descriptions are copied in as Input or Output Specifications.
  - Non-standard data transfer — Copy records into program variable storage using the COPY SUBSCHEMA RECORDS statement. Descriptions can also be copied into the program for individual records by using the COPY DS RECORD statement. In either case, these record descriptions are copied in as data structures.
- Binding a run unit
  - Standard data transfer — Programs are automatically signed on to CA-IDMS when the program is run.
  - Non-standard data transfer — The BIND RUN-UNIT statement is not allowed under combined data transfer. If a BIND RUN-UNIT statement has been included in the program, the DMLR precompiler issues an error message for the statement.
- Binding database records
  - Standard data transfer — Database records named in an IDMS FILE ID statement are bound automatically.
  - Non-standard data transfer — Use the COPY SUBSCHEMA BINDS statement.

Database records can also be bound individually by using the BIND RECORD functional DML statement.

- Readying database areas
  - Standard data transfer — Areas associated with records named in an IDMS FILE ID statement are readied automatically by the interface routine IDMSRPGI. This routine readies an area in exclusive update mode if records in this area are stored or modified by the program; otherwise, the area is readied in shared retrieval usage mode.

    You can suppress the automatic readying of areas using the SR option in the SUBSCHEMA/SCHEMA statement. When this option is specified, all areas accessed by the program (i.e., those associated with standard data transfer, or both) are automatically readied according to the default usage modes defined in the subschema. READY statements are not allowed in programs that specify the SR option.

    The READY statement cannot be used to ready any area associated with standard data transfer (regardless of whether or not the area is also associated with non-standard data transfer). An error message is generated for READY statements that affect such areas.
  - Non-standard data transfer — Areas are readied by using the READY DML statement.
- Releasing database areas
  - Standard data transfer — Performed by a FINISH file automatically generated in the program by the DMLR precompiler. The DMLR precompiler automatically provides the necessary File and Output Specifications for this file. Note that the FINISH file terminates database processing at the conclusion of the standard data transfer within a program, regardless of any non-standard data transfer also taking place within the program.
  - Non-standard data transfer — The DML FINISH statement is not allowed under combined data transfer. If a finish statement appears in the Calculation Specifications, the DMLR precompiler issues an error message for the statement.

L.7 Compiling and link-editing

This section contains the JCL used to prepare RPG II source code containing DML statements for execution. Link-edit considerations are also discussed.

L.7.1 Compiling an RPG II Program

To compile an RPG II program under the DMLR precompiler, execute the program IDMSDMLR, and then execute the following:

1. The RPG II compiler
2. The assembler (in order to prepare the CA-IDMS runtime interface routine, IDMSRPGI)
3. The linkage editor

IDMSDMLR: Input to IDMSDMLR consists of:

- A source program written in RPG II/DML
- Protocol/control information
- Dictionary record descriptions

Output from IDMSDMLR is as follows:

- A source RPG II program
- A DML source listing and diagnostics
- Assembler source code for the IDMSRPGI interface routine

RPG II compiler: Input to the RPG II compiler consists of the source program produced by IDMSDMLR.

Output is as follows:

- An object program
- RPG II listings

Assembler: Input to the assembler consists of the assembler code for the IDMSRPGI interface routine.

Output from the assembler is object code.

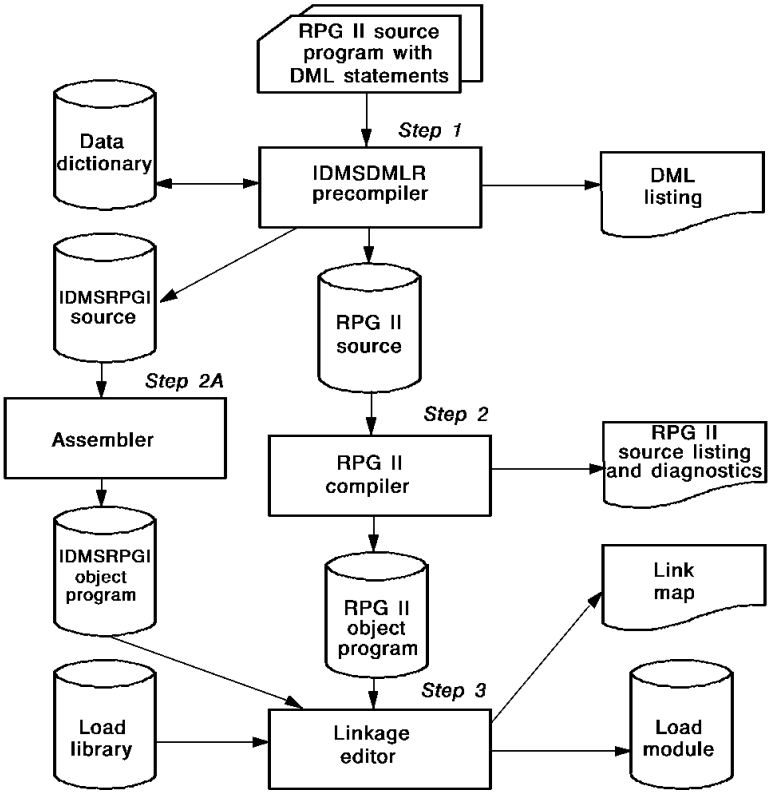
Linkage editor: Input to the linkage editor consists of the:

- Object program produced by the RPG II compiler
- IDMSRPGI object code

Output is as follows:

- A load module
- A link-edit map

Compiling an RPG II program: The following diagram illustrates the steps involved in compiling an RPG II program.



