

# CA-IDMS®

---

Utilities  
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

---

<b>How to use this Manual</b> . . . . .	xv
<b>Chapter 1. Overview</b> . . . . .	1-1
1.1 About this chapter . . . . .	1-3
1.2 Utility statements . . . . .	1-4
1.3 Utility programs . . . . .	1-5
1.4 Operating mode . . . . .	1-6
1.5 JCL considerations . . . . .	1-7
<b>Chapter 2. Utility Operations</b> . . . . .	2-1
2.1 About this chapter . . . . .	2-3
2.2 CA-IDMS/DB utilities . . . . .	2-4
2.2.1.1 Utility descriptions . . . . .	2-4
2.2.1.2 SQL and non-SQL . . . . .	2-8
<b>Chapter 3. Batch Command Facility Considerations</b> . . . . .	3-1
3.1 About this chapter . . . . .	3-3
3.2 Statement coding considerations . . . . .	3-4
<b>Chapter 4. Utility Statements</b> . . . . .	4-1
4.1 ARCHIVE JOURNAL . . . . .	4-7
4.1.1 Description . . . . .	4-7
4.1.2 Usage . . . . .	4-9
4.1.3 JCL considerations . . . . .	4-10
4.1.4 Example . . . . .	4-11
4.1.5 Sample output . . . . .	4-11
4.1.6 For more information . . . . .	4-11
4.2 ARCHIVE LOG . . . . .	4-12
4.2.1 Description . . . . .	4-12
4.2.2 Usage . . . . .	4-12
4.2.3 JCL considerations . . . . .	4-13
4.2.4 Example . . . . .	4-13
4.2.5 Sample output . . . . .	4-13
4.2.6 For more information . . . . .	4-14
4.3 BACKUP . . . . .	4-15
4.3.1 Description . . . . .	4-15
4.3.2 Usage . . . . .	4-16
4.3.3 JCL considerations . . . . .	4-17
4.3.4 Examples . . . . .	4-17
4.3.5 Sample output . . . . .	4-18
4.3.6 For more information . . . . .	4-19
4.4 BUILD . . . . .	4-20
4.4.1 Description . . . . .	4-20
4.4.2 Usage . . . . .	4-23
4.4.3 JCL considerations . . . . .	4-25
4.4.4 Examples . . . . .	4-25
4.4.5 Sample output . . . . .	4-26

4.4.6 For more information	4-26
4.5 CLEANUP	4-27
4.5.1 Description	4-27
4.5.2 Usage	4-28
4.5.3 JCL considerations	4-28
4.5.4 Example	4-28
4.5.5 Sample output	4-29
4.5.6 For more information	4-29
4.6 CONVERT PAGE	4-30
4.6.1 Description	4-30
4.6.2 Usage	4-31
4.6.3 JCL considerations	4-33
4.6.4 Examples	4-33
4.6.4.1 Example 1	4-33
4.6.4.2 Example 2	4-34
4.6.4.3 Example 3	4-34
4.6.4.4 Example 4	4-35
4.6.4.5 Bad Example 4	4-35
4.6.5 Sample output	4-36
4.6.6 For more information	4-37
4.7 EXPAND PAGE	4-38
4.7.1 Description	4-38
4.7.2 Usage	4-39
4.7.3 JCL considerations	4-39
4.7.4 Example	4-39
4.7.5 Sample output	4-40
4.7.6 For more information	4-40
4.8 EXTRACT JOURNAL	4-41
4.8.1 Description	4-41
4.8.2 Usage	4-44
4.8.3 JCL considerations	4-47
4.8.4 Example	4-47
4.8.5 Sample output	4-47
4.9 FASTLOAD	4-50
4.9.1 Description	4-50
4.9.2 Usage	4-51
4.9.3 JCL considerations	4-59
4.9.4 Example	4-61
4.9.5 Sample output	4-61
4.9.6 For more information	4-62
4.10 FIX ARCHIVE	4-63
4.10.1 Description	4-63
4.10.2 Usage	4-63
4.10.3 JCL considerations	4-64
4.10.4 Examples	4-64
4.10.5 Sample output	4-64
4.10.6 For more information	4-65
4.11 FIX PAGE	4-66
4.11.1 Description	4-66
4.11.2 Usage	4-67
4.11.3 JCL considerations	4-67

4.11.4	Examples	4-67
4.11.5	Sample output	4-68
4.11.6	For more information	4-69
4.12	FORMAT	4-70
4.12.1	Description	4-70
4.12.2	Usage	4-71
4.12.3	JCL considerations	4-73
4.12.4	Examples	4-73
4.12.5	Sample output	4-74
4.12.6	For more information	4-75
4.13	INSTALL STAMPS	4-76
4.13.1	Description	4-76
4.13.2	Usage	4-76
4.13.3	JCL considerations	4-77
4.13.4	Examples	4-77
4.13.5	Sample output	4-77
4.13.6	For more information	4-77
4.14	LOAD	4-78
4.14.1	Description	4-78
4.14.2	Usage	4-86
4.14.3	JCL considerations	4-87
4.14.4	Examples	4-88
4.14.5	Sample output	4-88
4.14.6	For more information	4-92
4.15	MAINTAIN INDEX	4-93
4.15.1	Description	4-93
4.15.2	Usage	4-96
4.15.3	JCL considerations	4-106
4.15.4	Examples	4-108
4.15.5	Sample output	4-109
4.15.6	For more information	4-109
4.16	MERGE ARCHIVE	4-110
4.16.1	Description	4-110
4.16.2	Usage	4-111
4.16.3	JCL considerations	4-112
4.16.4	Example	4-112
4.16.5	Sample output	4-112
4.16.6	For more information	4-113
4.17	PRINT INDEX	4-114
4.17.1	Description	4-114
4.17.2	Usage	4-117
4.17.3	JCL considerations	4-117
4.17.4	Examples	4-117
4.17.5	Sample output	4-118
4.17.6	For more information	4-120
4.18	PRINT JOURNAL	4-121
4.18.1	Description	4-121
4.18.2	Usage	4-123
4.18.3	JCL considerations	4-123
4.18.4	Example	4-123

4.18.5	Sample output	4-123
4.18.6	For more information	4-124
4.19	PRINT LOG	4-125
4.19.1	Description	4-125
4.19.2	Usage	4-128
4.19.3	JCL considerations	4-128
4.19.4	Examples	4-128
4.19.5	Sample output	4-129
4.19.6	For more information	4-130
4.20	PRINT PAGE	4-131
4.20.1	Description	4-131
4.20.2	Usage	4-134
4.20.3	JCL considerations	4-134
4.20.4	Example	4-135
4.20.5	Sample output	4-135
4.20.6	For more information	4-136
4.21	PRINT SPACE	4-137
4.21.1	Description	4-137
4.21.2	Usage	4-138
4.21.3	JCL considerations	4-139
4.21.4	Examples	4-139
4.21.5	Sample output	4-139
4.21.6	For more information	4-141
4.22	PUNCH	4-142
4.22.1	Description	4-142
4.22.2	Usage	4-142
4.22.3	JCL considerations	4-143
4.22.4	Example	4-143
4.22.5	Output	4-143
4.22.6	For more information	4-143
4.23	RELOAD	4-144
4.23.1	Description	4-144
4.23.2	Usage	4-146
4.23.3	JCL considerations	4-154
4.23.4	Example	4-155
4.23.5	Sample output	4-156
4.23.6	For more information	4-157
4.24	RESTORE	4-158
4.24.1	Description	4-158
4.24.2	Usage	4-159
4.24.3	JCL considerations	4-159
4.24.4	Examples	4-160
4.24.5	Sample output	4-160
4.24.6	For more information	4-161
4.25	RESTRUCTURE CONNECT	4-162
4.25.1	Description	4-162
4.25.2	Usage	4-163
4.25.3	JCL considerations	4-164
4.25.4	Examples and sample output	4-164
4.25.5	For more information	4-165
4.26	RESTRUCTURE SEGMENT	4-166

4.26.1	Description	4-166
4.26.2	Usage	4-167
4.26.3	JCL considerations	4-170
4.26.4	Examples and sample output	4-170
4.26.5	For more information	4-171
4.26.6	Callable Restructure Utility	4-171
4.26.6.1	Considerations	4-171
4.26.6.2	Parameters	4-172
4.27	ROLLBACK	4-173
4.27.1	Description	4-173
4.27.2	Usage	4-176
4.27.3	JCL considerations	4-178
4.27.4	Examples	4-179
4.27.5	Sample output	4-179
4.27.6	For more information	4-179
4.28	ROLLFORWARD	4-180
4.28.1	Description	4-180
4.28.2	Usage	4-184
4.28.3	JCL considerations	4-187
4.28.4	Examples	4-188
4.28.5	Sample output	4-188
4.28.6	For more information	4-188
4.29	TUNE INDEX	4-189
4.29.1	Description	4-189
4.29.2	Usage	4-190
4.29.3	JCL considerations	4-191
4.29.4	Example	4-191
4.29.5	Sample output	4-191
4.29.6	For more information	4-192
4.30	UNLOAD	4-193
4.30.1	Description	4-193
4.30.2	Usage	4-195
4.30.2.1	How UNLOAD works	4-196
4.30.2.2	The unload/reload subschemas	4-197
4.30.2.3	The unload/reload SEGMENT	4-198
4.30.2.4	The unload/reload DMCL	4-198
4.30.2.5	General procedure for UNLOAD and RELOAD	4-199
4.30.3	JCL considerations	4-200
4.30.4	Example	4-200
4.30.5	Sample output	4-201
4.30.6	For more information	4-201
4.31	UNLOCK	4-202
4.31.1	Description	4-202
4.31.2	Usage	4-202
4.31.3	JCL considerations	4-202
4.31.4	Examples	4-203
4.31.5	Sample output	4-203
4.31.6	For more information	4-203
4.32	UPDATE STATISTICS	4-204
4.32.1	Description	4-204

4.32.2	Usage	4-205
4.32.3	JCL considerations	4-206
4.32.4	Example	4-206
4.32.5	Sample output	4-207
4.32.6	For more information	4-207
4.33	VALIDATE	4-208
4.33.1	Description	4-208
4.33.2	Usage	4-210
4.33.3	JCL considerations	4-211
4.33.4	Example	4-212
4.33.5	Sample output	4-212
4.33.6	For more information	4-212
<b>Chapter 5. Utility Programs</b>		5-1
5.1	IDMSCALC	5-3
5.1.1	Description	5-3
5.1.2	Usage	5-3
5.1.3	Calling the IDMSCALC routine	5-3
5.1.4	For more information	5-4
5.2	IDMSDBAN	5-5
5.2.1	Description	5-5
5.2.2	Syntax	5-5
5.2.3	Input parameter statements	5-6
5.2.4	Usage	5-11
5.2.5	JCL Considerations	5-15
5.2.6	Example	5-16
5.2.7	Sample output	5-16
5.2.8	For more information	5-21
5.3	IDMSDIRL	5-22
5.3.1	Description	5-22
5.3.2	Syntax	5-23
5.3.3	Input parameter statements	5-23
5.3.4	Usage	5-24
5.3.5	JCL considerations	5-25
5.3.6	Examples	5-25
5.3.7	Sample output	5-25
5.3.8	For more information	5-25
5.4	IDMSLOOK	5-27
5.4.1	Description	5-27
5.4.2	Syntax	5-27
5.4.3	Input parameter statements	5-28
5.4.4	Usage	5-35
5.4.5	JCL considerations	5-36
5.4.6	Examples	5-36
5.4.7	Sample output	5-36
5.4.8	For more information	5-47
5.5	IDMSRPTS	5-48
5.5.1	Description	5-48
5.5.2	Syntax	5-51
5.5.3	Statements	5-52
5.5.4	Usage	5-60

5.5.5	JCL considerations	5-61
5.5.6	Examples	5-61
5.5.7	Sample output	5-62
5.5.8	For more information	5-68
5.6	IDMSRSTC	5-69
5.6.1	Description	5-69
5.6.2	Syntax	5-69
5.6.3	Input parameter statements	5-70
5.6.4	Usage	5-72
5.6.5	Example	5-74
5.6.6	Sample output	5-74
5.6.7	For more information	5-74
<b>Chapter 6.</b>	<b>OS/390 JCL</b>	<b>6-1</b>
6.1	About this chapter	6-3
6.2	CA-IDMS Batch Command Facility	6-4
6.3	Utility statements	6-6
6.3.1	ARCHIVE JOURNAL	6-6
6.3.2	ARCHIVE LOG	6-6
6.3.3	BACKUP	6-7
6.3.4	BUILD	6-7
6.3.5	CLEANUP	6-8
6.3.6	CONVERT PAGE	6-9
6.3.7	EXPAND PAGE	6-9
6.3.8	EXTRACT JOURNAL	6-10
6.3.9	FASTLOAD	6-10
6.3.10	FIX ARCHIVE	6-12
6.3.11	FIX PAGE	6-12
6.3.12	FORMAT	6-13
6.3.13	INSTALL STAMPS	6-13
6.3.14	LOAD	6-14
6.3.15	MAINTAIN INDEX	6-15
6.3.16	MERGE ARCHIVE	6-16
6.3.17	PRINT INDEX	6-17
6.3.18	PRINT JOURNAL	6-17
6.3.19	PRINT LOG	6-18
6.3.20	PRINT PAGE	6-18
6.3.21	PRINT SPACE	6-18
6.3.22	PUNCH	6-19
6.3.23	RELOAD	6-20
6.3.24	RESTORE	6-23
6.3.25	RESTRUCTURE CONNECT	6-23
6.3.26	RESTRUCTURE SEGMENT	6-24
6.3.27	ROLLBACK	6-24
6.3.28	ROLLFORWARD	6-25
6.3.29	TUNE INDEX	6-25
6.3.30	UNLOCK	6-25
6.3.31	UNLOAD	6-26
6.3.32	UPDATE STATISTICS	6-27
6.3.33	VALIDATE	6-27

6.4	Utility programs	6-29
6.4.1	IDMSDBAN	6-29
6.4.2	IDMSDIRL	6-30
6.4.3	IDMSLOOK	6-31
6.4.4	IDMSRPTS	6-31
6.4.5	IDMSRSTC	6-32
6.4.6	IDMSRSTT	6-34
<b>Chapter 7.</b>	<b>VSE/ESA JCL</b>	7-1
7.1	About this chapter	7-3
7.2	=COPY Facility	7-4
7.3	IDMSLBLBS Procedure	7-5
7.4	SYSIDMS parameter file	7-6
7.5	CA-IDMS Batch Command Facility	7-7
7.6	Utility statements	7-9
7.6.1	ARCHIVE JOURNAL	7-9
7.6.2	ARCHIVE LOG	7-9
7.6.3	BACKUP	7-9
7.6.4	BUILD	7-10
7.6.5	CLEANUP	7-10
7.6.6	CONVERT PAGE	7-11
7.6.7	EXPAND PAGE	7-11
7.6.8	EXTRACT JOURNAL	7-11
7.6.9	FASTLOAD	7-12
7.6.10	FIX ARCHIVE	7-14
7.6.11	FIX PAGE	7-14
7.6.12	FORMAT	7-14
7.6.13	INSTALL STAMPS	7-16
7.6.14	LOAD	7-16
7.6.15	MAINTAIN INDEX	7-17
7.6.16	MERGE ARCHIVE	7-19
7.6.17	PRINT INDEX	7-19
7.6.18	PRINT JOURNAL	7-19
7.6.19	PRINT LOG	7-20
7.6.20	PRINT PAGE	7-20
7.6.21	PRINT SPACE	7-20
7.6.22	PUNCH	7-21
7.6.23	RELOAD	7-21
7.6.24	RESTORE	7-23
7.6.25	RESTRUCTURE CONNECT	7-24
7.6.26	RESTRUCTURE SEGMENT	7-24
7.6.27	ROLLBACK	7-25
7.6.28	ROLLFORWARD	7-25
7.6.29	TUNE INDEX	7-26
7.6.30	UNLOCK	7-26
7.6.31	UNLOAD	7-27
7.6.32	UPDATE STATISTICS	7-27
7.6.33	VALIDATE	7-28
7.7	Utility programs	7-29
7.7.1	IDMSDBAN	7-29
7.7.2	IDMSDIRL	7-30

7.7.3	IDMSLOOK	7-32
7.7.4	IDMSRPTS	7-33
7.7.5	IDMSRSTC	7-34
7.7.6	IDMSRSTT	7-36
7.8	IDMSLBLE Procedure	7-37
<b>Chapter 8.</b>	<b>CMS Commands</b>	8-1
8.1	About this chapter	8-3
8.2	CA-IDMS Batch Command Facility	8-4
8.3	Utility statements	8-6
8.3.1	ARCHIVE JOURNAL	8-6
8.3.2	ARCHIVE LOG	8-6
8.3.3	BACKUP	8-7
8.3.4	BUILD	8-7
8.3.5	CLEANUP	8-8
8.3.6	CONVERT PAGE	8-8
8.3.7	EXPAND PAGE	8-9
8.3.8	EXTRACT JOURNAL	8-9
8.3.9	FASTLOAD	8-10
8.3.10	FIX ARCHIVE	8-11
8.3.11	FIX PAGE	8-11
8.3.12	FORMAT	8-12
8.3.13	INSTALL STAMPS	8-12
8.3.14	LOAD	8-13
8.3.15	MAINTAIN INDEX	8-14
8.3.16	MERGE ARCHIVE	8-15
8.3.17	PRINT INDEX	8-15
8.3.18	PRINT JOURNAL	8-16
8.3.19	PRINT LOG	8-16
8.3.20	PRINT PAGE	8-16
8.3.21	PRINT SPACE	8-17
8.3.22	PUNCH	8-17
8.3.23	RELOAD	8-18
8.3.24	RESTORE	8-21
8.3.25	RESTRUCTURE CONNECT	8-21
8.3.26	RESTRUCTURE SEGMENT	8-22
8.3.27	ROLLBACK	8-22
8.3.28	ROLLFORWARD	8-23
8.3.29	TUNE INDEX	8-23
8.3.30	UNLOCK	8-23
8.3.31	UNLOAD	8-24
8.3.32	UPDATE STATISTICS	8-24
8.3.33	VALIDATE	8-25
8.4	Utility programs	8-27
8.4.1	IDMSDBAN	8-27
8.4.2	IDMSDIRL	8-28
8.4.3	IDMSLOOK	8-29
8.4.4	IDMSRPTS	8-30
8.4.5	IDMSRSTC	8-31
8.4.6	IDMSRSTT	8-32

<b>Chapter 9. BS2000/OSD JCL</b>	9-1
9.1 About this chapter	9-3
9.2 General JCL considerations	9-4
9.3 =COPY Facility	9-5
9.4 CA-IDMS Batch Command Facility	9-6
9.5 Utility statements	9-8
9.5.1 ARCHIVE JOURNAL	9-8
9.5.2 ARCHIVE LOG	9-8
9.5.3 BACKUP	9-9
9.5.4 BUILD	9-9
9.5.5 CLEANUP	9-10
9.5.6 CONVERT PAGE	9-10
9.5.7 EXPAND PAGE	9-10
9.5.8 EXTRACT JOURNAL	9-11
9.5.9 FASTLOAD	9-11
9.5.10 FIX ARCHIVE	9-13
9.5.11 FIX PAGE	9-14
9.5.12 FORMAT	9-14
9.5.13 INSTALL STAMPS	9-15
9.5.14 LOAD	9-16
9.5.15 MAINTAIN INDEX	9-17
9.5.16 MERGE ARCHIVE	9-18
9.5.17 PRINT INDEX	9-18
9.5.18 PRINT JOURNAL	9-19
9.5.19 PRINT LOG	9-19
9.5.20 PRINT PAGE	9-19
9.5.21 PRINT SPACE	9-20
9.5.22 PUNCH	9-20
9.5.23 RELOAD	9-21
9.5.24 RESTORE	9-23
9.5.25 RESTRUCTURE CONNECT	9-24
9.5.26 RESTRUCTURE SEGMENT	9-24
9.5.27 ROLLBACK	9-25
9.5.28 ROLLFORWARD	9-25
9.5.29 TUNE INDEX	9-27
9.5.30 UNLOCK	9-27
9.5.31 UNLOAD	9-27
9.5.32 UPDATE STATISTICS	9-28
9.5.33 VALIDATE	9-29
9.6 Utility programs	9-30
9.6.1 IDMSDBAN	9-30
9.6.2 IDMSDIRL	9-31
9.6.3 IDMSLOOK	9-32
9.6.4 IDMSRPTS	9-33
9.6.5 IDMSRSTC	9-34
9.6.6 IDMSRSTT	9-35
<b>Appendix A. Format Program for FASTLOAD</b>	A-1
A.1 About this appendix	A-3
<b>Appendix B. IDMSRSTT Macro Statements</b>	B-1

B.1 Overview	B-3
B.2 IDMSRSTT BUFSIZE	B-5
B.3 IDMSRSTT RECNAME	B-6
B.4 IDMSRSTT SETPTR	B-9
B.5 IDMSRSTT FIELD	B-11
B.6 IDMSRSTT END	B-14
B.7 END	B-15
<b>Index</b>	<b>X-1</b>



# How to use this Manual

---

## What this manual is about

This manual describes reference material for using utility statements and utility programs with CA-IDMS. For each statement and program, the manual provides:

- Syntax diagram
- Parameter explanations
- Usage notes
- CA-IDMS Batch Command Facility (IDMSBCF) considerations
- JCL considerations
- Examples
- Sample output for each statement and program

Additionally, appendixes contain generic JCL for submitting utility statements through the batch command facility, and for running the utility programs. The required CA-IDMS files are listed with each utility statement and program.

## Who should use this manual

This manual is intended for:

- **Database administrators (DBAs)** responsible for maintaining CA-IDMS/DB databases
- **Data communications administrators (DCAs)** responsible for maintaining CA-IDMS/DC and CA-IDMS/UCF (DC/UCF) systems

# What this manual contains

This manual contains the following:

- **Chapters 1 through 3** introduce the utility statements and programs, explain how to submit the statements and programs to CA-IDMS/DB, and present coding considerations.
- **Chapter 4** presents syntax, usage information, JCL considerations, and examples for each utility statement. The utility statements are in alphabetical order by statement name.
- **Chapter 5** presents syntax, usage information, JCL considerations, and examples for each utility program. The utility programs are in alphabetical order by program name.
- **Chapters 6 through 9** presents generic JCL for the batch command facility and sample operating system-specific JCL.

## Related CA-IDMS documentation

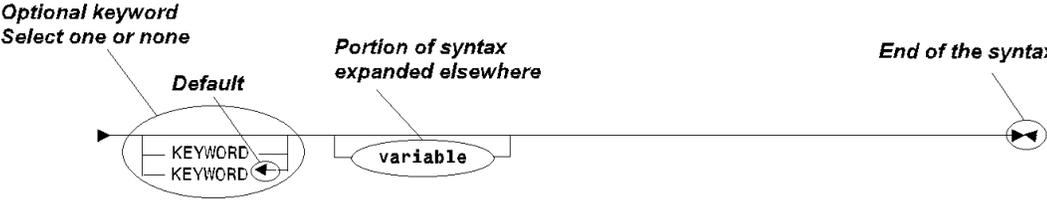
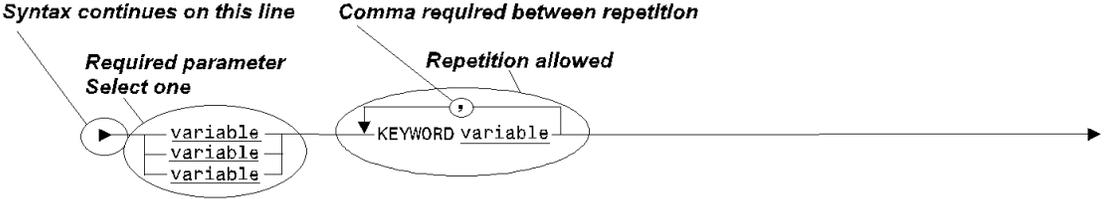
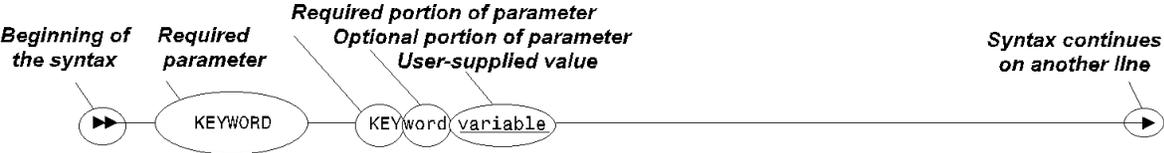
- **For additional information on using utility statements**, refer to *CA-IDMS Database Administration* and *CA-IDMS System Operations*.
- **For information on using the batch command facility**, refer to *CA-IDMS Command Facility*.

# 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.
<u>underlined lowercase</u>	Represents a value that you supply.
←	Points to the default in a list of choices.
<b>lowercase bold</b>	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 , parameter ▶	Shows that you must enter a comma between repetitions of the parameter.

# Sample Syntax Diagram





# Chapter 1. Overview

---

- 1.1 About this chapter . . . . . 1-3
- 1.2 Utility statements . . . . . 1-4
- 1.3 Utility programs . . . . . 1-5
- 1.4 Operating mode . . . . . 1-6
- 1.5 JCL considerations . . . . . 1-7



---

## 1.1 About this chapter

The CA-IDMS utilities assist the database administrator in performing database maintenance and backup and recovery functions.

There are two ways to execute utilities:

- You execute most utilities by submitting utility statements through the CA-IDMS Command Facility (either in batch through program IDMSBCF or online using the task code OCF).
- The remaining utilities are executed as separate programs.

This chapter presents a list of utility statements and programs.

## 1.2 Utility statements

You submit the following utilities as statements to the CA-IDMS Batch Command Facility (IDMSBCF). These statements can only be submitted to the batch command facility except the INSTALL STAMPS, TUNE INDEX, and UPDATE STATISTICS utility statements which can also be submitted to the online command facility (OCF).

ARCHIVE JOURNAL  
ARCHIVE LOG  
BACKUP  
BUILD  
CLEANUP  
CONVERT PAGE  
EXPAND PAGE  
EXTRACT JOURNAL  
FASTLOAD  
FIX ARCHIVE  
FIX PAGE  
FORMAT  
INSTALL STAMPS  
LOAD  
MAINTAIN INDEX  
MERGE ARCHIVE  
PRINT INDEX  
PRINT JOURNAL  
PRINT LOG  
PRINT PAGE  
PRINT SPACE  
PUNCH  
RELOAD  
RESTORE  
RESTRUCTURE CONNECT  
RESTRUCTURE SEGMENT  
ROLLBACK  
ROLLFORWARD  
TUNE INDEX  
UNLOAD  
UNLOCK  
UPDATE STATISTICS  
VALIDATE

## 1.3 Utility programs

You submit the following utilities to CA-IDMS/DB as separate programs.

IDMSCALC (called as a subroutine from user programs)  
IDMSDBAN  
IDMSDIRL  
IDMSLOOK  
IDMSRPTS  
IDMSRSTC

## 1.4 Operating mode

With few exceptions, the utilities should be run while operating in local mode.

The following utilities can be run while operating in local mode or under the central version:

- INSTALL STAMPS
- TUNE INDEX
- UPDATE STATISTICS
- IDMSDIRL
- IDMSRPTS
- IDMSRSTC

---

## 1.5 JCL considerations

In the discussion of each utility, the files required to run a utility are identified under the heading "JCL considerations".

Additionally, Chapters 6 through 9 present sample JCL for each utility by operating system.

**Batch command facility JCL:** In each JCL chapter, the basic JCL to execute the CA-IDMS batch command facility (IDMSBCF) is presented first. IDMSBCF JCL must include definitions of the input and output files CA-IDMS/DB needs to perform the requested operations. For each operating system, sample file assignments for these input and output files are presented in alphabetical order by utility.

►►For more detailed information on using the CA-IDMS Batch Command Facility, refer to *CA-IDMS Command Facility*.

**Operating system-specific JCL:** Sample JCL for submitting utility statements and programs is presented in the following chapters:

- Chapter 6, "OS/390 JCL"
- Chapter 7, "VSE/ESA JCL"
- Chapter 8, "CMS Commands"
- Chapter 9, "BS2000/OSD JCL"

**SYSIDMS parameter file** The SYSIDMS parameter file is added to the JCL stream of batch jobs running in local mode or under the central version. You can use SYSIDMS parameters to specify:

- Physical requirements of the environment, such as the DMCL and the database or dictionary to use at runtime
- Runtime directives that assist in application execution such as, activating the IDMSQSAM facility
- Operating system-dependent file information, such as overriding a block size for a file in a VSE/ESA environment

When executing utility statements through the batch command facility, you use SYSIDMS parameters, as appropriate, to specify database name, dictionary name, DMCL name and other required information.

You should be familiar with SYSIDMS parameters and the function of the SYSIDMS parameter file before running CA-IDMS utilities.

For a complete discussion of SYSIDMS parameters, see *CA-IDMS Database Administration*.

**IDMSLBS procedure for VSE/ESA JCL:** A procedure containing file assignments for CA-IDMS dictionary and database files, disk journals, and the SYSIDMS parameter file for use in the CA-IDMS/DB VSE/ESA environment is provided during the CA-IDMS installation. A copy of this procedure appears in Chapter 7, “VSE/ESA JCL.”

Only the file definitions for work files, SYSCTL files, and tape and archive journal files appear in the sample JCL in Chapter 7, “VSE/ESA JCL.”

# Chapter 2. Utility Operations

---

- 2.1 About this chapter . . . . . 2-3
- 2.2 CA-IDMS/DB utilities . . . . . 2-4
  - 2.2.1.1 Utility descriptions . . . . . 2-4
  - 2.2.1.2 SQL and non-SQL . . . . . 2-8



---

## 2.1 About this chapter

This chapter presents utility statements and programs by database and system operation functions.

## 2.2 CA-IDMS/DB utilities

You submit utility statements and programs to CA-IDMS/DB to request the following types of operations:

- Backup and recovery
- CA-IDMS system log maintenance
- Database area maintenance
- Database loading and restructuring
- Database integrity checking
- Database reporting
- Enhancing SQL data access
- Load module management

The utility statements and programs are presented below by database and system operation functions.

### 2.2.1.1 Utility descriptions

**Backup and recovery utilities:** CA-IDMS/DB provides the following utilities for backup and recovery operations:

---

<b>Utility</b>	<b>Purpose</b>
ARCHIVE JOURNAL	Offload disk journal files to archive files
BACKUP	Back up database areas
EXTRACT JOURNAL	Extracts AFTR images from archived journal file and writes them to an extract file; the extract file can be used as input to the ROLLFORWARD utility
FIX ARCHIVE	Rewrite a tape journal file
FIX PAGE	Verify and/or modify the contents of a database page
MERGE ARCHIVE	Merge archived journal files of data sharing group members. The output file can be used as input to the ROLLFORWARD, ROLLBACK, EXTRACT JOURNAL, and MERGE ARCHIVE utility statements.
PRINT JOURNAL	Report on transaction activity
RESTORE	Restore backed up database areas
ROLLBACK	Restore files or areas to earlier states using journal information
ROLLFORWARD	Update a backup copy of a file or area using journal information
UNLOCK	Remove locks from an area

►► For more information on backup and recovery operations, see the sections on the individual utilities in this document or refer to *CA-IDMS Database Administration*.

**Log maintenance utilities:** CA-IDMS/DB provides the following utilities for maintaining the DC/UCF system log:

---

<b>Utility</b>	<b>Purpose</b>
ARCHIVE LOG	Offload system log to archive file
PRINT LOG	Print all or part of a system log or archive log

►► For more information on maintaining the system log, see the sections on the individual utilities in this document or refer to *CA-IDMS System Operations*.

**Area maintenance utilities:** CA-IDMS/DB provides the following utilities for database and journal file maintenance:

<b>Utility</b>	<b>Purpose</b>
CLEANUP	Erase logically deleted records
EXPAND PAGE	Increase page size for a database file
FIX PAGE	Verify and/or modify the contents of a database page
FORMAT	Prepare a file, area or segment for use by CA-IDMS/DB
INSTALL STAMPS	Store synchronization stamps for an SQL-defined database
PRINT PAGE	Print the contents of database pages
PRINT SPACE	Report on space utilization in areas
TUNE INDEX	Walk a sorted index in order to cause the adoption of orphaned index records
UNLOCK	Remove locks from an area
UPDATE STATISTICS	Update statistics used by CA-IDMS/DB to optimize access to an SQL-defined database

►► For more information on database area maintenance, see the chapters on the individual utilities in this document or refer to *CA-IDMS Database Administration*.

**Database loading and restructuring utilities:** CA-IDMS/DB provides the following utilities for loading and restructuring a database:

<b>Utility</b>	<b>Purpose</b>
BUILD	Build or rebuild indexes and build referential constraints for an SQL-defined database
CONVERT PAGE	Change the page range for an area or the maximum number of records that can be stored on a page of an area
FASTLOAD	Load data into a non-SQL defined database for the first time
IDMSDIRL	Load the IDMSNTWK version 1 schema and the IDMSNWKA subschema into a data dictionary
IDMSRSTC	Generate IDMSRSTT macro statements for restructuring a non-SQL defined database
LOAD	Load data into an SQL-defined database
MAINTAIN INDEX	Build, rebuild, or delete indexes in a non-SQL defined database
RELOAD	Reload a database unloaded by UNLOAD
RESTRUCTURE CONNECT	Connect new prior and owner pointers in existing sets in a non-SQL defined database
RESTRUCTURE SEGMENT	Modify record occurrences to match new schema specifications
UNLOAD	Unload all or part of a database
VALIDATE	Check referential constraints for an SQL-defined database

►► For more information on loading data into the database, see the chapters on the individual utilities in this document or refer to *CA-IDMS Database Administration*.

**Integrity checking utilities:** CA-IDMS/DB provides the following utilities for checking database integrity:

<b>Utility</b>	<b>Purpose</b>
IDMSDBAN	Analyze the structure of an existing non-SQL defined database
PRINT INDEX	Report on system owned indexes and indexed sets
VALIDATE	Check referential constraints for an SQL-defined database

►► For more information on checking database integrity, see the chapters on the individual utilities in this document or refer to *CA-IDMS Database Administration*.

**Database reporting utilities:** CA-IDMS/DB provides the following utilities for reporting on database structure and contents:

IDMSDBAN	Analyze the structure of an existing non-SQL defined database
IDMSLOOK	Report on the contents of load modules
IDMSRPTS	Report on information stored in a data dictionary
PRINT INDEX	Report on system owned indexes and indexed sets
PRINT JOURNAL	Report on transaction activity
PRINT PAGE	Print the contents of database pages
PRINT SPACE	Report on space utilization in areas

### 2.2.1.2 SQL and non-SQL

Some utilities can only be used for SQL or non-SQL databases. Others can be used without regard for the type of database you are using.

#### SQL database only:

BUILD  
INSTALL STAMPS  
LOAD  
VALIDATE

#### Non-SQL database only:

CLEANUP  
FASTLOAD  
MAINTAIN INDEX  
RESTRUCTURE CONNECT  
RESTRUCTURE SEGMENT  
IDMSRSTC

#### For both SQL and non-SQL databases:

ARCHIVE JOURNAL  
ARCHIVE LOG  
BACKUP  
CONVERT PAGE  
EXPAND PAGE  
EXTRACT JOURNAL  
FIX ARCHIVE  
FIX PAGE

FORMAT  
MERGE ARCHIVE  
PRINT INDEX  
PRINT JOURNAL  
PRINT LOG  
PRINT PAGE  
PRINT SPACE  
PUNCH  
RELOAD  
RESTORE  
ROLLBACK  
ROLLFORWARD  
TUNE INDEX  
UNLOAD  
UNLOCK  
UPDATE STATISTICS  
IDMSDBAN  
IDMSCALC  
IDMSDIRL  
IDMSLOOK  
IDMSRPTS



# Chapter 3. Batch Command Facility Considerations

---

- 3.1 About this chapter . . . . . 3-3
- 3.2 Statement coding considerations . . . . . 3-4



---

## 3.1 About this chapter

This chapter presents general coding considerations when using the CA-IDMS Batch Command Facility (IDMSBCF) to execute utility statements.

For a complete description of the Batch Command Facility, see *CA-IDMS Command Facility*.

## 3.2 Statement coding considerations

**Statement components:** Utility statements consist of:

- **Keywords** that:
  - Identify the action requested by the statement (for example, BACKUP or PRINT SPACE)
  - Specify the type of entity (for example, AREA or SEGMENT) that is the object of the requested action
  - Place qualifications on the requested action, either by themselves (for example, SHARE or NO REPORT) or in conjunction with user-supplied values (for example, START AT 1999-12-08-06.00.00)
- **User-supplied values** that:
  - Identify specific occurrences of entities (for example, the area EMP\_SPACE or the database segment DEMOSEG)
  - Specify data values (for example, 983 or 'Boston')
- **Separators** that separate keywords and user-supplied values from one another. A separator can be a space, a comment, or the end of a line.

Separators are *not* required:

- Before or after a value in single quotation marks
- A comma (,)
- An equal sign (=)
- Left and right parentheses ()
- A period (.)
- A semicolon (;)

**Delimiting statements:** When you use the batch command facility to submit utility statements, you must terminate each statement with a semicolon (;). You can enter the semicolon either on the same line as the rest of the statement or on a separate line. For example, the two statements shown below are equivalent:

```
format area emp-region-area;  
  
format area emp-region-area  
;
```

**Continuing statements:** You can code utility statements on one or more lines. No special character is required to indicate that a statement continues on the next line.

**Quotation marks around identifiers:** In a utility statement, you must enclose a database entity identifier in double quotation marks if the identifier includes significant lowercase characters, special characters (except a dash (-)), or blanks. Place the quotation marks only around the individual identifier for which they are required (for example, SALESSEG."EST%\_SPACE").

Note that all identifiers can contain dashes even when not quoted. For example, SALES-SEG."EST%\_SPACE".

►► For more information on the use of quotation marks with identifiers, refer to *CA-IDMS SQL Reference*.

For more information on database entity identifiers, see *CA-IDMS Database Administration*.

**Maximum statement length:** A utility statement can be at most 8,192 bytes long. If you are using only single-byte characters, the maximum number of characters equals the maximum number of bytes. If any user-supplied values contain double-byte characters, the maximum number of characters is less than the maximum number of bytes.