L.7.2 JCL

Central version JCL: The MVS and DOS JCL used to compile and link edit the DMLR source statements under the CA-IDMS central version and local mode is shown below.

IDMSDMLR (MVS)

```
//* ********************************************************************
//* **                      PRECOMPILE RPG II PROGRAM                 **
//* ********************************************************************
//precomp  EXEC PGM=IDMSDMLR,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmgs    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYS001   DD   UNIT=disk,SPACE=(TRK,(10,10))
//SYS002   DD   UNIT=disk,SPACE=(TRK,(10,10))
//SYS003   DD   UNIT=disk,SPACE=(TRK,(10,10))
//SYSPCH   DD   DSN=&&SOURCE,DISP=(NEW,PASS,DELETE),
//              UNIT=disk,SPACE=(TRK,(10,5),RLSE),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYS004   DD   DSN=&&rpgi,DISP=(NEW,PASS),
//              UNIT=disk,SPACE=(TRK,(5,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//dcmsg    DD  DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSLST   DD   SYSOUT=A
//SYSIDMS  DD   *

DMCL=dmcl-name
DICTNAME=dictionary-name
Other SYSIDMS parameters, as appropriate
/*

//SYSIPT  DD   *


RPGII DML source statements
//* ********************************************************************
//* **                     COMPILE RPGII PROGRAM                      **
//* ********************************************************************
//rpgcmp   EXEC PGM=RPGII,REGION=160,
//              PARM='DECK,NOLOAD,NOLIB,BUF=50000,SIZE=150K'
//STEPLIB  DD   DSN=sys1.rpglib,DISP=SHR
//SYSUT1   DD   UNIT=disk,SPACE=(TRK,(10,5))
//SYSUT2   DD   UNIT=disk,SPACE=(TRK,(10,5))
//SYSUT3   DD   UNIT=disk,SPACE=(TRK,(10,5))
//SYSPUNCH DD   DSN=&&object,UNIT=disk,DISP=(NEW,PASS),
//              SPACE=(80,(400,40))
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSN=&&source,DISP=(OLD,DELETE)
//* ********************************************************************
//* **                     ASSEMBLER                                  **
//* ********************************************************************
//ASSEMBLE   EXEC ASMGC
//ASM.SYSLIB DD
//           DD   DSN=idms.maclib,DISP=SHR
//ASM.SYSIN  DD   DSN=&&rpgi,DISP=(OLD,DELETE)
```

```
//* ******************************************************************
//* **                     LINK PROGRAM MODULE                      **
//* ******************************************************************
//link     EXEC PGM=IEWL,REGION=96K,
//              PARM='XREF,LIST,LET,SIZE=(96K,12K)'
//SYSUT1   DD   UNIT=disk,SPACE=(TRK,(20,5))
//SYSLIB   DD   DSN=sys1.rpglib,DISP=SHR
//loadlib  DD   DSN=idms.loadlib,DISP=SHR
//SYSLMOD  DD   DSN=user.loadlib,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSLIN   DD   DSN=&&object,DISP=(OLD,DELETE)
//         DD   DSN=&&loadset,DISP=(OLD,DELETE)
//SYSIN    DD   *
 INCLUDE loadlib(IDMS)      required
 INCLUDE loadlib(IDMSCANC)  required for BATCH and DC-BATCH
                                  if using IDMS-STATUS module
 INCLUDE loadlib(IDMSOPTI)  optional;
                                  BATCH and DC-BATCH only
 ENTRY   userentry
 NAME    userprog(R)
/*
```

1024K	size of region to run IDMSDMLC
idms.dba.loadlib	data set name of the load library containing the DMCL and database name table load modules
idms.loadib	data set name of the load library containing the executable CA-IDMS Release 12.0 modules
sysctl	ddname of SYSCTL file
idms.sysctl	data set name of SYSCTL file
disk	symbolic device name for work files
&&source	name of temporary data set output from DMLR precompiler (source program)
dcmsg	ddname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	data set name of the system message (DDLDCMSG) area
SYSIDMS	name of the parameter file where you specify physical requirements of the environment, runtime directives, or operating system-dependent file information.
dmcl-name	specifies the name of the dictionary the DMLF precompiler should access
dictionary-name	identifies the CA-IDMS system to bind to at runtime
sys1.rpglib	data set name of the RPG II library
&&object	name of temporary data set output from RPG II compiler
user.loadlib	user application load library

`userentry`	name of program entry point
`idms.maclib`	data set name of CA-IDMS macro library
`userprog`	name of user application program
`&&loadset`	name of temporary data set output from assembler

Note: Depending on the CV operating environment, an IDMSOPTI module link edited with IDMSDMLR can be used in place of or in addition to the SYSCTL file.

Parameter option available to code in the EXEC statement of the precompile step is:

`DICTNAME=dictionary-name`

where *dictionary-name* specifies the name of the dictionary the precompiler should access.

▸▸ For a complete description of all the parameter option, see *CA-IDMS SQL Programming*.

For a complete description of SYSIDMS parameters and syntax, see *CA-IDMS Navigational DML Programming* and *CA-IDMS Database Administration*.

Local mode JCL: To execute the IDMSDMLR precompiler in local mode remove the SYSCTL statement from step1 and replace it with the following:

IDMSDMLR (Local)

```
//dictdb   DD DSN=idms.appldict.ddldml,DISP=SHR
//sysjrnl  DD DSN=idms.tapejrnl,DISP=(NEW,KEEP),UNIT=tape
```

`dictdb`	ddname of the application dictionary DDLDML area
`idms.appldict.ddldml`	data set name of the application dictionary DDLDML area
`sysjrnl`	ddname of journal file
`idms.tapejrnl`	data set name of journal file

Note: In local mode you will also need to specify the name of the DMCL in the SYSIDMS parameter file unless you're using the default name of IDMSDMCL. In addition, you must include the load library containing the DMCL in the STEPLIB concatenation:

```
//STEPLIB  DD  DSN=idms.dbalib,DISP=SHR
//         DD  DSN=idms.loadlib,DISP=SHR
```

L.8 Link-edit considerations

The modules involved in the link edit of an application program contain six external references. Some must be resolved, others can be left unresolved, depending on the mode of operation.

The following table lists and explains the external references.

Reference	Referenced By	Resolved By	Comments
ABORT	Application program	IDMSCANC	Should be resolved.
IDMS	Application Program	IDMS	Must be resolved.
IDMSOPTI (1)	IDMS	IDMSOPTI module	Must be resolved if using the central version without a SYSCTL file

(1)IDMSOPTI is a weak external reference (WXTRN).

L.9 Standard RPG II data transfer return codes

After servicing a program request, the interface routine generates a return code that indicates the status of the request, and places this code in register 15.

If an RPG II program is terminated by the interface, the interface stores registers 0 through 15 in memory. The programmer can examine the error code returned to register 15 in the memory dump to determine the cause of the error. This register can be found easily by locating the constant ERROREGS in the text portion of the dump; registers 0 through 15 are stored immediately to the right of this constant.

Normal return codes: The following return codes reflect normal processing conditions and do not force the program to terminate:

Status code	Meaning
00	The request has been executed successfully.
04	The end of the file has been reached.

Error conditions: Return codes of 08 or higher, however, reflect error conditions. When one of these return codes is detected in register 15, the HO indicator in the RPG II program is set on and the program is forces to terminate.

With the exception of a return code of 8, all of the return codes indicate that inconsistency exists between the code generated by the DMLR precompiler and the interface routine, or between RPG II and the interface routine. The return codes that indicate error conditions are listed in the following table.

Status code	Meaning
08	An unexpected CA-IDMS error status has been returned to the ERST field of the IDMS communications block. (The programmer should check the value of the ERST field in the dump for this error status.)
12	The interface has been invoked, but CA-IDMS processing is specified as non-standard only.
16	The request is for a file not specified in the interface's CA-IDMS file descriptor tables (which are generated by the DMLR precompiler).
20	The interface's common record is too short to contain the longest record to be accessed.
24	The request passed from RPG II is for other than a read, write, or close.

Status code	Meaning
28	A read has been requested for a file for which end-of-file has already been encountered; or a request has been made for a file for which an error condition has been indicated; or a request has been made for a file that has been closed.
32	A read has been requested by RPG II for file specified as output only.
36	A write has been requested by RPG II for a file specified as input only.
40	The database access technique specified in the interface's IDMS file descriptor table is not supported.
44	A request has been made for a logical record, but the program has not specified logical-record access (i.e., has not specified L as a database access technique).
48	A write has been issued by RPG II for an input-output record, but no prior read has been issued.
4092	The interface has been invoked, but should not have been.

Appendix M. Optional Online COBOL Functionality

M.1 About this appendix M-3
M.2 COBOL II and LE COBOL task management M-4
M.3 Suppression of certain program checks M-7
M.4 Loading VS-COBOL programs into XA storage M-10

M.1 About this appendix

Several APARs have been written that affect the performance and/or functionality of COBOL programs in the online CA-IDMS/DC system. This appendix discusses the effects of the various APARs.

Note: This discussion applies only to online programs running in a CA-IDMS/DC/UCF region. It does not apply to batch programs or to programs running in a CICS region or under control of another TP monitor, even if such programs access an IDMS database via LOCAL or CV mode.

M.2 COBOL II and LE COBOL task management

Several optional APARs have to do with the management of a COBOL II task thread or a LE COBOL process(environment) and enclave. To better understand the concept of a COBOL task thread, first consider a batch COBOL job in which IDMS is **not** involved. When a COBOL II program is first invoked, the COBOL support code causes the load of a small program called IGZCTCO. As the COBOL II runtime system is built, control information is placed in the copy of IGZCTCO that has been loaded into the address space. If the top level program (call it program A) issues a CALL IDENTIFIER to a second COBOL program (call it program B), the COBOL II support code finds the existing copy of IGZCTCO. Program B is entered using the same COBOL II environment. The first time program B is entered, its WORKING STORAGE is initialized according to any VALUE clauses coded.

If program B does a GOBACK to program A and then program A issues a second CALL IDENTIFIER to program B, program B is normally entered with the same WORKING STORAGE values left from the previous invocation. The VALUE clauses are **not** reinitialized.

In Language Environment for OS/390, the concepts of the LE process and enclave are roughly analagous to the COBOL II task thread. See IBM documentation for a more complete discussion of these concepts.