# Chapter 4. Utility Statements

---

4.1 ARCHIVE JOURNAL	4-7
4.1.1 Description	4-7
4.1.2 Usage	4-9
4.1.3 JCL considerations	4-10
4.1.4 Example	4-11
4.1.5 Sample output	4-11
4.1.6 For more information	4-11
4.2 ARCHIVE LOG	4-12
4.2.1 Description	4-12
4.2.2 Usage	4-12
4.2.3 JCL considerations	4-13
4.2.4 Example	4-13
4.2.5 Sample output	4-13
4.2.6 For more information	4-14
4.3 BACKUP	4-15
4.3.1 Description	4-15
4.3.2 Usage	4-16
4.3.3 JCL considerations	4-17
4.3.4 Examples	4-17
4.3.5 Sample output	4-18
4.3.6 For more information	4-19
4.4 BUILD	4-20
4.4.1 Description	4-20
4.4.2 Usage	4-23
4.4.3 JCL considerations	4-25
4.4.4 Examples	4-25
4.4.5 Sample output	4-26
4.4.6 For more information	4-26
4.5 CLEANUP	4-27
4.5.1 Description	4-27
4.5.2 Usage	4-28
4.5.3 JCL considerations	4-28
4.5.4 Example	4-28
4.5.5 Sample output	4-29
4.5.6 For more information	4-29
4.6 CONVERT PAGE	4-30
4.6.1 Description	4-30
4.6.2 Usage	4-31
4.6.3 JCL considerations	4-33
4.6.4 Examples	4-33
4.6.4.1 Example 1	4-33
4.6.4.2 Example 2	4-34
4.6.4.3 Example 3	4-34
4.6.4.4 Example 4	4-35
4.6.4.5 Bad Example 4	4-35
4.6.5 Sample output	4-36
4.6.6 For more information	4-37

---

4.7 EXPAND PAGE	4-38
4.7.1 Description	4-38
4.7.2 Usage	4-39
4.7.3 JCL considerations	4-39
4.7.4 Example	4-39
4.7.5 Sample output	4-40
4.7.6 For more information	4-40
4.8 EXTRACT JOURNAL	4-41
4.8.1 Description	4-41
4.8.2 Usage	4-44
4.8.3 JCL considerations	4-47
4.8.4 Example	4-47
4.8.5 Sample output	4-47
4.9 FASTLOAD	4-50
4.9.1 Description	4-50
4.9.2 Usage	4-51
4.9.3 JCL considerations	4-59
4.9.4 Example	4-61
4.9.5 Sample output	4-61
4.9.6 For more information	4-62
4.10 FIX ARCHIVE	4-63
4.10.1 Description	4-63
4.10.2 Usage	4-63
4.10.3 JCL considerations	4-64
4.10.4 Examples	4-64
4.10.5 Sample output	4-64
4.10.6 For more information	4-65
4.11 FIX PAGE	4-66
4.11.1 Description	4-66
4.11.2 Usage	4-67
4.11.3 JCL considerations	4-67
4.11.4 Examples	4-67
4.11.5 Sample output	4-68
4.11.6 For more information	4-69
4.12 FORMAT	4-70
4.12.1 Description	4-70
4.12.2 Usage	4-71
4.12.3 JCL considerations	4-73
4.12.4 Examples	4-73
4.12.5 Sample output	4-74
4.12.6 For more information	4-75
4.13 INSTALL STAMPS	4-76
4.13.1 Description	4-76
4.13.2 Usage	4-76
4.13.3 JCL considerations	4-77
4.13.4 Examples	4-77
4.13.5 Sample output	4-77
4.13.6 For more information	4-77
4.14 LOAD	4-78

---

4.14.1	Description	4-78
4.14.2	Usage	4-86
4.14.3	JCL considerations	4-87
4.14.4	Examples	4-88
4.14.5	Sample output	4-88
4.14.6	For more information	4-92
4.15	MAINTAIN INDEX	4-93
4.15.1	Description	4-93
4.15.2	Usage	4-96
4.15.3	JCL considerations	4-106
4.15.4	Examples	4-108
4.15.5	Sample output	4-109
4.15.6	For more information	4-109
4.16	MERGE ARCHIVE	4-110
4.16.1	Description	4-110
4.16.2	Usage	4-111
4.16.3	JCL considerations	4-112
4.16.4	Example	4-112
4.16.5	Sample output	4-112
4.16.6	For more information	4-113
4.17	PRINT INDEX	4-114
4.17.1	Description	4-114
4.17.2	Usage	4-117
4.17.3	JCL considerations	4-117
4.17.4	Examples	4-117
4.17.5	Sample output	4-118
4.17.6	For more information	4-120
4.18	PRINT JOURNAL	4-121
4.18.1	Description	4-121
4.18.2	Usage	4-123
4.18.3	JCL considerations	4-123
4.18.4	Example	4-123
4.18.5	Sample output	4-123
4.18.6	For more information	4-124
4.19	PRINT LOG	4-125
4.19.1	Description	4-125
4.19.2	Usage	4-128
4.19.3	JCL considerations	4-128
4.19.4	Examples	4-128
4.19.5	Sample output	4-129
4.19.6	For more information	4-130
4.20	PRINT PAGE	4-131
4.20.1	Description	4-131
4.20.2	Usage	4-134
4.20.3	JCL considerations	4-134
4.20.4	Example	4-135
4.20.5	Sample output	4-135
4.20.6	For more information	4-136
4.21	PRINT SPACE	4-137

---

4.21.1	Description	4-137
4.21.2	Usage	4-138
4.21.3	JCL considerations	4-139
4.21.4	Examples	4-139
4.21.5	Sample output	4-139
4.21.6	For more information	4-141
4.22	PUNCH	4-142
4.22.1	Description	4-142
4.22.2	Usage	4-142
4.22.3	JCL considerations	4-143
4.22.4	Example	4-143
4.22.5	Output	4-143
4.22.6	For more information	4-143
4.23	RELOAD	4-144
4.23.1	Description	4-144
4.23.2	Usage	4-146
4.23.3	JCL considerations	4-154
4.23.4	Example	4-155
4.23.5	Sample output	4-156
4.23.6	For more information	4-157
4.24	RESTORE	4-158
4.24.1	Description	4-158
4.24.2	Usage	4-159
4.24.3	JCL considerations	4-159
4.24.4	Examples	4-160
4.24.5	Sample output	4-160
4.24.6	For more information	4-161
4.25	RESTRUCTURE CONNECT	4-162
4.25.1	Description	4-162
4.25.2	Usage	4-163
4.25.3	JCL considerations	4-164
4.25.4	Examples and sample output	4-164
4.25.5	For more information	4-165
4.26	RESTRUCTURE SEGMENT	4-166
4.26.1	Description	4-166
4.26.2	Usage	4-167
4.26.3	JCL considerations	4-170
4.26.4	Examples and sample output	4-170
4.26.5	For more information	4-171
4.26.6	Callable Restructure Utility	4-171
4.26.6.1	Considerations	4-171
4.26.6.2	Parameters	4-172
4.27	ROLLBACK	4-173
4.27.1	Description	4-173
4.27.2	Usage	4-176
4.27.3	JCL considerations	4-178
4.27.4	Examples	4-179
4.27.5	Sample output	4-179
4.27.6	For more information	4-179

---

4.28 ROLLFORWARD	4-180
4.28.1 Description	4-180
4.28.2 Usage	4-184
4.28.3 JCL considerations	4-187
4.28.4 Examples	4-188
4.28.5 Sample output	4-188
4.28.6 For more information	4-188
4.29 TUNE INDEX	4-189
4.29.1 Description	4-189
4.29.2 Usage	4-190
4.29.3 JCL considerations	4-191
4.29.4 Example	4-191
4.29.5 Sample output	4-191
4.29.6 For more information	4-192
4.30 UNLOAD	4-193
4.30.1 Description	4-193
4.30.2 Usage	4-195
4.30.2.1 How UNLOAD works	4-196
4.30.2.2 The unload/reload subschemas	4-197
4.30.2.3 The unload/reload SEGMENT	4-198
4.30.2.4 The unload/reload DMCL	4-198
4.30.2.5 General procedure for UNLOAD and RELOAD	4-199
4.30.3 JCL considerations	4-200
4.30.4 Example	4-200
4.30.5 Sample output	4-201
4.30.6 For more information	4-201
4.31 UNLOCK	4-202
4.31.1 Description	4-202
4.31.2 Usage	4-202
4.31.3 JCL considerations	4-202
4.31.4 Examples	4-203
4.31.5 Sample output	4-203
4.31.6 For more information	4-203
4.32 UPDATE STATISTICS	4-204
4.32.1 Description	4-204
4.32.2 Usage	4-205
4.32.3 JCL considerations	4-206
4.32.4 Example	4-206
4.32.5 Sample output	4-207
4.32.6 For more information	4-207
4.33 VALIDATE	4-208
4.33.1 Description	4-208
4.33.2 Usage	4-210
4.33.3 JCL considerations	4-211
4.33.4 Example	4-212
4.33.5 Sample output	4-212
4.33.6 For more information	4-212



## 4.1 ARCHIVE JOURNAL

### 4.1.1 Description

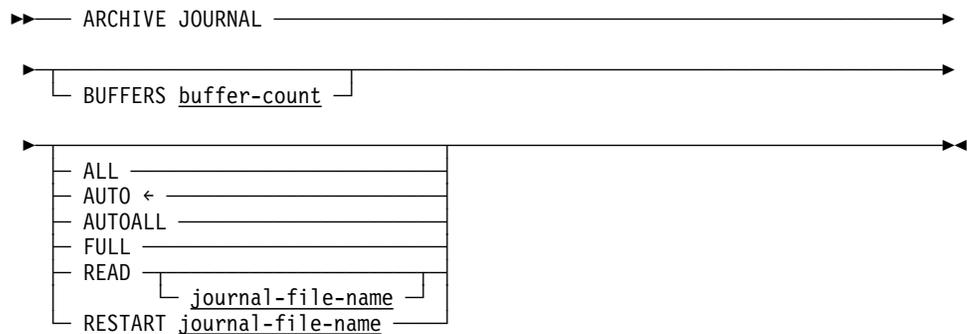
**Purpose:** The ARCHIVE JOURNAL utility offloads, to one or more archive journal files, the entries in one or more disk journal files.

The options you can choose depend upon the type of recovery being performed and whether or not the DC/UCF system is active.

**Authorization:**

To	You need this privilege	On
Archive one or more journal files	USE	The DMCL

**Syntax**



**Parameters**

**BUFFERS**

Specifies the number of buffer pages to be used during condense processing.

Condense processing involves copying the before image of unfinished transactions back to the disk journal file after it is offloaded. Condense processing of journal segments takes place when *both* of the following apply:

- You specify *AUTO* and
- The DC/UCF system is active or after an abnormal system termination

The buffer is used for storing before images of unfinished transactions before copying the images back to the disk journal file.

The buffer page size is equal to the block size of the disk journal file.

**buffer-count**

An integer in the range 2 through 32,767. The default is 5.

**ALL**

Specifies that all non-empty disk journal files are to be offloaded, starting with the file containing the oldest entries.

**Note:** While the DC/UCF system is active or after an abnormal system shutdown, *do not* specify ALL unless you are beginning a manual recovery of the entire system.

After offloading the journal files, they are marked as empty.

Before images of unfinished transactions *are not* rewritten or condensed after offloading the journal files. These before images are needed if you need to recover from an abnormal shutdown.

**AUTO**

Directs the ARCHIVE JOURNAL utility to select a single file to offload:

- While the DC/UCF system is active or after an abnormal system termination, the oldest full disk journal file is selected.

After offloading the file, it is condensed.

After condensing the selected file, a new, empty journal segment is created.

If no full disk journal file exists, no file is offloaded or condensed.

- After a normal DC/UCF system shutdown, the oldest non-empty disk journal file is selected.

After offloading the file, it is marked as empty.

**AUTOALL**

- While the DC/UCF system is active or after an abnormal system shutdown, the oldest full disk journal file is offloaded. The file is condensed after it is offloaded and a new empty journal segment is created.

- After a normal DC/UCF shutdown, specifies that all non-empty disk journal files are to be offloaded, starting with the file containing the oldest entries.

After offloading the journal files, they are marked as empty.

**FULL**

Directs the ARCHIVE JOURNAL utility to offload all full disk journal files associated with the database, starting with the file containing the oldest entries. After offloading each file, it is marked as empty.

Before images of unfinished transactions *are not* condensed and rewritten. These before images are needed if you need to recover from an abnormal shutdown.

**Note:** Do not specify FULL while the DC/UCF system is active or after an abnormal system shutdown, unless you are beginning a manual recovery of the entire system.

**READ**

Directs the ARCHIVE JOURNAL utility to offload a single disk journal file without condensing it or marking it as empty afterwards.

If you do not specify a file name, the oldest non-empty disk journal file is offloaded.

**journal-file-name**

The name of the disk journal file to be read.

**RESTART**

Directs the ARCHIVE JOURNAL utility to restart an archive journal operation that terminated abnormally. If the operation failed:

- While a disk journal file was being *offloaded*, the offload operation is restarted.
- While a disk journal file was being *condensed*, the condense operation is restarted.

**journal-file-name**

The name of the disk journal file being offloaded at the time of the abend.

## 4.1.2 Usage

**Summary of offload parameter options:** In summary, if the DC/UCF system is *not* active, all disk journal files are offloaded and marked empty.

If the DC/UCF system is active, only the oldest full disk journal file is processed (as if the AUTO option is specified).

<b>Parameter</b>	<b>While DC/UCF active</b>	<b>After abnormal termination</b>	<b>After normal shutdown</b>
ALL	Don't specify.	Don't specify.	All non-empty journal database files, beginning with the oldest file, are offloaded.  Offloaded files are marked as empty.
AUTO	Oldest full disk journal file is offloaded.  Condenses offloaded file, then creates a new, empty journal segment.		Oldest non-empty disk journal file is offloaded; offloaded file marked as empty.
AUTOALL	Offloads and condenses the oldest full disk journal file, and then creates a new, empty journal segment.	All non-empty disk journal files, beginning with the oldest file, are offloaded.  Offloaded files are marked as empty.	

**How to submit the ARCHIVE JOURNAL statement:** You submit the ARCHIVE JOURNAL statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**You normally specify AUTO:** Normally, you archive journal files as they become full. For this purpose, specify AUTO.

The other options are used only in special circumstances, such as recovering damaged files.

### 4.1.3 JCL considerations

When you submit an ARCHIVE JOURNAL statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The disk journal files
- The archive journal files.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.1.4 Example

While your DC/UCF system is active, you can initiate a job that will offload full disk journal files. To do this, specify the AUTO option of the ARCHIVE JOURNAL statement:

```
archive journal auto;
```

### 4.1.5 Sample output

After successful completion of the ARCHIVE JOURNAL statement submitted with the AUTO option, the CA-IDMS Batch Command Facility produces the following listing:

```
IDMSBCF                                IDMS Batch Command Facility                                09/17/99  PAGE 1
ARCHIVE JOURNAL ;

      JOURNAL DISK FILES STATUS REPORT

FILENAME          SEGMENT  LORBN  HIRBN  FULL  ACTIVE  STATUS  CV ACTIVE
SYSJRN2           77      10  1000  NO    YES    NON-AJNL  YES
                  75      8    9
SYSJRN1           76      10  1000  YES   NO     NON-AJNL  YES
                  74      8    9

WILL SELECT SYSJRN1          FOR OFFLOADING

PERCENTAGE DISTRIBUTION PER PAGE
PERCENT      NO OF PAGES
0-10         430
11-20        15
21-30        185
31-40        15
41-50         5
51-60        160
61-70         3
71-80        29
81-90         3
91-100       145

DISK BLOCKS OFFLOADED          990
TAPE BLOCKS WRITTEN THIS SEGMENT 32
TOTAL TAPE BLOCKS WRITTEN       32

END OF JOURNAL ARCHIVE      1999-09-17-14.38.43.924089
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

### 4.1.6 For more information

- **On journaling procedures**, see *CA-IDMS Database Administration*.
- **On system generation journal parameters**, see *CA-IDMS System Generation*.

## 4.2 ARCHIVE LOG

### 4.2.1 Description

**Purpose:** The ARCHIVE LOG utility offloads, to an archive file, the contents of the DC/UCF system log.

**Authorization:**

To	You need this privilege	On
To archive a system log	DBAWRITE	The SYSTEM.DDLDCLOG area in the dictionary associated with the DC/UCF system whose log you want to archive

**Syntax**

```

▶▶— ARCHIVE LOG —┬─ PRINT ─┘

```

**Parameters**

**PRINT**

Optionally, prints a copy of the contents of the archived log. If you do not specify PRINT, a copy of the archived log is *not* printed.

### 4.2.2 Usage

**How to submit the ARCHIVE LOG statement:** You submit an ARCHIVE LOG statement to CA-IDMS/DB only through the batch command facility. You must be running CA-IDMS/DB in local mode, without journaling.

**When to use ARCHIVE LOG:** Use the ARCHIVE LOG utility only when the system log is being written to the DDLDCLOG area.

**When not to use ARCHIVE LOG:** If the system log is assigned to one or two sequential files, you should use the appropriate operating system utility (for example, IEBGENER for OS/390 systems or DITTO for VSE/ESA systems) to archive the contents of the log file.

**Archiving the log for an active system:** When you submit an ARCHIVE LOG statement while the DC/UCF system is active, CA-IDMS/DB archives the contents of the DDLDCLOG area up to, but not including, the page to which the system is currently writing.

### 4.2.3 JCL considerations

When you submit an ARCHIVE LOG statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The log area (DDLDCLOG)
- The message area (DDLDCMSG)
- The dummied journal file
- The archive log file being created

▶▶ Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.2.4 Example

The ARCHIVE LOG statement below requests that the contents of the DC/UCF system log be offloaded to an archive file and to print the contents of the archived log.

```
archive log print;
```

### 4.2.5 Sample output

The ARCHIVE LOG utility produces the following standard listing:

## 4.2 ARCHIVE LOG

```
IDMSBCF                                IDMS Batch Command Facility                                09/19/99  PAGE 1
ARCHIVE LOG;
CA-IDMS-DB/DC Print Log Utility          CA-IDMS-DB/DC is a Proprietary Software Product    DATE       TIME       PAGE
CAGJF0       Release 15.0                Licensed from Computer Associates International      09/19/99   09:31:04   1

*** PAGE 000030002 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000031001 STATUS 0000 091799 14.20.00 6 S
*** PAGE 000031500 STATUS 0000 091699 10.19.58 6 S
*** PAGE 000031750 STATUS 0000 091699 14.30.01 6 S
*** PAGE 000031875 STATUS 0000 091699 16.46.34 6 S
*** PAGE 000031937 STATUS 0000 091699 17.54.59 6 S
*** PAGE 000031968 STATUS 0000 091699 18.34.58 6 S
*** PAGE 000031984 STATUS 0000 091699 18.54.59 6 S
*** PAGE 000031992 STATUS 0000 091699 19.04.58 6 S
*** PAGE 000031996 STATUS 0000 091699 19.10.00 6 S
*** PAGE 000031998 STATUS 0000 091699 19.14.58 6 S
*** PAGE 000031999 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000032000 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000030002 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000032000 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000031001 STATUS 0000 091799 14.20.00 6 S
*** PAGE 000031500 STATUS 0000 091699 10.19.58 6 S
*** PAGE 000031250 STATUS 0000 091799 16.02.43 6 S
*** PAGE 000031375 STATUS 0000 091799 16.24.57 6 S
*** PAGE 000031437 STATUS 0000 091799 16.30.42 6 S
*** PAGE 000031468 STATUS 0000 091999 09.25.12 6 L
*** PAGE 000031484 STATUS 0000 091699 09.59.59 6 S
*** PAGE 000031476 STATUS 0000 091699 09.49.58 6 S
*** PAGE 000031472 STATUS 0000 091999 09.29.58 6 L
*** PAGE 000031474 STATUS 0000 091699 09.44.59 6 S
*** PAGE 000031473 STATUS 0000 091999 09.29.58 6 L
*** PAGE 000031474 STATUS 0000 091699 09.44.59 6 S
*** PAGE 000031473 STATUS 0000 091999 09.29.58 6 L
*** PAGE 000031474 STATUS 0000 091699 09.44.59 6 S
*** PAGE 000030474 STATUS 0000 091799 05.04.59 6 S
*** PAGE 000030973 STATUS 0000 091799 14.17.00 6 S
*** PAGE 000031223 STATUS 0000 091799 15.59.04 6 S
*** PAGE 000031348 STATUS 0000 091799 16.18.18 6 S
*** PAGE 000031410 STATUS 0000 091799 16.30.41 6 S
*** PAGE 000031441 STATUS 0000 091799 16.30.43 6 S
*** PAGE 000031457 STATUS 0000 091999 09.25.05 4 L
*** PAGE 000031449 STATUS 0000 091799 16.30.43 6 S
*** PAGE 000031453 STATUS 0000 091799 16.30.44 6 S
*** PAGE 000031455 STATUS 0000 091799 16.30.44 6 S
*** PAGE 000031456 STATUS 0000 091999 09.23.57 1 L
*** PAGE 000031456 STATUS 0000 091999 09.23.57 1 L
*** PAGE 000031473 STATUS 0000 091999 09.29.58 6 L
*** PAGE 000031473 STATUS 0000 091999 09.29.58 6 L
*** PAGE 000031472 STATUS 0000 091999 09.29.58 6 L
*** DDLDLOG AREA FROM PAGES 0030001 TO 0032000
*** FIRST AND LAST PAGES SELECTED ARE 0031456 AND 0031473

ARCHIVE LOG IS COMPLETE
Status = 0
AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

### 4.2.6 For more information

- On defining the DC/UCF system log, refer to *CA-IDMS System Generation*.
- On maintaining the DC/UCF system log, refer to *CA-IDMS System Operations*.
- On statistics written to the DC/UCF system log, refer to *CA-IDMS Reports*.

## 4.3 BACKUP

### 4.3.1 Description

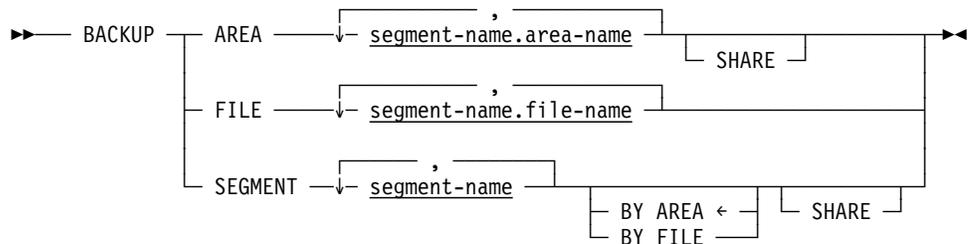
**Purpose:** The BACKUP utility copies one or more areas in a database to a backup file. The backup file can be used later as input for a restore operation.

**Note:** The format of files produced by the BACKUP utility is *not* compatible with the format of backup files produced by the 10.2 IDMSDUMP utility program. Files produced by BACKUP can only be used by the RESTORE utility and files produced by IDMSDUMP can only be used with IDMSRSTR.

**Authorization:**

To	You need this privilege	On
Back up an area	DBAREAD	The area
Back up a file	DBAREAD	The area(s) to which the file maps

**Syntax**



**Parameters**

**AREA**

Directs the BACKUP utility to back up one or more areas. Multiple area names must be separated by commas.

**segment-name**

The name of the segment associated with an area to be backed up.

**area-name**

The name of an area.

### **SHARE**

Specifies that no locks are to be placed on the named areas. When you specify SHARE, each specified area is backed up regardless of whether a lock has been placed on the area by another program. SHARE allows the named areas to be backed up while another job is updating those areas at the same time, i.e., it needs to be specified when taking a "hot backup" during which the areas that are being backed up are being updated by transactions executing under the central version.

By default, if you do not specify SHARE, an external lock is placed on each specified area for the duration of the backup operation. If an external lock cannot be placed on an area, that area will not be backed up, and the backup operation will terminate with an error. Thus, no more areas will be backed up. An external lock means that it is placed physically using the normal SMP lock.

**Note:** When you specify SHARE and *do not* vary the affected areas for retrieval only, the copy of the database created by the BACKUP utility may not be usable for restore operations.

### **FILE**

Directs the BACKUP utility to back up one or more files.

Multiple file names must be separated by commas.

#### **segment-name**

The name of a segment associated with a file to be backed up.

#### **file-name**

The name of a file.

### **SEGMENT segment-name**

The name of the segment to be backed up.

### **BY AREA**

Specifies that each area defined within the segment is to be backed up. AREA is the default.

### **BY FILE**

Specifies that each file within the segment is to be backed up.

**Note:** The SHARE option is only valid for area processing. When you specify the BACKUP SEGMENT command with the BY FILE option, the SHARE option is ignored. If you specify the BY FILE option with the BACKUP SEGMENT command and the SHARE option is omitted, area locks are not set.

## 4.3.2 Usage

**How to submit the BACKUP statement:** You submit the BACKUP statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**When to use SHARE:** You can specify SHARE and get a backup usable for restore operations if you vary the affected area(s) for retrieval only. This prevents other users from changing the contents of the area during the backup operation.

If you specify `SHARE` without varying the area(s) for retrieval only, you should exercise extreme caution when making use of the backup.

**BACKUP by file does not lock areas:** When you back up by file, the `BACKUP` utility does not lock the associated area(s). Therefore, to preserve the integrity of the area, you should vary affected areas for retrieval only.

**RESTORE the same object you backed up:** If you back up by area, restore by area. If you back up by file, restore by file.

**Number of BACKUP statements per BCF-job:** Only one `BACKUP` statement per `BCF-job` is allowed. Specifying multiple `BACKUP` statements for the same `BCF-job` will result in all `BACKUP` files, except for the last one, being overwritten. This is caused by the way syntax parsing has been implemented, i.e., for each `BACKUP` statement the `BACKUP` utility (i.e., module `IDMSUBKP`) is called that at the beginning will open the `BACKUP` file specified by `<ddname/filename/linkname>` `SYS001` in `OUTPUT` mode and will close it before returning to the syntax parsing module.

### 4.3.3 JCL considerations

When you submit a `BACKUP` statement to `CA-IDMS/DB` through the batch command facility, the `JCL` to execute the facility must include statements to define:

- The files associated with the areas to be backed up
- The archive file which will contain the backup

►► Refer to the chapter pertaining to your operating system for generic `JCL` to execute the batch command facility.

### 4.3.4 Examples

**Back up by area:** The following example directs the `BACKUP` utility to back up three database areas.

```
backup area empdemo.emp-demo-region,
           empdemo.org-demo-region,
           empdemo.ins-demo-region;
```

**Back up by file:** The following example directs the `BACKUP` utility to back up three database files.

```
backup file empdemo.empdemo,
           empdemo.orgdemo,
           empdemo.insdemo;
```

**Back up by segment:** The following example directs the `BACKUP` utility to back up all areas in the `empdemo` segment.

```
backup segment empdemo;
```

### 4.3.5 Sample output

**Back up by area:** When the backup by area operation in the above example is completed, the BACKUP utility provides the following report.

```
IDMSBCF 15.0                                CA-IDMS Batch Command Facility          10/30/99  PAGE 1

  BACKUP AREA EMPDEMO.EMP-DEMO-REGION,
           EMPDEMO.ORG-DEMO-REGION,
           EMPDEMO.INS-DEMO-REGION;

UT015006 BACKUP file created on 1999-10-30-16.08.53.303879
UT015005 Max Archive record size is 4,280
UT000038 Starting BACKUP of area EMPDEMO.EMP-DEMO-REGION
UT000040 BACKUP complete
UT000038 Starting BACKUP of area EMPDEMO.ORG-DEMO-REGION
UT000040 BACKUP complete
UT000038 Starting BACKUP of area EMPDEMO.INS-DEMO-REGION
UT000040 BACKUP complete

Status = 0      SQLSTATE = 00000
```

**Output from back up by file:** When the backup by file operation in the above example is completed, the BACKUP utility provides the following report.

```
IDMSBCF 15.0                                CA-IDMS Batch Command Facility          10/30/99  PAGE 3

  BACKUP FILE EMPDEMO.EMPDEMO,
           EMPDEMO.ORGDEMO,
           EMPDEMO.INSDEMO;

UT015006 BACKUP file created on 1999-10-30-16.09.36.080684
UT015005 Max Archive record size is 4,280
UT000039 Starting BACKUP of file EMPDEMO.EMPDEMO
UT000040 BACKUP complete
UT000039 Starting BACKUP of file EMPDEMO.ORGDEMO
UT000040 BACKUP complete
UT000039 Starting BACKUP of file EMPDEMO.INSDEMO
UT000040 BACKUP complete

Status = 0      SQLSTATE = 00000
```

**Output from back up by segment:** When the backup by segment operation in the above example is completed, the BACKUP utility provides the following report.

```
IDMSBCF 15.0                                CA-IDMS Batch Command Facility                10/30/99  PAGE 5

  BACKUP SEGMENT EMPDEMO;

UT015006 BACKUP file created on 1999-10-30-16.10.04.315493
UT015005 Max Archive record size is 4,280
UT000038 Starting BACKUP of area EMPDEMO.EMP-DEMO-REGION
UT000040 BACKUP complete
UT000038 Starting BACKUP of area EMPDEMO.INS-DEMO-REGION
UT000040 BACKUP complete
UT000038 Starting BACKUP of area EMPDEMO.ORG-DEMO-REGION
UT000040 BACKUP complete

Status = 0          SQLSTATE = 00000
```

### 4.3.6 For more information

- **On using the SHARE option**, refer to *CA-IDMS Database Administration*.
- **On varying areas**, refer to *CA-IDMS System Tasks and Operator Commands*.

## 4.4 BUILD

### 4.4.1 Description

**Purpose:** The BUILD utility statement builds indexes and referential constraints linked through an index on tables that are being loaded with a phased or stepped LOAD.

The BUILD utility can also be used to reorganize existing indexes.

The BUILD utility works only on tables in an SQL-defined database.

---

Type of BUILD	What it does
Complete BUILD	Runs all four steps
Stepped BUILD	Runs one step at a time with intermediate file sorting required between each step

---

**Authorization:**

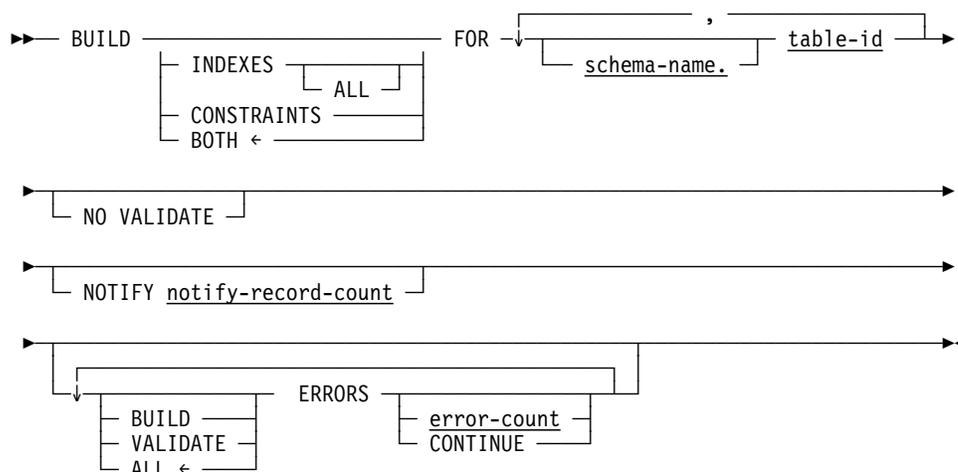
---

To	You need this privilege	On
Build indexes and/or referential constraints on a table	INSERT	The indexed or referencing table

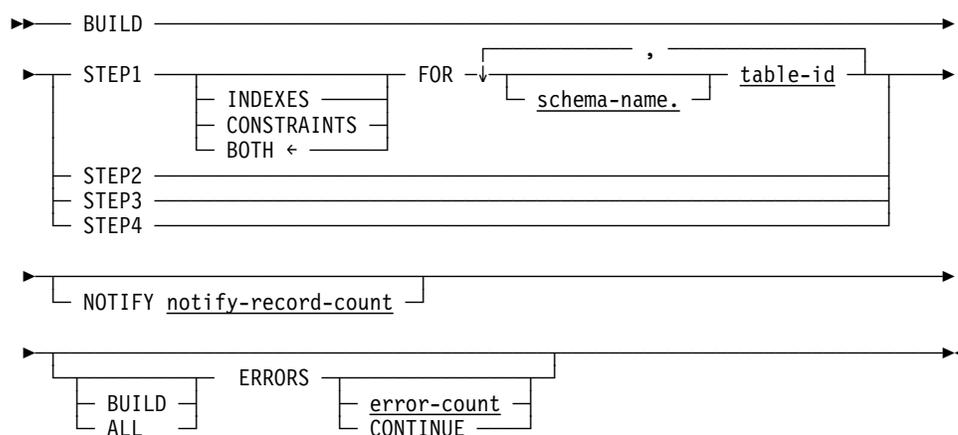
---

**Syntax**

*Syntax for complete BUILD:*



#### Syntax for stepped BUILD



### Parameters

#### INDEXES

Directs the BUILD utility to build the indexes only.

By default, if you do not specify what is to be built, both indexes and constraints are built.

#### ALL

Directs the BUILD utility to build all indexes (clustered and non-clustered). If ALL is not specified, only non-clustered indexes will be built.

#### CONSTRAINTS

Directs the BUILD utility to build referential constraints only.

By default, if you do not specify what is to be built, both indexes and constraints are built.

#### BOTH

Directs the BUILD utility to build both indexes and relationships. BOTH is the default.

**FOR**

Specifies the table for which indexes and/or constraints are to be built.

**schema-name.**

The name of the schema that defines the table.

**table-id**

The identifier of the table.

**NO VALIDATE**

Directs the BUILD utility not to validate referential constraints.

If you specify NO VALIDATE, you will have to execute the VALIDATE utility before you can use the table(s).

By default, the validation is performed.

You can specify NO VALIDATE only for a complete BUILD.

**NOTIFY**

Directs the BUILD utility to send a message to the operator whenever a specified number of records are processed.

The message states the phase and step currently being executed and the number of records that have been processed.

**notify-record-count**

The number of records to process before sending a message.

**BUILD ERRORS**

When errors are detected, directs the BUILD utility to either continue processing or stop after a specified number of errors are detected.

By default, processing is stopped after the first error is detected.

Detected errors are listed in the report generated by the BUILD utility.

**VALIDATE ERRORS**

When errors are detected in the validation process, directs the BUILD utility to either continue processing or stop processing after a specified number of errors are encountered.

By default, processing is stopped after the first error is detected.

Detected errors are listed in the report generated by the BUILD utility.

You can specify VALIDATE ERRORS only for a complete BUILD.

**ALL ERRORS**

Directs the BUILD utility to either continue when any errors are detected, or stop after a specified number of errors are detected.

By default, processing is stopped after the first error is detected.

Detected errors are listed in the report generated by the BUILD utility.

**error-count**

The number of errors to detect before stopping.

If you are doing a complete BUILD, you can specify different values for *error-count* for different kinds of errors.

**CONTINUE**

Indicates that processing should continue regardless of the number of errors detected.

**STEP $n$** 

Directs the BUILD utility to perform only the  $n$ th step of the index or constraint building process.

By default, if you do not specify a step, all four steps are performed as a single operation, and is considered a complete BUILD.

**STEP1**

Directs the BUILD utility to perform only STEP1 of the BUILD process. STEP1 sweeps the area containing the specified table(s), creating an intermediate work file. The file contains the information needed later to build the index structures.

If you specified the EXTRACT option in STEP1 of the LOAD utility, you do not need to run BUILD STEP1.

In this case, the intermediate work file (SYS003) that is output from the LOAD utility can be used as the input file (SYS002) to STEP2 of the BUILD utility.

**STEP2**

Directs the BUILD utility to perform only STEP2 of the BUILD process. STEP2 determines the database key of the referenced table rows.

**STEP3**

Directs the BUILD utility to perform only STEP3 of the BUILD process. STEP3 creates the index structures needed for both indexes and constraints.

**STEP4**

Directs the BUILD utility to perform only STEP4 of the BUILD process. STEP4 updates the prefix(es) of the affected referencing table rows.

## 4.4.2 Usage

**How to submit the BUILD statement:** You submit the BUILD statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**When to use BUILD:** Use the BUILD utility after loading one or more tables using a phased or stepped LOAD.

You can also use the BUILD utility at any time to reorganize existing indexes on tables in an SQL-defined database.

**When not to use BUILD:** There is no need to run the BUILD utility if you loaded the table(s) with a complete LOAD. The indexes and constraints have already been built.

If the table is not part of an SQL-defined database, you cannot use the BUILD utility.

**When to specify NO VALIDATE:** The BUILD utility validates all referential constraints on the tables being worked on, not just the constraints currently being built. If all tables referenced by those specified by *table-id* have not yet been loaded, defer validation by specifying NO VALIDATE.

►► For help in deciding which options to specify, refer to *CA-IDMS Database Administration*.

**BUILD utility uses intermediate work files:** Each step of the build process, except BUILD STEP4, produces intermediate work files to be used by the next step. If you run a complete BUILD without separating steps, data is sorted in the intermediate files between the steps automatically. If you run a stepped BUILD, you must run the intermediate sorts.

**Note:** When running a complete BUILD, SYS002 and SYS003 must point to the *same* intermediate file. If the database being processed is so large that the intermediate file must be a multi-volume file it is required that all extents of the file are physically allocated prior to the initiation of the BUILD utility. If this cannot be done then run a stepped BUILD. When running a stepped BUILD, SYS002 and SYS003 must point to different intermediate files. The data that is output in SYS003 by each step is input to the next step in SYS002.

The following table shows the output of the steps of the BUILD process:

Step	Output	Size
STEP1	SYS003	For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24
	SYSPCH contains sort parameters	80 bytes
STEP2	SYS003	For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24
	SYSPCH contains sort parameters	80 bytes
STEP3	SYS003	For each prefix: 56 bytes
	SYSPCH contains sort parameters	80 bytes

**Sorting intermediate work files:** If you run the load process in steps or phases, use the sort parameters in the SYSPCH file to sort the intermediate files.

### 4.4.3 JCL considerations

When you submit a BUILD statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The dictionary containing table definitions
- The files containing the tables and indexes to be processed
- Intermediate work files to be used by BUILD
- Sort work files are needed if doing a complete BUILD

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.4.4 Examples

The example below instructs the BUILD utility to perform a complete BUILD for the LOAD.M and LOAD.M2 sample tables. The NO VALIDATE option specifies that a VALIDATE should not be performed and ERRORS CONTINUE indicates that processing should continue regardless of the number of errors.

```

build for load.m,
      load.m2
errors continue
no validate;

```

### 4.4.5 Sample output

The report below was generated after executing the BUILD statement in the example above.

```

IDMSBCF                                IDMS Batch Command Facility

*DEBUG IDMS OFF
CONNECT TO SYSCAT;
Status = 0
SET BATCH
  HEADINGS OFF WIDTH PAGE 79 UNDERLINE '-'
  SQLCODE ERROR
  COMPRESS ON;

UNLOCK AREA SYSSQL.DDL CAT;
Status = 1      Extended Reason Code = 2367      Messages follow:
DB002367 C1M353: Area SYSSQL.DDL CAT was not locked.
UNLOCK AREA SYSSQL.DDL CATX;
Status = 1      Extended Reason Code = 2367      Messages follow:
DB002367 C1M353: Area SYSSQL.DDL CATX was not locked.

-- **** Load data into Tables      ****
*DEBUG IDMS ON

BUILD FOR LOAD.M,
      LOAD.M2
      ERRORS CONTINUE
      NO VALIDATE;
IDMSLOAD - CAGJF0      SWEEP DATABASE      99-11-07-12.45.16
IDMSLOAD - 3 records processed for table LOAD.M
IDMSLOAD - 3 intermediate records for index LOAD.IX_M
IDMSLOAD - 3 records processed for table LOAD.M2
IDMSLOAD - 3 intermediate records for index LOAD.IX1_M2
IDMSLOAD - 3 intermediate records for index LOAD.IX2_M2
IDMSLOAD - 15 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - SWEEP DATABASE      processing completed
IDMSLOAD - CAGJF0      CONNECT UP INDEXES      99-11-07-12.45.23
IDMSLOAD - 6 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - CONNECT UP INDEXES      processing completed

AutoCommit will COMMIT transaction

Command Facility ended with warnings

```

### 4.4.6 For more information

- **On designing indexes and referential constraints in SQL-defined databases,** see *CA-IDMS Database Design*.
- **On defining indexes and referential constraints in SQL-defined databases,** see *CA-IDMS Database Administration*.

## 4.5 CLEANUP

### 4.5.1 Description

**Purpose:** The CLEANUP utility physically erases logically deleted records from all or some areas in a database segment.

The CLEANUP utility works only with non-SQL defined databases.

**Authorization:**

To	You need this privilege	On
Clean up an area	DBAWRITE	The area
Clean up a segment	DBAWRITE	All areas within the segment

**Syntax**

```

▶— CLEANUP SEGMENT segment-name —————▶
▶— USING subschema-name —————▶
▶— [ AREA [ area-name ] ] —————▶

```

**Parameters**

**SEGMENT**

Specifies the segment containing the areas to be processed.

**segment-name**

The name of the segment.

**USING**

Specifies a subschema that describes all the sets, records, and areas that are related to the logically deleted records being processed.

**subschema-name**

The name of the subschema.

**AREA**

Specifies one or more areas within the segment to process.

By default, if you do not specify any areas, all areas in the specified segment are processed.

**area-name**

The name of an area within the specified segment.

## 4.5.2 Usage

**How to submit the CLEANUP statement:** You submit the CLEANUP statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**How CLEANUP works:** An area sweep is performed on each specified area. When a logically deleted record is found, each set in which the record is a member is processed. Every logically deleted record in the set is first disconnected and then physically erased. When all sets in which the original record was a member are processed, the sweep of the area is resumed.