Now let us return to the discussion of COBOL II in an online CA-IDMS/DC/UCF system. When COBOL II support was first introduced for CA-IDMS/DC/UCF, every invocation of a COBOL program via a TRANSFER RETURN from another COBOL program caused a new IGZCTCO to be loaded. The COBOL II task thread was recreated. Thus if Task A invokes COBOL II program X, which does a TRANSFER CONTROL ten times to program Y, the COBOL II task thread was built eleven times -- once for program X and ten times for program Y. Moreover, if program Y terminated with a DC RETURN instead of a GOBACK, all the storage associated with each invocation was preserved until task termination. This causes serious overhead of CPU and potentially of storage utilization.

To reduce the overhead of constantly creating new COBOL II environments, the CA-IDMS/DC/UCF COBOL II support was modified to load only one copy of IGZCTCO per task. Using this method, if main program X issues a TRANSFER CONTROL ten times to program Y, the COBOL II environment is built only once. The drawback is that certain functionality is changed. In particular, WORKING STORAGE is not reinitialized each time program Y is entered. Also, recursive TRANSFER CONTROL (Program X issues TRANSFER CONTROL TO X) is not allowed. Since some existing applications depended on those features, optional APARs were developed to allow use of one method or the other. Unfortunately, the CA-IDMS/DC/UCF default methodology changed from release to release and sometimes within one release.

Note: IGZCTCO is handled differently in CA-IDMS/DC/UCF systems that are operating with an IBM Language Environment runtime system (such as LE for OS/390). COBOL II programs can be used in these systems, but prior to

release 14.1, a new IGZCTCO was used for every TRANSFER to a COBOL II program.

Beginning with release 14.1, online COBOL II programs in an LE runtime environment will run most efficiently if RHDCLEFE is defined in the CA-IDMS/DC/UCF Sysgen as documented in the release 14.1 *Features Guide*. This gives functionality similar to that documented for the "single IGZCTCO" method shown below. In that case, the COBOL II program is handled as though it were compiled under LE COBOL.

The following table describes how to utilize each of the two methods for the latest maintenance of all currently supported CA-IDMS/DC/UCF releases. This table supercedes the documentation in any previous APARs or PDCs. Note that the table is divided into several sections depending on the COBOL compiler level and the runtime level.

Part 1: This part of the table contains programs compiled under COBOL II and using COBOL II runtime libraries.

Release	Method 1 (Most CPU efficient) Use single IGZCTCO per task	Method 2 (Special functionality) Use new IGZCTCO each TRANSFER
10.21PS	Default	Apply optional APAR 88-06-1105
12.01	Default (see note below)	Apply optional APAR LS12053.
14.0 and later	Turn on optional bit 49 in RHDCOPTF.	Default.

Note: In release 12.01, prior to maintenance level 9607, it is necessary to apply APAR GO97250 to obtain the default condition shown above. With application of GO97250, optional APAR GS19348 is obsolete.

Note: Optional bit 49 is not valid in a LE/370 runtime environment.

Part 2: This part of the table contains programs compiled under COBOL II and using LE runtime libraries. RHDCLEFE is not in use.

Release	Method 1 (Most CPU efficient) Use single IGZCTCO per task	Method 2 (Special functionality) Use new IGZCTCO each TRANSFER
10.21PS and later	Not available	Default

Part 3: This part of the table contains programs compiled under COBOL II and using LE runtime libraries. RHDCLEFE is in use.

Release	Method 1 (Most CPU efficient) Reuse same process/enclave	Method 2 (Special functionality) Use new process/enclave
10.21PS	Not available	Default
12.01	Not available	Default
14.0	Not available	Default.
14.1 and later	Default	Optional bit 196.

Part 4: This part of the table contains programs compiled under LE COBOL and using LE runtime libraries.

Release	Method 1 (Most CPU efficient) Reuse same process/enclave (see first note below)	Method 2 (Special functionality) Use new process/enclave (see first note below)
10.21PS	Not available	Default
12.01	Not available	Default
14.0	Default	Optional bit 196. (see second note below)
14.1 and later	Default	Optional bit 196.

Note: When using RHDCLEFE with release 14.1 and later, the default is to preserve both the LE environment (process) and the LE enclave when invoking the same program multiple times in the same DC task. When not using RHDCLEFE, the environment is preserved for LE COBOL programs, but not the enclave.

In release 14.0, prior to maintenance level 9810, it is necessary to apply APAR LS40957 in order for optional bit 196 to have any effect. That APAR is automatically applied at level 9810 and above.

M.3 Suppression of certain program checks

Various conditions may arise in a COBOL program that can cause program checks. Some conditions (e.g. an attempt to update protected storage) will always result in a program check. Other conditions can be controlled by setting the program mask in the PSW. The program mask consists of 4 bits:

- Bit 1: Fixed-point overflow
- Bit 2: Decimal overflow
- Bit 3: Exponent underflow
- Bit 4: Significance

If a bit is off, the corresponding interrupt is masked. The condition will not cause a program check. Invalid data may occur unless special programming is done to handle the potential condition. If a bit is on, the condition will cause a program check.

See the *IBM System/370 Extended Architecture Principle of Operations* for more information on the program mask.

The normal mode of operation for online CA-IDMS/DC/UCF programs is to run with a program mask of B'1110'. This allows all program checks to occur except for significance exceptions. Significance exceptions generally do not occur in COBOL programs. The mask is set to this value for other languages that need to mask significance exceptions to operate properly.

Several APARs and RHDCOPTF bits have been provided for different releases of CA-IDMS/DC/UCF to allow program mask settings other than the default. The most common need for these APARs are programs that expect decimal overflows and that have ON SIZE clauses to handle the overflow condition. Such programs will not operate properly using the default value.

The APARs or optional bit settings that affect the program mask are as shown below. Note that these APAR and bit settings affect all user mode programs, not just COBOL programs. They are presented here because the optional settings are most commonly used for specialized COBOL applications.

Release	APAR	Effect
12.01	GS27987	This optional APAR is only for release 12.01 clients at a maintenance level lower than 9707. It sets the program mask to B'1011'. This masks decimal overflow interrupts, but allows all other interrupts to cause a program check.

Release	APAR	Effect
12.01	GS97248	This optional APAR is only for release 12.01 clients at a maintenance level lower than 9707. It modifies the mask bits set in APAR GS27987 to B'1010'. This masks decimal overflow and significance interrupts, but allows all other interrupts to cause a program check.
12.01	LS39045	This optional APAR is only for release 12.01 clients at a maintenance level 9707 or higher. It sets the program mask to B'1011'. This masks decimal overflow interrupts, but allows all other interrupts to cause a program check. See the description in the APAR for information on how to modify the APAR to set the program mask to any desired value.
14.0 14.1	LS31790	After applying this required APAR, a new set of optional bits can be used to set the program mask on initial entry into an application program.

The rest of this section explains the effect of LS31790 in more detail.

If OPT00184 is defined, then OPT00148 will be ignored and the program mask will be set based on #DEFOPTF bits 185-188 as follows:

- If OPT00185 is defined, fixed-point overflow exceptions will result in an interrupt (program check). When it is not defined, fixed-point overflows will not result in an interrupt.
- If OPT00186 is defined, decimal overflow exceptions will result in an interrupt (program check). When it is not defined, decimal overflows will not result in an interrupt.
- If OPT00187 is defined, exponent underflow exceptions will result in an interrupt (program check). When it is not defined, exponent underflows will not result in an interrupt.
- If OPT00188 is defined, significance exceptions will result in an interrupt (program check). When it is not defined, significance exceptions will not result in an interrupt.

If OPT00184 is not defined and OPT00148 is defined, then the program mask will be set to binary 1010, i.e., fixed-point overflow and exponent underflow will cause an interrupt, but decimal overflow and significance exceptions will not.

If neither OPT00184 nor OPT00148 is defined, then the program mask will be set to binary 1110, i.e., fixed-point overflow, decimal overflow, and exponent underflow exceptions will cause an interrupt, but significance exceptions will not. This is the standard DC/UCF default.

Note that this APAR only affects the program mask for initial entry into a user mode program each time it is invoked. If the application program changes its program mask and then makes a system call (a database OBTAIN, for example), the user mode program mask will be saved when the transition is made from user mode to system mode. The program mask is always set to binary 1110 while the DC/UCF system is running in system mode. When the system has satisfied the request and returns to user mode, the latest user-mode program mask will be restored.

For sites that use ON SIZE checking to handle expected decimal overflow conditions, the recommended RHDCOPTF bit setting is OPTF148. This will result in a program mask of B'1010' and will prevent program checks from decimal overflow and significance exceptions.

Note: LS31790 is a required APAR. It is automatically applied at maintenance level 9810 for release 14.0 and on all release 14.1 base and maintenance levels.

M.4 Loading VS-COBOL programs into XA storage

VS-COBOL II and LE COBOL programs can and normally should be linked with AMODE 31 and RMODE ANY. Old-style VS-COBOL programs, which run in batch, must run with AMODE 24 and RMODE 24. However, when running online VS-COBOL programs in a CA-IDMS/DC/UCF region, it is permissable to run with AMODE 31 and RMODE 24. This is the normal recommended AMODE/RMODE setting for online VS-COBOL program. This allows the WORKING STORAGE for VS-COBOL programs to be allocated in XA storage. Since multiple copies of WORKING STORAGE may be allocated simultaneously (when multiple tasks are running that use the same program), this feature considerably reduces the amount of below-the-line storage required.

Some sites have a large number of COBOL programs and may wish to link VS-COBOL programs with AMODE 31 and RMODE ANY. This allows the programs to be loaded into the 31-bit (above-the-line) PROGRAM POOL. Caution should be used before utilizing this feature. If a program that is loaded above the line issues a COBOL verb that causes an illegal SVC to be issued or if the program is compiled with illegal compile options, the entire CA-IDMS/DC/UCF region may be abended. In some cases, the entire operating system may be abended. Illegal COBOL verbs and compile options are listed in Chapter 1, “Introduction to CA-IDMS Data Manipulation Language” on page 1-1.