**When to use CLEANUP:** Use the CLEANUP utility:

- To erase deleted records in sets with no prior pointers
- Before running the RESTRUCTURE SEGMENT utility
- Before running the UNLOAD or RELOAD utilities

**When not to use CLEANUP:** Records in sets *with* prior pointers are erased when they are deleted. If all the sets in an area have prior pointers, you never need to use the CLEANUP utility.

**Journaling:** You can use journaling while executing the CLEANUP utility to allow for recovery with the ROLLBACK utility.

## 4.5.3 JCL considerations

When you submit a CLEANUP statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The files containing the areas to be processed.
- The journal files of the DMCL you are using. If you are not journaling, these should be dummied out.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.5.4 Example

The following example directs the CLEANUP utility to erase any logically deleted records in the EMP-DEMO-REGION area.

```
cleanup segment empdemo using empss01
      area emp-demo-region;
```

### 4.5.5 Sample output

When the CLEANUP operation is completed, the following listing is provided.

```
IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1

  CLEANUP SEGMENT EMPDEMO USING EMPSS01
    AREA EMPDEMO.EMP-DEMO-REGION;
UT000038 Starting CLEANUP of area EMPDEMO.EMP-DEMO-REGION
UT011020 CLEANUP completed.  Pages read=10  Records read=5  Ldel records read=2
UT011021 LDEL Record name BB  Found=2  Removed=2
Status = 0

AutoCommit will COMMIT transaction
```

### 4.5.6 For more information

- On preventing logically deleted records, see *CA-IDMS Database Administration*.



page changes. Unless restricted by the FILE parameter, all files for this area are scanned for page range changes and copied to new files.

**target-segment**

Names the segment with the new page range and maximum records per page definitions. If you omit target-segment, it defaults to the source segment name. You should use this when the target segment is in the same DMCL as the source.

**target-seg.area**

Names the area that contains the new page range and maximum records per page definitions. If you omit this, it defaults to source-seg.area name. If you omit the area, it defaults to the source area name. You should use this when the target area is in the same DMCL as the source.

**target-dmcl**

Names the DMCL that contains the target segment or area definitions. Use this parameter when the target segment or area is defined in a different DMCL than the current one.

**filename**

Restricts the conversion to the selected source files. If you do not specify the FILE parameter, then all files for selected segments and areas are converted. However, all target file definitions must have ddnames that are different from the source file definitions.

**new-ddname**

Names the JCL ddname used for the output file. This name must be unique in the job step. If you don't specify new-ddname, the ddname in the target file definition is used. In this case the name of the target file must match the source file.

**not-interval**

Specifies the notify interval. After every not-interval record read, a message is issued to the console stating how far the job has progressed. If omitted, the default interval is 10,000 records. If specified as zero (0), no notify message is issued.

## 4.6.2 Usage

You may process an entire segment or one or more individual areas using this utility. The utility first identifies the changes that need to be made by comparing old and new definitions for the segment or areas. Once all changes have been identified, it converts one or more files. It is possible to run the utility in parallel to process several files concurrently.

This utility allows you to reassign the page range of an area so that you can consolidate free pages. It also allows you to change the maximum number of records on a page, thus permitting more effective use of the space on a page and more flexibility in choosing page sizes.

In order to use the feature, you must define one or more new segments and include them in a DMCL, or alter the definition of an existing segment and include it in a new

DMCL. The utility requires both the old and new definitions of the affected segment or areas to be available at runtime.

Once the definitions are available, you can execute the utility. There must be enough disk space available to hold all converted files. The converted files can also be written to tape and then copied back to the original database files. Be sure to back up the original database beforehand.

Once all files have been converted, you must make the new segment or area definitions effective.

**Page range and radix changes:** Page range and radix changes are determined by comparing all source and target area definitions. Regardless of which files are selected for conversion, all differences are applied to all files selected for conversion.

**File conversion:** File conversion is performed against all files in the specified segment or areas, unless the FILE parameter is specified, in which case only specified files are converted.

File conversion consists of copying a page from the source file to the target file while making any required changes. Page number changes are accomplished by subtracting the old low page and adding the new low page. Maximum records per page changes are effected by breaking up a dbkey into a line and page number using the old information and reassembling it using the new information.

**Note:** If no changes apply to a selected file it is copied anyway.

The Convert Page Utility also has the following usage considerations:

- If an area that is being converted has a cross-area set or linked constraint, then you must convert all areas touched by the set or link. You must determine what files are affected, and therefore what files must be converted.
- If you need to convert multiple files, the files do not all have to be converted in the same job.
- As long as all changes are identified to each job, you can convert each affected file separately. It may be desirable to run several jobs in parallel to reduce the time required to convert an area.

***Important!** It is important that you identify all page range changes that affect a file before the file is converted. Depending on the nature of the page range change, it may not be possible to apply a partial change to a previously converted file.*

- If you omit a change identification, then you will need to reapply all changes to the original file, not the converted file.
- You cannot change the number of pages assigned to an area, nor lower the maximum number of records per page to less than the actual number of records stored on any page of an area.

- This utility must run in local mode and with update activity quiesced on the affected areas.
- The source and target files must match in page size and number of blocks. (The source file is copied block for block.) If the target file does not match then the converted file will not be usable by the new file definition.

### 4.6.3 JCL considerations

All input and output ddnames must be unique within the same job step. If the target DMCL files do not have unique ddnames, use the INTO option for each output file.

JCL must include DD statements for the output files. If a file is not defined to use dynamic file allocation, then DD statements for the input files are also required.

►► Refer to the chapter pertaining to your operating system generic JCL to execute the batch command facility.

### 4.6.4 Examples

The EMPDEMO database has three areas: the Emp-demo-region, Org-demo-region, and the Ins-demo-region. There are cross-area sets between the Emp and Org-demo-regions, and between the Emp and Ins-demo-regions, but there are no sets between the Ins-demo-region and the Org-demo-region; therefore, if you need to change the:

- Page range for the Org-demo-region, you must convert both the Emp-demo-region and the Org-demo-region.
- Page range for the Ins-demo-region, you must convert both the Emp-demo-region and the Ins-demo-region.
- Page range for the Emp-demo-region, you need to convert all three areas.

#### 4.6.4.1 Example 1

The first step in converting an area's page range is to define the new segment with all changes. This new segment could be included in the current DMCL if its name is different from the old segment, if the page group and page ranges are different from the old ones, and if the ddnames are different from the old ones.

After the DMCL is updated, the required syntax to do all conversions would be:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO;
```

The Convert Page Utility scans both segments and determines which areas are being changed. Because the output files have unique ddnames, they do not need to be identified, but all three files that are in the EMPDEMO segment are converted, even if the changes do not affect all files.

To restrict the process, you could code the following:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO
  FILE EMPDEMO-FILE, ORGDEMO-FILE;
```

This syntax would only convert the EMPDEMO and ORGDEMO files. If the INSDemo file were not affected by any changes, then this would be all that would be needed. However, if the INSDemo file were affected, it could be converted with a separate job as follows:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO
  FILE INSDemo-FILE;
```

This works because all changes were identified to both jobs, because all changes are contained in the single segment.

When completed, the DMCL would have to be modified a second time to remove the old segment definition and to possibly rename the new segment to the old one, to possibly change the new page group to the old page group, and to possibly change the ddnames back to the old ones.

#### 4.6.4.2 Example 2

Another approach is to define a new DMCL with all changes made to the existing segment.

In this case you would not have to modify the DMCL a second time, but more syntax is required for the conversion process, as follows:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO EMPDEMO DMCL NEWDMCL
  FILE EMPDEMO-FILE INTO EMPDDX,
      INSDemo-FILE INTO INSDDX,
      ORGDEMO-FILE INTO EMPDDX;
```

This syntax requires that you give each output file a unique ddname for the run only, because the defined ddnames in the target DMCL are the same as the source DMCL. However, once converted, the old DMCL is discarded, and the new one is renamed and the old one is replaced.

Again, if all three files are not affected, the number of files converted can be reduced by not naming the unaffected files.

#### 4.6.4.3 Example 3

If area syntax were being used, the syntax could be as follows:

```
CONVERT PAGE IN AREA
  EMPDEMO.EMP-DEMO-REGION TO NEWDEMO,
  EMPDEMO.INS-DEMO-REGION TO NEWDEMO,
  EMPDEMO.ORG-DEMO-REGION TO NEWDEMO;
```

Or if the areas were in a different DMCL:

```

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                  EMPDEMO.INS-DEMO-REGION,
                  EMPDEMO.ORG-DEMO-REGION
DMCL NEWDMCL
FILE EMPDEMO-FILE INTO EMPDDX,
   INSDEMO-FILE INTO INSDDX,
   ORGDEMO-FILE INTO EMPDDX;

```

This syntax works because all areas affected by the changes are identified. Again, this converts all files even if some are not affected.

#### 4.6.4.4 Example 4

To only convert affected files, specify only the files that should be converted:

```

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION TO NEWDEMO,
                  EMPDEMO.INS-DEMO-REGION TO NEWDEMO,
                  EMPDEMO.ORG-DEMO-REGION TO NEWDEMO
FILE EMPDEMO-FILE, INSDEMO-FILE;

```

Or if the areas were in a different DMCL:

```

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                  EMPDEMO.INS-DEMO-REGION,
                  EMPDEMO.ORG-DEMO-REGION
DMCL NEWDMCL
FILE EMPDEMO-FILE INTO EMPDDX,
   INSDEMO-FILE INTO INSDDX;

```

This syntax would only cause the Emp-demo-region and the Ins-demo-region to be converted.

If you decided that the Org-demo-region also needed to be converted, you could run the following:

```

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                  EMPDEMO.INS-DEMO-REGION,
                  EMPDEMO.ORG-DEMO-REGION
DMCL NEWDMCL
FILE ORGDEMO-FILE INTO ORGDDX;

```

This syntax works because all three areas were identified to compare phase. Even though the files were converted in separate jobs, all jobs knew about all changes.

#### 4.6.4.5 Bad Example 4

The wrong way to do the previous example would be to reduce the areas considered for change, instead of the files to be converted:

```

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                  EMPDEMO.INS-DEMO-REGION,
DMCL NEWDMCL
FILE EMPDEMO-FILE INTO EMPDDX,
   INSDEMO-FILE INTO INSDDX;

```

This syntax works only if the Org-demo-region is unaffected by any changes. Only the Emp-demo-region and the Ins-demo-region are converted. But if it were later discovered that the Org-demo-region required conversion, you would have to reconvert the Emp-demo-region.

```

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                EMPDEMO.INS-DEMO-REGION,
                EMPDEMO.ORG-DEMO-REGION
                DMCL NEWDMCL
                FILE EMPDEMO-FILE INTO EMPDDX,
                ORGDEMO-FILE INTO ORGDXX;

```

Note that Ins-demo-region did not have to be reconverted because it was unaffected by changes to the Org-demo-region, but because the Emp-demo-region is affected by changes to the Ins-demo-region; Ins-demo-region is included in the list of changed areas.

### 4.6.5 Sample output

```

IDMSBCF 15.0 CA-IDMS Batch Command Facility 12/29/99 PAGE 1

CONVERT PAGE
  AREA EMPDEMO.EMP-DEMO-REGION TO EMPXDEMO,
    EMPDEMO.ORG-DEMO-REGION TO ORGXDEMO,
    EMPDEMO.INS-DEMO-REGION TO INSXDEMO
  DMCL EMPXDMCL
  NOTIFY 10
;
UT018000 Convert Page Utility Starting:
UT018001 Source DMCL: LRDTDMCL Target DMCL: EMPXDMCL
UT018003 Comparing source area: EMPDEMO.EMP-DEMO-REGION
UT018004 with target area: EMPXDEMO.EMP-DEMO-REGION
UT018012 Area added to change table.
UT018013 SOURCE AREA Range: 75,001 to 75,100 Radix: 8
UT018013 TARGET AREA Range: 75,001 to 75,100 Radix: 7
UT018003 Comparing source area: EMPDEMO.ORG-DEMO-REGION
UT018004 with target area: ORGXDEMO.ORG-DEMO-REGION
UT018012 Area added to change table.
UT018013 SOURCE AREA Range: 75,151 to 75,200 Radix: 8
UT018013 TARGET AREA Range: 75,001 to 75,050 Radix: 9
UT018003 Comparing source area: EMPDEMO.INS-DEMO-REGION
UT018004 with target area: INSXDEMO.INS-DEMO-REGION
UT018012 Area added to change table.
UT018013 SOURCE AREA Range: 75,101 to 75,150 Radix: 8
UT018013 TARGET AREA Range: 75,001 to 75,050 Radix: 8
UT018005 File EMPDEMO.EMPDEMO has been selected for processing
UT018006 Source Ddname: EMPDEMO Target Ddname: EMPXDEMO
UT018005 File EMPDEMO.INSDEMO has been selected for processing
UT018006 Source Ddname: INSDemo Target Ddname: INSXDEMO
UT018005 File EMPDEMO.ORGDEMO has been selected for processing
UT018006 Source Ddname: ORGDEMO Target Ddname: ORGXDEMO
Starting CONVERSION of file EMPDEMO.EMPDEMO
UT001006 Program IDMSUCON processed 10 records
UT001006 Program IDMSUCON processed 20 records
UT001006 Program IDMSUCON processed 30 records
UT001006 Program IDMSUCON processed 40 records
UT001006 Program IDMSUCON processed 50 records
UT001006 Program IDMSUCON processed 60 records
UT001006 Program IDMSUCON processed 70 records
UT001006 Program IDMSUCON processed 80 records
UT001006 Program IDMSUCON processed 90 records
UT001006 Program IDMSUCON processed 100 records
CONVERSION complete
  Pages Read: 100 Modified: 100
  Records Read: 346 Modified: 346
  Dbkeys Read: 2,847 Modified: 2,847 Null: 1
  Errors Page: 0 Dbkey: 0

```

```
Starting CONVERSION of file EMPDEMO.INSDEMO
UT001006 Program IDMSUCON processed 110 records
UT001006 Program IDMSUCON processed 120 records
UT001006 Program IDMSUCON processed 130 records
UT001006 Program IDMSUCON processed 140 records
UT001006 Program IDMSUCON processed 150 records
CONVERSION complete
  Pages  Read: 50 Modified: 50
  Records Read: 84 Modified: 84
  Dbkeys Read: 494 Modified: 494 Null: 0
  Errors Page: 0 Dbkey: 0
Starting CONVERSION of file EMPDEMO.ORGDEMO
UT001006 Program IDMSUCON processed 160 records
UT001006 Program IDMSUCON processed 170 records
UT001006 Program IDMSUCON processed 180 records
UT001006 Program IDMSUCON processed 190 records
UT001006 Program IDMSUCON processed 200 records
CONVERSION complete
  Pages  Read: 50 Modified: 50
  Records Read: 195 Modified: 195
  Dbkeys Read: 1,285 Modified: 1,285 Null: 65
  Errors Page: 0 Dbkey: 0
UT001006 Program IDMSUCON processed 200 records
Status = 0          SQLSTATE = 000000

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

## 4.6.6 For more information

- **On page ranges**, see the *Database Administration Guide*.
- **On max records per page**, see the *Database Administration Guide*.

## 4.7 EXPAND PAGE

### 4.7.1 Description

**Purpose:** The EXPAND PAGE utility increases the page size of an area by transferring a database file to a new file with an expanded block size.

**Authorization:**

To	You need this privilege	On
Expand the pages of a file	DBAWRITE	All areas that map to the file.

**Syntax**

```

▶— EXPAND page for FILE segment-name.file-name —————▶
▶— INTO ddname —————▶
▶— NEWSIZE new-page-size —————▶▶

```

**Parameters**

**FILE**

Specifies the file whose pages are to be expanded.

**segment-name**

The name of the segment associated with the file.

**file-name**

The name of the file.

**INTO**

Specifies the external name of the output file.

**ddname**

The external OS/390 or CMS name of the new file.

**filename**

The external VSE/ESA name of the new file.

**linkname**

The external BS2000/OSD name of the new file.

**NEWSIZE**

Specifies the new page size.

**new-page-size**

The new page size, in bytes. The new page size must be a multiple of four and greater than the current page size.

## 4.7.2 Usage

**How EXPAND PAGE works:** A page at a time from the old file is read. The page is expanded adding the new space before the footer section of the page. The space management entry for the page is adjusted to reflect the added space. The space management pages are also expanded, but they continue to have the same number of entries as before.

**How to submit the EXPAND PAGE statement:** You submit the EXPAND PAGE statement to CA-IDMS/DB only through the batch command facility. You must be running CA-IDMS/DB in local mode. You must also vary affected areas to retrieval mode or offline in all currently executing DC/UCF systems.

**If an area maps to more than one file:** You can expand pages in only one file at a time. If you are expanding the pages for an area that maps to more than one file, you must run the EXPAND PAGE utility once for each file the area is associated with.

**Regenerate DMCL modules:** After running the EXPAND PAGE utility, you must:

- Alter the page size of the area to the new page size in all files where the area is used.
- Regenerate all DMCL's containing the file's segment.

The pages are expanded but the space management pages are not relocated. Therefore, when you redefine an area whose page size has been expanded, you must use the ORIGINAL PAGE SIZE clause of the AREA statement.

## 4.7.3 JCL considerations

When you submit an EXPAND PAGE statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the files to be processed.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.7.4 Example

The following example directs the EXPAND PAGE utility to increase the size of the EMPDEMO.EMPDEMO file.

```
expand file empdemo.empdemo
      into newfile
      newsize 4820;
```

### 4.7.5 Sample output

The EXPAND PAGE utility produces the listing below after successful completion of the statement in the above example.

```
IDMSBCF                                IDMS Batch Command Facility                09/18/99  PAGE 1
EXPAND FILE EMPDEMO.EMPDEMO INTO NEWFILE NEWSIZE 4820;
OUT000039 Starting Expansion of file EMPDEMO.EMPDEMO
OUT015007 SMI based on 4,276 characters for area EMPDEMO.EMP-DEMO-REGION
UT015008      Low page 75,001  High page 75,100  Page group 0
UT000040 Expansion complete
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

### 4.7.6 For more information

- **On recompiling DMCL modules and for guidelines on when to use the EXPAND PAGE utility**, see *CA-IDMS Database Administration*.



**AREA**

Specifies that segment-name.area-name dbkeys in the specified area will be included in the extract file.

**FILE**

Specifies that segment-name.file-name dbkeys in the specified file will be included in the extract file.

**COMPLETED**

If you do not specify a STOP time, only after images for run units that have completed by the end of the archive file will be written to the extract file.

If you do specify a STOP time, only after images for run units that have completed by the first checkpoint at or following the stop time will be written to the extract file.

**ALL**

If you do not specify a STOP time, after images for all run units will be written to the extract file.

If you do specify a STOP time, after images for all run units will be written to the extract file up to the point at or after the stop time where there are no active transactions. If such a point is not found by the end of the archive file all after images will be written.

**TO**

Specifies the external name to be used for the extract file.

**ddname**

The external OS/390 or CMS name for the extract file.

**file-name**

The external VSE/ESA name for the extract file.

**link-name**

The external BS2000/OSD name for the extract file.

**SYS002**

The default name for all systems.

**VERIFY QUIESCE**

Specifies the level of quiesce point verification that will be performed.

**NONE**

Specifies that no quiesce point verification should be done. This is the default if neither VERIFY QUIESCE nor START AT is specified. If START AT is specified, NONE is treated as LIMITED.

**LIMITED**

Specifies that limited quiesce point verification be performed. This is the default if VERIFY QUIESCE is not specified and START AT is specified.

**FULL**

Specifies that the strictest form of quiesce point verification be performed. This is the default if VERIFY QUIESCE is specified without indicating the verification level.

**START AT**

Specifies that the operation should start only after reaching a specified date and time in the journal file. Only images for transactions that begin after the specified start time will be processed. By default, processing starts at the beginning of the journal file.

**STOP AT**

Specifies that the operation should stop as soon as possible after reaching a specified date and time in the journal file.

By default, processing stops when the end of the journal file is reached.

**date**

Specifies the date, in one of the following formats:

- *yyyy-mm-dd*
- *mm/dd/yyyy*
- *dd.mm.yyyy*

In these formats, the following rules apply:

- **Yyyy** specifies the year. *Yyyy* must be an integer in the range 0001 through 9999. Leading zeros are optional.
- **Mm** specifies the month within the year. *Mm* must be an integer in the range 01 through 12. Leading zeros are optional.
- **Dd** specifies the day within the month. *Dd* must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of *yyyy*, *mm*, and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

**date-time**

Specifies the date and time, where:

- The format for specifying the DATE-TIME is:

*yyyy-mm-dd-hh.mm.ss.ffffff*

- The rules for specifying the DATE component of DATE-TIME are the same as for DATE described above. The rules for specifying the TIME component of DATE-TIME are:

- **Hh** specifies the hour on a 24-hour clock. *Hh* must be an integer in the range 00 through 23. Leading zeros are optional.
- **Mm** specifies the number of minutes past the hour. *Mm* must be an integer in the range 00 through 59. Leading zeros are optional.
- **Ss** specifies the number of seconds past the minute. *Ss* must be an integer in the range 00 through 59. Leading zeros are optional.
- **Fffffff** specifies the number of millionths of a second past the specified second. *Fffffff* is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

**time**

Specifies the time in one of the following formats:

- *hh.mm.ss*
- *hh:mm:ss*

The rules for specifying TIME are the same as those listed for DATE-TIME above.

When TIME is specified, the date defaults to the current date.

## 4.8.2 Usage

**How to submit EXTRACT JOURNAL:** EXTRACT JOURNAL must be submitted through the batch command facility and in local mode.

**When to use EXTRACT JOURNAL:** EXTRACT JOURNAL is most commonly used on a daily basis or after each archive journal is created. It is used as part of a plan for recovering a file or area on a device that takes a hardware hit.

►► For more information, see the 'FROM EXTRACT' option on the ROLLFORWARD command. It allows ROLLFORWARD processing to be done during a recovery situation when time is critical extract processing to be done during a non-critical time in case it is needed.

**How EXTRACT JOURNAL works:** EXTRACT JOURNAL works much the same as ROLLFORWARD using the sort option, except that instead of applying the latest after image directly to the database, the image is written to the extract file. The input archive journal is read and after images for selected areas are sorted. Only the latest image is saved and the rest are discarded. By saving only the latest after images for only selected areas, the resulting extract file is smaller and presorted, making subsequent processing by ROLLFORWARD go much faster.

**Multiple input archive journals:** Multiple input archive journals can be used as input provided they are read in the correct order. They do not need to be merged into a single file. With restrictions, each archived journal can also be processed with separate EXTRACT JOURNAL jobs, and the resulting extract files can be concatenated as input to one ROLLFORWARD job.

**COMPLETE and ALL with STOP time considerations:** If the COMPLETE option or the ALL option with a STOP time is specified, the resulting extract file will not contain any information for run units that were active at the end of the archive file. Therefore, an extract file created with these options should never be used as input with extract files created from subsequent archive journals. Use these options only if you intend this extract file to be the last in a series of extract files that will be used as input to a ROLLFORWARD.

**Multiple output extract files from one archive journal:** To create multiple extract files from one archive journal (one for each database segment, for instance) EXTRACT JOURNAL must be run multiple times, once for each group of database areas or files that you may wish to recover separately.

Since ROLLFORWARD has the ability to select images from an input extract file by area or file, you do not need an extract file for each area or file. For example, you can create an extract file for each critical database segment, and if a recovery is needed for one file in that segment, you can use ROLLFORWARD to recover just that file from the segment extract file. This speeds recovery while minimizing the number of extract tapes you need to maintain.

**KSDS native vsam records:** EXTRACT JOURNAL does not support KSDS native vsam records.

**Controlling the starting point of the extract operation:** If no START AT parameter is specified, the extract operation starts with the first journal image on the input archive file(s). If START AT is specified, the extract operation starts with the first checkpoint record (BGIN, COMT, ENDJ, ABRT or CKPT) whose timestamp is on or after the specified time.

Specifying a start time may save recovery time and also circumvent issues associated with aborted recovery units that span the start of the input file. It further enables the extract operation to begin processing at a quiesce point that does not correspond with a journal file boundary.

If START AT is specified, quiesce point verification is always performed even if VERIFY QUIESCE NONE is specified. Limited verification will be performed unless full verification is specifically requested.

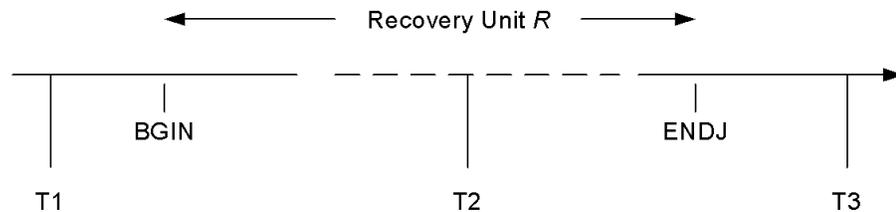
**Quiesce point verification:** A quiesce point is a point in time during which there is no update activity for some portion of a database. In order to perform a correct recovery, you must begin the recovery operation at a quiesce point for the portion of the database being recovered. In order to assist in this effort, the extract journal utility provides three levels of quiesce point verification: none, limited, and full.

None means that no quiesce point verification is performed. It is appropriate for extract operations whose input is not expected to coincide with a quiesce point. For example, if extracting journal information incrementally, then quiesce point verification could be used when processing the archive files produced immediately after a quiesce operation, but must not be used when processing subsequent archive files.

The limited and full options both enable quiesce point verification. They do this by checking for the existence of spanned recovery units which are recovery units that are active at the start of the extract operation. A recovery unit is represented by journal images starting with a BGIN or COMT checkpoint and ending with a COMT, ENDJ, or ABRT checkpoint. If a spanned recovery unit updates the specified portion of the database, then the extract operation is not starting at a quiesce point. If this situation is

detected and either limited or full quiesce point verification is in effect, the extract operation will terminate with an error.

However, it is not always possible to know whether a spanned recovery unit affects the specified portion of the database or not. If the initial BGIN or COMT checkpoint record for a recovery unit is not contained on the archive file being processed, then it is not possible to determine whether it updated the specified portion of the database. Such a recovery unit is referred to as an indoubt recovery unit.



The above time line illustrates what is meant by an indoubt recovery unit. The journal images for recovery unit *R* are written to the journal file at the times shown. If the archive file includes images starting at T1, then *R* is not an indoubt recovery unit because the archive file contains all journal images written for *R* since its inception. Similarly, if the archive file starts at time T3, *R* is not an indoubt recovery unit, because the archive file contains no images for *R* whatsoever. However, if the archive file starts at time T2, then *R* is an indoubt recovery unit, since the archive file does not contain all journal images written since its inception.

If an indoubt recovery unit does not span the start of the rollforward operation, its existence doesn't matter. But if an indoubt recovery unit is also a spanned recovery unit then the extract operation may not be starting at a quiesce point.

The action taken if an indoubt spanned recovery unit is encountered depends on whether limited or full quiesce point verification is in effect. Under full verification, the extract operation will terminate with an error. Under limited verification, a warning message will be issued identifying the recovery unit, but processing will continue. Warning messages produced under limited verification should be examined to ensure that the identified recovery units in fact did not affect the specified portion of the database. If there is any doubt, the PRINT JOURNAL utility statement should be used to gain more information about the indoubt recovery units. If after researching the situation, it is found that an indoubt recovery unit did update the specified portion of the database, the resulting extract file must not be used for recovery purposes. You must locate a quiesce point corresponding to a backup of the specified portion of the database and begin the extract operation from that point.

**When to use full or limited verification:** Full quiesce point verification should only be used if you expect that no indoubt spanned recovery units are active at the starting point of the extract operation. The only way to guarantee this is to process the archive files that were created immediately following a quiesce of update activity across all areas. One way to establish such a quiesce point is to shutdown a central version. Another way is to use the DCMT QUIESCE command and specify a DBNAME that includes every area in the DMCL.

Limited quiesce point verification can be used when processing the archive files produced immediately following a quiesce operation for the portion of the database for which the extract is being performed. One way to do this is to use the DCMT QUIESCE SEGMENT command to quiesce a segment and then use the limited quiesce point verification when extracting records for that segment.

### 4.8.3 JCL considerations

When you submit an EXTRACT JOURNAL statement to CA-IDMS/DB through the batch command facility, in addition to the standard JCL required for the batch command facility, you must also include statements to define:

- SYS001 to point to the input archived journal file.
- SYS002 or the DDname specified in the JCL to point to the output extract file. The output extract file is a standard variable length record file. Specify a block size that is at least as large as the block size of the input journal.
- Any sort work files needed by your local sort.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.8.4 Example

The following statement directs the EXTRACT JOURNAL utility to create an extract file from all after images in the EMPDEMO segment. The default ddname of SYS002 is used.

```
extract journal for segment empdemo all;
```

### 4.8.5 Sample output

The first listing below was generated after submitting the sample EXTRACT JOURNAL statement in the above example. The extract file was then used as input to the ROLLFORWARD utility to restore the EMPDEMO segment. The listing from the ROLLFORWARD utility is presented after the EXTRACT JOURNAL listing.

**EXTRACT JOURNAL listing:**

## 4.8 EXTRACT JOURNAL

```
IDMSBCF                                IDMS Batch Command Facility                02/03/99  PAGE 1

EXTRACT JOURNAL FOR
  AREA EMPDEMO.EMP-DEMO-REGION,
      EMPDEMO.INS-DEMO-REGION,
      EMPDEMO.ORG-DEMO-REGION
  ALL
  TO SYS002
  STOP AT '1999-06-30-00.00.00.000000'
  ;

ROLLFORWARD STARTED 1999-02-03-12.49.19.614065
RU_ID 00000001 PGM_ID EMPLOAD QUIESCE LEVELS 01 UPD 00 BGIN 1999-02-03-11.37.12.830234
RU_ID 00000001 PGM_ID EMPLOAD QUIESCE LEVELS 00 UPD 00 ENDJ 1999-02-03-11.39.30.533458

IMAGES SELECTED FOR AREA EMPDEMO.EMP-DEMO-REGION          1,034
IMAGES SELECTED FOR AREA EMPDEMO.INS-DEMO-REGION          196
IMAGES SELECTED FOR AREA EMPDEMO.ORG-DEMO-REGION          587

TOTAL IMAGES WRITTEN                1817

JOURNAL INPUT COUNTS:
BLOCK COUNT      705  BACKWARD      0
RECORD COUNT    5526  BACKWARD      0
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

**Listing from ROLLFORWARD:**

```
IDMSBCF                                IDMS Batch Command Facility                02/03/99  PAGE 1

ROLLFORWARD
  SEGMENT EMPDEMO
  FROM EXTRACT SYS002
  ;

ROLLFORWARD STARTED 1999-02-03-12.50.40.754380
IMAGE FILE CREATED ON 1999-02-03-12.49.19.614065
EXTRACT FILE IMAGES CREATED FROM 1999-02-03-11.37.12.830234 TO 1999-02-03-11.39.30.533458

RECORDS RESTORED TO AREA EMPDEMO.EMP-DEMO-REGION          1,034
RECORDS RESTORED TO AREA EMPDEMO.INS-DEMO-REGION          196
RECORDS RESTORED TO AREA EMPDEMO.ORG-DEMO-REGION          587

TOTAL RECORDS RESTORED          1817

JOURNAL IMAGES READ    1,818
Status = 0
IDMSBCF                                IDMS Batch Command Facility                02/03/99  PAGE 2

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

## 4.9 FASTLOAD

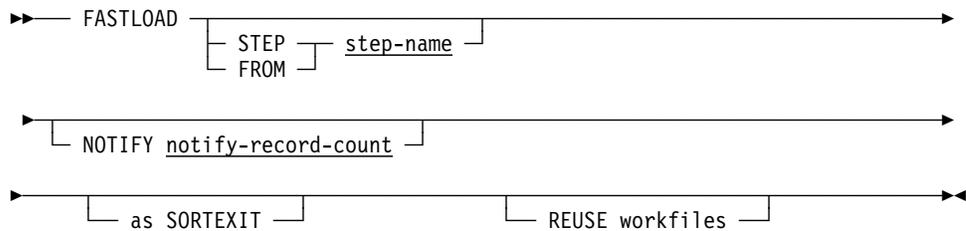
### 4.9.1 Description

**Purpose:** The FASTLOAD utility loads data into a non-SQL defined database for the first time.

**Authorization:**

To	You need this privilege	On
FASTLOAD into a segment	DBAWRITE	The segment

**Syntax**



**Parameters**

**STEP**

Specifies that only one step of the entire FASTLOAD process should be executed.

If you do not specify STEP, all steps are performed.

**FROM**

Specifies that FASTLOAD processing should begin at a specified step and that all remaining steps should be completed.

**step-name**

The name of the step to execute.

The name must be one of the following:

- SORT1
- IDMSDBL2
- SORT3
- IDMSDBL3
- SORT4
- IDMSDBL4

**NOTIFY**

Directs the FASTLOAD utility to send a message to the system console after a specified number of records are read in the current step.

If you omit the NOTIFY option or specify 0, no messages are sent. Only the standard messages that go to the SYSLST file are sent when a step is completed.

**notify-record-count**

The number of records to read before sending a message.

If you specify zero, the only message sent is the message that is always sent to the SYSLST file at the end of each step. The message indicates the number of records processed during the step.

**as SORTEXIT**

Causes each DBLx step in the utility to return its input data directly from the preceding sort instead of having the sort write the data to a workfile. This option eliminates one workfile for each sort and saves the I/O it takes to write then read the workfile.

**REUSE workfiles**

Causes each step in the utility to reuse an existing workfile, if possible, when writing its output data, instead of writing to a new one for each step. This reduces the number of workfiles that need to be allocated.

## 4.9.2 Usage

**How to submit the FASTLOAD statement:** You submit the FASTLOAD statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**When to use the FASTLOAD utility:** Use the FASTLOAD utility to load a non-SQL defined database for the first time.

**When not to use the FASTLOAD utility:** To reload a non-SQL defined database that has been unloaded, use RELOAD.

To load an SQL-defined database, use LOAD.

**Note:** FASTLOAD cannot process mixed page groups and will issue an error message if mixed page groups are encountered. If your subschema binds to multiple page groups, you must select a subset of areas to process that are all in the same page group. You must use multiple invocations of the utility to process different page groups.

**Before running FASTLOAD:** To use the FASTLOAD utility, follow these guidelines:

1. Write a format program to prepare input data and create a subschema that describes the records and sets to be loaded
2. Link edit the format program with IDMSDBLU and IDMS
3. Execute the format program

4. Execute the FASTLOAD utility to complete the loading process

**The format program:** The format program builds record and set ownership descriptors for each record to be loaded. These descriptors are used when loading the records.

**The subschema:** The subschema must:

- Include all records being loaded and all sets in which the records participate
- Allow the areas being loaded, and all areas with set connections to those areas, to be readied in exclusive update mode

**Run FASTLOAD all at once or in steps:** The processing initiated by the FASTLOAD utility has six steps, which you can run one at a time or as a single process. Each step generates output for use by the next step. Three of the steps are sorts to prepare the data for use by the next step. You can use your own sorting program or direct the FASTLOAD utility to sort the data during processing. If you run the steps as a single process, sorting is performed automatically.

**When to run FASTLOAD in steps:** The most common reason to use the FASTLOAD utility in steps is to cut the work into pieces, each piece requiring less time to run than the whole process.

You may also decide to use the FASTLOAD utility in steps in order to use your own sorting programs between the steps, or you can run the sort steps on a different machine than that holding the database.

**Restarting IDMSDBL2, IDMSDBL3 or IDMSDBL4:** If a problem arises while running IDMSDBL2 or IDMSDBL3, *do not* simply fix the cause of the problem and rerun the step. In addition to fixing the cause of the problem, you must do one of the following:

- Reinitialize the database and begin again at the IDMSDBL2 step
- If you backed up the database before running the step, you can run the step again, using the backup

These steps change the database, and if a problem arises, you need to undo the changes before running a step over again.

If a correctable abend occurs while running IDMSDBL4 (i.e., a time-out abend), you can restart the fastload operation at the IDMSDBL4 step after unlocking the areas involved. Because the IDMSDBL4 step modifies existing records, when it is restarted it will simply modify the same records in the same way.

**SORTXIT and FROM/STEP:** When using the FROM and STEP options with the SORTXIT option, each pair of SORT $n$  and DBL $x$  steps are considered to be one step. If either half of the SORT $n$ /DBL $x$  is specified on a FROM or STEP option, processing will start with the SORT $n$  step and the DBL $x$  step will also be executed. For example:

- FROM IDMSDBL3 will start with step SORT3 and will continue to the end.
- STEP SORT3 will run steps SORT3 and IDMSDBL3.

**SORTEXT/REUSE WORKFILE restart considerations:** Since SORTEXT combines each SORT $n$  step with the DBL $x$  step that follows it, if a failure occurs in the DBL $x$  step, a restart (if a restart is possible) must begin with the sort step and the input to the step will be resorted. Non-SORTEXT mode will take longer to run but can be restarted after the sort in this case. Therefore, if restart time is more critical than normal runtime do not run the utility as a sortexit.

If the REUSE WORKFILE option is used with SORTEXT, some input workfiles will be used as output files in the same step. Therefore if these two options are used together and a failure occurs, the utility must be restarted from the beginning.

**Workfile Considerations for restarting a failed FASTLOAD:** If the FASTLOAD command fails, depending on the reason for failure, restart the command at the failing step using the "FROM step-name" syntax. You can only restart a step if the input files to that step are intact and valid.

To prepare for a possible restart when running a one-step FASTLOAD, the Intermediate work files should have a disposition that preserves the dataset in the event of an abend, for example, "DISP=(NEW,CATLG,CATLG)."

To restart FASTLOAD at a particular step, the input files to that step must have a disposition to specify that the files already exist, for example, "DISP=OLD."

To determine which files were input to a given step refer to the "Intermediate Work File" tables under "JCL Considerations." Partially created output files should be deleted before restarting the job and the original disposition should be used in the restart job for example, "DISP=(NEW,CATLG,CATLG)."

The SYSPCH file contains sort parameter information for sort steps. It is an output file to IDMSDBL $n$  steps, but is not read unless restarting or running in step mode. So during a normal run the SYSPCH file should be treated as a normal output file, for example, "DISP=(NEW,CATLG,CATLG)." However restarting is not as straightforward. If the previous job failed in an IDMSDBL $x$  step, the SYSPCH file was an output file and should be deleted before restarting. But if the failure occurred in a SORT $x$  step, the contents of the SYSPCH file should contain the same values that were input to the SORT $x$  step. In this case the SYSPCH file should be preserved and defined as a SYS001 input file to the restart step.

When the SORTEXT option is used the SORT $x$  and IDMSDBL $x$  steps are combined. If a failure occurs in this mode the SYSPCH file should normally be preserved and used as a SYS001 input file to the restart. However, there is a small window at the end of a IDMSDBL $x$  step where the SYSPCH file is opened for output and new SORT parameters are written. If the job fails at this point the entire SORT $x$ /IDMSDBL $x$  step must be restarted, but the SYSPCH file will not be valid as a SYS001 input file. In this case the sort parameters must be recreated by hand or the job must be restarted at

an earlier IDMSDBL $x$  step if possible. One way to avoid this situation is to run in step mode when running SORTEXIT mode.

The RELDCTL dataset is always an input file to the first step of a FASTLOAD whether being restarted or not.