If online VS-COBOL programs are to be linked RMODE(ANY) and run under Release 12.01 or earlier, an optional APAR must be applied. No optional APAR is required for release 14.0 and above, but the cautions listed above must be observed. The optional APARs are as follows:

Release	APAR
10.21PS (MVS)	90-09-1003
10.21PS (VSE)	Not available
12.01 (MVS)	CS82390
12.01 (VSE)	GS53516

Index

Numerics

18-byte communications blocks J-3

A

ABEND 5-19
abend exits 5-212—5-213
ACCEPT 5-20—5-21, 5-30
ACCEPT BIND ADDRESS 5-22
ACCEPT DATABASE STATISTICS 5-23—5-24
ACCEPT DB-KEY FROM CURRENCY 5-25—5-26
ACCEPT DB-KEY RELATIVE TO CURRENCY 5-27—5-28
ACCEPT page-info-location 5-29
ACCEPT PROCEDURE CONTROL LOCATION 5-31
ACCEPT TRANSACTION STATISTICS 5-32—5-34
AID keys
  See attention ID keys
application program information block 5-31, 5-39
ATTACH 5-35—5-36
attention ID keys 5-133
  DC-AID-CONDITION-NAMES 5-135
automatic editing/error handling 5-141, 5-180, 5-181, 5-182
AUTOSTATUS protocols 3-33—3-34, 4-6, 5-41, 5-43, 5-130

B

basic mode
  READ TERMINAL 5-198—5-200
  WRITE TERMINAL 5-251—5-253
  WRITE THEN READ TERMINAL 5-254—5-257
batch processing 1-3
BIND MAP 5-37—5-38
BIND PROCEDURE 5-39—5-40
BIND RECORD 5-41—5-42
BIND RUN-UNIT 5-43—5-46
BIND TASK 5-47
BIND TRANSACTION STATISTICS 5-48
boolean selection criteria
  See WHERE clause (LRF)
BS2000 JCL A-16

C

CA-IDMS/DC I-3
call expansions
  See call formats
call formats D-3—D-14
CALL statements
  database 4-6
  DC 4-6
  DC-BATCH 4-6
  Non-DC TP monitors 4-6
CHANGE PRIORITY 5-49
CHECK TERMINAL 5-50—5-52
checkpoints
  COMMIT 5-53—5-54
  FINISH 5-110—5-111
  ROLLBACK 5-207—5-208
CMS commands A-19
COBOL compiler 1-9, A-4
COBOL II M-4
COBOL program, samples
  batch B-3
  online C-3
comment generation 2-5
COMMIT 5-53—5-54
communications blocks 3-4—3-17
communications blocks, 18-byte 4-6, J-3
comparison operators
  See WHERE clause (LRF)
compiler options 2-3—2-7
  comment generation 2-5
  dictionary ready override 2-4
  list generation 2-6
  log suppression 2-7

compiler-directive statements 4-3—4-24
DATA DIVISION 4-9—4-21
ENVIRONMENT DIVISION 4-5—4-8
FILE SECTION 4-9
IDENTIFICATION DIVISION 4-4
MAP SECTION 4-10—4-12
PROCEDURE DIVISION 4-22—4-24
SCHEMA SECTION 4-10
WORKING-STORAGE/LINKAGE SECTIONs 4-12—4-21
compiling CA-IDMS programs 1-9
CONNECT 5-55—5-57
control statements
FINISH 5-110—5-111
IF 5-130—5-132
KEEP 5-143—5-144
READY 5-201—5-203
ROLLBACK 5-207—5-208
COPY IDMS MAP-BINDS
See COPY IDMS statement
COPY IDMS statement
COPY IDMS MAP-BINDS 5-37
COPY IDMS SUBSCHEMA-BINDS 4-22, 5-41, 5-43
in FILE SECTION of DATA DIVISION 4-9
in MAP SECTION of DATA DIVISION 4-10
in PROCEDURE DIVISION 4-22
in WORKING-STORAGE/LINKAGE SECTIONs of DATA DIVISION 4-12
COPY IDMS SUBSCHEMA-BINDS
See COPY IDMS statement
cursor position
See also INQUIRE MAP
MODIFY MAP 5-178

D

data communications 1-3
DATA DIVISION 4-9—4-21
FILE SECTION 4-9
MAP SECTION 4-10—4-12
SCHEMA SECTION 4-10
WORKING-STORAGE/LINKAGE SECTIONs 4-12—4-21
data structure diagram G-3
date
See GET TIME
DBNAME parameter
in BS2000 JCL A-16
DC RETURN 5-58—5-62
DC-BATCH 3-4, 3-12, 3-18, 3-33, 4-6, 4-7, 4-16, 5-47, 5-118, 5-192
DCMT VARY DYNAMIC PROGRAM H-5
DCMT VARY DYNAMIC TASK 5-35
DDLDCQUE area 5-190
DDLDCRUN area 5-190
DDLDCSCR area 5-193
DDLDML area 2-4, 2-7
DEBUG option 4-6
DELETE QUEUE 5-63—5-64
DELETE SCRATCH 5-65—5-66
DELETE TABLE 5-67—5-68
DEQUEUE 5-69—5-70
destination
SEND MESSAGE 5-209
WRITE PRINTER 5-248
dictionary ready override 2-4
See also READY
DISCONNECT 5-71—5-73
DML compiler
execution of A-4
keywords E-3
with non-DC TP monitor F-4
DML usage mode 1-5, 4-12
DMLC processor
See precompiler
DMLIST (compiler option) 2-6
dump
ABEND 5-19
SNAP 5-218—5-219

E

ECB
See event control block (ECB)
END LINE TERMINAL SESSION 5-74
END TRANSACTION STATISTICS 5-75—5-76
ENDPAGE 5-77
ENQUEUE 5-78—5-80
ENVIRONMENT DIVISION 4-5—4-8
ERASE 5-81—5-86
ERASE (LRF) 5-87—5-88
error detection 3-29—3-34
See also automatic editing/error handling
See also AUTOSTATUS protocols
ERROR-STATUS field 3-18
event control block (ECB) 5-189, 5-214, 5-232
executing CA-IDMS programs 1-10

F

FILE SECTION 4-9
FIND/OBTAIN 5-89
FIND/OBTAIN CALC/DUPLICATE 5-90—5-91
FIND/OBTAIN CURRENT 5-92—5-94
FIND/OBTAIN DB-KEY 5-95—5-97
FIND/OBTAIN OWNER 5-98—5-100
FIND/OBTAIN WITHIN SET USING SORT KEY 5-101—5-103
FIND/OBTAIN WITHIN SET/AREA 5-104—5-109
FINISH 5-110—5-111
FREE STORAGE 5-112—5-113

G

GET 5-114—5-115
GET QUEUE 5-116—5-119
GET SCRATCH 5-120—5-122
GET STORAGE 5-123—5-127
GET TIME 5-128—5-129

I

IDENTIFICATION DIVISION 4-4
IDMS communications block 3-4—3-10
  See also status codes
  field descriptions 3-5
  figure 3-5
  update (figure) 3-8
IDMS communications block, 18-byte J-3
IDMS-ABORT SECTION 3-32
IDMS-CONTROL SECTION 4-5—4-8
IDMS-DC communications block 3-12—3-17
  See also status codes
  field descriptions 3-13
  figure 3-12
IDMS-DC communications block, 18-byte J-3
IDMS-RECORDS statement 4-7
IDMS-STATUS routine 3-29—3-32
IF 5-130—5-132
  AUTOSTATUS considerations 5-130
INQUIRE MAP
  See also mapping mode
  general discussion 5-133
  moving map-related data 5-133—5-136
  testing for cursor position 5-138
  testing for global map input conditions 5-136—5-137
  testing for input non-zero status conditions 5-138—5-141
integrated indexing
  RETURN 5-204, 5-206

J

journal file
  COMMIT 5-53—5-54
  FINISH 5-110—5-111
  ROLLBACK 5-207—5-208
  WRITE JOURNAL 5-234—5-235

K

KEEP 5-143—5-144
KEEP LONGTERM 5-145—5-150
kept storage
  FREE STORAGE 5-112—5-113
  GET STORAGE 5-123—5-127
keywords E-3—E-5

L

LE COBOL M-4
LE-compliant language compilers
  executing programs under CA-IDMS/DC I-6
  supported compilers I-4
  supported functions I-10
  unsupported functions I-11
  using I-3—I-11
line mode
  READ LINE FROM TERMINAL 5-195—5-197
  WRITE LINE TO TERMINAL 5-236—5-239
list generation 2-6
LOAD TABLE 5-151—5-154
locks
  See record locks
log suppression 2-7
Logical Record Facility
  logical-record clauses 5-258—5-265
  MODIFY 5-174—5-175
  OBTAIN 5-186—5-188
  ON clause 5-262—5-264
  status codes 5-264—5-265
  STORE 5-228—5-229
  WHERE clause 5-258—5-262
logical terminal 5-20, 5-48, 5-145, 5-210
logical-record clauses
  general discussion 5-258
  ON clause 5-262—5-264
  WHERE 5-258—5-262

logical-record request control (LRC) block 3-10—3-12
field descriptions 3-11
figure 3-10
LR usage mode 1-5, 4-13
LRC block
See logical-record request control (LRC) block
LRF
See Logical Record Facility

M

map
attributes 5-182
field list 5-140
message area 5-162
modifying 5-176
MAP IN 5-155—5-159
MAP OUT 5-160—5-166
MAP OUTIN 5-167—5-170
map request block (MRB) 5-37, 5-176
MAP SECTION 4-10—4-12
mapping mode
INQUIRE MAP 5-133—5-142
MAP IN 5-155—5-159
MAP OUT 5-160—5-166
MAP OUTIN 5-167—5-170
MODIFY MAP 5-176—5-185
STARTPAGE 5-220—5-222
MIXED usage mode 1-5, 4-13
MODE statement
See operating mode
modification statements
MODIFY 5-171—5-173
STORE 5-223—5-227
MODIFY 5-171—5-173
MODIFY (LRF) 5-174—5-175
MODIFY MAP 5-176—5-185
MRB
See map request block (MRB)
MVS JCL A-5

N

native mode
MAP IN 5-155
MAP OUT 5-161
WRITE PRINTER 5-246, 5-247
navigational DML statements 1-5
NODENAME parameter
in BS2000 JCL A-16
NODMLIST (compiler option) 2-6
non-navigational DML statements 1-6
See also Logical Record Facility

O

OBTAIN
See FIND/OBTAIN statements
OBTAIN (LRF) 5-186—5-188
ON clause (LRF) 5-262—5-264
expanded syntax 5-263—5-264
online DML statements 1-7
online processing 1-3
operating mode 4-5—4-6
operator's console 5-240