**The steps of FASTLOAD:** The FASTLOAD utility consists of the following steps which you can run separately or as a single operation:

Step	Description
SORT1	Sorts the output file from the execution of the format program
IDMSDBL2	<ul style="list-style-type: none"> <li>■ Stores records in the database using the output from SORT1, but does not connect any sets</li> <li>■ Creates an intermediate work file for use by SORT3</li> <li>■ Prints statistics on the records written to the database</li> </ul>
SORT3	Sorts the file produced by IDMSDBL2
IDMSDBL3	<ul style="list-style-type: none"> <li>■ Establishes pointers for each chained set in which each record participates (that is, builds the record prefix) but does not write the prefixes to the database</li> <li>■ Creates an intermediate work file for use by SORT4</li> <li>■ Builds indexes in the database</li> </ul>
SORT4	Sorts the file produced by IDMSDBL3
IDMSDBL4	Inserts the record prefixes by performing a serial sweep of the database

**Each step has input and output:**

Step	Input	Output
SORT1	<ul style="list-style-type: none"> <li>■ SYS001 contains sort control parameters from the SYSPCH file after running the format program</li> <li>■ SYS002 from running the format program</li> </ul>	SYS004 contains the sorted contents of SYS002

Step	Input	Output
IDMSDBL2	<ul style="list-style-type: none"> <li>■ SYS004 from SORT1</li> <li>■ RELDCTL file from the format program; RELDCTL contains control and set information</li> </ul>	<ul style="list-style-type: none"> <li>■ SYSPCH contains sort control parameters</li> <li>■ SYS005 contains set membership information</li> <li>■ SYSLST contains statistics on records written to the database</li> </ul>
<p><b>Note:</b> Overflow statistics may be fewer than expected. In order to improve performance during a fastload, IDMSDBL2 uses the dbkey of the previously stored record as a 'direct' dbkey if the next record to be stored has the same new target page. This reduces the number of overflow conditions.</p>		
SORT3	<ul style="list-style-type: none"> <li>■ SYS001 contains sort control parameters from IDMSDBL2. If running all steps at once, no sort parameters are needed.</li> <li>■ SYS005 from IDMSDBL2</li> </ul>	SYS009 contains the sorted contents of SYS005
IDMSDBL3	<ul style="list-style-type: none"> <li>■ SYS009 from SORT3</li> <li>■ RELDCTL file from the format program; RELDCTL contains control and set information</li> </ul>	<ul style="list-style-type: none"> <li>■ SYSPCH contains sort control parameters</li> <li>■ SYS010 contains pointer descriptors</li> </ul>
SORT4	<ul style="list-style-type: none"> <li>■ SYS001 contains sort control parameters from IDMSDBL3. If running all steps at once, no sort parameters are needed.</li> <li>■ SYS010 from IDMSDBL3</li> </ul>	SYS011 contains the sorted contents of SYS010
IDMSDBL4	<ul style="list-style-type: none"> <li>■ SYS011 from SORT4</li> <li>■ RELDCTL file from the format program; RELDCTL contains control and set information</li> </ul>	

**Note:** This table describes the input and output files as if FASTLOAD were executing without the SORTEXIT and REUSE options. For the effect of these parameters, see "JCL considerations" later in this chapter.

**Sort output after each step:** If you run the FASTLOAD utility a step at a time, you must sort the contents of the intermediate work files. You can use your own sort program or direct the FASTLOAD utility to perform the sorts for you. If you use your own sort program, do not execute the FASTLOAD utility sort steps.

You can use the sort parameters in SYSPCH from the format program, IDMSDBL2, and IDMSDBL3 as the starting point for coding your sort parameters.

Sort the intermediate work files as follows:

Sort name	File to sort	Sort order	Sort on	Begins at
SORT1	SYS002	Descending	16 bytes	Byte 5
SORT3	SYS005	Ascending	16 + (2 x <i>n</i> ) <i>n</i> = the length of the longest sort or CALC key in the subschema	Byte 5
SORT4	SYS010	Ascending	12 bytes	Byte 5

#### The format program:

**Functions:** The format program must perform the following functions for each record to be loaded:

- Builds a record occurrence descriptor
- Builds one owner descriptor for each set in which the record is an automatic member
- Builds one owner descriptor for each set in which the record is a manual member if the record is to be connected to the set at load time
- Calls IDMSDBLU, passing the record occurrence descriptor as the first argument and the owner descriptors as the remaining arguments

**Output:** IDMSDBLU uses the information provided by the format program to create an output file for use as input by the FASTLOAD utility.

**The format program specifies the subschema, SEGMENT, and DMCL:** As part of preparation for FASTLOAD operations, the format program calls IDMSDBLU. On the FIRST CALL ONLY, the subschema, segment name, and the DMCL name must be identified for use by IDMSDBLU.

For example,

```

01 PARMLIST-1.
02 SUBSCHEMA-NAME PIC X(8) VALUE 'EMPSS01'.
02 SEGMENT-NAME   PIC X(8) VALUE 'EMPDEMO'.
02 DMCL-NAME      PIC X(8) VALUE 'EMPDACL'.

CALL 'IDMSDBLU' USING PARMLIST-1.

```

The subschema, segment name, and DMCL must exist and be accessible at the time that the format program passes their names to IDMSDBLU, as well as when they are to be used by the FASTLOAD utility.

**Record descriptors:** Record occurrence descriptors built by the format program must be aligned on a doubleword and must contain the following fields:

Field	Usage	Size	Description
1	Char	18 bytes	The record name of the record being loaded. The format program must initialize this field before calling IDMSDBLU.
2	Char	6 bytes	Binary zeros
3	Binary	4 bytes	Record ID of the record occurrence being loaded. The format program must initialize this field before calling IDMSDBLU.
4	Binary	4 bytes	Suggested page number for storage of the record occurrence being loaded. Responsibility for supplying the page number depends on the location mode of the record: <ul style="list-style-type: none"> <li>▪ If the location mode is either CALC or VIA a CALC record, IDMSDBLU determines and returns the suggested page number.</li> <li>▪ If the location mode is DIRECT, the format program must initialize this field either with an actual page number or with the value -1. In the latter case, IDMSDBLU returns as the suggested page number the number of the first page in the range to which the record is assigned.</li> <li>▪ If the location mode is VIA a VIA record, the format program must process the owner record first and save the suggested page number of the owner record to initialize this field.</li> </ul>
5	Binary	4 bytes	Binary zeros.

Field	Usage	Size	Description
6	Binary	4 bytes	Serial number that uniquely identifies the record occurrence being loaded. IDMSDBLU generates and returns this value.
7	Character	4 bytes	Error status. IDMSDBLU returns error-status codes that parallel those returned by CA-IDMS/DB. The format program must ensure that the returned value is 0000 before proceeding.
8	Character	Size of the largest record to be loaded	Actual record occurrence, left justified. The format program must initialize this field before calling IDMSDBLU.

After the last record is passed to IDMSDBLU, it is necessary to call IDMSDBLU one final time. This call must have the RECORD descriptor with the record id field set to a -1 as its parameter.

**Owner descriptors:** Owner descriptors built by the format program must be aligned on a fullword and must contain the following fields:

Field	Usage	Size	Description
1	Character	16 bytes	Name of the set in which the record being loaded is a member, left justified. The format program must initialize this field before calling IDMSDBLU.
2	Binary	4 bytes	Serial number of the owner record. The format program must initialize this field only if the owner record has a location mode of CALC and duplicates are allowed. In this case, the format program must process the owner record first and save the returned serial number (field 4 of the record occurrence descriptor) to initialize the field.  The format program does not need to initialize this field for system-owned indexes.
3	CALC owner: as defined in the schema	1 through 256 bytes	CALC key of the owner record, left justified. The format program must initialize this field before calling IDMSDBLU.

Field	Usage	Size	Description
	Non-CALC owner: binary	4 bytes	Serial number of the owner record. In this case, the format program must process the owner record first and save the returned serial number (field 4 of the record occurrence descriptor) to initialize the field.  The format program does not need to initialize this field for system-owned indexes.

### 4.9.3 JCL considerations

When you submit a FASTLOAD utility to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- Files containing the areas to be loaded
- The intermediate work files
- Sort space

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

**Intermediate work files:** The following tables indicate which work files are created and read by the different utility steps depending on the use of the SORTEXIT and REUSE WORKFILE options.

Step	Input	Output
FASTLOAD: NOT sortexit mode and NOT reusing workfiles		
SORT1	SYS002	SYS004
IDMSDBL2	SYS004	SYS005
SORT3	SYS005	SYS009
IDMSDBL3	SYS009	SYS010
SORT4	SYS010	SYS011
IDMSDBL4	SYS011	
FASTLOAD: NOT sortexit mode and REUSING workfiles		
SORT1	SYS002	SYS004
IDMSDBL2	SYS004	SYS005
SORT3	SYS005	SYS004
IDMSDBL3	SYS004	SYS005

Step	Input	Output
SORT4	SYS005	SYS004
IDMSDBL4	SYS004	
FASTLOAD: SORTEXTIT mode and NOT reusing workfiles		
SORT1/IDMSDBL2	SYS002	SYS005
SORT3/IDMSDBL3	SYS005	SYS010
SORT4/IDMSDBL4	SYS010	
FASTLOAD: SORTEXTIT mode and REUSING workfiles		
SORT1/IDMSDBL2	SYS002	SYS005
SORT3/IDMSDBL3	SYS005	SYS005
SORT4/IDMSDBL4	SYS005	

**Note:** The RELDCTL file is read in steps IDMSDBL2, IDMSDBL3, and IDMSDBL4.

**Work file JCL Considerations for STEP mode:** FASTLOAD normally runs as a single step but runs as separate steps using the "STEP step-name" syntax. When running in step mode, input files should have dispositions to state that the file already exists, for example, "DISP=OLD."

Preserve output files on successful completion but not when the job fails, for example, "DISP=(NEW,CATLG,DELETE)."

See the "Intermediate work file" table to determine which files are input and which files are output, and when they are used.

The RELDCTL file is always input to every step.

The SYSPCH file is created by an IDMSDBLx step and used as input to a SORTx step. When used as input it is defined as SYS001.

Work file record lengths:

- The RELDCTL file is a fixed length file with a record length of 60 bytes.
- The SYSPCH file is a fixed length file with a record length of 80 bytes.
- All SYS.xxx files are variable length files. The record length can vary from one step to the next, from one job to the next. Do not code an LRECL value in the JCL just code a BLKSIZE value. A BLKSIZE value should be chosen based on the optimal size for the device being used, for example, 1/2 track if disk or 32k if tape.

## 4.9.4 Example

The following example directs the FASTLOAD utility to perform an initial load of a sample CA-IDMS/DB non-SQL defined database.

```
fastload;
```

The following command directs FASTLOAD to run all steps as a sortexit and to reuse workfiles:

```
fastload as sortexit reuse workfiles;
```

## 4.9.5 Sample output

After successful completion of the FASTLOAD utility, the CA-IDMS Batch Command Facility produces the following listing:

```

IDMSBCF                                IDMS Batch Command Facility                                10/02/99  PAGE 1

  SET BATCH WIDTH PAGE 80;
  Status = 0
  FASTLOAD;
  UT010002 BEGINNING PROCESSING FOR STEP SORT1
  UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
  UT009002    12 RECORDS WERE READ FROM SYS002
  UT009003    12 RECORDS WERE WRITTEN TO SYS004
  UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
  UT010003 STEP SORT1    HAS COMPLETED SUCCESSFULLY
  UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL2
  UT005001 IDMSDBL2 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
  UT005002 DATABASE LOAD STATISTICS
  DATABASE LOADED ON 10/02/99 AT 154852
  PAGES READ ..... 4
  PAGES WRITTEN ..... 5
  PAGES REQUESTED ..... 7
  CALC RCDS IN TARGET PAGE . 5
  CALC RCDS OVERFLOWED ..... 1
  VIA RCDS IN TARGET PAGE .. 3
  VIA RCDS OVERFLOWED ..... 0
  LINES REQUESTED BY IDMS .. 26
  RCDS MADE CURRENT OF R/U . 9
  CALLS TO IDMS ..... 13
  FRAGMENTS STORED ..... 0
  RECORDS RELOCATED ..... 0
  UT005003    12 INTERMEDIATE RECORDS WERE WRITTEN TO SYS005
  UT005004    SYS005 RECORD LENGTH IS 44
  UT005005    NO DATABASE ERRORS WERE ENCOUNTERED
  UT005006 IDMSDBL2 RELEASE 15.0 PROCESSING COMPLETED
  UT010003 STEP IDMSDBL2 HAS COMPLETED SUCCESSFULLY
  UT010002 BEGINNING PROCESSING FOR STEP SORT3
  UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
  UT009002    12 RECORDS WERE READ FROM SYS005
  UT009003    12 RECORDS WERE WRITTEN TO SYS009
  UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
  UT010003 STEP SORT3    HAS COMPLETED SUCCESSFULLY

```

## 4.9 FASTLOAD

---

```
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT006007 12 INTERMEDIATE RECORDS WERE READ FROM SYS009
UT006002 12 INTERMEDIATE RECORDS WERE WRITTEN TO SYS010
UT006005 NO DATABASE ERRORS WERE ENCOUNTERED
UT006006 IDMSDBL3 RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBL3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT4
UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
UT009002 12 RECORDS WERE READ FROM SYS010
UT009003 12 RECORDS WERE WRITTEN TO SYS011
UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT4 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL4
UT007001 IDMSDBL4 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT007004 NO DATABASE ERRORS WERE ENCOUNTERED
UT007005 NO LOGIC ERRORS WERE ENCOUNTERED
UT007002 12 RECORDS WERE READ FROM SYS011
UT007006 IDMSDBL4 RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBL4 HAS COMPLETED SUCCESSFULLY
UT010001 DATABASE LOAD HAS COMPLETED SUCCESSFULLY
Status = 0
```

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings

### 4.9.6 For more information

- **On loading a non-SQL defined database**, see *CA-IDMS Database Administration*.

## 4.10 FIX ARCHIVE

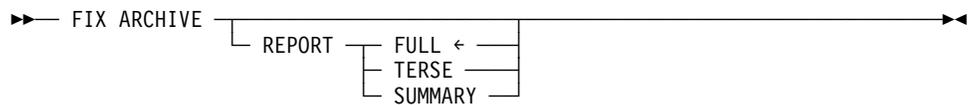
### 4.10.1 Description

**Purpose:** The FIX ARCHIVE utility rewrites a tape journal file. Typically, you rewrite a tape journal file in use at the time of an abnormal system shutdown to make the file usable by the ROLLBACK utility.

**Authorization:**

To	You need this privilege	On
Rewrite a tape journal file	USE	The DMCL

**Syntax**



**Parameters**

**REPORT**

Specifies the amount of detail that is to appear on the report.

**FULL**

Specifies that all details are to be reported. This includes for every transaction: checkpoints, database statistics, and area usage. In addition, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.

**TERSE**

Indicates that only transaction checkpoints and summary information is produced.

**SUMMARY**

Indicates that only final summary information is produced.

### 4.10.2 Usage

**How to submit the FIX ARCHIVE statement:** You submit the FIX ARCHIVE statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**How CA-IDMS/DB rewrites a tape journal file:** When rewriting a tape journal file, CA-IDMS/DB:

- Writes an end-of-file marker on the output tape

- Writes ABRT checkpoints for all run units active at the time of the abnormal termination
- Prints checkpoints and program statistics for run units whose activity is recorded in the journal file
- Identifies quiesce points for use in a rollback or rollforward operation

**Multivolume journal files:** The FIX ARCHIVE statement must process all unarchived volumes of a tape journal file in a single run.

**Archived journal files:** In most cases, you do not need to run the FIX ARCHIVE utility statement against archived journal files. Here are some exceptions:

- To write ABRT checkpoints for incomplete run units that exist on an archived journal tape
- To merge multiple journal tapes onto one tape for certain rollback and rollforward operations that require multiple journal tapes to be one contiguous file

### 4.10.3 JCL considerations

When you submit a FIX ARCHIVE statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The tape journal file to be rewritten which is on SYS001
- The rewritten tape journal file which is on SYS002

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.10.4 Examples

The following statement directs the FIX ARCHIVE statement to rewrite a tape journal file.

```
fix archive;
```

### 4.10.5 Sample output

When the FIX ARCHIVE utility runs successfully, the following listing is produced:

```

IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1
FIX ARCHIVE;
RU_ID      1  PGM_ID EMPLOAD QUIESECE LEVELS 1  UPD 0  BGIN 1999-09-18-15.52.35.481748
RU_ID      1  PGM_ID EMPLOAD QUIESECE LEVELS 0  UPD 0  ENDJ 1999-09-18-15.55.53.333490

STATISTICS FOR EMPLOAD      RU_ID      1

PAGES READ          969  PAGES WRITTEN          847  PAGES REQUESTED          2567  CALC TARGET          186
CALC OVERFLOW       0    VIA TARGET          439  VIA OVERFLOW            0    LINES REQUESTED      6042
RECS CURRENT        1307  CALLS TO IDMS        1461  FRAGMENTS STORED        0    RECS LOCATED         0
LOCKS REQUESTED     0    SELECT LOCKS         0    UPDATE LOCKS            0

START TIME: 1999-09-18-15.52.35.481748  TIME OF LAST COMMIT: NONE                                NUMBER OF COMMITS:      0

TABLESPACES OPENED      BEFORE  AFTER  SINCE LAST COMMIT  USAGE MODE
EMPDEMO.ORG-DEMO-REGION  825    825    825    825  SHARED UPDATE
EMPDEMO.INS-DEMO-REGION  115    115    115    115  SHARED UPDATE
EMPDEMO.EMP-DEMO-REGION  1314   1314   1314   1314  SHARED UPDATE

BLOCK COUNT      702  RECORD COUNT      5512

DATABASE IN QUIESCE AT END OF FILE
DATABASE IN UPDATE QUIESCE AT END OF FILE

ACTIVE PROGRAMS AT STOP TIME WERE:
NONE

DATA BASE MAY NOT NEED TO BE RECOVERED
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings

```

## 4.10.6 For more information

- On journaling, see *CA-IDMS Database Administration*.

## 4.11 FIX PAGE

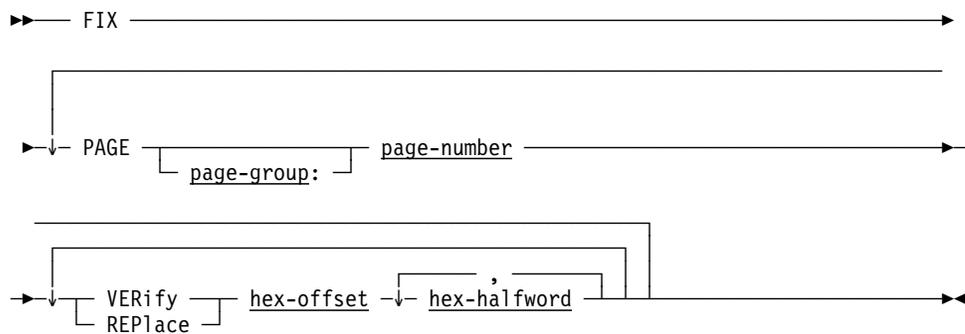
### 4.11.1 Description

**Purpose:** The FIX PAGE utility verifies, and optionally modifies, the contents of a database page.

**Authorization:**

To	You need this privilege	On
Modify a page in an area	DBAWRITE	The area
Verify a page in an area	DBAWRITE	The area

**Syntax**



**Parameters**

**PAGE**

Specifies the page containing the data to be verified or replaced.

**page-group:**

The page group of the area containing the page you want to verify or replace.

If you do not specify a page group, page group zero is used.

**page-number**

The number of a page in an area included in the DMCL module.

**VERify**

Directs the FIX PAGE utility to verify that the data starting at the specified offset is equal to the specified halfwords.

**REPlace**

Directs the FIX PAGE utility to replace the data starting at the specified offset with the specified halfwords.

**hex-offset**

A 4-digit hexadecimal offset from the beginning of the page.

The offset identifies the starting position of the data to be verified or replaced. An offset of 0000 indicates that the data begins with the first byte on the page.

**hex-halfword**

A 4-digit hexadecimal value representing two bytes of data (one halfword) to be used:

- **For comparison to the data on the page**, when *hex-halfword* occurs in a VERIFY parameter
- **As a replacement for data on the page**, when *hex-halfword* occurs in a REPLACE parameter

You can specify up to 19 halfwords in a single VERIFY or REPLACE parameter. Multiple halfwords must be separated by commas.

## 4.11.2 Usage

**How to submit the FIX PAGE statement:** You submit the FIX PAGE statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**If a VERIFY fails:** If you specify VERIFY and REPLACE, CA-IDMS/DB verifies existing data before making any modifications to the contents of a page. If any of the data supplied in the VERIFY parameter does not match the existing contents of the page, data replacements are not made on the page, and the FIX PAGE operation terminates with an error.

**Vary areas offline:** Before executing the FIX PAGE utility, vary the area(s) containing the pages offline to all DC/UCF systems.

## 4.11.3 JCL considerations

When you submit a FIX PAGE statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the files containing the pages to be processed.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.11.4 Examples

**Verifying page contents:** The FIX PAGE statement below requests verification of four bytes of data at the hexadecimal offset 0030 and two bytes of data at the hexadecimal offset 0048 on page 75,003:

```
fix page 75003
  verify 0030 0125,5F0F
  verify 0048 7822;
```

**Replacing data on a single page:** The FIX PAGE statement below verifies and replaces data at one offset on page 75,020. If the data being verified is incorrect, FIX PAGE does not replace any data on the page.

```
fix page 75020
  verify 0066 C1D9,D440
  replace 0066 D3C5,C740;
```

**Replacing data on multiple pages:** The FIX PAGE statement below verifies and replaces data on three pages. FIX PAGE replaces the data on each specified page only if all the data being verified on the page is correct.

```
fix page 224521
  verify 0200 89EE,F2C3
  replace 0200 89E5
page 263942
  verify 00C2 440A,1254,339B
  verify 0110 5B2A,872F,AA23
  replace 00C2 3F24,85D2,1087
  replace 0110 5B24,8733,2842
page 263957
  verify 0124 8924,3258
  replace 0124 0000,3268;
```

### 4.11.5 Sample output

The listing below was generated after successfully replacing data on page 75020 in example two above.

```
IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1
FIX PAGE 75020
VER 0066 C1D9,D440
REP 0066 D3C5,C740;
PAGE 75,020          PAGE GROUP 0          AVAILABLE SPACE 3,952
-000000 0001250C 01250C01 01250C01 0F700000 01250C00 01250C00 01254801 01252D01 *.....*
000020 0125BB01 0124FA02 0125A803 0125A801 01256702 01256701 01250C02 01250C02 *.....Y...Y.*
000040 01250C03 01250C03 01250C01 01250C01 01250317 01250317 F0F4F5F7 C8C1D9D9 *.....0457HARR*
000060 E8404040 404003C5 C7404040 40404040 40404040 40F7F740 E2E4D5E2 C5E340E2 *Y   LEG           77 SUNSET S*
000080 E3D9C9D7 40404040 40D5C1E3 C9C3D240 40404040 40404040 DAC1F0F2 F1F7F840 *TRIP  NATICK      MA02178 *
0000A0 404040F6 F1F7F4F3 F2F0F9F2 F3F0F5F0 F2F8F7F7 F0F1F4F7 F7F7F1F2 F0F1F0F0 * 6174320923050 2877014777120100*
0000C0 F0F0F0F0 F3F4F0F4 F0F50000 01250C01 01250C01 01250C01 0125130F 0125BD05 *0000340405.....*
0000E0 0125BD05 F7F7F1F2 F0F1F7F8 F0F6F0F1 F5F30046 00000C00 7C000C00 0C000000 *...771201780601 53.....@.....*
000100 01250C01 01250C01 01250C01 0125AF05 0125AF03 F0F4F5F8 F0F8F0F8 00000000 *.....04580808....*
000120 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
000140 --SAME--
001080 00000000 00000000 00000000 01A90100 001C0014 01A400CC 00340018 019F0010 *.....Z.. ..U.....*
0010A0 00BC0048 00010004 000C0008 00280000 0001250C *.....*
-
- 1      4      0 0004      75,020-001      75,020-001
- 415    116    1 0010      75,020-000      75,020-000      75,080-001      75,053-001      75,195-001      75,002-002
      75,176-003      75,176-001      75,111-002      75,111-001      75,020-002      75,020-002
      75,020-003      75,020-003      75,020-001      75,020-001      75,011-023      75,011-023
      *0457HARRY LEG           77 SUNSET STRIP NATICK      MA02178      617432092305028*
      *770147771201000000340405..*
- 420    28      2 00CC      75,020-001      75,020-001      75,020-001      75,027-015      75,197-005      75,197-005
      *77120178060153.....@.....*
- 425    8      3 0100      75,020-001      75,020-001      75,020-001      75,183-005      75,183-003
      *04580808*
Status = 0
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings
```

### 4.11.6 For more information

- **On database pages**, see *CA-IDMS Database Administration*.

## 4.12 FORMAT

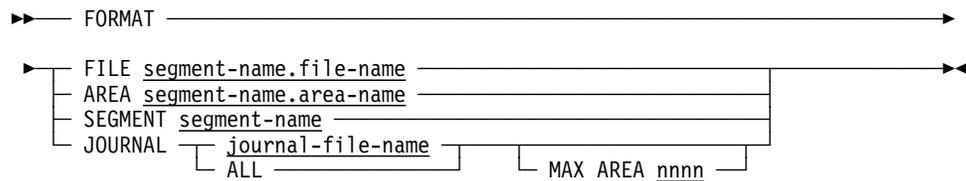
### 4.12.1 Description

**Purpose:** Prepares a database file, area, segment, or a disk journal file for use by CA-IDMS/DB.

**Authorization:**

To FORMAT	You need this privilege	On
An area	DBAWRITE	The area
A segment	DBAWRITE	All areas of the segment
A file	DBAWRITE	All areas that map to the file
A disk journal file	USE	The DMCL

**Syntax**



**Parameters**

**FILE**

Formats the specified file.

**segment-name**

Identifies the segment associated with the file.

**file-name**

The name of a database file described by the DMCL module.

**AREA**

Formats the files associated with the specified area. Only the portion of each file that maps to pages in the area is formatted.

If you are formatting an SQL-defined area, synchronization time stamps are stored for the area.

**Note:** You can use the AREA parameter only when you are reformatting an area that has been formatted at least once before.

**segment-name**

The name of the segment associated with the area.

**area-name**

The name of an area included in the DMCL module.

**SEGMENT**

Formats all files associated with the specified segment.

**segment-name**

The name of a segment included in the DMCL module.

**Note:** If you are formatting a segment of an SQL-defined database, synchronization time stamps for all areas in the segment are also stored.

**JOURNAL**

Formats the specified disk journal file.

**journal-file-name**

The name of a disk journal file included in the DMCL module.

**ALL**

Formats all disk journals in the DMCL.

**MAX AREA nnn**

The maximum number of areas to define for the journal, where *nnn* is an integer from 1 to 32,767. The actual number of areas that CA-IDMS can handle may be higher because of rounding and the size of a journal block.

## 4.12.2 Usage

**How to submit the FORMAT statement:** You submit a FORMAT statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**When to use FORMAT:** You must use the FORMAT utility to prepare:

- Database files before loading any data into the database
- Disk journal files before any journaling to the files occurs

When necessary, you also use the FORMAT statement to *reformat* database files and areas and disk journal files.

**When not to use the FORMAT statement:** Do not try to reformat a file or area that is currently active under a DC/UCF system.

**When formatting SQL-defined segments:** When formatting an area or segment that contains SQL-defined tables, be sure you are connected to the catalog segment where the table definitions reside. You can issue an explicit CONNECT to the DBNAME of the dictionary containing the catalog segment or to the actual segment where the table definitions reside.

**How FORMAT formats a database file or area:** The FORMAT statement formats a database file or area into pages using information contained in the DMCL module. When formatting a database file or area, the FORMAT statement:

- Establishes space management pages (SMPs)
- Initializes the space management entry for each database page
- Establishes a header and footer on each database page
- Sets all data portions of each database page to binary zeros
- Stores synchronization stamps when formatting areas or segments of an SQL-defined database

**Restriction on the AREA parameter:** When formatting a database by file or by segment, a sequential access method (QSAM) is used. When formatting a database by area, a direct access method is used. Because the database must be formatted into blocks by a sequential access method before it can be processed using a direct access method, you can use the AREA parameter of the FORMAT statement only when you are *reformatting* the area.

**Area format depends on the area lock:** To prevent inadvertent formatting of an area that is being updated by another application, the area is locked for the duration of the operation, when formatting by area. If the area is already locked, the format will not take place. If a local mode application abandons while updating an area, the lock may remain on the area. In this case, you can either explicitly unlock the area using the UNLOCK utility, or you can format by file.

**Formatting by segment does not lock areas:** If you format by segment, the FORMAT utility does not lock the areas involved.

**Reformatting an area:** You can reformat a database by file, by area, or by segment. Processing by file is more efficient than processing by area. However, if you format by file in an SQL-defined database, you must run the INSTALL STAMPS utility for each affected area or segment.

**How FORMAT formats a disk journal file:** The FORMAT utility formats a disk journal file into blocks according to the journal file definition in a DMCL module. The FORMAT utility writes a journal header record at the beginning of each block and sets the remainder of the block to binary zeros.

**Formatting disk journal files:** You should format a given disk journal file only once, when the file is first allocated.

**When to use MAX AREA:** Normally when a journal is formatted, CA-IDMS creates a fixed number of JHDA blocks. A JHDA block stores the ready status of areas for warmstart purposes. The size of a journal block and the number of JHDAs limit the number of areas that CA-IDMS can handle.

The new MAX AREA option lets you format a journal that can handle more areas without increasing the size of a journal block. Ideally, the size of a journal block should be optimized to improve runtime efficiency, and should not be affected by the number of areas that may exist.

The MAX AREA option can also reduce the number of default JHDA blocks that CA-IDMS creates, which frees journal space.

**Native VSAM files:** Native VSAM files that are to be accessed by CA-IDMS/DB are not structured like CA-IDMS/DB database files. Do not use the FORMAT statement against native VSAM files.

### 4.12.3 JCL considerations

When you submit a FORMAT statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The database files to be processed (or that map to the areas to be processed)
- The journal files to be processed
- The dictionary containing table definitions, if formatting all or part of an SQL-defined database by area or by segment.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.12.4 Examples

**Formatting a segment:** The illustration below shows a segment that includes three files:

FILE_1	FILE_2	FILE_3
SEGMENT_A		

To format all the files associated with the segment illustrated above, you could use the following FORMAT statement:

```
format segment segment_a;
```

**Reformatting by database file:** The left-hand illustration below shows two areas, each of which maps to a single database file. The right-hand illustration shows a single area that maps to two database files.

FILE_1	FILE_2
AREA_A	AREA_B

FILE_1	FILE_2
AREA_A	

The FORMAT utility can reformat the portions of the database illustrated above either by file or by area. However, because processing by file is more efficient, you would use the following FORMAT statements to do the job in both cases:

```
format file segment_a.file_1;
format file segment_a.file_2;
```

**Reformatting by area:** The left-hand illustration below shows two areas that map to a single database file. The right-hand illustration shows an area that maps to two database files, one of which also contains another area.



To reformat AREA\_A in either illustration above, you must use the following FORMAT statement:

```
format area segment_a.area_a;
```

**Formatting a journal file:** The following FORMAT statement requests formatting of the disk journal named SYSJRNL1:

```
format journal sysjrn1;
```

**Formatting by SEGMENT:** In the following example three database files are formatted for an SQL-defined database using the SEGMENT option of the FORMAT utility.

Note that a CONNECT to the segment or database name containing the table definitions is issued. This is necessary so that the INSTALL STAMPS utility can automatically install area and table stamps during the FORMAT operation.

```
connect to syssql;
format segment userdb;
```

## 4.12.5 Sample output

**Formatting by SEGMENT:** The listing below was generated after the successful completion of the FORMAT SEGMENT USERDB example above.

```

IDMSBCF                                IDMS Batch Command Facility                                09/17/99  PAGE 5
CONNECT TO SYSSQL;
Status = 0
FORMAT SEGMENT USERDB;

File USERDB.EMPF1                      blocks 1 to 50.
Area USERDB.EMP_AREA                   pages 5,001 to 5,050.
Page size in file 4,096.

File USERDB.ORGF1                      blocks 1 to 50.
Area USERDB.ORG_AREA                   pages 5,051 to 5,100.
Page size in file 4,096.

File USERDB.EMPIX                      blocks 1 to 50.
Area USERDB.EMPIX_AREA                 pages 5,101 to 5,150.
Page size in file 2,000.

Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings

```

### 4.12.6 For more information

- **On allocating CA-IDMS/DB files** refer to *CA-IDMS Database Administration*.

## 4.13 INSTALL STAMPS

### 4.13.1 Description

**Purpose:** The INSTALL STAMPS utility stores synchronization stamps in an area of an SQL-defined database that was reformatted by file.

**Authorization:**

To	You need this privilege	On
Install stamps for an area	DBAWRITE	The area

**Syntax**

▶— INSTALL STAMPS INTO AREA segment-name.area-name —▶

**Parameters**

**AREA**

Directs the INSTALL STAMPS utility to install synchronization stamps in an area.

**segment-name**

The name of the segment containing the area.

**area-name**

The name of the area.

### 4.13.2 Usage

**How to submit the INSTALL STAMPS statement:** You submit the INSTALL STAMPS statement to CA-IDMS/DB using either the batch command facility or the online command facility.

**When to use INSTALL STAMPS:** Use the INSTALL STAMPS utility only if the area was reformatted using the file option of the FORMAT utility statement.

**When not to use INSTALL STAMPS:** If you have used the area or segment option of the FORMAT utility to initialize the area, do not run the INSTALL STAMPS utility. The synchronization stamps have already been installed.

Do not use the INSTALL STAMPS utility on areas of non-SQL defined databases.

### 4.13.3 JCL considerations

When you submit a `INSTALL STAMPS` statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The file(s) that map to the area to be processed
- The dictionary that defines the table(s) in the area
- The journal file(s) if you are running CA-IDMS/DB in local mode

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.13.4 Examples

The following example directs the `INSTALL STAMPS` utility to store synchronization stamps in the `EMPLDEMO.EMPLAREA` area.

```
install stamps into area empldemo.emplarea;
```

### 4.13.5 Sample output

After successfully storing synchronization stamps in the `EMPLDEMO.EMPLAREA` area, the following listing is generated.

```
IDMSBCF                                IDMS Batch Command Facility                10/01/99  PAGE 1
CONNECT TO SQLDEMO;
Status = 0
FORMAT FILE EMPLDEMO.EMPF1;

File EMPLDEMO.EMPF1                blocks 1 to 50.
Area EMPLDEMO.EMPLAREA            pages 5,001 to 5,050.
Page size in file 4,096.

Status = 0
INSTALL STAMPS INTO AREA EMPLDEMO.EMPLAREA ;
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

### 4.13.6 For more information

- **On synchronization stamps**, see *CA-IDMS Database Administration* and *CA-IDMS SQL Reference*.

## 4.14 LOAD

### 4.14.1 Description

**Purpose:** The LOAD utility loads data into an SQL-defined database.

The process of loading a database has three major phases:

Phase	Description
Load	Loads the data
Build	Builds indexes and linked indexed referential constraints
Validate	Ensures the validity of referential constraints

Each phase is composed of multiple steps.

You can run the LOAD utility in several ways:

Type of LOAD	What it does
Complete LOAD	Runs all three phases
Phased LOAD	Runs either the LOAD and BUILD phases or just the LOAD phase
Stepped LOAD	Runs the first or second step of the LOAD phase, with intermediate file sorting required between each step

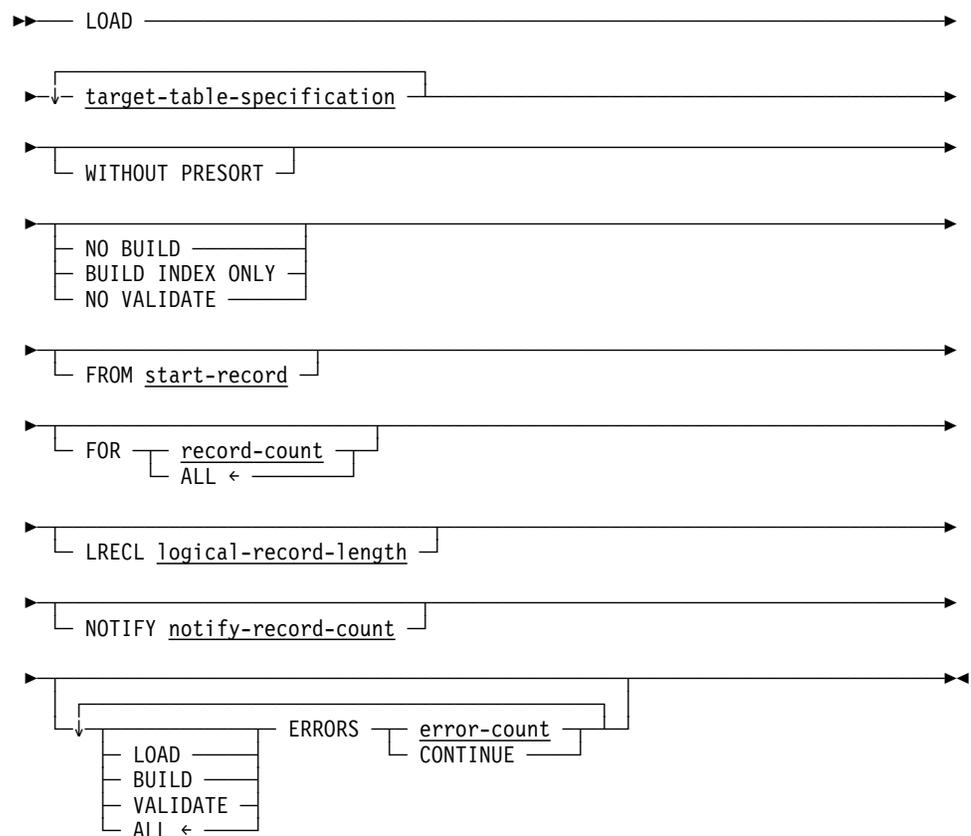
►► For detailed information on the phases and steps involved in loading a database, and help in deciding what kind of load you should run, see *CA-IDMS Database Administration*.

#### Authorization:

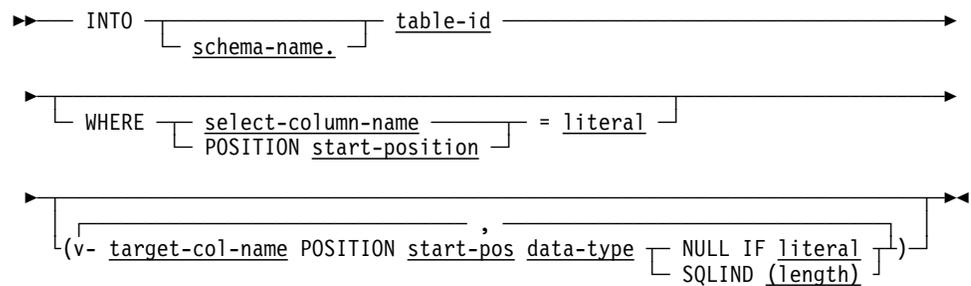
To	You need this privilege	On
Load into a table	INSERT	The table

#### Complete LOAD:

#### Syntax for complete or phased LOAD:



#### Expanded syntax for target table specification:



#### Parameters for complete or phased LOAD

##### target-table-specification

Specifies the tables and columns to be loaded.

See parameter descriptions under "Parameters for target table specification" below.

##### **WITHOUT PRESORT**

Directs the LOAD utility not to sort the input data in SYS001 before starting to load.

By default, if you do not specify **WITHOUT PRESORT**, the LOAD utility will sort the input data into target page sequence before beginning to load.

►► For help in deciding whether to suppress the presort, refer to *CA-IDMS Database Administration*.

### **NO BUILD**

Directs the LOAD utility to perform neither the BUILD nor the VALIDATE phase.

If you specify NO BUILD, you must run the BUILD utility before you can use the table(s).

### **BUILD INDEX ONLY**

Directs the LOAD utility to build indexes but not referential constraints.

If you specify BUILD INDEX ONLY you may have to run the BUILD utility before you can use the table(s).

### **NO VALIDATE**

Directs the LOAD utility to stop before validating referential constraints.

If you specify NO VALIDATE, you will have to run the VALIDATE utility before you can use the table(s).

### **FROM**

Directs the LOAD utility to begin processing input data from a specified record in the input file.

By default, if you do not specify a *start-record*, the LOAD utility will begin with the first record of the input file.

### **start-record**

The number of the first record to process.

### **FOR**

Directs the LOAD utility to stop after processing a specified number of records from the input file.

By default, if you do not specify a FOR option, the LOAD utility will continue until it has processed the last input record.

### **record-count**

The number of records to process before stopping.

### **ALL**

Directs the LOAD utility to continue processing until it has loaded the last record in the input file.

All is the default.

### **LRECL**

Specifies that the SYS001 input records are fixed length records.

By default, if you do not specify LRECL, the LOAD utility assumes that the SYS001 records are variable length.

### **logical-record-length**

The length, in bytes, of the fixed length input records.

**NOTIFY**

Directs the LOAD utility to send a message to the operator whenever a specified number of records are processed.

The message states the phase and step currently being executed and the number of records processed.

By default, the LOAD utility will not notify you of its progress until it is finished.

**notify-record-count**

The number of records to process before sending a message.

**ALL ERRORS**

Directs the LOAD utility either to continue when any errors are detected or stop after a specified number of errors are detected.

By default, the LOAD utility will stop after the first error is detected.

Detected errors are listed in the report generated by LOAD and sent to the SYSLST file.

**LOAD ERRORS**

Directs the LOAD utility either to continue when errors are detected in the LOAD process or stop after a specified number of errors are detected.

By default, the LOAD utility will stop after the first error is detected.

Detected errors are listed in the report generated by the LOAD utility and sent to the SYSLST file.

**BUILD ERRORS**

Directs the LOAD utility either to continue when errors are detected while indexes and referential constraints are being built or stop after a specified number of errors are detected.

By default, the LOAD utility will stop after the first error is detected.

Detected errors are listed in the report generated by the LOAD utility and sent to the SYSLST file.

**VALIDATE ERRORS**

Directs the LOAD utility either to continue when errors are detected in the validation or stop after a specified number of errors are detected.

By default, the LOAD utility will stop after the first error is detected.

Detected errors are listed in the report generated by the LOAD utility and sent to the SYSLST file.

**error-count**

The number of errors to detect before stopping.

**CONTINUE**

Directs the LOAD utility to continue regardless of the number of errors detected.

**Parameters for target table specification**

**INTO**

Specifies a table in which to load data.

**schema-name**

The name of the schema that contains the table.

**table-id**

The identifier of the table.

**WHERE**

Directs the LOAD utility to insert a row into the table from an input record only if a field beginning at a specified position in the input record, or the contents of an input column, equals a given literal value.

**select-column-name**

The name of the column whose contents must equal a literal value.

The column name you specify must also be a column name of the table and must be named in the column list of the target table specification if target column names are used.

**literal**

The value the column must contain.

The data type of the literal must be comparable to that of the column.

►► For information on comparable data types, refer to *CA-IDMS SQL Reference*.

**POSITION start-pos**

The beginning position of the field whose value is to be tested.

**literal**

The value the field must contain. The literal must be a character or hexadecimal literal value.

**target-column-name**

Specifies the columns of the table for which input values are present in the input file.

If no column names are specified, the input file must contain values for all columns of the table; the order, data type, and null indications must exactly match those of the table as defined in the dictionary.

If you specify more than one *target-column-name*, specify them in increasing order; the position of each must be greater than the sum of:

- The position of the one listed before
- *plus*
- The size of the data type for the one listed before

**POSITION**

Specifies the position of the column value in the input record.

**start-pos**

The position of the first byte of the value relative to one.

**data-type**

Specifies the SQL data type of the column value in the input record.

►► For a list of SQL data types and for the SQL standards of data-type specification, refer to *CA-IDMS SQL Reference*.

**NULL IF**

Directs the LOAD utility to substitute a null value if it encounters a specified input value.

**literal**

The value for which a null will be substituted. The data type of the literal must be comparable to the data type of the column.

**SQLIND**

Indicates that a null indicator immediately follows the data value on the input file.

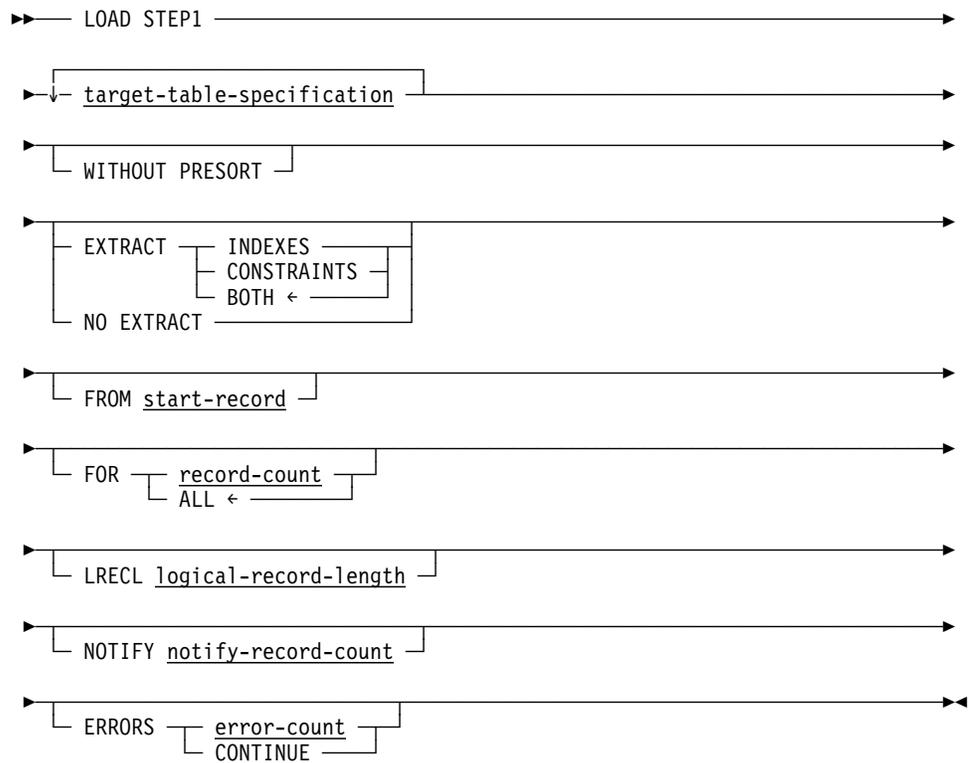
The possible values of the null indicator and their meanings are as follows:

<b>Value</b>	<b>Meaning</b>
Binary zeros	The column value is not null
Binary X'FF's	The column value is null

**(length)**

The length of the indicator. The length must be 1, 2, or 4.

**Stepped LOAD:****Syntax for STEP 1**



### Parameters for STEP 1

#### LOAD STEP1

Directs the LOAD utility to perform only the first step of the load process.

#### target-table-specification

Specifies the tables and columns to be loaded.

For the syntax expansion and parameter descriptions of target-table-specification, see "Complete LOAD" earlier in this chapter.

#### WITHOUT PRESORT

Directs the LOAD utility not to sort the input data before starting to load.

By default, if you do not specify WITHOUT PRESORT, the LOAD utility will sort the input data before beginning to load. The records are sorted so that target pages will be loaded in order.

►► For help in deciding whether to suppress the presort, refer to *CA-IDMS Database Administration*.

If you specify WITHOUT PRESORT, then do not run LOAD STEP2. After LOAD STEP1 is completed, and you have sorted the output of LOAD STEP1, run the BUILD utility.

#### EXTRACT

Directs LOAD to extract information needed for building indexes and/or referential constraints.

By default, if you do not specify otherwise, LOAD STEP1 will extract the information needed for building both indexes and indexed referential constraints. In this case, you can begin BUILD processing with BUILD STEP2.

### INDEXES

Directs the LOAD utility to extract only the information needed for building indexes. Thus the LOAD utility will not extract the information needed for building indexed referential constraints.

### CONSTRAINTS

Directs the LOAD utility to extract only the information needed for building linked indexed referential constraints. Thus, the LOAD utility will not extract the information needed for building indexes.

### BOTH

Directs the LOAD utility to extract the information needed for building both indexes and referential constraints. In this case, when you run the BUILD utility, you can begin with BUILD STEP2.

BOTH is the default.

### NO EXTRACT

Directs the LOAD utility not to extract information needed for building either indexes or referential constraints. In this case, run STEP1 of the BUILD utility to do the extraction.

**Note:** The remaining parameters are identical to the like-named parameters for a complete LOAD presented earlier in this chapter. Refer to this section for parameter descriptions.

### Syntax for STEP 2

```

▶— LOAD STEP2 —————▶
  └──────────────────┘
    NOTIFY notify-record-count

```

### Parameters for STEP 2:

#### LOAD STEP2

Directs the LOAD utility to perform only the second step of the load process.

Execute this only after executing LOAD STEP1 without specifying WITHOUT PRESORT.

**Note:** The remaining parameters are identical to the like-named parameters for a complete LOAD presented earlier in this chapter. Refer to this section for parameter descriptions.

## 4.14.2 Usage

**How to submit the LOAD statement:** You submit the LOAD statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**How LOAD works:** The LOAD utility reads records sequentially from an input file whose external name is SYS001.

The specifications in the syntax tell the LOAD utility how to interpret each record in SYS001 and which table(s) to load. The LRECL parameter controls whether the input file is fixed or variable length.

**When to use LOAD:** Use the LOAD utility to load an SQL-defined database for the first time.

**When not to use LOAD:** Do not use the LOAD utility to load a non-SQL defined database. Instead, use the FASTLOAD utility.

**When to use a phased or stepped LOAD:** Considerations for using the LOAD utility for a phased or stepped load and detailed loading procedures are discussed in *CA-IDMS Database Administration*.

**LOAD utility uses intermediate work files:** Each step of the load process produces intermediate work files to be used by later steps. If you run a complete LOAD without separating steps or phases, the LOAD utility sorts data in the intermediate files between the steps automatically. If you run a phased or stepped LOAD, you must run the intermediate sorts.

**Note:** When running a complete or phased LOAD, SYS002 and SYS003 must point to the same intermediate file. When the database being processed is so large that the intermediate file must be a multi-volume dataset it is required that all extents are physically allocated before jobstep initiation. If this is not possible then a stepped LOAD should be used. When running a stepped LOAD, SYS002 and SYS003 must point to different intermediate files. The data that is output in SYS003 by each step is input to the next step in SYS002.

The following table shows the output of LOAD STEP1 and LOAD STEP2:

Step	Output	Size
STEP1 (WITH PRESORT)	SYS003	For each record:  (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + (MAX RECORD OCCURRENCE SIZE) + 24
	SYSPCH contains sort parameters	80 bytes
STEP1 (WITHOUT PRESORT)	SYS003	For each record:  (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24
	SYSPCH contains sort parameters	80 bytes
STEP2	SYS003	For each record:  (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24

**Sorting intermediate work files:** If you run the load process in steps or phases, use the sort parameters in the SYSPCH file to sort the intermediate files.

**Checking error messages:** Use *CA-IDMS Messages and Codes* to locate messages associated with return codes received from the LOAD utility. Additionally, other useful information about any errors that occurred during LOAD processing are generated on the listing produced by the LOAD utility.

### 4.14.3 JCL considerations

When you submit a LOAD statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The dictionary containing table definitions
- The files containing the areas associated with the tables being loaded
- The intermediate work files
- Sort work files are needed if doing a complete LOAD

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.14.4 Examples

**Phased LOAD:** The sample statement below instructs the LOAD utility to load data from an input file into various tables when a record in the input file at position 60 matches the where clause criteria.

```
load into load.a where position 60 = 'a'
      into load.b where position 60 = 'b'
      into load.h where position 60 = 'h'
no validate lrecl 80;
```

**Complete LOAD:** The example below instructs the LOAD utility to perform a complete load of data from an input file into various tables when a record in the input file at position 60 matches the where clause criteria. In addition, it will load data into two tables when a record in the input file matches the character string associated with the column name in the where clause.

---

```
load into load.c where position 60 = 'C'
      into load.d where position 60 = 'D'
      into load.e where position 60 = 'E'
      into load.f where position 60 = 'F'
      into load.g where position 60 = 'G'
      into load.m where mchar = 'm1'
(mchar   position 1   char(2),
mvchar   position 3   varchar,
mbin     position 7   binary(2),
mhalf    position 9   smallint,
mfull    position 11  integer,
mlong    position 15  longint,
mdec     position 23  dec(2,1),
mdecu    position 25  unsigned dec(3,3),
mnum     position 27  num(2,1),
mnumu    position 29  unsigned numeric(3,3),
mdate    position 32  date,
mtime    position 42  time,
mts      position 50  timestamp)

      into load.m2 where position 1 = 'M2'
(mgraph   position 3   graphic,
mvgraph   position 5   vargraphic(4),
mreal     position 15  real,
mfloat1   position 19  float(4),
mfloat2   position 23  float(50),
mdp       position 31  double precision)
errors continue
lrecl 80;
```

---

### 4.14.5 Sample output

**Complete LOAD:** The report below was generated after executing the LOAD statement in example 1 above.

```

IDMSBCF                                IDMS Batch Command Facility

CONNECT TO SYSCAT;
Status = 0
SET BATCH
  HEADINGS OFF WIDTH PAGE 79 UNDERLINE '-'
  SQLCODE ERROR
  COMPRESS ON;
*DEBUG IDMS OFF

-- **** FORMAT segments      ****

FORMAT SEGMENT USERDB;

File USERDB.EMPF1                blocks 1 to 50.
Area USERDB.EMP_AREA             pages 5,001 to 5,050.
Page size in file 4,096.

File USERDB.ORGF1                blocks 1 to 50.
Area USERDB.ORG_AREA             pages 5,051 to 5,100.
Page size in file 4,096.

File USERDB.EMPIX                blocks 1 to 50.
Area USERDB.EMPIX_AREA           pages 5,101 to 5,150.
Page size in file 2,000.

-- **** Load data into Tables  ****

*DEBUG IDMS ON

LOAD INTO LOAD.A   WHERE POSITION 60 = 'A'

      INTO LOAD.B   WHERE POSITION 60 = 'B'

      INTO LOAD.H   WHERE POSITION 60 = 'H'

NO VALIDATE      LRECL 80;
IDMSLOAD - CAGJF0  PRESORT TABLES FOR LOAD   99-10-29-09.23.53
IDMSLOAD - 1 records processed for table LOAD.H
IDMSLOAD - 1 records processed for table LOAD.B
IDMSLOAD - 1 records processed for table LOAD.A
IDMSLOAD - 7 intermediate records were written to SYS003
IDMSLOAD -   largest SYS003 record size is 136 characters
IDMSLOAD - PRESORT TABLES FOR LOAD   processing completed
IDMSLOAD - CAGJF0  LOAD TABLES AFTER SORT   99-10-29-09.24.01
IDMSLOAD - 1 records processed for table LOAD.A
IDMSLOAD - 1 records processed for table LOAD.B
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_B
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_B
IDMSLOAD - 1 intermediate records for constraint LOAD.AB_S2
IDMSLOAD - 1 records processed for table LOAD.H
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_H
IDMSLOAD - 3 records were stored in the database
IDMSLOAD - 12 intermediate records were written to SYS003
IDMSLOAD -   largest SYS003 record size is 56 characters

```

```

IDMSLOAD - LOAD TABLES AFTER SORT   processing completed
IDMSLOAD - CAGJF0  FILL IN OWNERS DBKEY   99-10-29-09.24.04

IDMSLOAD - 12 intermediate records were written to SYS003
IDMSLOAD -   largest SYS003 record size is 56 characters
IDMSLOAD - FILL IN OWNERS DBKEY   processing completed
IDMSLOAD - CAGJF0  CONNECT UP INDEXES   99-10-29-09.24.10
IDMSLOAD - 9 intermediate records were written to SYS003
IDMSLOAD -   largest SYS003 record size is 56 characters
IDMSLOAD - CONNECT UP INDEXES   processing completed
IDMSLOAD - CAGJF0  BUILD RECORD PREFIX   99-10-29-09.24.12
IDMSLOAD - BUILD RECORD PREFIX   processing completed

```

```

AutoCommit will COMMIT transaction
Command Facility ended with warnings

```

**Complete LOAD specifying column name:** The report below was generated after executing the LOAD statement in example 2 above.

Note that some errors occurred during the load of tables LOAD.M and LOAD.M2. The LOAD utility provides useful information about the type and location of the error. Additional status information can be found in *CA-IDMS Messages and Codes*.

```

IDMSBCF                                IDMS Batch Command Facility

*DEBUG IDMS OFF
CONNECT TO SYSCAT;
Status = 0
SET BATCH
  HEADINGS OFF WIDTH PAGE 79 UNDERLINE '-'
  SQLCODE ERROR
  COMPRESS ON;

-- **** Load data into Tables ****
*DEBUG IDMS ON

LOAD INTO LOAD.C   WHERE POSITION 60 = 'C'

      INTO LOAD.D   WHERE POSITION 60 = 'D'

      INTO LOAD.E   WHERE POSITION 60 = 'E'

      INTO LOAD.F   WHERE POSITION 60 = 'F'

      INTO LOAD.G   WHERE POSITION 60 = 'G'

      INTO LOAD.M WHERE MCHAR = 'M1'
(MCHAR  POSITION 1  CHAR(2),
MVCHAR  POSITION 3  VARCHAR,
MBIN    POSITION 7  BINARY(2),
MHALF   POSITION 9  SMALLINT,
MFULL   POSITION 11 INTEGER,
MLONG   POSITION 15 LONGINT,
MDEC    POSITION 23 DEC(2,1),
MDECU   POSITION 25 UNSIGNED DEC(3,3),
MNUM    POSITION 27 NUM(2,1),
MNUMU   POSITION 29 UNSIGNED NUMERIC(3,3),
MDATE   POSITION 32 DATE,
MTIME   POSITION 42 TIME,
MTS     POSITION 50 TIMESTAMP
)

      INTO LOAD.M2 WHERE POSITION 1 = 'M2'
(MGRAPH  POSITION 3  GRAPHIC,
MVGRAPH  POSITION 5  VARGRAPHIC(4),
MREAL   POSITION 15 REAL,
MFLOAT1  POSITION 19 FLOAT(4),
MFLOAT2  POSITION 23 FLOAT(50),
MDP     POSITION 31 DOUBLE PRECISION
)
  ERRORS CONTINUE
  LRECL 80;

```

```

IDMSLOAD - CAGJF0   PRESORT TABLES FOR LOAD   99-10-29-09.51.27
SQLCODE = -4   Extended reason code = 1026   Messages follow:
DB001026 C-4M322: Data conversion error
IDMSLOAD - error in table LOAD.M - I/P record sequence number 10
IDMSLOAD - error in column 32 - column name is MDATE
ERROR RECORD --> +0   D4F10001 C2F1FFFF 00020000 00020000 M1..B1.....
                  +10  00000000 0002022C 022FF2C2 F0F2F2E7 .....2B022X
                  +20  E7E7E760 E7E760E7 E7E7E74B F3F04BF0 XXX-XX-XXXX.30.0
                  +30  F1F1F3F9 F060F0F1 60F0F260 F1F64BF3 11390-01-02-16.3
                  +40  F04BF0F1 4BF0F0F0 F0F040   0.01.00000
SQLCODE = -4   Extended reason code = 1025   Messages follow:
DB001025 C-4M322: Data exception
IDMSLOAD - error in table LOAD.M2 - I/P record sequence number 16
IDMSLOAD - error in column 5 - column name is MVGRAPH
ERROR RECORD --> +0   D4F2C7F4 0004C7F1 C7F2C7C7 C7C70000 M2G4..G1G2GGGG..
                  +10  00000000 00000000 00000000 00000000 .....
                  +20  00000000 0000   .....
IDMSLOAD - 3 records processed for table LOAD.M2
IDMSLOAD - 3 records processed for table LOAD.M
IDMSLOAD - 1 records processed for table LOAD.G
IDMSLOAD - 1 records processed for table LOAD.F
IDMSLOAD - 1 records processed for table LOAD.E
IDMSLOAD - 1 records processed for table LOAD.D
IDMSLOAD - 1 records processed for table LOAD.C
IDMSLOAD - 19 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 136 characters
IDMSLOAD - PRESORT TABLES FOR LOAD processing completed
IDMSLOAD - CAGJF0   LOAD TABLES AFTER SORT   99-10-29-09.51.46
IDMSLOAD - 1 records processed for table LOAD.C
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_C
IDMSLOAD - 1 intermediate records for constraint LOAD.BC_S2
IDMSLOAD - 1 records processed for table LOAD.D
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_D
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_D
IDMSLOAD - 1 records processed for table LOAD.E
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_E
IDMSLOAD - 1 records processed for table LOAD.F
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_F
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_FB
IDMSLOAD - 1 records processed for table LOAD.G
IDMSLOAD - 3 records processed for table LOAD.M
IDMSLOAD - 3 intermediate records for constraint LOAD.IX_M
IDMSLOAD - 3 records processed for table LOAD.M2
IDMSLOAD - 3 intermediate records for constraint LOAD.IX1_M2
IDMSLOAD - 3 intermediate records for constraint LOAD.IX2_M2
IDMSLOAD - 11 records were stored in the database
IDMSLOAD - 34 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - LOAD TABLES AFTER SORT processing completed
IDMSLOAD - CAGJF0   FILL IN OWNERS DBKEY     99-10-29-09.51.55

```

```
IDMSLOAD - 34 intermediate records were written to SYS003
IDMSLOAD -   largest SYS003 record size is 56 characters
IDMSLOAD -   FILL IN OWNERS DBKEY      processing completed
IDMSLOAD - CAGJF0   CONNECT UP INDEXES  99-10-29-09.52.13
IDMSLOAD - 19 intermediate records were written to SYS003
IDMSLOAD -   largest SYS003 record size is 56 characters
IDMSLOAD -   CONNECT UP INDEXES      processing completed
IDMSLOAD - CAGJF0   BUILD RECORD PREFIX 99-10-29-09.52.24
IDMSLOAD -   BUILD RECORD PREFIX     processing completed
IDMSLOAD - CAGJF0   VALIDATE INDEXES STEP 1 99-10-29-09.52.26
IDMSLOAD - 1 records processed for table LOAD.C
IDMSLOAD - 0 intermediate records for constraint LOAD.BC_S1
IDMSLOAD - 0 intermediate records for constraint LOAD.BC_S2
IDMSLOAD - 0 intermediate records for constraint LOAD.BC_S3
IDMSLOAD - 1 records processed for table LOAD.D
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_DD
IDMSLOAD - 1 intermediate records for constraint LOAD.LOAD_AD
IDMSLOAD - 2 records processed for table LOAD.E
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_AE
IDMSLOAD - 0 records processed for table LOAD.F
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_EF
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_BF
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_HF
IDMSLOAD - 1 records processed for table LOAD.G
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_BG
IDMSLOAD - 0 records processed for table LOAD.M
IDMSLOAD - 0 records processed for table LOAD.M2
IDMSLOAD - 19 intermediate records were written to SYS003
IDMSLOAD -   largest SYS003 record size is 56 characters
IDMSLOAD -   VALIDATE INDEXES STEP 1   processing completed
IDMSLOAD - CAGJF0   VALIDATE INDEXES STEP 2 99-10-29-09.53.24
IDMSLOAD -   VALIDATE INDEXES STEP 2   processing completed
Status = 1      Extended Reason Code = 2991      Messages follow:
DB002991 C1M349: Error detected doing a LOAD/BUILD/VALIDATE statement

AutoCommit will COMMIT transaction

Command Facility ended with warnings
```

## 4.14.6 For more information

- **On procedures for loading a database**, see *CA-IDMS Database Administration*.
- **On SQL data-type specifications**, see *CA-IDMS SQL Reference*.

## 4.15 MAINTAIN INDEX

### 4.15.1 Description

**Purpose:** The MAINTAIN INDEX utility builds, rebuilds or deletes one or more indexes in a non-SQL defined database.

You can run the MAINTAIN INDEX utility all at once or break it into steps.

#### Authorization:

To	You need this privilege	On
Process an index in a segment	DBAWRITE	All areas of the segment

#### Syntax

```

▶— MAINTAIN INDEX —————▶
  ▶— IN SEGMENT segment-name —————▶
  ▶— USING subschema-name —————▶
      └─ NEWSUB new-subschema-name ─┘
  ▶— NOTIFY notify-record-count —————▶
  └─ ───────────────────────────────────▶
      └─ STEP step-name ─┘
        └─ FROM ─┘

  └─ ───────────────────────────────────▶
      └─ BUILD index-name ─┘
        └─ DELETE index-name ─┘
          └─ REBUILD index-name FROM ─┘
              └─ INDEX ─┘
                └─ ALLrows ─┘
                  └─ MEMbers ─┘

  └─ ───────────────────────────────────▶
      └─ as SORTEXIT ─┘
        └─ REUSE workfiles ─┘

```

#### Parameters

##### IN SEGMENT

Specifies the segment containing the index(es) to be processed.

##### segment-name

The name of the segment.

**USING**

Specifies the subschema that defines the index(es).

**Note:** If processing an index associated with an ASF table, specify the name of the table's default subschema, RUnnnnnn where *nnnnnn* is the table's definition number preceded by zeroes.

If processing an ASF index that resides in ASF's definition area, IDMSR-AREA, which specify the IDMSRSSA subschema.

**subschema-name**

The name of the subschema.

If building an index, this is the name of the subschema defining the index to be built.

If deleting an index, this is the name of the subschema defining the index to be deleted.

If rebuilding an index, this is the name of a subschema describing either the existing index structure or the new index structure.

►►For more information, see "Usage" later in this section.

**NEWSUB**

Specifies the subschema that defines one or more indexes to be rebuilt.

Use this clause only when rebuilding an index according to a new definition.

**new-subschema-name**

The name of the new subschema.

**NOTIFY**

A message is sent to the system console after a specified number of records have been processed in the current step.

**notify-record-count**

The number of records to process before sending a message.

By default or if 0 is specified, no message is sent to the system console except the standard message sent at the end of each step indicating the number of records processed during the step.

Notify messages are displayed by steps IDMSTABX, IDMSDBL3, and IDMSDBL4 when a notify-record-count is specified.

**STEP**

Directs the MAINTAIN INDEX utility to execute only one step of the index maintenance process.

By default, if you do not specify STEP, all steps are performed.

If the specified step is any step except IDMSTABX, all other parameters are ignored. In this case, the information normally provided by the other parameters is obtained from the intermediate work files.

**FROM**

Specifies that MAINTAIN INDEX processing should begin at a specified step and complete all remaining steps.

If the specified restart step is any step except IDMSTABX, all other parameters are ignored. In this case, the information normally provided by the other parameters is obtained from the intermediate work files.

**step-name**

The name of the first or only step to execute.

The name must be one of the following:

- IDMSTABX
- SORT3
- IDMSDBL3
- SORT4
- IDMSDBL4

**BUILD**

For system-owned indexes only, directs the MAINTAIN INDEX utility to add all member record occurrences to the specified index.

Both the index and *all* associated areas and files must be defined in the subschema specified in the USING SUBSCHEMA-NAME clause.

**DELETE**

For system-owned indexes only, directs the MAINTAIN INDEX utility to:

- Disconnect all members from the specified index
- Remove the SR7 and SR8 records for the index from the database

**REBUILD**

For system-owned indexes only, directs the MAINTAIN INDEX utility to rebuild an existing, non-empty index.

**index-name**

The name of the index to process.

**FROM**

Identifies the records to use in rebuilding an index.

Use this parameter only with REBUILD.

**INDEX**

Directs the MAINTAIN INDEX utility to connect only members of the existing index to the new index.

**ALLrows**

Directs the MAINTAIN INDEX utility to:

- Sweep the area of the database that contains eligible members
- Connect all eligible member occurrences to the rebuilt index

**MEMbers**

Directs the MAINTAIN INDEX utility to:

- Sweep the area of the database that contains the member record type
- Determine if record occurrences found, participate in the index and if they do, connect them to the rebuilt index
- Connect all member record occurrences to the rebuilt index (this is the same as the ALLROWS option), for unlinked system-owned indexes only

**as SORTEXIT**

Causes each DBLx step in the utility to return its input data directly from the preceding sort instead of having the sort write the data to a workfile. This option eliminates one workfile for each sort and saves the I/O it takes to write then read the workfile.

**REUSE workfiles**

Causes each step in the utility to reuse an existing workfile, if possible, when writing its output data, instead of writing to a new one for each step. This reduces the number of workfiles that need to be allocated.

## 4.15.2 Usage

**How to submit the MAINTAIN INDEX utility:** You submit the MAINTAIN INDEX utility to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

All areas affected by the index must be varied offline.

**When to use MAINTAIN INDEX:** Use the MAINTAIN INDEX utility to process indexes in a non-SQL defined database. This includes indexes associated with individual ASF tables and ASF indexes that reside in the IDMSR-AREA, specify the IDMSRSSA subschema.

**When not to use MAINTAIN INDEX:** To process indexes in an SQL-defined database, use the LOAD or BUILD utility.

**Multiple operations in one execution:** You can perform maintenance on multiple indexes in one execution of the MAINTAIN INDEX utility. However, you should perform only one operation on an index within the same execution. For example, do not DELETE and BUILD the same index at the same time.

**Subschema re-entrancy:** If the subschemas specified in the MAINTAIN INDEX utility reside in a load library, they must not be linked with the reentrant attribute, nor can they reside in the LPA (OS) or SVA (DOS). If the subschemas are loaded from a dictionary load area, these issues are not relevant.

**Mixed Page Groups:** MAINTAIN INDEX cannot process mixed page groups and will issue an error message if mixed page groups are encountered. If your subschema binds to multiple page groups, you must select a subset of areas to process that are all

in the same page group. You must use multiple invocations of the utility to process different page groups.

**Rebuilding indexes:** When rebuilding an index, MAINTAIN INDEX may need a subschema that describes the old index, the new index, or both depending on the changes (if any) being made to the index structure. The following table specifies the REBUILD option and subschema to use based on the function that the rebuild operation is performing:

<b>Function</b>	<b>REBUILD option</b>	<b>Subschema specification</b>
Re-organize an existing index (for example, after deleting many member occurrences)	FROM INDEX	USING <i>old-subschema-name</i>
Rebuild a damaged index	FROM MEMBERS	USING <i>old-subschema-name</i>
Modify the following index tuning options: <ul style="list-style-type: none"> <li>▪ Key compression</li> <li>▪ Number of entries in an SR8 record</li> <li>▪ Index displacement</li> <li>▪ Location of an index (its area or page range)</li> <li>▪ Linked or unlinked attribute</li> </ul>	FROM INDEX	USING <i>old-subschema-name</i> NEW SUB <i>new-subschema-name</i>
Change the following index characteristics: <ul style="list-style-type: none"> <li>▪ Sort key</li> <li>▪ Collating sequence</li> <li>▪ Sorted or unsorted attribute</li> <li>▪ Duplicates option</li> </ul>	FROM MEMBERS	USING <i>new-subschema-name</i>
Rebuild an index from all member occurrences without making index changes	FROM ALLROWS	USING <i>old-subschema-name</i>
Rebuild an index from all member occurrences making index changes	FROM ALLROWS	USING <i>new-subschema-name</i>

Where there is a choice between using FROM INDEX or FROM MEMBERS, consider the following:

- FROM INDEX is generally more efficient because only the index structure is read, rather than every member record occurrence
- FROM INDEX preserves the order of entries with duplicate key values; FROM MEMBERS or FROM ALLROWS, rebuilds the index with duplicate entries in db-key sequence
- FROM INDEX requires that the index exist and be readable

**Changing symbolic index values:** The following values can be supplied as a symbolic index parameter in the physical area definition:

- The number of entries in an SR8 (INDEX BLOCK CONTAINS)
- The number of pages bottom-level SR8s are displaced from top-level SR8s (DISPLACEMENT)

If these values are changed in the physical definition, you can (if desired) re-organize the index to reflect the new values by rebuilding it using a DMCL containing the updated segment definition.

**Changing subarea page range:** The page range in which a system-owned index resides can be specified by using a symbolic subarea parameter in the physical area definition. If this value is changed, execute MAINTAIN INDEX twice:

- The first execution must use a DMCL with the old values for the symbolic parameters and specify STEP IDMSTABX to indicate that only the first step of index maintenance is to be performed
- The second execution must use a new DMCL (with the same name as the old DMCL) and specify FROM SORT3 to indicate that all remaining steps of index maintenance are to be performed

To insure that the correct DMCL is used in each case, you can either change load libraries or rename the new DMCL to have the same name as the old DMCL.

**Adding or removing indexes:** If adding or removing a linked system-owned index or a user-owned index set, you must use RESTRUCTURE SEGMENT to add or remove index and optional owner pointers in the member record.

**Sorted indexes:** Adding, removing, or changing the sort key of a sorted index (system- or user-owned) may change the control length of the record. If it does, and the record is compressed or variable in length, you must also use RESTRUCTURE SEGMENT to adjust the control length of the record in the database.

**Duplicate index entries:** When building an index (or rebuilding an index with the FROM ALLROWS or MEMBERS option), MAINTAIN INDEX stores duplicate index entries in the order in which the corresponding member record occurrences exist in the database (that is, in db-key sequence). When rebuilding an index with the FROM INDEX option, the order of duplicate index entries is maintained in the rebuilt index.

**SORTEXIT and FROM/STEP:** When using the FROM and STEP options with the SORTEXIT option, each pair of SORT $n$  and DBL $x$  steps are considered to be one step. If either half of the SORT $n$ /DBL $x$  is specified on a FROM or STEP option, processing will start with the SORT $n$  step and the DBL $x$  step will also be executed. For example:

- FROM IDMSDBL3 will start with step SORT3 and will continue to the end.
- STEP SORT3 will run steps SORT3 and IDMSDBL3.

**SORTEXIT/REUSE WORKFILE restart considerations:** Since SORTEXIT combines each SORT $n$  step with the DBL $x$  step that follows it, if a failure occurs in the DBL $x$  step, a restart (if a restart is possible) must begin with the sort step and the input to the step will be resorted. Non-SORTEXIT mode will take longer to run but can be restarted after the sort in this case. Therefore, if restart time is more critical than normal runtime do not run the utility as a sortexit.

If the REUSE WORKFILE option is used with SORTEXIT, some input workfiles will be used as output files in the same step. Therefore if these two options are used together and a failure occurs, the utility must be restarted from the beginning.

**Workfile Considerations for restarting a failed MAINTAIN INDEX:** If the MAINTAIN INDEX command fails, depending on the reason for failure, restart the command at the failing step using the "FROM step-name" syntax. You can only restart a step if the input files to that step are intact and valid.

To prepare for a possible restart when running a one-step MAINTAIN INDEX, the Intermediate work files should have a disposition that preserves the dataset in the event of an abend, for example, "DISP=(NEW,CATLG,CATLG)."

To restart MAINTAIN INDEX at a particular step, the input files to that step must have a disposition to specify that the files already exist, for example, "DISP=OLD."

To determine which files were input to a given step refer to the "Intermediate Work File" tables under "JCL Considerations." Partially created output files should be deleted before restarting the job and the original disposition should be used in the restart job for example, "DISP=(NEW,CATLG,CATLG)."

The SYSPCH file contains sort parm information for sort steps. It is an output file to IDMSDBL $n$  steps, but is not read unless restarting or running in step mode. So during a normal run the SYSPCH file should be treated as a normal output file, for example, "DISP=(NEW,CATLG,CATLG)." However restarting is not as straightforward. If the previous job failed in an IDMSDBL $x$  step, the SYSPCH file was an output file and should be deleted before restarting. But if the failure occurred in a SORT $x$  step, the contents of the SYSPCH file should contain the same values that were input to the SORT $x$  step. In this case the SYSPCH file should be preserved and defined as a SYS001 input file to the restart step.

When the SORTEXIT option is used the SORT $x$  and IDMSDBL $x$  steps are combined. If a failure occurs in this mode the SYSPCH file should normally be preserved and

used as a SYS001 input file to the restart. However, there is a small window at the end of a IDMSDBL $x$  step where the SYSPCH file is opened for output and new SORT parameters are written. If the job fails at this point the entire SORT $x$ /IDMSDBL $x$  step must be restarted, but the SYSPCH file will not be valid as a SYS001 input file. In this case the sort parameters must be recreated by hand or the job must be restarted at an earlier IDMSDBL $x$  step if possible. One way to avoid this situation is to run in step mode when running SORTEXIT mode.

The RELDCTL dataset is always an input file to the first step of a MAINTAIN INDEX whether being restarted or not.

**The steps of MAINTAIN INDEX:** The MAINTAIN INDEX utility consists of the following steps which you can run separately or as a single operation:

---

<b>Step</b>	<b>Description</b>
IDMSTABX	<ul style="list-style-type: none"><li>■ Interprets the control statements</li><li>■ Reads the database</li><li>■ Deletes system-owned indexes</li><li>■ Writes index descriptors to the SYS003 file</li><li>■ Writes set descriptors to the RELDCTL file</li><li>■ Writes sort control parameters to the SYSPCH file</li></ul>
SORT3	Sorts the contents of the SYS003 file and puts the results in SYS004.
IDMSDBL3	<ul style="list-style-type: none"><li>■ Erases old index structures, if there are any</li><li>■ Builds the new index structures</li><li>■ Establishes pointers for user-owned index sets, and puts them in the SYS005 file</li><li>■ Establishes index pointers for system-owned linked indexes and puts them in the SYS005 file</li></ul>
SORT4	Sorts the contents of the SYS005 file, and puts the results in SYS006.
IDMSDBL4	Fills in prefix pointers for user-owned indexes and linked system-owned indexes.

---

**Each step has input and output:**

Step	Input	Output
IDMSTABX	<ul style="list-style-type: none"> <li>■ Control parameters from your submission of the utility</li> <li>■ The database</li> </ul>	<ul style="list-style-type: none"> <li>■ SYS003 contains index membership information</li> <li>■ SYSPCH contains sort parameters</li> <li>■ RELDCTL file</li> </ul> <p><b>Note:</b> The SYS003 record length can be calculated as follows: (largest sortkey length in the subschema * 2) + 28</p>
SORT3	<ul style="list-style-type: none"> <li>■ SYS003 from IDMSTABX</li> <li>■ SYS001 contains the sort parameters from IDMSTABX SYSPCH file.</li> </ul> <p>If running in step mode or if it is the first step in a sort, no sort parameters are needed.</p>	SYS004 contains the sorted contents of SYS003
IDMSDBL3	<ul style="list-style-type: none"> <li>■ SYS004 from SORT3</li> <li>■ RELDCTL file</li> </ul>	<ul style="list-style-type: none"> <li>■ SYS005 contains prefix pointer information for user-owned indexes</li> <li>■ SYSPCH contains sort parameters</li> </ul> <p><b>Note:</b> The SYS005 record length is 28.</p>
SORT4	<ul style="list-style-type: none"> <li>■ SYS005 from IDMSDBL3</li> <li>■ SYS001 contains the sort parameters from IDMSDBL3 SYSPCH file.</li> </ul> <p>If running in step mode or if this is the first step in a sort, sort control parameters are not needed.</p>	SYS006 contains the sorted contents of SYS005

Step	Input	Output
IDMSDBL4	<ul style="list-style-type: none"> <li>■ SYS006 from SORT4</li> <li>■ RELDCTL file</li> </ul>	SYSLST contains a summary of the results of the MAINTAIN INDEX operation

**Note:** This table describes the input and output files when executing MAINTAIN INDEX *without* the SORTEXIT and REUSE options. For the impact of running MAINTAIN INDEX with these options, see "JCL considerations" later in this section.

**Sort output after each step:** If you execute the MAINTAIN INDEX utility a step at a time, you must use the sort parameters to sort the contents of the intermediate work files. You can use your own sort program or IDMSSORT.

Sort the intermediate work files as follows:

Sort name	File to sort	Sort order	Sort on	Begins at
SORT3	SYS003	Ascending	16 + (2 x <i>n</i> )  <i>n</i> is the length of the longest sort or CALC key in the subschema	Byte 5
SORT4	SYS005	Ascending	12 bytes	Byte 5

**Note:** If running in step mode, the sort parameters generated by IDMSTABX and IDMSDBL3 are not sufficient for stand alone sort programs running under VSE/ESA. If you want to use your own sort program, you must add "WORK=" parameters to specify more than one sort work file.

### Maintaining indexes

**General procedure:** In general, the procedure for changing indexes is as follows:

#### Step 1: Create a new schema and global subschema, if necessary

**Note:** These steps are not necessary if the only change being made is to the value of symbolic parameters associated with the index.

1. Create a new schema which is identical to the original schema.
2. Create a global subschema for the new schema with a name which is different from that of any other subschema in the dictionary. Include in the subschema all areas, records and sets associated with the schema.
3. Make the necessary changes to the new schema definition.
4. Validate the schema.

5. Regenerate the global subschema.

**Step 2: Modify the segment and DMCL, if necessary**

**Note:** Segment and DMCL modification is necessary only if adding or changing the values of symbolic index or subarea parameters associated with the index or adding a new area in which to store the index.

1. Make the appropriate changes in the segment definition. Make sure that subareas and other symbolics are defined appropriately.
2. Generate, punch, and link all DMCLs containing the altered segment.

**Step 3: Make changes to the index**

1. Backup the area(s).
2. Use the MAINTAIN INDEX utility to change a system-owned index.
3. Use a user-written program in conjunction with IDMSTBLU and the MAINTAIN INDEX utility to change a user-owned index.
4. Verify the change with IDMSDBAN or a retrieval program, CA-OLQ, or CA-CULPRIT.
5. Backup the altered area(s).

**Step 4: Complete the change**

If schema changes were necessary:

1. Update the original schema in the same way that the copy was changed.
2. Regenerate all subschemas associated with the original schema that are affected by the change.
3. Recompile all access modules affected by the change, using the ALTER ACCESS MODULE statement with the REPLACE ALL option.

As appropriate, make the new subschemas, DMCL(s) and area(s) available to your runtime environment.

**Maintaining user-owned indexes:** To build, rebuild, or delete user-owned indexes, you must write a program which calls IDMSTBLU and passes information about the indexes to be operated on and the owner and member record occurrences participating in the indexes.

Once the program executes, complete the operation by executing the MAINTAIN INDEX utility, specifying FROM SORT3. Use the SYSPCH, SYS002 and RELDCTL work files generated by IDMSTBLU as input to the MAINTAIN INDEX utility as SYS001, SYS003, and RELDCTL respectively.

**Creating the user-written program:** The following considerations apply when writing your program:

1. Include in your program the following descriptors:

- The global subschema describing the index
  - The owner and member record descriptions
  - Descriptions of the IDMSTBLU parameters outlined below
2. The general logic of the program should:
- As the first call to IDMSTBLU, pass the subschema descriptor:  
CALL IDMSTBLU USING SUBSCTYP.
  - For each occurrence of a user-owned indexed set, identify the owner by passing an owner descriptor:

CALL IDMSTBLU USING OWNERTYP.

**Note:** If more than one set per owner is to be processed during a single execution of the user-written program, multiple owner descriptors must be passed to IDMSTBLU. For example, to rebuild two indexed sets in which REC-A is owner and REC-B is member, as the second and third calls to IDMSTBLU:

```
CALL IDMSTBLU USING OWNERTYP.  
                :.....+2 REBUILD IXSET-1 REC-A's dbkey  
CALL IDMSTBLU USING OWNERTYP.  
                :.....+2 REBUILD IXSET-2 REC-A's dbkey
```

In this example, subsequent REC-B member descriptors passed to IDMSTBLU should contain 2 occurrences of set name and owner dbkey information (one occurrence for IXSET-1 and one occurrence for IXSET-2).

- For each record that participates as a member of an indexed set to be processed, pass a member descriptor and the member record occurrence:

CALL IDMSTBLU USING MEMBERTYP member-record.

- As the last call to IDMSTBLU, pass the end-of-file descriptor:

CALL IDMSTBLU USING EOFTYP.

The owner and member information can be obtained either from the database or a user input file.

As information is passed to IDMSTBLU, it creates work files needed to build, rebuild, or drop the specified indexed sets.

3. Link edit your program with IDMSTBLU. Also, link it with IDMS if your program binds a rununit.

### IDMSTBLU parameters

**Subschema descriptor:** The subschema descriptor identifies the subschema and segment that contain the indexed sets. If rebuilding an index and changing its characteristics, the identified subschema and segment must describe the new index definition.

The subschema descriptor also identifies the name of the DMCL to be used during MAINTAIN INDEX execution. It must be the same as the name of the DMCL

specified in the SYSIDMS parameter file used to execute the MAINTAIN INDEX utility.

Descriptor type	Fullword binary value 1
Subschema name	8-byte character
Segment name	8-byte character
DMCL name	8-byte character

For example:

```
01 SUBSCTYP.
  02 FILLER PIC S9(8) COMP VALUE +1.
  02 ssnm PIC X(8) VALUE 'EMPSS01'.
  02 segnm PIC X(8) VALUE 'EMPSEG'.
  02 dmc1nm PIC X(8) VALUE 'NEWDMCL'.
```

**Owner descriptor:** The owner descriptor describes each record that participates as an owner in an indexed set to be processed:

Descriptor type	Fullword binary value 2
Requested function	8-character value: 'BUILD ', 'REBUILD ', 'DELETE ', 'EXTEND '
Set name	16-character set name
Owner db-key	Fullword binary owner db-key (zero if owner is an SR7 record)

For example:

```
01 OWNERTYP.
  02 FILLER PIC S9(8) COMP VALUE +2.
  02 OFUNC PIC X(8) VALUE SPACES.
  02 OSET PIC X(16) VALUE SPACES.
  02 ODBKEY PIC S9(8) COMP SYNC VALUE +0.
```

**Member descriptor:** The member descriptor describes each record that participates as a member in one or more indexed sets to be processed:

Descriptor type	Fullword binary value 3
Record name	16-character record name
Record db-key	Fullword binary db-key
Number of sets	Fullword binary value that specifies the number of sets in which the record participates as a member

Repeat the following two fields once for every indexed set in which the record participates as a member:

Set name	16-character set name
Owner db-key	Fullword binary owner db-key value

For example, the following illustrates a member that participates in user-owned indexed sets:

```
01  MEMBRTYP.
    02  FILLER PIC S9(8) COMP VALUE +3.
    02  MREC   PIC X(16) VALUE SPACES.
    02  MDBKEY PIC S9(8) COMP SYNC VALUE +0.
    02  MSETS  PIC S9(8) COMP VALUE +2.
    02  MSET-INFO OCCURS n TIMES.
        04  MSET   PIC X(16).
        04  MODBKEY PIC S9(8) COMP SYNC.
```

**Note:** n represents the number of indexed sets that are actually being processed (i.e., those for which an owner descriptor was previously passed to IDMSTBLU and in which MREC participates as a member).

**Database record descriptor:** The database record descriptor describes each member record type in an indexed set to be processed:

For example:

```
01  COPY IDMS RECORD EMPLOYEE.
01  COPY IDMS RECORD SKILL.
01  COPY IDMS RECORD EXPERTISE.
01  COPY IDMS RECORD JOB.
```

**End-of-file descriptor:** The end-of-file descriptor serves as an end-of-file indicator:

Descriptor type	Fullword binary value -1 (X'FFFFFFFF')
-----------------	----------------------------------------

For example:

```
01  EOFTYP.
    02  FILLER PIC S9(8) COMP VALUE -1.
```

### 4.15.3 JCL considerations

When you submit a MAINTAIN INDEX utility to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The files containing the areas to be processed
- The intermediate work files
- Sort space

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

**Work file JCL Considerations for STEP mode:** MAINTAIN INDEX normally runs as a single step but runs as separate steps using the "STEP step-name" syntax. When running in step mode, input files should have dispositions that state the file already exists, for example, "DISP=OLD."

Preserve output files on successful completion but not when the job fails, for example, "DISP=(NEW,CATLG,DELETE)."

See the "Intermediate work file" table to determine which files are input and which files are output, and when they are used.

The RELDCTL file is always input to every step.

The SYSPCH file is created by an IDMSDBLx step and used as input to a SORTx step. When used as input it is defined as SYS001.

Work file record lengths:

- The RELDCTL file is a fixed length file with a record length of 60 bytes.
- The SYSPCH file is a fixed length file with a record length of 80 bytes.
- All SYSxxx files are variable length files. The record length can vary from one step to the next, from one job to the next. Do not code an LRECL value in the JCL just code a BLKSIZE value. A BLKSIZE value should be chosen based on the optimal size for the device being used, for example, 1/2 track if disk or 32k if tape.

**Intermediate work files:** The following tables indicate which work files are created and read by the different utility steps depending on the use of the SORTEXIT and REUSE WORKFILE options.

Step	Input	Output
MAINTAIN INDEX: NOT sortexit mode and NOT reusing workfiles		
IDMSTABX		SYS003
SORT3	SYS003	SYS004
IDMSDBL3	SYS004	SYS005
SORT4	SYS005	SYS006
IDMSDBL4	SYS006	
MAINTAIN INDEX: NOT sortexit mode and REUSING workfiles		
IDMSTABX		SYS003
SORT3	SYS003	SYS004
IDMSDBL3	SYS004	SYS003
SORT4	SYS003	SYS004
IDMSDBL4	SYS004	
MAINTAIN INDEX: SORTEXTIT mode and NOT reusing workfiles		
IDMSTABX		SYS003
SORT3/IDMSDBL3	SYS003	SYS005
SORT4/IDMSDBL4	SYS005	
MAINTAIN INDEX: SORTEXTIT mode and REUSING workfiles		
IDMSTABX		SYS003
SORT3/IDMSDBL3	SYS003	SYS003
SORT4/IDMSDBL4	SYS003	

#### 4.15.4 Examples

The following example directs the MAINTAIN INDEX utility to rebuild the SKILL-NAME-NDX in the EMPDEMO segment.

```
maintain index in segment empdemo
  using empss01
  rebuild "skill-name-ndx" from members;
```

The following command directs MAINTAIN INDEX to rebuild the SKILL-NAME-NDX, to run all steps as a sortexit, and to reuse workfiles:

```
maintain index in segment empdemo using empss01
as sortexit reuse workfiles
rebuild skill-name-ndx from members;
```

### 4.15.5 Sample output

The following listing is generated after successful completion of the MAINTAIN INDEX utility statement in the above example.

```

IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1

MAINTAIN INDEX IN SEGMENT EMPDEMO USING EMPSS01
REBUILD "SKILL-NAME-NDX" FROM MEMBERS ;
UT010002 BEGINNING PROCESSING FOR STEP IDMSTABX
UT012001 IDMSTABX RELEASE 15.0 TAPE CAGJF0 STARTED
UT012004 69 INTERMEDIATE RECORDS WERE WRITTEN TO SYS003
UT012006 SYS003  MAXIMUM RECORD SIZE IS 78
UT012007 IDMSTABX RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSTABX HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
UT009002 69 RECORDS WERE READ FROM SYS003
UT009003 69 RECORDS WERE WRITTEN TO SYS004
UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT3  HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT006007 69 INTERMEDIATE RECORDS WERE READ FROM SYS004
UT006002 68 INTERMEDIATE RECORDS WERE WRITTEN TO SYS005
UT006005 NO DATABASE ERRORS WERE ENCOUNTERED
UT006006 IDMSDBL3 RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBL3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT4
UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
UT009002 68 RECORDS WERE READ FROM SYS005
UT009003 68 RECORDS WERE WRITTEN TO SYS006
UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT4  HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL4
UT007001 IDMSDBL4 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT007004 NO DATABASE ERRORS WERE ENCOUNTERED
UT007005 NO LOGIC ERRORS WERE ENCOUNTERED
UT007002 68 RECORDS WERE READ FROM SYS006
UT007006 IDMSDBL4 RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBL4 HAS COMPLETED SUCCESSFULLY
UT013001 DATABASE INDEX/ASF TABLE HAS COMPLETED SUCCESSFULLY
Status = 0

```

### 4.15.6 For more information

- On designing indexes, see *CA-IDMS Database Design*.
- On defining and maintaining indexes, see *CA-IDMS Database Administration*.

## 4.16 MERGE ARCHIVE

### 4.16.1 Description

**Purpose:** This utility is used to merge the archived journal files of data sharing group members that are sharing update access to data. It can also be used to merge archive and local mode journal files to simplify a subsequent recovery operation.

The output file created by the merge utility can be used as input to the ROLLFORWARD, ROLLBACK, EXTRACT JOURNAL, and MERGE ARCHIVE utility statements.

The merge utility also produces a report identifying global quiesce points within the set of merged journal images.

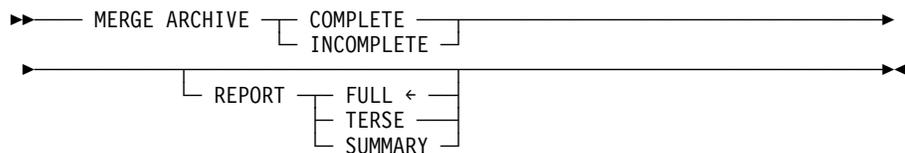
Under certain circumstances, this utility must be used if members of a data sharing group are sharing update access to data.

For more information on when the use of this utility is mandated, refer to Backup and Recovery in *CA-IDMS Database Administration*.

#### Authorization

To	You need this privilege	On
merge archived journal files	USE	The DMCL

#### Syntax



#### Parameters

##### COMPLETE

Indicates that all journal files that contain images needed for recovery have been included as input. ABRT checkpoint records will be added to the merged output file for all transactions still active at the end of the process. The output file may then be used as input to the ROLLBACK, ROLLFORWARD, or EXTRACT JOURNAL utility functions.

**INCOMPLETE**

Indicates that only a subset of the journal files have been included as input. The journal files are merged, but no ABRT checkpoints are generated. The output file may only be used as input to an EXTRACT JOURNAL or a subsequent MERGE ARCHIVE. You should not use the output file as input to ROLLBACK or ROLLFORWARD.

**REPORT**

Specifies the amount of detail that is to appear on the report produced by the merge utility.

**FULL**

Specifies that all details are to be reported. This includes for every transaction: checkpoints, database statistics, and area usage. In addition, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.

**TERSE**

Indicates that only transaction checkpoints and summary information is produced.

**SUMMARY**

Indicates that only final summary information is produced.

## 4.16.2 Usage

**Processing flow:** MERGE ARCHIVE uses two input files: SYS001 and JRNM01 and one output file: SYS002. It proceeds by sorting the contents of SYS001 in chronological sequence and then merging the results of the sort with the contents of JRNM01. The resulting merged journal images are then processed and written to SYS002. Control records are also written to SYS002 indicating the range of images for each member that are included on the merged file and an indication of whether or not the output file was created with the COMPLETE option.

**Input files:** SYS001 is used to supply input that is not in chronological sequence. Typically, archive files from one or more data sharing members are concatenated together as input to SYS001, although it is also possible to include one or more merged files as input. The order of concatenation is not relevant.

JRNM01 is used to supply a single merged archive file. Multiple files cannot be concatenated as input to JRNM01. If no merged archive file exists, then JRNM01 should be specified as dummy.

**Incremental merging:** To minimize recovery time, journal files can be merged periodically and the output from each merge operation used as input to a subsequent MERGE ARCHIVE.

When merging journal files incrementally, specify the INCOMPLETE option on every MERGE ARCHIVE execution except the final one. The final merge operation before executing a ROLLFORWARD or ROLLBACK utility statement must specify the COMPLETE option.

**Using disk files:** It is possible to use disk files for merged journal files. This can be beneficial for incremental merging, since all intermediate merged files can reside on disk. Only the final merged file (the one created with the COMPLETE option) may need to be written to tape so that ROLLFORWARD or ROLLBACK can process it.

**Incomplete transactions:** If a transaction is encountered whose initial checkpoint record (BGIN) is not contained on the input files being processed, a warning message is written that will result in a return code of 4. This is not necessarily an error, since missing journal images can be merged at a later time; however, the missing journal records may need to be provided before the merged file can be used for recovery purposes. For more information, refer to the ROLLFORWARD or ROLLBACK utility commands.

### 4.16.3 JCL considerations

When submitting a MERGE JOURNAL statement through the batch command facility, in addition to the standard JCL required for the batch command facility, you must also include statements to define:

- SYS001 to point to the concatenated set of archived journal files and/or merged journal files
- JRNM01 to point to a single merged journal file. If no such file exists, JRNM01 must be specified as DUMMY
- SYS002 to point to the merged output file. The output file will have the block size that is specified for the archive journal file in the DMCL used for the merge
- Any sort work files needed by your local sort

Refer to the appropriate chapter for your operating system for generic JCL to execute the batch command facility.

### 4.16.4 Example

The following statement directs the MERGE ARCHIVE utility to create a merged output file of all archived journal records and to write ABRT checkpoint records for any transaction still active when all input has been processed.

```
merge archive complete;
```

### 4.16.5 Sample output

The following is output generated after submitting a MERGE ARCHIVE statement to the batch command facility.

```

MERGE ARCHIVE COMPLETE REPORT TERSE ;
NODE SYSTEM72 RU_ID 46 PGM_ID DBCRUPD QUIESCE LEVELS 1 UPD 0 BGIN 2000-03-02-04.34.55.920431 GLBQU
NODE SYSTEM74 RU_ID 42 PGM_ID DBCRUPD QUIESCE LEVELS 1 UPD 0 BGIN 2000-03-02-04.34.55.930616 GLBQU
NODE SYSTEM74 RU_ID 43 PGM_ID DBCRUPD QUIESCE LEVELS 2 UPD 1 BGIN 2000-03-02-04.34.55.932905
NODE SYSTEM74 RU_ID 45 PGM_ID DBCRUPD QUIESCE LEVELS 3 UPD 2 BGIN 2000-03-02-04.34.55.998525
NODE SYSTEM72 RU_ID 46 PGM_ID DBCRUPD QUIESCE LEVELS 0 UPD 0 ENDJ 2000-03-02-04.34.56.785356
NODE SYSTEM72 RU_ID 50 PGM_ID DBCRUPD QUIESCE LEVELS 1 UPD 0 BGIN 2000-03-02-04.34.57.062785
NODE SYSTEM72 RU_ID 51 PGM_ID DBCRUPD QUIESCE LEVELS 2 UPD 1 BGIN 2000-03-02-04.34.57.137212
NODE SYSTEM72 RU_ID 52 PGM_ID DBCRUPD QUIESCE LEVELS 3 UPD 2 BGIN 2000-03-02-04.34.57.148888
NODE SYSTEM72 RU_ID 53 PGM_ID DBCRUPD QUIESCE LEVELS 4 UPD 3 BGIN 2000-03-02-04.34.57.152170
NODE SYSTEM74 RU_ID 42 PGM_ID DBCRUPD QUIESCE LEVELS 2 UPD 2 ENDJ 2000-03-02-04.34.57.206724
NODE SYSTEM73 RU_ID 49 PGM_ID DBCRUPD QUIESCE LEVELS 1 UPD 0 BGIN 2000-03-02-04.34.57.429637
...
NODE SYSTEM73 RU_ID 1499 PGM_ID DBCRUPD QUIESCE LEVELS 6 UPD 6 ABRT 2000-03-02-04.41.20.714545
NODE SYSTEM73 RU_ID 1458 PGM_ID DBCRUPD QUIESCE LEVELS 5 UPD 5 ENDJ 2000-03-02-04.41.20.729313
NODE SYSTEM73 RU_ID 1359 PGM_ID DBCRUPD QUIESCE LEVELS 4 UPD 4 ABRT 2000-03-02-04.41.21.070481
NODE SYSTEM74 RU_ID 1226 PGM_ID DBCRUPD QUIESCE LEVELS 0 UPD 0 ENDJ 2000-03-02-04.41.21.096618
NODE SYSTEM73 RU_ID 1472 PGM_ID DBCRUPD QUIESCE LEVELS 3 UPD 3 ENDJ 2000-03-02-04.41.21.293535
NODE SYSTEM73 RU_ID 1498 PGM_ID DBCRUPD QUIESCE LEVELS 2 UPD 2 ABRT 2000-03-02-04.41.26.665860
NODE SYSTEM73 RU_ID 1448 PGM_ID DBCRUPD QUIESCE LEVELS 1 UPD 1 ABRT 2000-03-02-04.41.26.807159
NODE SYSTEM73 RU_ID 1392 PGM_ID DBCRUPD QUIESCE LEVELS 0 UPD 0 ENDJ 2000-03-02-04.41.26.820245 GLBQU

```

ACTIVE PROGRAMS AT STOP TIME WERE:  
NONE

DATABASE IN QUIESCE AT END OF FILE  
DATABASE IN UPDATE QUIESCE AT END OF FILE

DATA BASE MAY NOT NEED TO BE RECOVERED

SYS001 BLOCK COUNT	40	RECORD COUNT	6194
JRNM01 BLOCK COUNT	342	RECORD COUNT	52235
SYS002 BLOCK COUNT	382	RECORD COUNT	58429

## 4.16.6 For more information

- On manual recovery, refer to *CA-IDMS Database Administration*
- On data sharing, refer to *CA-IDMS System Operations*

## 4.17 PRINT INDEX

### 4.17.1 Description

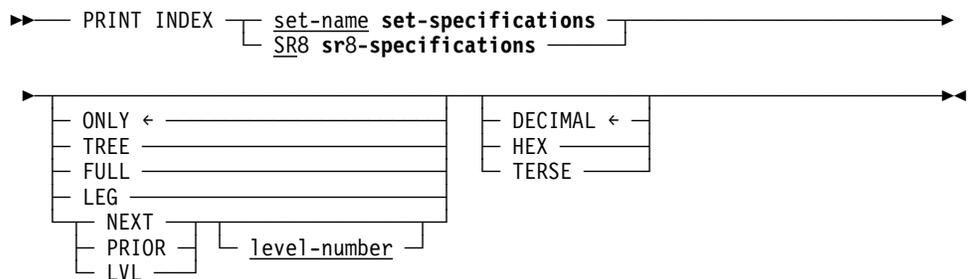
**Purpose:** The PRINT INDEX utility reports on the structure of system-owned indexes and indexed sets. Using the PRINT INDEX utility, you can review:

- The number of levels in an index
- The contents of the fixed and variable portions of one or more SR8 records in an index
- The amount of available space on the page containing each SR8 in an index

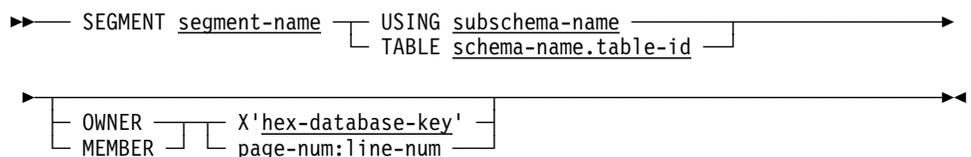
**Authorization:**

To	You need this privilege	On
Report on indexes in a segment	DBAREAD	The area containing the index and the area(s) containing records referenced by the index.

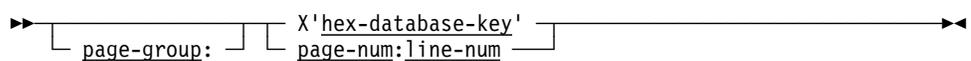
**Syntax**



*Expansion of set-specifications*



*Expansion of SR8-specifications*



**Parameters**

**set-name**

The name of the system-owned index or indexed set on which the PRINT INDEX statement is to report.

When processing a system-owned index, processing begins at the first SR8 record in the SR7-SR8 set.

**SEGMENT**

Specifies the segment containing the index structures to be reported.

**segment-name**

The name of the segment.

**USING subschema-name**

Specifies the name of the subschema in which the named indexed set is included.

**TABLE schema.table-id**

Specifies the name of a table.

**OWNER**

For the USING *subschema-name* clause, directs the PRINT INDEX utility to locate the index through the owner db-key.

By default, if the index is a user owned index, and you do not specify OWNER, all occurrences of the set will be printed.

**MEMBER**

For the USING *subschema-name* clause, directs the PRINT INDEX utility to locate the index through the index pointer in the specified member db-key.

**X'hex-database-key'**

The hexadecimal database key of an owner or member record in the specified indexed set.

**page-num**

The page number of an owner or member record in the specified indexed set.

**line-num**

The line number of an owner or member record in the specified indexed set.

**SR8**

Identifies the index to be processed by specifying an index of an SR8 record in the index.

**page-group**

Identifies the page group of the SEGMENT where the index resides.

**X'database-key'**

The hexadecimal database key of the SR8 record.

**page-num**

The page number of the SR8 record.

**line-num**

The line number of the SR8 record.

**ONLY**

Directs the PRINT INDEX utility to report only on the SR8 record used as the entry point into the index.

ONLY is the default when you do not specify a portion of the index structure to report on.

**TREE**

Directs the PRINT INDEX utility to report on all the SR8 records in the index, starting with the top-level SR8. SR8s are processed by following the next pointers.

**FULL**

Directs the PRINT INDEX utility to report on:

- All the SR8 records in the index, starting with the top-level SR8. SR8s are processed by following the next pointers.
- The database key, index pointer value, and orphan condition of each member record in the index. Member records are processed by walking the bottom level of the index.

**LEG**

Directs the PRINT INDEX utility to report on the SR8 records connected by up pointers, starting with the SR8 used as the entry point into the index.

For an unsorted index or for an entry SR8 that is the top-level SR8 in a sorted index, specifying LEG has the same affect as specifying ONLY.

**NEXT**

Directs the PRINT INDEX utility to report on the SR8 records connected by next pointers in a single level of the index, starting with the SR8 used as an entry point into the index.

**PRIOR**

Directs the PRINT INDEX utility to report on the SR8 records connected by prior pointers in a single level of the index, starting with the SR8 used as an entry point into the index.

**LEVEL**

Directs the PRINT INDEX utility to report on all the SR8 records in a single level of the index.

**level-number**

The index level to report on; an integer in the range 0 through 255.

By default, if you do not specify an index level, the PRINT INDEX utility reports on the SR8s in the level of the SR8 record used as the entry point into the index.

**DECIMAL**

Directs the PRINT INDEX utility to print both the fixed and variable portions of each SR8 record in the report. Symbolic keys in the variable portion of each SR8 are printed in decimal (display) format.

DECIMAL is the default when you do not specify the way in which the contents of the SR8s in the index are to be printed.

**HEX**

Directs the PRINT INDEX utility to print both the fixed and variable portions of each SR8 record in the report. Symbolic keys in the variable portion of each SR8 are printed in hexadecimal format.

**TERSE**

Directs the PRINT INDEX utility to print only the fixed portion of each SR8 record in the report.

## 4.17.2 Usage

**How to submit the PRINT INDEX statement:** You submit the PRINT INDEX statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**When to use PRINT INDEX:** The PRINT INDEX utility can help you determine whether indexes need to be rebuilt. For example, you should consider rebuilding an index when the PRINT INDEX utility report on the index indicates either of the following:

- The number of index levels is greater than was calculated for the original index structure
- Twenty-five percent or more of the member records are orphans

**Hexadecimal display of symbolic keys:** The HEX parameter of the SET/SR8 statement is useful when the symbolic key for the index is a nondisplayable data type, such as binary or packed.

## 4.17.3 JCL considerations

When you submit a PRINT INDEX utility to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The database files that contain the indexes and member records to be accessed

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.17.4 Examples

**Printing an entire index:** The following example directs the PRINT utility to report on the EMP-NAME-NDX using the FULL option.

```
print index "emp-name-ndx" segment empdemo using empss01 full;
```

**Printing the bottom level of an index:** The following example directs the PRINT utility to report on the SKILL-NAME-NDX using the LEVEL and TERSE options.

## 4.17 PRINT INDEX

```
print index "skill-name-ndx" segment empdemo using empss01
member x'00eb0703' level0 terse;
```

**Printing individual SR8 record:** The following example directs the PRINT utility to report on a specific SR8 record.

```
print index sr8 0:60154:2 next 2 hex;
```

**Printing an index from an SQL-defined database:** The following example directs the PRINT utility to report on the DEPT\_EMPL index that is part of the DEMO.DEPT table.

```
print index dept_empl segment userdb table demo.dept full;
```

### 4.17.5 Sample output

**Printing an entire index:** The PRINT INDEX utility generates the following report after successful completion of the statement in the "Printing an entire index" example above.

```
IDMSBCF                                IDMS Batch Command Facility                                09/16/99  PAGE 1
PRINT INDEX "EMP-NAME-NDX" SEGMENT EMPDEMO USING EMPSS01 FULL;
SET=EMP-NAME-NDX  OWNER=SR7              ODBK=00EA6201 SR8 N00EA6203 SR8 P00EA6204 ASC CUSH=33  SYM TKL=24  COMP
                MEMBER=EMPLOYEE
L1  00EA6203  NUME=2                      U=FFFFFFF N=00EA6202 P=00EA6201 RECL=1388  SPA=1384
                00EA6202  JENSEN          JULIE      00EA6204  ZEDI          BETSY
L0  00EA6202  NUME=27  ORPH=19              U=00EA6203 N=00EA6204 P=00EA6203 RECL=668   SPA=1384
                00EA790D  ANDALE          ROY        00EA770A  ANGELO        MICHAEL    00EA8A04  ARM          HARRY
                00EA8201  BANK          MONTE     00EA7901  BLOOMER      JUNE       00EA8705  BOWER       CHARLES
                00EA7D0A  BREEZE        C.        00EA6F01  CLOTH        TERRY     00EA6901  CLOUD       BETH
                00EA8E01  CRANE         HERBERT   00EA8204  CROW         CAROLYN   00EA6D0A  DONOVAN     ALAN
                00EA6C01  DOUGH         JANE      00EA8301  FERNDALE     JANE       00EA6708  FINN        PHINEAS
                00EA6A0A  FITZHUGH     TOM       00EA7401  FONRAD       GEORGE    00EA7C01  GALLWAY     JAMES
                00EA8A01  GARDNER      ROBIN     00EA7601  GARFIELD    JENNIFER  00EA7501  GRANGER     PERCY
                00EA8501  HEAROWITZ    VLADIMIR  00EA7A06  HENDON      HENRIETTA 00EA8F01  HUTTON     EDWARD
                00EA8007  JACKSON      JOCK      00EA6A01  JACOBI       JAMES     00EA7A01  JENSEN     JULIE
                00EA6204  NUME=29      U=00EA6203 N=00EA6201 P=00EA6202 RECL=740   SPA=1384
.
.
.
```

```
MEM 00EA790D          U=00EA6202
     00EA770A          U=00EA6202
     00EA8A04          U=00EA6202
     00EA8201          U=00EA6202
     00EA7901          U=00EA6202
     .
     .
     .
```

```
00EA7D01  *ORPHAN*OF*  U=00EA6202
00EA8A07          U=00EA6204
00EA6B01          U=00EA6204
00EA7914          U=00EA6204
00EA8001  *ORPHAN*OF*  U=00EA6202
00EA8101  *ORPHAN*OF*  U=00EA6202
00EA7D06  *ORPHAN*OF*  U=00EA6202
.
.
.
Status = 0
```



## 4.17 PRINT INDEX

```
IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1
SET BATCH WIDTH PAGE 80 ;
Status = 0
PRINT INDEX DEPT_EMPL SEGMENT USERDB TABLE DEMO.DEPT FULL ;
SET=DEPT_EMPL OWNER=DEPT ODBK=0013C101 SR8 N0013C113
SR8 P0013C113 UNS CUSH=4
MEMBER=EMPL
LO 0013C113 NUME=1 U=FFFFFFFF N=0013C101 P=0013C101
RECL=40 SPA=3048
00139701
MEM 00139701 U=0013C113
SET=DEPT_EMPL OWNER=DEPT ODBK=0013D103 SR8 N0013D104
SR8 P0013D104 UNS CUSH=4
MEMBER=EMPL
LO 0013D104 NUME=1 U=FFFFFFFF N=0013D103 P=0013D103
RECL=40 SPA=3760
0013B801
MEM 0013B801 U=0013D104
SET=DEPT_EMPL OWNER=DEPT ODBK=0013D301 SR8 N0013D319
SR8 P0013D319 UNS CUSH=4
MEMBER=EMPL
LO 0013D319 NUME=2 U=FFFFFFFF N=0013D301 P=0013D301
RECL=44 SPA=2428
0013AE01 0013B501
MEM 0013AE01 U=0013D319
0013B501 U=0013D319
SET=DEPT_EMPL OWNER=DEPT ODBK=0013D319 ***** SET IS EMPT
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

### 4.17.6 For more information

- On designing indexes, see *CA-IDMS Database Design*.
- On defining and maintaining indexes, see *CA-IDMS Database Administration*.

## 4.18 PRINT JOURNAL

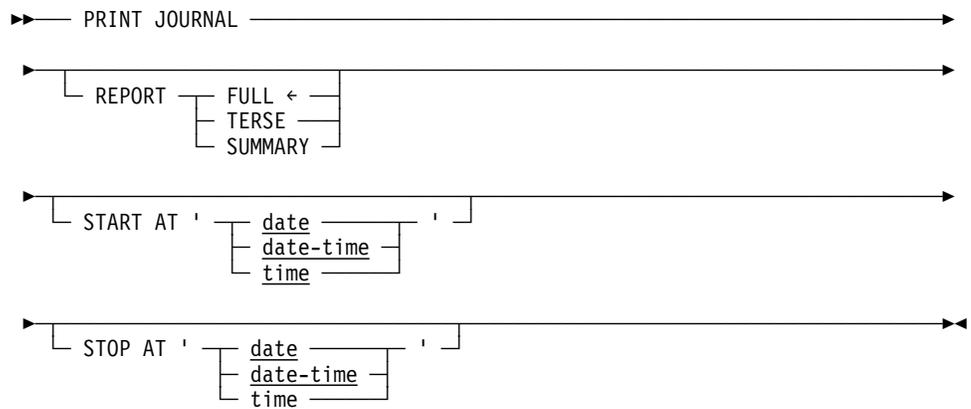
### 4.18.1 Description

**Purpose:** The PRINT JOURNAL utility reports on transaction checkpoints in an archive journal file.

**Authorization:**

To	You need this privilege	On
Report on checkpoints for a journal file	USE	The DMCL associated with the journal file

**Syntax**



**Parameters**

**REPORT**

Specifies the amount of detail that is to appear on the report.

**FULL**

Specifies that all details are to be reported. This includes for every transaction: checkpoints, database statistics, and area usage. In addition, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.

**TERSE**

Indicates that only transaction checkpoints and summary information is produced.

**SUMMARY**

Indicates that only final summary information is produced.

**START AT**

Directs the PRINT JOURNAL utility to report only on checkpoints written for transactions that started on or after the indicated date and time.

By default, if you do not specify a start date or date and time, processing begins with the first checkpoint written in the archive journal file.

**STOP AT**

Directs the PRINT JOURNAL utility to report on checkpoints written only for transactions that started on or before the indicated date and time.

By default, if you do not specify a stop date or date and time, processing ends with the last checkpoint written to the archive journal file.

**date**

Specifies the date, in one of the following formats:

- *yyyy-mm-dd*
- *mm/dd/yyyy*

In these formats, the following rules apply:

- *Yyyy* specifies the year. *Yyyy* must be an integer in the range 0001 through 9999. Leading zeros are optional.
- *Mm* specifies the month within the year. *Mm* must be an integer in the range 01 through 12. Leading zeros are optional.
- *Dd* specifies the day within the month. *Dd* must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of *yyyy*, *mm*, and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

**date-time**

Specifies the date and time, where:

- The format for specifying the date and time are:  
*yyyy-mm-dd-hh.mm.ss.ffffff*
- The rules for specifying the date component of DATE-TIME are the same as for the DATE option described above. Rules for specifying the TIME component are:
  - *Hh* specifies the hour on a 24-hour clock. *Hh* must be an integer in the range 00 through 23. Leading zeros are optional.
  - *Mm* specifies the number of minutes past the hour. *Mm* must be an integer in the range 00 through 59. Leading zeros are optional.
  - *Ss* specifies the number of seconds past the minute. *Ss* must be an integer in the range 00 through 59. Leading zeros are optional.
  - *Fffffff* specifies the number of millionths of a second past the specified second. *Fffffff* is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

**time**

Specifies the time, in the following format:

- *hh:mm:ss*

The rules for specifying *time* are the same as those listed for DATE-TIME above.

When specifying *time*, the date defaults to the current date.

## 4.18.2 Usage

**How to submit the PRINT JOURNAL statement:** You submit the PRINT JOURNAL statement to CA-IDMS/DB only through the batch command facility. When submitting PRINT JOURNAL statements, you must run the batch command facility in local mode.

## 4.18.3 JCL considerations

When you submit a PRINT JOURNAL statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the archive journal file.

▶▶ Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.18.4 Example

The following example directs the PRINT JOURNAL utility to report on all transaction checkpoints in the archive journal file beginning with the first checkpoint.

```
print journal;
```

## 4.18.5 Sample output

The PRINT JOURNAL utility produces the following report.