P

pageable maps
See mapping mode
Performance improvements I-12
POST 5-189
Precompiler
compiler options 2-3—2-7
compiler-directive statements 4-3—4-24
execution of 1-9
general discussion 1-9
precompiler options
See compiler options
preprocessor
See precompiler
print
classes 5-246
destinations 5-246
queues 5-246
PROCEDURE DIVISION 4-22—4-24
program activity statistics
See log suppression
Program checks M-7
program expansion element (PXE) 3-12, 4-18
program management
DELETE TABLE 5-67—5-68
LOAD TABLE 5-151—5-154
SET ABEND EXIT 5-212
SET ABEND EXIT (STAE) 5-213
TRANSFER CONTROL 5-230—5-231
program registration 4-4, 5-43
PROTECTED-UPDATE (compiler option) 2-4
PROTOCOL clause 4-5
See also operating mode

protocols
AUTOSTATUS 3-33—3-34, 4-6, 5-41, 5-43, 5-130
PROTOCOL clause 4-5
standard protocols (table) 4-7
USER-DEFINED 3-34—3-35
PUT QUEUE 5-190—5-192
PUT SCRATCH 5-193—5-194
PXE
See program expansion element (PXE)

Q

queue management
GET QUEUE 5-116—5-119
PUT QUEUE 5-190—5-192
queues
BIND TASK 5-47
DEQUEUE 5-69—5-70
ENQUEUE 5-78—5-80

R

READ LINE FROM TERMINAL 5-195—5-197
READ TERMINAL 5-198—5-200
READY 5-201—5-203
record locks
KEEP 5-144
KEEP LONGTERM 5-150
recovery
ROLLBACK 5-207—5-208
WRITE JOURNAL 5-234—5-235
RETRIEVAL (compiler option) 2-4
retrieval statements
FIND/OBTAIN 5-89
FIND/OBTAIN CALC/DUPLICATE 5-90—5-91
FIND/OBTAIN CURRENT 5-92—5-94
FIND/OBTAIN DB-KEY 5-95—5-97
FIND/OBTAIN OWNER 5-98—5-100
FIND/OBTAIN WITHIN SET USING SORT KEY 5-101—5-103
FIND/OBTAIN WITHIN SET/AREA 5-104—5-109
GET 5-114—5-115
OBTAIN (LRF) 5-186—5-188
RETURN 5-204—5-206
RHDCLEFE I-12
ROLLBACK 5-207—5-208

S

SCHEMA SECTION 4-10
SCHEMA-COMMENTS (compiler option) 2-5
scratch management
GET SCRATCH 5-120—5-122
PUT SCRATCH 5-193—5-194
selection criteria
See WHERE clause (LRF)
SEND MESSAGE 5-209—5-211
Sequential Processing Facility
RETURN 5-204—5-206
SET ABEND EXIT 5-212
SET ABEND EXIT (STAE) 5-213
SET TIMER 5-214—5-217
SNAP 5-218—5-219
source statement listing
See compiler options
See list generation
STARTPAGE 5-220—5-222
status codes
database major codes 3-18
database minor codes 3-19
DC major codes 3-24
DC minor codes 3-24
storage management
FREE STORAGE 5-112—5-113
GET STORAGE 5-123—5-127
STORE 5-223—5-227
STORE (LRF) 5-228—5-229
subschema usage modes
DML 1-5, 4-12
LR 1-5, 4-13
MIXED 1-5, 4-13
suppress log
See log suppression
Suppression of program checks M-7

T

tables
DELETE TABLE 5-67—5-68
LOAD TABLE 5-151—5-154
task management
COBOL II M-4
LE COBOL M-4
POST 5-189
WAIT 5-232—5-233
teleprocessing monitors
notes to users of F-3—F-5
protocols for use with (table) 4-7
terminal management
INQUIRE MAP 5-133—5-142
MAP IN 5-155—5-159

terminal management *(continued)*
MAP OUT 5-160—5-166
MAP OUTIN 5-167—5-170
MODIFY MAP 5-176—5-185
READ LINE FROM TERMINAL 5-195—5-197
READ TERMINAL 5-198—5-200
STARTPAGE 5-220—5-222
WRITE PRINTER 5-246—5-250
WRITE TERMINAL 5-251—5-253
WRITE THEN READ TERMINAL 5-254—5-257
time
See GET TIME
time interval
See SET TIMER
time management
GET TIME 5-128—5-129
SET TIMER 5-214—5-217
transaction statistics block (TSB)
ACCEPT TRANSACTION STATISTICS 5-32—5-34
BIND TRANSACTION STATISTICS 5-48
END TRANSACTION STATISTICS 5-75—5-76
TRANSFER CONTROL 5-230—5-231
NORETURN (XCTL) parameter 5-230
RETURN (LINK) parameter 5-230
TSB
See transaction statistics block (TSB)

U

user storage
FREE STORAGE 5-112—5-113
GET STORAGE 5-123—5-127
utilities
IDMSRPTS G-3
utility functions
ACCEPT 5-20—5-21, 5-30
ACCEPT page-info-location 5-29
KEEP LONGTERM 5-145—5-150
SEND MESSAGE 5-209—5-211
SNAP 5-218—5-219
WRITE LOG 5-240—5-245

V

VS-COBOL M-10
VSE JCL A-9

W

WAIT 5-232—5-233
WCC
See write control character (WCC)
WHERE clause (LRF) 5-258—5-262
expanded syntax 5-258—5-262
WORKING-STORAGE/LINKAGE SECTIONs 4-12—4-21
write control character (WCC) 5-178
WRITE JOURNAL 5-234—5-235
WRITE LINE TO TERMINAL 5-236—5-239
WRITE LOG 5-240—5-245
WRITE PRINTER 5-246—5-250
WRITE TERMINAL 5-251—5-253
WRITE THEN READ TERMINAL 5-254—5-257

X

XA storage M-10