## 4.18 PRINT JOURNAL

```
IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1

PRINT JOURNAL;
RU_ID      1  PGM_ID EMPLOAD QUIESECE LEVELS 1  UPD  0  BGIN 1999-09-18-15.52.35.481748
RU_ID      1  PGM_ID EMPLOAD QUIESECE LEVELS 0  UPD  0  ENDJ 1999-09-18-15.55.53.333490

STATISTICS FOR EMPLOAD      RU_ID      1

PAGES READ          969  PAGES WRITTEN          847  PAGES REQUESTED      2567  CALC TARGET          186
CALC OVERFLOW       0    VIA TARGET          439  VIA OVERFLOW          0    LINES REQUESTED     6042
RECS CURRENT        1307  CALLS TO IDMS      1461  FRAGMENTS STORED     0    RECS LOCATED        0
LOCKS REQUESTED     0    SELECT LOCKS          0    UPDATE LOCKS          0

START TIME: 1999-09-18-15.52.35.481748  TIME OF LAST COMMIT: NONE                                NUMBER OF COMMITS: 0

TABLESPACES OPENED          BEFORE  AFTER  SINCE LAST COMMIT  BEFORE  AFTER  USAGE MODE
EMPDEMO.ORG-DEMO-REGION      825    825    825    825    SHARED UPDATE
EMPDEMO.INS-DEMO-REGION      115    115    115    115    SHARED UPDATE
EMPDEMO.EMP-DEMO-REGION     1314   1314   1314   1314   SHARED UPDATE

BLOCK COUNT      702  RECORD COUNT    5512

DATABASE IN QUIESCE AT END OF FILE
DATABASE IN UPDATE QUIESCE AT END OF FILE

ACTIVE PROGRAMS AT STOP TIME WERE:
NONE

DATA BASE MAY NOT NEED TO BE RECOVERED
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

### 4.18.6 For more information

- **On journaling and transaction checkpoints**, see *CA-IDMS Database Administration*.

## 4.19 PRINT LOG

### 4.19.1 Description

**Purpose:** The PRINT LOG utility prints all or selected portions of the DC/UCF system log or an archive log file created by the ARCHIVE LOG statement.

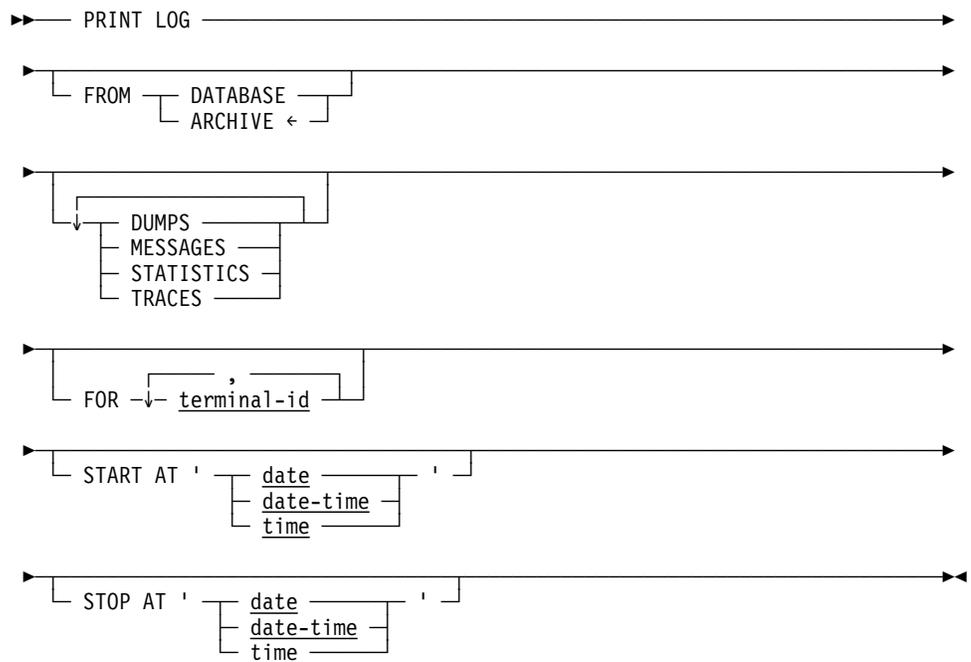
You can select portions for printing based on:

- Type of information
- Logical terminal identifier
- Date and time

**Authorization:**

To	You need this Privilege	On
To print a system or archive log	DBAREAD	The SYSTEM.DDLDCLOG area of the database associated with the DC/UCF system whose log you want to print

**Syntax**



**Parameters**

**FROM**

Specifies whether print log information is to be printed from the SYSTEM.DDLDCLOG area or from an archive log file.

**DATABASE**

Prints log information from the SYSTEM.DDLDCLOG area.

**ARCHIVE**

Prints log information from an archive log file.

**DUMPS**

Includes snap dumps in the printed information.

**MESSAGES**

Includes DC/UCF system messages in the printed information.

**STATISTICS**

Includes DC/UCF system statistics in the printed information.

**TRACES**

Includes user traces in the printed information.

**Note:** If you specify one or more of DUMPS, MESSAGES, STATISTICS, and TRACES, only the specified type of information is printed. If you do not specify any of these options, all types of information are printed.

**FOR**

Prints only log information associated with one or more specified logical or physical terminals.

By default, if you do not specify FOR, log information associated with all logical or physical terminals is printed.

**terminal-id**

- **logical** - The identifier of a logical terminal defined to the DC/UCF system.
- **physical** - The identifier of a physical terminal defined to the DC/UCF system. This parameter will direct the system to select only SYSTEM STATISTICS PTERM records.

You can specify up to 32 logical or physical terminals.

**START AT**

Prints only log information recorded at or after the specified time.

By default, if you do not specify START AT, information from the beginning of the system log or archive log file is printed.

**STOP AT**

Prints only log information recorded at or before the specified time.

By default, if you do not specify STOP AT, all information recorded in the system log or archive log file (starting at the time specified in the START parameter, if any) is printed.

**date**

Specifies the date, in one of the following formats:

- *yyyy-mm-dd*
- *mm/dd/yyyy*

In these formats, the following rules apply:

- **Yyyy** specifies the year. *Yyyy* must be an integer in the range 0001 through 9999. Leading zeros are optional.
- **Mm** specifies the month within the year. *Mm* must be an integer in the range 01 through 12. Leading zeros are optional.
- **Dd** specifies the day within the month. *Dd* must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of *yyyy*, *mm*, and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

**date-time**

Specifies the date and time, where:

- The format for specifying the DATE-TIME is:

*yyyy-mm-dd-hh.mm.ss.ffffff*

- The rules for specifying the DATE component of DATE-TIME are the same as for DATE described above. The rules for specifying the TIME component of DATE-TIME are:
  - **Hh**: specifies the hour on a 24-hour clock. *Hh* must be an integer in the range 00 through 23. Leading zeros are optional.
  - **Mm** specifies the number of minutes past the hour. *Mm* must be an integer in the range 00 through 59. Leading zeros are optional.
  - **Ss** specifies the number of seconds past the minute. *Ss* must be an integer in the range 00 through 59. Leading zeros are optional.
  - **Fffffff** specifies the number of millionths of a second past the specified second. *Fffffff* is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

**time**

Specifies the time in one of the following formats:

- *hh:mm:ss*
- *hh:mm:ss.f*

The rules for specifying TIME are the same as those listed for DATE-TIME above.

When specifying only the TIME option, the date defaults to the current date.

## 4.19.2 Usage

**How to submit the PRINT LOG statement:** You submit a PRINT LOG statement to CA-IDMS/DB only through the batch command facility. You can only run the batch command facility in local mode.

**When to use PRINT LOG:** Use the PRINT LOG utility only when the system log is being written to the DDLDCLOG area of the data dictionary.

**When not to use PRINT LOG:** If the system log is assigned to one or more sequential files, you should use the appropriate operating system utility (for example, IEBGENER for OS/390 systems or DITTO for VSE/ESA systems) to print the contents of the log file.

## 4.19.3 JCL considerations

When you submit a PRINT LOG statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The system log area (DDLDCLOG) if you specify FROM DATABASE
- The archive log file whose contents you want to print if you specify FROM ARCHIVE
- The system message area (DDLDCMSG)
- The dummied journal file

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.19.4 Examples

**Printing everything except dumps:** The PRINT LOG statement below requests printing of all messages, statistics, and user traces currently in the data dictionary log area.

```
print log from database
  messages
  statistics
  traces;
```

**Printing all information for a specified time period:** The PRINT LOG statement below requests printing of all information recorded in the system log from January 18, 1999, at 8:00 p.m. until just before 3:00 a.m. on January 19, 1999. The PRINT LOG utility will retrieve this information from the archive log file.

```
print log from archive
  start at '1999-1-18-20.00.00'
  stop at '1999-1-19-02.59.59.999999';
```

**Printing user traces for a logical terminal:** The PRINT LOG statement below requests printing of all user traces for logical terminal LTVTM05 beginning at 2:30 p.m. on the current day. The PRINT LOG utility will retrieve this information from the data dictionary log area.

```
print log from database
  traces
  for ltvtm05
  start at '14.30.00';
```

## 4.19.5 Sample output

The PRINT LOG report below requests the printing of all information recorded in the system log from September 19, 1999, at 8:10 a.m. until just after 2:20 p.m. on September 19, 1999. The PRINT LOG utility will retrieve this information from the archive log file.

```
IDMSBCF                                IDMS Batch Command Facility                                09/19/99  PAGE 1
PRINT LOG FROM DATABASE START AT '1999-09-19-08.10.00.00000'
STOP AT '1999-09-19-14.20.10.00000';

*** PAGE 000030002 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000031001 STATUS 0000 091799 14.20.00 6 S
*** PAGE 000031500 STATUS 0000 091699 10.19.58 6 S
*** PAGE 000031750 STATUS 0000 091699 14.30.01 6 S
*** PAGE 000031875 STATUS 0000 091699 16.46.34 6 S
*** PAGE 000031937 STATUS 0000 091699 17.54.59 6 S
*** PAGE 000031968 STATUS 0000 091699 18.34.58 6 S
*** PAGE 000031984 STATUS 0000 091699 18.54.59 6 S
*** PAGE 000031992 STATUS 0000 091699 19.04.58 6 S
*** PAGE 000031996 STATUS 0000 091699 19.10.00 6 S
*** PAGE 000031998 STATUS 0000 091699 19.14.58 6 S
*** PAGE 000031999 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000032000 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000030002 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000032000 STATUS 0000 091699 19.14.59 6 S
*** PAGE 000031001 STATUS 0000 091799 14.20.00 6 S
*** PAGE 000031500 STATUS 0000 091699 10.19.58 6 S
*** PAGE 000031250 STATUS 0000 091799 16.02.43 6 S
.
.
.
*** PAGE 000031469 STATUS 0000 091999 09.25.55 6 L
*** PAGE 000031468 STATUS 0000 091999 09.25.12 6 L
*** DDLDCLOG AREA FROM PAGES 0030001 TO 0032000
*** FIRST AND LAST PAGES SELECTED ARE 0031456 AND 0031469
```

## 4.19 PRINT LOG

```
CA-IDMS-DB/DC Print Log Utility          CA-IDMS-DB/DC is a Proprietary Software Product    DATE       TIME       PAGE
CAGJF0      Release 15.0                Licensed from Computer Associates International    09/19/99   09:26:36   2

091999 09.23.57 IDMS DC013002 V72 TO ATTACHING DATABASE RESOURCE CONTROLLER
091999 09.23.57 IDMS DC200131 V72 T1 Lock Manager Initialization Complete
091999 09.23.57 IDMS DC200007 V72 T1 3005 Error Accessing Area - Not Available. SYSDEF.DDLCLSCR
091999 09.23.57 IDMS DC201001 V72 T1 CA-IDMS/DB: 72 Started
091999 09.23.57 IDMS DC200009 V72 T1 CA-IDMS/DB Active 09:23:41 91.262
091999 09.23.57 IDMS DC013003 V72 TO OPENING SYSTEM RUN UNITS
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 TO ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC050001 V72 TO DCLLOG IS 00% FULL
.
.
.
```

```
CA-IDMS-DB/DC Print Log Utility          CA-IDMS-DB/DC is a Proprietary Software Product    DATE       TIME       PAGE
CAGJF0      Release 15.0                Licensed from Computer Associates International    09/19/99   09:26:36   3

091999 09.24.03 IDMS DC021010 V72 LOAD OF RESIDENT MODULE RHDCUX28 FAILED -- CODE 20
091999 09.24.03 IDMS DC021010 V72 LOAD OF RESIDENT MODULE F100D11S FAILED -- CODE 20
091999 09.24.03 IDMS DC013016 V72 TO SECURITY SYSTEM INITIALIZATION BEGINNING
091999 09.24.04 IDMS DC013017 V72 TO SECURITY SYSTEM INITIALIZATION COMPLETION CODE
091999 09.24.04 SYSTEM INITIALIZED ON 91262 AT 09:24:04.35
091999 09.24.04 MAP OF REGION
091999 09.24.04 RHDCOS00 000066E8 IDMSDBIO 00020060 IDMSDBMS 0002F030 OPT      00045870
091999 09.24.04 CSA      00045D70 CCE      0004E560 SCAAREA 00050280 RUA      00050700
091999 09.24.04 DDT      00053340 LTT      00053380 PTT      00055360 QDT      000582A0
091999 09.24.04 TDT      00058420 PDT      0005C8C0 TRCEBUFS 000D5860 TCA      000E5280
091999 09.24.04 DCEAREA 000E52B0 TCEAREA 000E5EB0 MPMODTBL 0013DDC0 ECBLIST 0013E3C0
091999 09.24.04 RCA      0013E640 RLEAREA 0013E6A0 RCEAREA 0014D3E8 DPEAREA 001679C8
091999 09.24.04 ILEAREA 0016CE28 SCT      0016D460 CSVCAREA 0016F200 PGMPOOL 00173000
091999 09.24.04 RENTPOOL 00180000 RHDCD04W 001D2E00 RHDCD05V 001D3400 RHDCD01B 001D6E00
091999 09.24.04 RHDCD06E 001D7C00 RHDCD0ZU 001D9A00 RHDCD0LV 001DBE00 STGPOOL 0045E000
091999 09.24.04 XALODBUF 0065E000 ABENDSTG 00666000 HIADDR  00666320 NLT      00960AA8
.
.
.
```

### 4.19.6 For more information

- On viewing the contents of the DC/UCF system log online, refer to *CA-IDMS System Tasks and Operator Commands*.
- On defining the DC/UCF system log, refer to *CA-IDMS System Generation*.
- On maintaining the DC/UCF system log, refer to *CA-IDMS System Operations*.
- On statistics written to the DC/UCF system log, refer to *CA-IDMS Reports*.



**page-group:**

Specifies the page group from which one or more pages is to be printed. Page group is concatenated with the **start-page** in the form **page-group:start-page**.

By default, if you do not specify a page group, page group zero is used.

**start-page**

The number of the page to print or the first in a range of pages to print.

**TO**

Specifies a range of pages to be printed.

**end-page**

The last page of a range of pages to be printed. **End-page** must be greater than or equal to **start-page**.

**FOR page-count pages**

Specifies that the indicated number of database pages is to be printed.

**Page-count** must be an integer in the range 1 through 32,768.

If you specify a number higher than the remaining number of pages in the area, printing will continue with the first page of the area. Processing will stop when the specified number of pages are printed or when all the pages in the area are printed, whichever comes first. No pages are printed more than once.

By default, if you do not specify TO or FOR, one page is printed.

**IN AREA**

Specifies pages from a specified area to be printed.

By default, if you specify IN AREA, but do not specify SUBAREA, all the pages in the specified area are printed.

**segment**

The name of the segment associated with the area whose pages are to be printed.

**area**

The name of the area whose pages are to be printed.

**SUBAREA**

Specifies that only the pages in a subarea are to be printed.

**subarea**

The name of the subarea whose pages are to be printed.

**FOR page-count pages**

Specifies the indicated number of database pages, starting with the first page of the (sub)area, to be printed.

**Page-count** must be an integer in the range 1 through 32,768.

If you specify a number higher than the number of pages in the (sub)area, processing will stop when all the pages in the (sub)area are printed.

By default, if you do not specify the FOR clause, all pages in the area or subarea are printed.

**FOR CALC key**

Specifies that one or more pages based on CALC keys will be printed.

**calc-options**

Specifies the CALC keys on which to base the selection of pages.

**DISPLAY**

Directs the PRINT PAGE utility to print the contents of the requested database pages in display format only.

By default, if you do not specify DISPLAY or HEX, the contents of the specified pages will be printed in both display and hexadecimal format.

**HEX**

Directs the PRINT PAGE utility to print the contents of the requested database pages in hexadecimal format only.

**BOTH**

Directs the PRINT PAGE utility to print the contents of the requested database pages in both display and hexadecimal format.

BOTH is the default.

**'character-key-value'**

Specifies a CALC key with a character string literal. The target page in the specified area is printed.

**X'hex-key-value'**

Specifies a CALC key with a hexadecimal literal. The target page in the specified area is printed.

**IN AREA**

Identifies the area from which the target page for the specified CALC key is to be printed.

If you specify neither SUBAREA nor CALC in calc-options, the page range of the area specified by IN AREA is used to determine the target page for the specified CALC key.

**segment-name**

The name of the segment associated with the area containing the target page to be printed.

**area-name**

The name of the area that contains the target page to be printed.

**SUBAREA**

Identifies the subarea of the area to be used in determining the target page for the specified CALC key.

**subarea-name**

The name of the subarea.

**CALC**

Identifies a page range of the area to be used in determining the target page for the specified CALC key.

**start-page**

The number of the first page in the page range.

**TO**

Identifies the end of the page range.

**end-page**

The number of the last page in the page range.

**FOR page-count pages**

Specifies the indicated number of database pages to be printed, starting with the target page for the specified CALC key.

**Page-count** must be an integer in the range 1 through 32,768.

If you specify a number higher than the remaining number of pages in the (sub)area, printing will continue with the first page of the (sub)area. Processing will stop when the specified number of pages is printed or when all the pages in the (sub)area are printed, whichever comes first. No pages are printed more than once.

By default, if you do not specify FOR or OVERFLOW, one page is printed.

**OVERFLOW**

Directs the PRINT PAGE utility to print (in addition to the target page) all pages in the specified (sub)area that contain records in the CALC chain of the target page for the specified CALC key due to overflow situations or that contain duplicates of the CALC key.

## 4.20.2 Usage

**How to submit the PRINT PAGE statement:** You submit the PRINT PAGE statement to CA-IDMS/DB only through the batch command facility. When submitting the PRINT PAGE utility, you must run the batch command facility in local mode.

## 4.20.3 JCL considerations

When you submit a PRINT PAGE statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the files containing the pages to be processed.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.20.4 Example

**Print a specific page in an area:** The following example directs PRINT PAGE to print page 75020.

```
print page 75020;
```

## 4.20.5 Sample output

The following report lists the contents of page 75020 in response to the PRINT PAGE statement in the above example.

```

IDMSBCF                                IDMS Batch Command Facility                09/18/99  PAGE 1
  SET BATCH WIDTH PAGE 80;
  Status = 0
  PRINT PAGE 75020;
  PAGE 75,020                PAGE GROUP 0                AVAILABLE SPACE 3,952
-000000  0001250C 01250C01 01250C01 0F700000  *.....*
000010  01250C00 01250C00 01254801 01252D01  *.....*
000020  0125BB01 0124FA02 0125A803 0125A801  *.....Y...Y.*
000030  01256702 01256701 01250C02 01250C02  *.....*
000040  01250C03 01250C03 01250C01 01250C01  *.....*
000050  01250317 01250317 F0F4F5F7 C8C1D9D9  *.....0457HARR*
000060  E8404040 4040C1D9 D4404040 40404040  *Y   ARM   *
000070  40404040 40F7F740 E2E4D5E2 C5E340E2  *   77 SUNSET S*
000080  E3D9C9D7 40404040 40D5C1E3 C9C3D240  *TRIP  NATICK *
000090  40404040 40404040 D4C1F0F2 F1F7F840  *   MA02178 *
0000A0  404040F6 F1F7F4F3 F2F0F9F2 F3F0F5F0  * 6174320923050*
0000B0  F2F8F7F7 F0F1F4F7 F7F7F1F2 F0F1F0F0  *2877014777120100*
0000C0  F0F0F0F0 F3F4F0F4 F0F50000 01250C01  *0000340405.....*
0000D0  01250C01 01250C01 0125130F 0125BD05  *.....*
0000E0  0125BD05 F7F7F1F2 F0F1F7F8 F0F6F0F1  *...771201780601*
0000F0  F5F30046 00000C00 7C000C00 0C000000  *53.....@.....*
000100  01250C01 01250C01 01250C01 0125AF05  *.....*
000110  0125AF03 F0F4F5F8 F0F8F0F8 00000000  *...04580808...*
000120  00000000 00000000 00000000 00000000  *.....*
000130  ---SAME--
001080  00000000 00000000 00000000 01A90100  *.....Z..*
001090  001C0014 01A400CC 00340018 019F0010  *....U.....*
0010A0  00BC0048 00010004 000C0008 00280000  *.....*
0010B0  0001250C
*
-      1      4      0 0004      75,020-001      75,020-001
-    415    116      1 0010      75,020-000      75,020-000      75,080-001
      75,053-001      75,195-001      75,002-002
      75,176-003      75,176-001      75,111-002
      75,111-001      75,020-002      75,020-002
      75,020-003      75,020-003      75,020-001
      75,020-001      75,011-023      75,011-023
      *0457HARRY   ARM   77 SUNSET STRIP   N*
      *ATICK      MA02178   6174320923050287701477712*
      *01000000340405..*
-    420     28     2 00CC      75,020-001      75,020-001      75,020-001
      75,027-015      75,197-005      75,197-005
      *77120178060153.....@.....*
-    425      8     3 0100      75,020-001      75,020-001      75,020-001
      75,183-005      75,183-003
      *04580808*

  Status = 0
  SET BATCH WIDTH PAGE 132;
  Status = 0

```

### 4.20.6 For more information

- **On defining CALC keys**, see *CA-IDMS Database Administration*.
- **On database pages**, see *CA-IDMS Database Administration*.



**area-name**

The name of the area.

**SEGMENT**

Directs the PRINT SPACE utility to report on space utilization in all areas of one or more segments.

**Note:** Native vsam files will be ignored if used in this utility.

**segment-name**

The name of the segment.

**FILE**

Directs the PRINT SPACE utility to report on space utilization for each area or portion of an area contained in the file. This option always produces a full report, whether or not you specify the FULL parameter.

**segment-name**

The name of the segment associated with the file.

**file-name**

The name of the file.

**FULL**

Directs the PRINT SPACE utility to base the space utilization report on information in the header of each page.

By default, if you do not specify FULL, the space utilization report is based on information in the SMPs of the specified areas.

**Note:** If the FULL option is not selected, the space utilization report is based solely on information on the space management pages (SMPs). The space utilization reported may vary widely from the actual space utilization as SMP statistics are not altered until the page referenced is at least 70% full.

## 4.21.2 Usage

**Not using the FULL option:** If the FULL option is not specified, the space utilization report is based on information on the Space Management Pages. The information reported may vary widely from the actual space utilization. For example, if all pages in a particular segment were 50% full, the SMP pages will indicate that each page is 100% available. When using the default on the PRINT SPACE utility, the report will indicate that the database is being used at a 0% level. When running the PRINT SPACE utility with the FULL option, the report will indicate that your database is 50% utilized.

**How to submit the PRINT SPACE statement:** You submit the PRINT SPACE statement to CA-IDMS/DB only through the batch command facility. When submitting PRINT SPACE statements, you must run the batch command facility in local mode.

**Logically-deleted records and reports:** PRINT SPACE BY FILE sequentially reads the files, letting you include only the files in the JCL stream that you want to process.

When you use this option, PRINT SPACE will not report relocated logically-deleted records as logically deleted. These records will be reported as normal records. Therefore, record space utilization reports for an area can produce different results when compared to the file report for the same page range.

### 4.21.3 JCL considerations

When you submit a PRINT SPACE statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the file(s) containing the area(s) to be processed.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.21.4 Examples

#### **PRINT SPACE by area with FULL option**

```
print space for area empdemo.emp-demo-area full;
```

#### **PRINT SPACE by file**

```
print space for file empdemo.empdemo;
```

### 4.21.5 Sample output

The following report is produced by the PRINT SPACE utility after processing the PRINT SPACE statement by area with the FULL option.

4.21 PRINT SPACE

```

IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1
SET BATCH WIDTH PAGE 80;
Status = 0
PRINT SPACE FOR AREA EMPDEMO.EMP-DEMO-AREA FULL;
                                AVAILABLE Space Distribution Report
                                AVAIL    NUMBER
                                SPACE    OF PAGES
AREA      EMPDEMO.EMP-DEMO-AREA
PAGE SIZE      512
PAGES      1,310,001 THRU      1,310,010
                                91-100%      7
                                81-90 %      2
                                71-80 %      0
                                61-70 %      0
                                51-60 %      0
                                41-50 %      0
                                31-40 %      0
                                21-30 %      0
                                11-20 %      0
                                00-10 %      0
                                SMPS      1
                                TOTAL      10
FILE      EMPDEMO.EMPDEMO
PAGES      1,310,001 THRU      1,310,010
BLOCKS      1 THRU      10
Total Space Allocated      5,120
Total Space Available (Percent)      4,128 (80%)
Total Space Used      992
AREA      EMPDEMO.EMP-DEMO-AREA      Distribution of USED Space Report
                                Maximum      Percent of
Record Type      Length      Occurrences      Total Space Used      Total Used
SR1002      40      3      120      12.09
**LD1002      24      3      72      7.25
Space Inv.      480      1      480      48.38
Overhead      32      10      320      32.25
                                *** logically deleted records FOUND ***
Status = 0

```

The following report is produced by the PRINT SPACE utility after processing the PRINT SPACE statement by file.

IDMSBCF	15.0	CA-IDMS Batch Command Facility	10/16/99	PAGE	1
PRINT SPACE FOR FILE EMPDEMO.EMPDEMO;					
AVAILABLE Space Distribution Report					
AREA	EMPDEMO.EMP-DEMO-REGION		AVAIL SPACE		NUMBER OF PAGES
PAGE SIZE	4,276				
PAGES	75,001 THRU	75,100			
			91-100%		78
			81-90 %		13
			71-80 %		7
			61-70 %		0
			51-60 %		0
			41-50 %		0
			31-40 %		1
			21-30 %		0
			11-20 %		0
			00-10 %		0
			SMPS		1
			TOTAL		100
FILE	EMPDEMO.EMPDEMO				
BLOCKS	1 THRU	100			
Total Space Allocated			427,600		
Total Space Available (Percent)			394,120	(92.17%)	
Total Space Used			33,480		
Logically Full Pages			0		
Total Space Unusable (Percent)			0	( 0.00%)	
AREA	EMPDEMO.EMP-DEMO-REGION	Distribution of USED Space Report			
Record Type	Maximum Length	Occurrences	Total Space Used	Percent of Total Used	
SR7	40	1	40	0.11	
SR8	1,396	3	2,820	8.42	
SR415	196	56	10,976	32.78	
SR420	60	68	4,080	12.18	
SR425	36	150	5,400	16.12	
SR460	40	68	2,720	8.12	
Space Inv.	4,244	1	4,244	12.67	
Overhead	32	100	3,200	9.55	
*** NO logically deleted records found ***					
Status = 0	SQLSTATE = 00000				

## 4.21.6 For more information

- On space utilization and database pages, see *CA-IDMS Database Administration*.

## 4.22 PUNCH

### 4.22.1 Description

**Purpose:** The PUNCH utility retrieves the DMCL or database name table load module from the dictionary, and writes them, in object module form, into the SYSPCH file.

**Authorization:**

To punch	You need this privilege	For
A DMCL	USE	The DMCL
A DBTABLE	USE	The DBTABLE

**Syntax**

```

▶— PUNCH —————▶
  └─ DMCL LOAD MODULE dmcl-name —————▶
     └─ DBTABLE LOAD MODULE dbtable-name ──▶
  
```

**Parameters**

**DMCL LOAD MODULE**

Directs the PUNCH utility to punch a DMCL load module from the dictionary.

**dmcl-name**

The load module name of the DMCL to be punched.

**DBTABLE LOAD MODULE**

Directs the PUNCH utility to punch a database name table load module from the dictionary.

**dbtable-name**

The load module name of the DBTABLE to be punched.

### 4.22.2 Usage

**How to submit the PUNCH statement:** You submit the PUNCH statement to CA-IDMS/DB only through the batch command facility.

### 4.22.3 JCL considerations

When you submit a PUNCH statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The dictionary from which the load module is to be punched (local mode only).
- The journal file(s) associated with the DMCL (local mode only). These can be dummied out.
- SYSPCH file.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.22.4 Example

The following example directs the PUNCH utility to retrieve the IDMSDMCL DMCL definition from the dictionary and write it, in object module form, to a SYSPCH file.

```
punch dmcl load module idmsdmcl;
```

### 4.22.5 Output

The CA-IDMS Batch Command Facility returns the following listing after successful completion of the PUNCH utility.

```
PUNCH DMCL LOAD MODULE IDMSDMCL;  
Status = 0
```

### 4.22.6 For more information

- **On defining DMCL and DBTABLE modules**, see *CA-IDMS Database Administration*.
- **On deleting DMCL and DBTABLE load modules**, see *CA-IDMS Database Administration*.

## 4.23 RELOAD

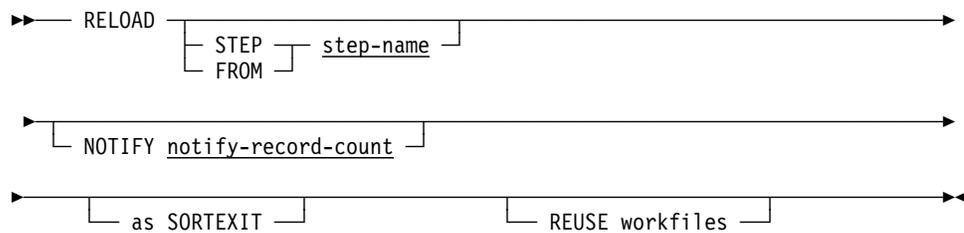
### 4.23.1 Description

**Purpose:** The RELOAD utility reloads a CA-IDMS/DB database using input created by the UNLOAD utility statement.

**Authorization:**

To	You need this privilege	On
Reload data into a segment	DBAWRITE	All areas associated with the segment

**Syntax**



**Parameters**

**STEP**

Specifies that only one step of the reload operation should be executed.

If you do not specify STEP, all steps in the reload process are performed as a single operation.

**step-name**

The name of the step to execute.

The name must be one of the following:

- SORT1
- IDMSDBL2
- SORT2
- IDMSDBLX
- SORT3
- IDMSDBL3
- SORT4

- IDMSDBL4

**FROM**

Specifies that RELOAD processing should begin at a specified step and complete all remaining steps.

If you specify FROM, all remaining steps are treated together as a single operation.

**step-name**

The name of the step to execute.

The name must be one of the following:

- SORT1
- IDMSDBL2
- SORT2
- IDMSDBLX
- SORT3
- IDMSDBL3
- SORT4
- IDMSDBL4

**Note:** See "Usage" below for an explanation of when you can restart steps IDMSDBL2, IDMSDBL3, or IDMSDBL4.

**NOTIFY**

A message is sent to the system console after a specified number of records have been processed in the current step.

**notify-record-count**

The number of records to process before sending a message.

By default or if 0 is specified, no message is sent to the system console except the standard message sent at the end of each step indicating the number of records processed during the step.

Notify messages are only displayed by steps IDMSDBL2, IDMSDBL3, and IDMSDBL4 when a NOTIFY-RECORD-COUNT is specified.

**as SORTEXIT**

Causes each DBLx step in the utility to return its input data directly from the preceding sort instead of having the sort write the data to a workfile. This option eliminates one workfile for each sort and saves the I/O it takes to write then read the workfile.

**REUSE workfiles**

Causes each step in the utility to reuse an existing workfile, if possible, when writing its output data, instead of writing to a new one for each step. This reduces the number of workfiles that need to be allocated.

## 4.23.2 Usage

**How to submit the RELOAD utility:** You submit the RELOAD utility to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**When to use RELOAD:** Use the RELOAD utility to reload a CA-IDMS/DB database that has been unloaded by UNLOAD.

When reloading an SQL-defined database, the database files must be formatted using the FILE option of the FORMAT utility statement so that AREA and TABLE stamps are properly reloaded.

**When not to use RELOAD:** Do not use the RELOAD utility to load a new non-SQL defined database; use the FASTLOAD utility.

Do not use the RELOAD utility to load a new SQL-defined database; use the LOAD utility.

**Run RELOAD all at once or in steps:** The reloading process has eight steps, which you can run one at a time or all together. Each step generates output for use by the next step. Four of the steps are sorts to prepare data for use by the following step.

You can run each step separately, in which case you can use your own sorting program. Alternatively, you can direct the RELOAD utility to do the sorting for you. If you run the steps as a single process, the RELOAD utility will do the sorting for you automatically.

**When to run RELOAD in steps:** The most common reason to run the RELOAD utility in steps is to cut the work into pieces, each piece requiring less time to run than the whole process.

You may also decide to run the RELOAD utility in steps in order to use your own sorting programs between the steps or you can run the sort steps on a different machine than that holding the database.

**Mixed Page Groups** RELOAD cannot process mixed page groups and will issue an error message if mixed page groups are encountered. You must use multiple invocations of the utility to process different page groups.

**RELOAD and ASF databases:** If the RELOAD utility is to be run against an ASF data or definition area, refer to the *CA-IDMS ASF User Guide* for more information.

**Restarting IDMSDBL2, IDMSDBL3, or IDMSDBL4:** If a problem arises while running IDMSDBL2 or IDMSDBL3, *do not* simply fix the cause of the problem and rerun the step. In addition to fixing the cause of the problem, you must do one of the following:

- Reinitialize the database and begin again at the IDMSDBL2 step
- If you backed up the database before running the step, you can run the step again, using the backup

These steps change the database, and if a problem arises, you need to undo the changes before running a step over again.

If a non-data related problem occurs while running IDMSDBL4, you can restart the job without restoring. If the input file (SYS011) still exists, unlock the area and run the RELOAD utility from IDMSDBL4. Any updates previously performed by the run of IDMSDBL4 that abended will be overlaid by the updates in the restarted IDMSDBL4.

►► For help in deciding whether to run the RELOAD utility all at once or in steps, refer to *CA-IDMS Database Administration*.

**Data from UNLOAD is in SYS001, SYS002, SYS003, and RELDCTL:** The information the RELOAD utility needs is in the SYS001, SYS002, SYS003, and RELDCTL files. UNLOAD knew these files as SYSPCH, SYS002, SYS003, and RELDCTL, respectively.

- The SYS001 file contains sort parameters.
  - Note:** If the RELOAD utility is started at a SORT step, in any mode, SYS001 should point to the sort parameters generated in a previous step.
- The SYS002 file contains:
  - The unloaded data from UNLOAD processing
  - The data to be loaded
  - Set membership information for each record
- The SYS003 file contains both system and user-owned index data from UNLOAD processing. If running in STEP mode, index data is not input until the SORT2 step.
- The RELDCTL file contains:
  - Set descriptor information
  - A control record containing subschema, segment, and DMCL information
  - Control information created during UNLOAD processing and used by RELOAD processing. If running in STEP mode, this file *must be input to every step*.

**SORTEXIT and FROM/STEP:** When using the FROM and STEP options with the SORTEXIT option, each pair of SORT $n$  and DBL $x$  steps are considered to be one step. If either half of the SORT $n$ /DBL $x$  is specified on a FROM or STEP option, processing will start with the SORT $n$  step and the DBL $x$  step will also be executed. For example:

- FROM IDMSDBL3 will start with step SORT3 and will continue to the end.
- STEP SORT3 will run steps SORT3 and IDMSDBL3.

**SORTEXIT/REUSE WORKFILE restart considerations:** Since SORTEXIT combines each SORT $n$  step with the DBL $x$  step that follows it, if a failure occurs in the DBL $x$  step, a restart (if a restart is possible) must begin with the sort step and the input to the step will be resorted. Non-SORTEXIT mode will take longer to run but can be restarted after the sort in this case. Therefore, if restart time is more critical than normal runtime do not run the utility as a sortexit.

If the REUSE WORKFILE option is used with SORTEXIT, some input workfiles will be used as output files in the same step. Therefore if these two options are used together and a failure occurs, the utility must be restarted from the beginning.

**Workfile Considerations for restarting a failed RELOAD:** If the RELOAD command fails, depending on the reason for failure, restart the command at the failing step using the "FROM step-name" syntax. You can only restart a step if the input files to that step are intact and valid.

To prepare for a possible restart when running a one-step RELOAD, the Intermediate work files should have a disposition that preserves the dataset in the event of an abend, for example, "DISP=(NEW,CATLG,CATLG)."

To restart RELOAD at a particular step, the input files to that step must have a disposition to specify that the files already exist, for example, "DISP=OLD."

To determine which files were input to a given step refer to the "Intermediate Work File" tables under "JCL Considerations." Partially created output files should be deleted before restarting the job and the original disposition should be used in the restart job for example, "DISP=(NEW,CATLG,CATLG)."

The SYSPCH file contains sort parameter information for sort steps. It is an output file to IDMSDBL $n$  steps, but is not read unless restarting or running in step mode. So during a normal run the SYSPCH file should be treated as a normal output file, for example, "DISP=(NEW,CATLG,CATLG)." However restarting is not as straightforward. If the previous job failed in an IDMSDBL $x$  step, the SYSPCH file was an output file and should be deleted before restarting. But if the failure occurred in a SORT $x$  step, the contents of the SYSPCH file should contain the same values that were input to the SORT $x$  step. In this case the SYSPCH file should be preserved and defined as a SYS001 input file to the restart step.

When the SORTEXIT option is used the SORT $x$  and IDMSDBL $x$  steps are combined. If a failure occurs in this mode the SYSPCH file should normally be preserved and used as a SYS001 input file to the restart. However, there is a small window at the end of a IDMSDBL $x$  step where the SYSPCH file is opened for output and new SORT parameters are written. If the job fails at this point the entire SORT $x$ /IDMSDBL $x$  step must be restarted, but the SYSPCH file will not be valid as a SYS001 input file. In this case the sort parameters must be recreated by hand or the job must be restarted at

an earlier IDMSDBLx step if possible. One way to avoid this situation is to run in step mode when running SORTEXIT mode.

The RELDCTL dataset is always an input file to the first step of a RELOAD whether being restarted or not.

**REUSE WORKFILE considerations:** Some tape volume management systems consider the reuse of a tape volume for second time output processing an error even in the same job and will not allow you to make this mistake. It results in rerunning the job over again without the REUSE option. You can sometimes avoid this by specifying a zero retention period for the tape output volume.

**Intermediate work files:** The following tables indicate which work files are created and read by the different utility steps depending on the use of the SORTEXIT and REUSE WORKFILE options.

Step	Input	Output
RELOAD: NOT sortexit mode and NOT reusing workfile		
SORT1	SYS002	SYS004
IDMSDBL2	SYS004	SYS005 SYS006
SORT2	SYS003 SYS006	SYS007
IDMSDBLX	SYS007	SYS008
SORT3	SYS005 SYS008	SYS009
IDMSDBL3	SYS009	SYS010
SORT4	SYS010	SYS011
IDMSDBL4	SYS011	
RELOAD: NOT sortexit mode and REUSING workfiles.		
SORT1	SYS002	SYS004
IDMSDBL2	SYS004	SYS005 SYS006
SORT2	SYS003 SYS006	SYS004
IDMSDBLX	SYS004	SYS006
SORT3	SYS005 SYS006	SYS004
IDMSDBL3	SYS004	SYS005

Step	Input	Output
SORT4	SYS005	SYS004
IDMSDBL4	SYS004	
RELOAD: SORTEXIT mode and NOT reusing workfiles.		
SORT1/IDMSDBL2	SYS002	SYS005 SYS006
SORT2/IDMSDBLX	SYS003 SYS006	SYS008
SORT3/IDMSDBL3	SYS005 SYS008	SYS010
SORT4/IDMSDBL4	SYS010	
RELOAD: SORTEXIT mode and REUSING workfiles.		
SORT1/IDMSDBL2	SYS002	SYS005 SYS006
SORT2/IDMSDBLX	SYS003 SYS006	SYS006
SORT3/IDMSDBL3	SYS005 SYS006	SYS005
SORT4/IDMSDBL4	SYS005	

**How RELOAD works:** The RELOAD utility consists of the following steps which you can run separately or as a single operation:

Step	Description
SORT1	Sorts the contents of SYS002.
IDMSDBL2	<ul style="list-style-type: none"> <li>■ Populates the database by means of an area sweep but does not connect any sets.</li> <li>■ Prints statistics on the records written to the database.</li> </ul> <p><b>Note:</b> Overflow statistics may be fewer than expected. In order to improve performance during a reload, IDMSDBL2 uses the dbkey of the previously stored record as a 'direct' dbkey if the next record to be stored has the same new target page. This reduces the number of overflow conditions.</p>
SORT2	Sorts the contents of SYS003 and SYS006.
IDMSDBLX	<ul style="list-style-type: none"> <li>■ Establishes pointers for each index in which each record participates but does not build the indexes in the database.</li> </ul>
SORT3	Sorts the contents of SYS005 and SYS008.

Step	Description
IDMSDBL3	<ul style="list-style-type: none"> <li>■ Establishes pointers for each chained set in which each record participates (that is, builds the record prefix) but does not write the prefixes to the database.</li> <li>■ Builds indexes in the database.</li> </ul>
SORT4	Sorts the contents of SYS010.
IDMSDBL4	Inserts the record prefixes by performing a serial sweep of the database.

**Each step has input and output:** Each step uses the output from an earlier step and generates output for use by the next step. The following table lists the input and output for each step:

Step	Input	Output	Size
First sort	SYS001 sort parameters from UNLOAD SYS001 or the previous IDMSDBL $n$ step SYSPCH		
SORT1	<ul style="list-style-type: none"> <li>■ SYS001 contains sort parameters from UNLOAD SYSPCH file</li> <li>■ SYS002 contains the record information from UNLOAD</li> </ul>	SYS004 contains the sorted record information	SYS004 record length = SYS002 record length from UNLOAD

Step	Input	Output	Size
IDMSDBL2	<ul style="list-style-type: none"> <li>■ SYS004 from SORT1</li> <li>■ RELDCTL file from UNLOAD utility</li> </ul>	<ul style="list-style-type: none"> <li>■ SYS005 contains member descriptors for chained sets</li> <li>■ SYS006 contains member descriptors for user owned index sets</li> <li>■ SYSPCH contains sort parameters</li> <li>■ SYSLST contains statistics report</li> </ul>	<p>SYS005 record length = 32 bytes</p> <p>SYS006 record length = 32 bytes</p>
SORT2	<ul style="list-style-type: none"> <li>■ SYS001 contains sort parameters from IDMSDBL2 SYSPCH*</li> <li>■ SYS003 contains index information from UNLOAD</li> <li>■ SYS006 contains member descriptors for user owned index sets from IDMSDBL2</li> </ul>	SYS007 contains the sorted contents of SYS003 and SYS006	SYS007 record length = larger of RELOAD's SYS006 or UNLOAD's SYS003

Step	Input	Output	Size
IDMSDBLX	<ul style="list-style-type: none"> <li>■ SYS007 contains sorted index information from SORT2</li> <li>■ RELDCTL file from UNLOAD utility</li> </ul>	SYS008 contains reformatted index information	SYS008 record length = same as SYS007
SORT3	<ul style="list-style-type: none"> <li>■ SYS001 contains sort parameters from IDMSDBL2 SYSPCH*</li> <li>■ SYS005 from IDMSDBL2</li> <li>■ SYS008 from IDMSDBLX</li> </ul>	SYS009 contains sorted index set descriptors	SYS009 record length = larger of RELOAD's SYS008 or IDMSDBL2's SYS005
IDMSDBL3	<ul style="list-style-type: none"> <li>■ SYS009 from SORT3</li> <li>■ RELDCTL file from UNLOAD utility</li> </ul>	<ul style="list-style-type: none"> <li>■ SYS010 contains prefix pointer information</li> <li>■ SYSPCH contains sort parameters</li> <li>■ SYSLST contains a statistics report</li> </ul>	SYS010 record length = 28 bytes
SORT4	<ul style="list-style-type: none"> <li>■ SYS001 contains sort parameters from IDMSDBL3 SYSPCH*</li> <li>■ SYS010 from IDMSDBL3</li> </ul>	SYS011 contains sorted prefix pointer information	SYS011 record length = same as SYS010 from prior step

Step	Input	Output	Size
IDMSDBL4	<ul style="list-style-type: none"> <li>■ SYS011 from SORT4</li> <li>■ RELDCTL file from UNLOAD utility</li> </ul>	SYSLST contains messages on the results of the reload operation	

\* The sort parameters noted in the above table are read from SYS001, as noted, only when you run the sort steps individually. If you run all steps together, or all steps from an intermediate restart (using FROM), the sort parameters are passed in-storage. If you restart from an abnormal termination, point the SYS001 file to the SYSPCH file of the last completed step.

**Sort output after each step:** If you run the RELOAD utility a step at a time, you must use the sort parameters in the SYSPCH file to sort the contents of the intermediate work files.

Sort the intermediate work files as follows:

Sort name	File to sort	Sort order	Sort on	Begins at
SORT1	SYS002	Descending	20 bytes	Byte 5
SORT2	SYS003 and SYS006	Ascending	24 bytes	Byte 5
SORT3	SYS005 and SYS008	Ascending	24 bytes	Byte 5
SORT4	SYS010	Ascending	12 bytes	Byte 5

**Note:** The generated sort parameters are insufficient for stand alone sort programs under VSE/ESA. If you want to use your own sort program, you must add 'WORK=' parameters to specify more than one file whose contents are to be sorted.

### 4.23.3 JCL considerations

When you submit a RELOAD statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- Files containing the areas to be reloaded
- The intermediate work files
- Sort space

---

**Work file JCL Considerations for STEP mode:** RELOAD normally runs as a single step but runs as separate steps using the "STEP step-name" syntax. When running in step mode, input files should have dispositions that state the file already exists, for example, "DISP=OLD."

Preserve output files on successful completion but not when the job fails, for example, "DISP=(NEW,CATLG,DELETE)."

See the "Intermediate work file" table to determine which files are input and which files are output, and when they are used.

The RELDCTL file is always input to every step.

The SYSPCH file is created by an IDMSDBLx step and used as input to a SORTx step. When used as input it is defined as SYS001.

Work file record lengths:

- The RELDCTL file is a fixed length file with a record length of 60 bytes.
  - The SYSPCH file is a fixed length file with a record length of 80 bytes.
  - All SYSxxx files are variable length files. The record length can vary from one step to the next, from one job to the next. Do not code an LRECL value in the JCL just code a BLKSIZE value. A BLKSIZE value should be chosen based on the optimal size for the device being used, for example, 1/2 track if disk or 32k if tape.
- ▶▶ Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

#### 4.23.4 Example

The following example directs the RELOAD utility to reload a previously unloaded non-SQL defined database.

```
reload;
```

The following command directs RELOAD to run all steps as a sortexit and to reuse workfiles:

```
reload as sortexit reuse workfiles;
```

### 4.23.5 Sample output

The RELOAD utility generates the following listing after the successful execution of the RELOAD statement in the above example. Note that the file the unloaded database is reloaded into is formatted first.

```

IDMSBCF                                IDMS Batch Command Facility                09/12/99  PAGE 1

SET BATCH WIDTH PAGE 80 ;
Status = 0
FORMAT FILE USERDB.EMPF1;

File USERDB.EMPF1                blocks 1 to 50.
Area USERDB.EMP AREA             pages 5,001 to 5,050.
Page size in file 4,096.

Status = 0
  RELOAD ;
UT010002 BEGINNING PROCESSING FOR STEP SORT1
UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
UT009002   105 RECORDS WERE READ FROM SYS002
UT009003   105 RECORDS WERE WRITTEN TO SYS004
UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT1 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL2
UT005001 IDMSDBL2 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT005002 DATABASE LOAD STATISTICS
  DATABASE LOADED ON 09/12/99 AT 123700
PAGES READ .....                19
PAGES WRITTEN .....              18
PAGES REQUESTED .....            20
CALC RCDS IN TARGET PAGE .       12
CALC RCDS OVERFLOWED .....       0
VIA RCDS IN TARGET PAGE ..       53
VIA RCDS OVERFLOWED .....        1
LINES REQUESTED BY IDMS ..       222
RCDS MADE CURRENT OF R/U .        64
CALLS TO IDMS .....              70
FRAGMENTS STORED .....            0
RECORDS RELOCATED .....           0

UT005003   77 INTERMEDIATE RECORDS WERE WRITTEN TO SYS006
UT005003   69 INTERMEDIATE RECORDS WERE WRITTEN TO SYS005
UT005004   SYS005 RECORD LENGTH IS 32
UT005005   NO DATABASE ERRORS WERE ENCOUNTERED
UT005006 IDMSDBL2 RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBL2 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT2
UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
UT009002   67 RECORDS WERE READ FROM SYS003
UT009002   77 RECORDS WERE READ FROM SYS006
UT009003   144 RECORDS WERE WRITTEN TO SYS007
UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT2 HAS COMPLETED SUCCESSFULLY

```

```
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBLX
UT008001 IDMSDBLX RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT008002 67 RECORDS WERE WRITTEN TO SYS008
UT008006 IDMSDBLX RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBLX HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
UT009002 69 RECORDS WERE READ FROM SYS005
UT009002 67 RECORDS WERE READ FROM SYS008
UT009003 136 RECORDS WERE WRITTEN TO SYS009
UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT006007 136 INTERMEDIATE RECORDS WERE READ FROM SYS009
UT006002 42 INTERMEDIATE RECORDS WERE WRITTEN TO SYS010
UT006005 NO DATABASE ERRORS WERE ENCOUNTERED
UT006006 IDMSDBL3 RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBL3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT4
UT009001 IDMSDBLY RELEASE 15.0 TAPE CAGJF0 SORT STARTED
UT009002 42 RECORDS WERE READ FROM SYS010
UT009003 42 RECORDS WERE WRITTEN TO SYS011
UT009004 IDMSDBLY RELEASE 15.0 SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT4 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL4
UT007001 IDMSDBL4 RELEASE 15.0 TAPE CAGJF0 PROCESSING STARTED
UT007004 NO DATABASE ERRORS WERE ENCOUNTERED
UT007005 NO LOGIC ERRORS WERE ENCOUNTERED
UT007002 42 RECORDS WERE READ FROM SYS011
UT007006 IDMSDBL4 RELEASE 15.0 PROCESSING COMPLETED
UT010003 STEP IDMSDBL4 HAS COMPLETED SUCCESSFULLY
UT010001 DATABASE RELOAD HAS COMPLETED SUCCESSFULLY
Status = 0
```

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings

## 4.23.6 For more information

- **On unloading and reloading a CA-IDMS/DB database**, see *CA-IDMS Database Administration*.

## 4.24 RESTORE

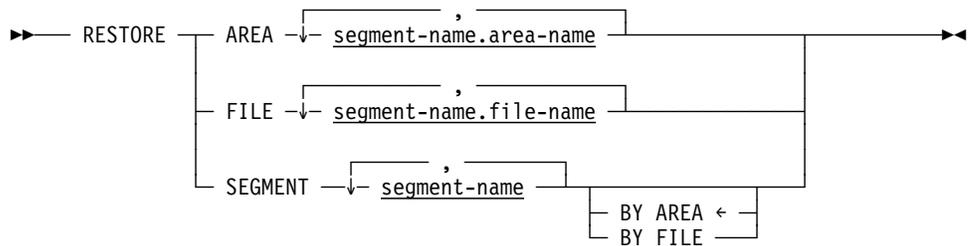
### 4.24.1 Description

**Purpose:** The RESTORE utility restores one or more areas in a database by copying back the contents of a file created by the BACKUP statement.

**Authorization:**

To	You need this privilege	On
Restore an area	DBAWRITE	The area
Restore a file	DBAWRITE	The area(s) to which the file maps
Restore a segment	DBAWRITE	The area(s) associated with the segment

**Syntax**



**Parameters**

**AREA**

Directs the RESTORE utility to restore one or more areas.

**segment-name**

The name of the segment associated with the area.

**area-name**

The name of the area.

**FILE**

Directs the RESTORE utility to restore one or more files.

**segment-name**

The name of the segment associated with the file.

**file-name**

The name of the file.

**SEGMENT segment-name**

The name of the segment to be restored.

**BY AREA**

Specifies that each area defined within the segment is to be restored. AREA is the default.

**BY FILE**

Specifies that each file within the segment is to be restored.

## 4.24.2 Usage

**How to submit the RESTORE statement:** You submit the RESTORE statement to CA-IDMS/DB only through the batch command facility. When submitting RESTORE statements, the batch command facility must be run in local mode.

**Vary areas offline:** Before running the RESTORE statement, vary all areas being restored offline to all DC/UCF systems. This will prevent all other jobs from accessing the areas until RESTORE processing is completed.

**Restoring by area requires locks on areas:** CA-IDMS/DB locks an area before restoring it. If the area is already locked, it is not restored and the RESTORE operation will terminate with an error. No more areas are restored. It is possible that a local mode application abended without releasing an area lock. In this case, use the UNLOCK statement to unlock the area.

**Restoring by file does not lock areas:** When you restore by file, CA-IDMS/DB does not lock the associated area(s).

**RESTORE the same object you backed up:** If you backed up by area, restore by area. If you backed up by file, restore by file.

**Restoring from IDMSDUMP** RESTORE can only restore files produced by the BACKUP utility. RESTORE *cannot* restore files produced by the 10.2 IDMSDUMP utility program. You must use the 10.2 IDMSRSTR utility program to restore files produced by IDMSDUMP.

## 4.24.3 JCL considerations

When you submit a RESTORE statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the file(s) containing the areas to be restored.

▶▶ Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.24.4 Examples

**Restoring by area:** The following example directs the RESTORE utility to restore three database areas.

```
restore area empdemo.emp-demo-region,
           empdemo.org-demo-region,
           empdemo.ins-demo-region;
```

**Restoring by file:** The following example directs the RESTORE utility to restore three database files.

```
restore file empdemo.empdemo.
           empdemo.orgdemo,
           empdemo.insdemo;
```

**Restoring by segment:** The following example directs the RESTORE utility to restore all areas in the empdemo segment.

```
restore segment empdemo;
```

## 4.24.5 Sample output

**Restoring a database by area:** The following listing is generated after successfully executing the statements in the, "Restoring a database by area" example above.

```
IDMSBCF 15.0                                CA-IDMS Batch Command Facility                10/30/99  PAGE 2

  RESTORE AREA EMPDEMO.EMP-DEMO-REGION,
             EMPDEMO.ORG-DEMO-REGION,
             EMPDEMO.INS-DEMO-REGION;

UT015006 BACKUP file created on 1999-10-30-16.08.53.303879
UT000038 Starting RESTORE of area EMPDEMO.EMP-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.ORG-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.INS-DEMO-REGION
UT000040 RESTORE complete

Status = 0          SQLSTATE = 00000
```

**Restoring a database by file:** The following listing is generated after successfully executing the statements in the, "Restoring a database by file" example above.

```
IDMSBCF 15.0                                CA-IDMS Batch Command Facility                10/30/99  PAGE 4

RESTORE FILE EMPDEMO.EMPDEMO,
           EMPDEMO.ORGDEMO,
           EMPDEMO.INSDEMO;

UT015006 BACKUP file created on 1999-10-30-16.09.36.080684
UT000039 Starting RESTORE of file EMPDEMO.EMPDEMO
UT000040 RESTORE complete
UT000039 Starting RESTORE of file EMPDEMO.ORGDEMO
UT000040 RESTORE complete
UT000039 Starting RESTORE of file EMPDEMO.INSDEMO
UT000040 RESTORE complete

Status = 0      SQLSTATE = 00000
```

**Restoring a database by segment:** The following listing is generated after successfully executing the statements in the, "Restoring a database by segment" example above.

```
IDMSBCF 15.0                                CA-IDMS Batch Command Facility                10/30/99  PAGE 6

RESTORE SEGMENT EMPDEMO;

UT015006 BACKUP file created on 1999-10-30-16.10.04.315493
UT000038 Starting RESTORE of area EMPDEMO.EMP-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.INS-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.ORG-DEMO-REGION
UT000040 RESTORE complete

Status = 0      SQLSTATE = 00000
```

#### 4.24.6 For more information

- On restoring a database, see *CA-IDMS Database Administration*.

## 4.25 RESTRUCTURE CONNECT

### 4.25.1 Description

**Purpose:** The RESTRUCTURE CONNECT utility statement connects new prior and owner pointers in existing sets. You execute the RESTRUCTURE CONNECT statement after executing the RESTRUCTURE SEGMENT statement as part of a database restructure operation. The RESTRUCTURE CONNECT statement uses the specifications in the base restructuring table and the information in the spill file generated by the RESTRUCTURE SEGMENT statement to make the pointer connections.

**Authorization:**

To	You need this privilege	On
Connect pointers in an area	DBAWRITE	The area
Connect pointers in a segment	DBAWRITE	All areas of the segment

**Syntax**

```

▶— RESTRUCTURE CONNECT SEGMENT segment-name
▶— USING subschema-name
  [ CONTINUE [ YES ] ]
  [ NO ← ]
  [ RSTTMOD [ rstt-modname ] ]
  [ IDMSRSTT ← ]
  [ AREA [ area-name ] ]

```

**Parameters**

**segment-name**

Identifies the segment containing the area(s) to be processed.

**USING**

Specifies the subschema that defines the records and sets being restructured.

**subschema-name**

The name of a subschema compiled under a schema that describes the database after restructuring.

**CONTINUE**

Specifies whether processing is to continue if an error condition is detected during processing.

By default, if you do not specify YES, RESTRUCTURE CONNECT will not continue processing when an error is detected.

**YES**

Specifies that processing should continue when an error is detected.

**NO**

Specifies that processing should stop when an error is detected.

NO is the default.

**RSTTMOD**

Specifies the base restructuring table that defines the changes in the database being made by the RESTRUCTURE SEGMENT and RESTRUCTURE CONNECT statements.

By default, if you do not specify a base restructuring table, IDMSRSTT is used.

**rstt-modname**

The name of the base restructuring table.

The default base restructuring table is IDMSRSTT.

**AREA**

Restricts processing to one or more specified areas.

By default, if you do not specify one or more areas, all areas in the specified segment that contain occurrences of the records or members of the sets being restructured are processed.

**area-name**

The name of an area.

## 4.25.2 Usage

**How to submit the RESTRUCTURE CONNECT statement:** You submit the RESTRUCTURE CONNECT statement to CA-IDMS/DB only through the batch command facility. You must be running CA-IDMS/DB in local mode.

**Vary areas offline:** Before submitting the RESTRUCTURE CONNECT statement, vary all affected areas offline.

**Connecting pointers for new sets:** The RESTRUCTURE CONNECT statement connects new pointers for existing sets only. To connect pointers for new sets, you must run a user-written program after the restructure process is complete.

### 4.25.3 JCL considerations

When you submit a RESTRUCTURE CONNECT statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The database file(s) that map to the area(s) to be processed.
- The file containing the assembled base restructuring table.
- The spill file generated by RESTRUCTURE SEGMENT. The size of the spill file should be a multiple of 32 with a maximum block size of 32,736.
- Sort files to sort the spill file in database key sequence.

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.25.4 Examples and sample output

To illustrate the use of the RESTRUCTURE CONNECT utility, the RESTRUCTURE SEGMENT utility is run first to add pointers to a set. The RESTRUCTURE CONNECT utility is then executed to update the prefix portion of affected records with pointers.

**Adding set pointers:** The example below directs the RESTRUCTURE SEGMENT utility to use the base restructuring table, LRDKRSTT, to add prior pointers to the owners and members of set A-B and owner pointers to the members of set A-B.

#### *Input to RESTRUCTURE SEGMENT*

```
restructure segment empdemo using restr01
rsttmod lrdkrstt continue yes;
```

**Output from RESTRUCTURE SEGMENT:** When the RESTRUCTURE SEGMENT operation is completed, the following listing is generated.

IDMSBCF	IDMS Batch Command Facility	09/18/99	PAGE 1
RESTRUCTURE SEGMENT EMPDEMO USING RESTR01			
RSTTMOD LRDKRSTT CONTINUE YES;			
UT000038 Starting Restructure of area EMPDEMO.EMP-DEMO-REGION			
UT000041 Completed processing of area EMPDEMO.EMP-DEMO-REGION, Pages read=10 Records read=6			
UT011023 Record name AA Found=2 Changed=2 Lde1=0			
0UT000038 Starting Restructure of area EMPDEMO.INS-DEMO-REGION			
UT000041 Completed processing of area EMPDEMO.INS-DEMO-REGION, Pages read=10 Records read=3			
UT011023 Record name BB Found=3 Changed=3 Lde1=0			
Status = 0			

**Connecting prior and owner pointers:** To complete this restructure operation, the RESTRUCTURE CONNECT utility is executed to connect the prior and owner pointers in all occurrences of set A-B after restructuring records A and B.

#### *Input to RESTRUCTURE CONNECT*

```
restructure connect segment empdemo using EMPSS01
rsttmod lrdkrstt continue yes;
```

*Output from RESTRUCTURE CONNECT:* When the RESTRUCTURE CONNECT operation is completed, the following listing is generated.

```
IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1

RESTRUCTURE CONNECT SEGMENT EMPDEMO USING EMPSS01
RSTTMOD LRDKRSTT CONTINUE YES;
UT000038 Starting Connect of area EMPDEMO.EMP-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.EMPDEMO-REGION, Pages read=10 Records read=6
UT011023 Record name AA Found=2 Changed=2 Ldel=0
UT000038 Starting Connect of area EMPDEMO.INSDEMO-REGION
UT000041 Completed processing of area EMPDEMO.INSDEMO-REGION, Pages read=10 Records read=3
UT011023 Record name BB Found=3 Changed=3 Ldel=0
Status = 0
```

### 4.25.5 For more information

- **On changing database components**, see *CA-IDMS Database Administration*.
- **On the RESTRUCTURE SEGMENT utility**, see 4.26, “RESTRUCTURE SEGMENT.”

## 4.26 RESTRUCTURE SEGMENT

### 4.26.1 Description

**Purpose:** The RESTRUCTURE SEGMENT utility modifies record occurrences to match new schema specifications.

Using the RESTRUCTURE SEGMENT utility, you can:

- Insert new data items anywhere in a record
- Delete existing data items
- Change the length and position of data items
- Change the format of a record from fixed length to variable length or from variable length to fixed length
- Compress or uncompress a record
- Add pointers for new or existing sets
- Delete pointers from existing sets
- Add or delete prior or owner pointers for existing sets

**Authorization:**

To	You need this privilege	On
Restructure an area	DBAWRITE	The area
Restructure a segment	DBAWRITE	All areas of the segment

**Syntax**

```

▶▶ RESTRUCTURE SEGMENT segment-name
▶ USING subschema-name
▶ [ CONTINUE [ YES ] ]
▶ [ NO ← ]
▶ [ RSTTMOD [ rstt-modname ] ]
▶ [ IDMSRSTT ← ]
▶ [ AREA [ area-name ] ]

```

**Parameters**

**segment-name**

The name of the segment to be restructured.

**USING**

Specifies the subschema that defines all the records and sets to be restructured.

**subschema-name**

The name of a subschema compiled under a schema that describes the database before restructuring.

**CONTINUE**

Specifies whether processing is to continue if an error condition is detected during processing.

By default, if you do not specify YES, processing stops when an error is detected.

**YES**

Directs processing to continue when an error is detected.

**NO**

Directs processing to stop when an error is detected.

NO is the default.

**RSTTMOD**

Specifies the base restructuring table that defines the changes to be made in the database.

By default, if you do not specify a base restructuring table, IDMSRSTT is used.

**rstt-modname**

The name of the base restructuring table.

The default base restructuring table is IDMSRSTT.

**AREA**

Restricts processing to one or more specified areas.

By default, if you do not specify one or more areas, all areas in the specified segment that contain occurrences of the records or members of the sets being restructured are processed.

**area-name**

The name of an area.

## 4.26.2 Usage

**How to submit the RESTRUCTURE SEGMENT statement:** You submit the RESTRUCTURE SEGMENT statement to CA-IDMS/DB only through the batch command facility. When submitting RESTRUCTURE SEGMENT statements, you must run the batch command facility in local mode.

**The base restructuring table:** The RESTRUCTURE SEGMENT statement modifies record occurrences according to the specifications in a **base restructuring table**. You assemble the base restructuring table from IDMSRSTT macro statements that define the changes to be made. Typically, you use the IDMSRSTC utility to

generate the IDMSRSTT macro statements; however, the statements can also be coded manually.

►►For a description of the IDMSRSTT macro statements, see Appendix B, “IDMSRSTT Macro Statements.”

**The spill file:** If you specify a *set-name* in the SETPTR statement of IDMSRSTT during RESTRUCTURE SEGMENT processing, a **spill file** of work records that describes new pointers being added to database records is generated. If the database restructure includes the addition of new prior pointers to existing sets, you use the spill file as input to the RESTRUCTURE CONNECT utility statement. The information in the spill file and the specifications in the base restructuring table are used during RESTRUCTURE SEGMENT processing to connect the new prior pointers.

**Database keys of restructured records:** During RESTRUCTURE SEGMENT processing all changes to the database are made in place (that is, no unload/reload occurs). As a result, database keys are not changed by a restructure operation.

**Back up the database:** Back up the database before performing a restructure operation. If the database restructure is unsuccessful, you can then restore the database from the backup copy.

**Remove logically deleted records:** You cannot use the RESTRUCTURE SEGMENT utility statement to make changes that will modify the prefix portion of a record if there are logically deleted records in the database area to be modified. To ensure there are no logically deleted records, use the CLEANUP utility to erase them before performing a restructure operation.

**Restructure ready mode:** All areas to be processed are readied in exclusive update mode for RESTRUCTURE SEGMENT processing.

**Modifying CALC and sort keys:** During a restructure operation, you should not modify CALC and sort keys.

►►For information on modifying database components, see *CA-IDMS Database Administration*.

### **Native VSAM data sets:**

You cannot use RESTRUCTURE SEGMENT to restructure records in native VSAM data sets.

**New pointers:** New pointers in an owner record are initialized to the database key of the owner record (indicating an empty set). In a member record, new pointers are initialized to -1.

To connect new pointers for existing sets, use the **RESTRUCTURE CONNECT** utility statement. To connect pointers for new sets, you must run a user-written program after the restructure process is complete.

**Consideration when using data compression routines:** If the named subschema references a data compression routine and a compressed record is involved in the restructure (i.e., changing the record to fixed length), the subschema will be modified. To modify the subschema, it must be nonreentrant. If necessary, relink the subschema as nonreentrant and run it from another load library.

If the subschema resides in a dictionary load area, this is not a consideration.

**Restructuring a database:** To restructure a database, following these guidelines:

1. **Back up the database.**
2. **Compile a new schema** that describes the database after restructuring. The new schema must have a different name or version number from the schema that describes the existing database.
3. **Execute the IDMSRSTC utility** to generate the IDMSRSTT macro statements that define the changes to be made to the database. Verify that the statements are correct; make any necessary modifications.
4. **Assemble the base restructuring table.**
5. **Link edit the base restructuring table.**
6. **Execute the RESTRUCTURE SEGMENT utility statement** to restructure the database. Use a subschema that was compiled under the old schema.
7. **Compile a subschema under the new schema.** Give the new subschema a temporary name to distinguish it from the old subschema.  
  
If no new pointers have been added to existing sets, skip to step 10.
8. **Execute the RESTRUCTURE CONNECT utility statement** to connect the new prior or owner pointers for existing sets in the restructured database. Use a subschema that was compiled under the new schema.
9. **Validate the restructure** using IDMSDBAN, CA-OLQ®, CA-CULPRIT®, or some other retrieval program.
10. **Recompile, under the new schema, all subschemas that use the changed records or sets.**
11. **Alter all access modules referencing changed records or owner and member records of changed sets.**
12. **Drop and recreate SQL views of records whose element descriptions have changed.**

### 4.26.3 JCL considerations

When you submit a RESTRUCTURE SEGMENT statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The database file(s) that map to the area(s) to be processed.
- The file containing the assembled base restructuring table.
- The spill file to be used as input to the RESTRUCTURE CONNECT utility statement. The size of the spill file should be a multiple of 32 with a maximum size of 32,736.

► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.26.4 Examples and sample output

**Adding set pointers:** The example below directs the RESTRUCTURE SEGMENT utility to use the base restructuring table, LRDKRSTT, to add prior pointers to the owners and members of set A-B and owner pointers to the members of set A-B.

*Input to RESTRUCTURE SEGMENT*

```
restructure segment empdemo using restr01
rsttmod lrdkrstt continue yes;
```

*Output from RESTRUCTURE SEGMENT:* When the RESTRUCTURE SEGMENT operation is completed, the following listing is generated.

IDMSBCF	IDMS Batch Command Facility	09/18/99	PAGE 1
RESTRUCTURE SEGMENT EMPDEMO USING RESTR01			
RSTTMOD LRDKRSTT CONTINUE YES;			
OUT000038	Starting Restructure of area EMPDEMO.EMP-DEMO-REGION		
UT000041	Completed processing of area EMPDEMO.EMP-DEMO-REGION,	Pages read=10	Records read=6
UT011023	Record name AA Found=2 Changed=2 Ldel=0		
OUT000038	Starting Restructure of area EMPDEMO.INS-DEMO-REGION		
UT000041	Completed processing of area EMPDEMO.INS-DEMO-REGION,	Pages read=10	Records read=3
UT011023	Record name BB Found=3 Changed=3 Ldel=0		
Status = 0			

**Connecting prior and owner pointers:** To complete this restructure operation, the RESTRUCTURE CONNECT utility is executed to connect the prior and owner pointers in all occurrences of set A-B after restructuring records A and B.

*Input to RESTRUCTURE CONNECT:*

```
restructure connect segment empdemo using EMPSS01
rsttmod lrdkrstt continue yes;
```

*Output from RESTRUCTURE CONNECT:* When the RESTRUCTURE CONNECT operation is completed, the following listing is generated.

```

IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1

RESTRUCTURE CONNECT SEGMENT EMPDEMO USING EMPSS01
RSTTMOD LRDKRSTT CONTINUE YES;
OUT000038 Starting Connect of area EMPDEMO.EMP-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.EMPDEMO-REGION, Pages read=10 Records read=6
UT011023 Record name AA Found=2 Changed=2 Ldel=0
OUT000038 Starting Connect of area EMPDEMO.INSDEMO-REGION
UT000041 Completed processing of area EMPDEMO.INSDEMO-REGION, Pages read=10 Records read=3
UT011023 Record name BB Found=3 Changed=3 Ldel=0
Status = 0

```

## 4.26.5 For more information

- **On changing a database definition**, see *CA-IDMS Database Administration*.
- **On CA-IDMS/DB physical database characteristics**, see *CA-IDMS Database Administration*.
- **On coding IDMSRSTT macro statements**, see Appendix B, “IDMSRSTT Macro Statements.”

## 4.26.6 Callable Restructure Utility

The Callable Restructure Utility (IDMSCRSU) is a version of the CA-IDMS Restructure Segment utility that is callable from a user-written program. It uses the standard base restructuring table to control the restructure of the data portion of a database record.

►► For more information on assembling a base restructuring table, see Appendix B, “IDMSRSTT Macro Statements.”

IDMSCRSU is called with the following register values and parms:

On Entry:

- R1 points to a four-word parmlist.
- R13 should point to a 36-word save and work area.
- R14 contains the return address.
- R15 contains the entry point address.

### 4.26.6.1 Considerations

IDMSCRSU does not support changes to pointers or calling database procedures. The assumption is the caller has obtained the database record in the old subschema format and now wants to store or modify a record in the new subschema format.

To save a search of the IDMSRSTT base restructuring table on every call, the caller can save the RREC address returned in R0 after the first call. This address can then be passed instead of the IDMSRSTT address on all subsequent calls for the same record type.

A 36-word register save area and work area is expected to be passed in R13. If called from a DC assembler program, you can use a "#CHKSTK =36" instruction to ensure there is enough room in the stack.

### 4.26.6.2 Parameters

**parm-1**

The address of the FSR (SR51) control block from the "old" subschema. This block should describe the record as it exists before the restructure takes place.

**parm-2**

The address of the IDMSRSTT base restructuring table that describes the changes being made to the database record. This table is generated from the IDMSRSTT macro. (See Appendix B.) The table is searched for the RREC block that matches the FSR name passed in parm-1.

- or -

The address of the RREC control block in the IDMSRSTT table. (No search is done.)

**parm-3**

The address of the data record to be restructured. Only the data should be passed, not pointers or internal fields.

**parm-4**

The address of a buffer where the restructured record will be returned. It should be large enough to hold the maximum size of the record being restructured as described in the 'new' subschema.

On exit:

- R0 contains the address of the IDMSRSTT RREC block for the record just restructured, if found.
- R15 contains one of the following return codes:
  - 0: Call successful - The output buffer contains the restructured record.
  - 4: FIELD=ALL was specified for the record in the IDMSRSTT table. There were no changes to the data. The output buffer does **not** contain a copy of the record; the caller should use the input record image.
  - 8: The record was not found in the IDMSRSTT table.

## 4.27 ROLLBACK

### 4.27.1 Description

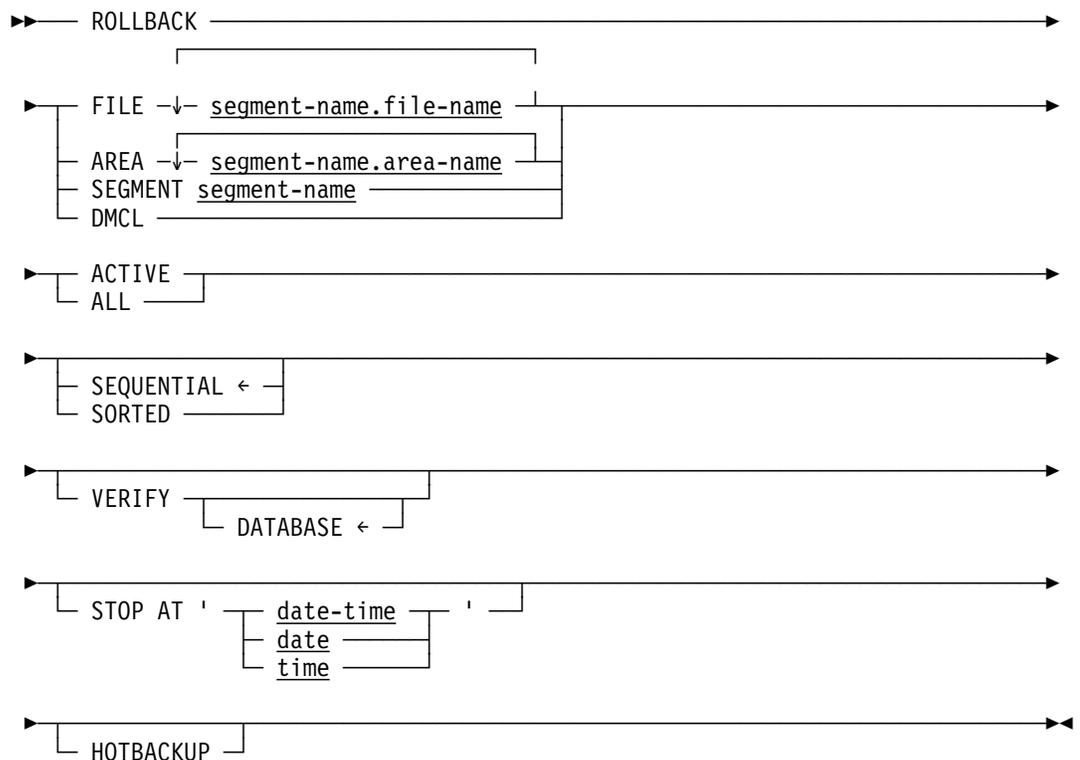
**Purpose:** The ROLLBACK utility restores all or part of a database to a previous state by applying before images from the journal file.

If requested, the ROLLBACK utility verifies the after images in the journal file against the contents of the database before applying the before images.

**Authorization:**

To	You need this privilege	On
Roll back an area	DBAWRITE	The area
Roll back a segment	DBAWRITE	All affected areas of the segment
Roll back a file	DBAWRITE	All affected areas associated with the file
Roll back a database	DBAWRITE	All affected areas of the database

**Syntax**



**Parameters****FILE**

Rolls back one or more files. Recovery by file does not unlock any areas associated with the files.

**segment-name**

The segment associated with the file.

**file-name**

The name of the file.

**AREA**

Rolls back and unlocks one or more specified areas.

**segment-name**

Identifies the segment associated with the area.

**area-name**

The name of the area.

**SEGMENT**

Rolls back and unlocks all areas associated with the specified segment.

**segment-name**

The name of the segment.

**DMCL**

Rolls back all areas defined in the DMCL identified in the SYSIDMS parameter file.

**ACTIVE**

- If you do not specify a STOP time, before images are applied only for transactions that remain open at the end of the journal file.
- If you do specify a STOP time, before images are applied for all transactions until the STOP time is reached, and then processing continues only with open transactions.

**ALL**

- If you do not specify a STOP time, before images are applied for all transactions in the journal file.
- If you do specify a STOP time, before images are applied until a point is reached at or before the specified time where there are no active transactions. If no such point is found, processing stops at the beginning of the journal file.

**SEQUENTIAL**

Applies before images in the sequence in which they occur starting from the end of the journal file, reading backwards.

SEQUENTIAL is the default.

**SORTED**

Sorts the before images in the journal file by database key and copies back only the earliest before image for any database key.

All the database keys on a given database page will be copied back together. Thus, each page will be accessed once.

By default, if you do not specify SORTED, the before images will be copied back sequentially.

**Note:** When recovering VSAM KSDS files, do not use the SORTED option. You must use SEQUENTIAL.

#### **VERIFY DATABASE**

Verifies the after images in the journal file before applying the before images. If an after image in the journal file does not match the contents of the database, the roll back operation will terminate with an error.

By default, if you do not specify VERIFY, before images are copied back without first verifying the after images.

The optional DATABASE keyword is included for consistency with the ROLLFORWARD utility statement and has no impact on the processing of the VERIFY option.

#### **STOP AT**

Directs processing to stop as soon as possible after reaching a specified date and time in the journal file.

By default, processing will stop when the beginning of the journal file is reached.

#### **date**

Specifies the date, in one of the following formats:

- *yyyy-mm-dd*
- *mm/dd/yyyy*
- *dd.mm.yyyy*

In these formats, the following rules apply:

- **Yyyy** specifies the year. *Yyyy* must be an integer in the range 0001 through 9999. Leading zeros are optional.
- **Mm** specifies the month within the year. *Mm* must be an integer in the range 01 through 12. Leading zeros are optional.
- **Dd** specifies the day within the month. *Dd* must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of *yyyy*, *mm*, and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

#### **date-time**

Specifies the date and time, where:

- The format for specifying the DATE-TIME is:

*yyyy-mm-dd-hh.mm.ss.ffffff*

- The rules for specifying the DATE component of DATE-TIME are the same as for DATE described above. The rules for specifying the TIME component of DATE-TIME are:
  - *Hh* specifies the hour on a 24-hour clock. *Hh* must be an integer in the range 00 through 23. Leading zeros are optional.
  - *Mm* specifies the number of minutes past the hour. *Mm* must be an integer in the range 00 through 59. Leading zeros are optional.
  - *Ss* specifies the number of seconds past the minute. *Ss* must be an integer in the range 00 through 59. Leading zeros are optional.
  - *Ffffff* specifies the number of millionths of a second past the specified second. *Ffffff* is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

**time**

Specifies the time in one of the following formats:

- *hh.mm.ss*
- *hh:mm:ss*

The rules for specifying TIME are the same as those listed for DATE-TIME above.

When TIME is specified, the date defaults to the current date.

**HOTBACKUP**

Forces ROLLBACK to also restore images for aborted run units.

## 4.27.2 Usage

**How to submit the ROLLBACK statement:** You submit the ROLLBACK statement to CA-IDMS/DB only through the batch command facility. When submitting ROLLBACK statements, you must run the batch command facility in local mode.

Vary any area being rolled back offline to all DC/UCF systems.

**When to use ROLLBACK:** The ROLLBACK utility is most commonly used after a local mode update job has abnormally terminated or has been run incorrectly, resulting in incorrect data being introduced into the database. In this case, use the ROLLBACK utility to restore the affected parts of the database to an earlier, uncorrupted, state.

Both the database and the journal must be available and readable in order to run the ROLLBACK utility.

**When not to use ROLLBACK:** If the database is unusable or incomplete because of a physical I/O error, then you cannot use the ROLLBACK utility.

In this case, use a backup copy of the database and run ROLLFORWARD.

►► For more information on when and how to use the ROLLBACK and ROLLFORWARD utilities, see *CA-IDMS Database Administration*.

**How ROLLBACK works:** The ROLLBACK utility starts with the current state of the database. It uses before images from a tape or journal file to restore progressively earlier states of the database, file, segment, or area.

You can run the ROLLBACK utility **sequentially**, applying before images in reverse order. Or you can run The ROLLBACK utility in **sorted** mode. In sorted mode, the ROLLBACK utility applies the earliest before image of any record being restored.

**ROLLBACK uses one journal file:** The journal file is a sequential file which may reside on disk or tape. If you have multiple journal files, you must consolidate them, in the order in which they were created, to a single file; and use the single file with the ROLLBACK utility.

**Disk journal file:** If the journal file is on disk, it cannot be read backwards in certain operating systems. To accommodate this, a SYSIDMS PARM has been created, ROLLBACK3490. To utilize this feature, the journal file must be sorted on the first GMT timestamp of each journal block, in a descending order. In this manner, the ROLLBACK utility, using the ROLLBACK3490 feature, reads each block forward, but is actually getting the blocks in descending order. Although the blocks are in descending order, the journal records within the blocks are in ascending order. As ROLLBACK reads each block in descending order, it processes each journal block from the end to the beginning. So even though the journal file is a sequential file on disk, sorted descending, ROLLBACK is processing the journal records in a descending order. To sort the journal blocks in descending order, use the following sort parm:

```
SORT FIELDS=(13,8,BI,D)
```

**Note:** In CMS, you do not need to use the ROLLBACK3490 parameter, but you must presort the journal images as described above regardless of whether the journal files reside on disk or tape.

**Note:** In BS2000/OSD, you do not need to use the ROLLBACK3490 parameter nor presort the data.

**Note:** If you decide to use the ROLLBACK3490 feature, or are using CMS, then you must sort the file in descending order prior to running the ROLLBACK utility, even if you are using 'sorted mode.'

**Tape journal file:** If you want the journal file on tape, you can put the file on tape by using the ARCHIVE JOURNAL utility, or by using a tape journal to begin with. If you have multiple tape journal files, you must concatenate them, in the order in which they were created, to a single file; and use the single file with the ROLLBACK utility.

If the tape file requires more than one tape volume, the volumes must be mounted in the order in which they were created.

**Note:** Depending on the version of OS/390 you are using, and on whether the tape volumes are labeled, you may need to mount either the first tape volume produced or all tape volumes produced before they can be read backwards.

**ROLLBACK and VSAM files:** You can run the ROLLBACK utility against most native VSAM files. However, because VSAM does not support deleting ESDS file records, the ROLLBACK statement cannot be used on a native VSAM ESDS file that might cause the ROLLBACK utility to try and delete a record. Therefore, a run unit that has added records to an ESDS file can only be recovered by restoring the file to an earlier time and then executing the ROLLFORWARD utility. For more information on CA-IDMS/DB and native VSAM files, refer to *CA-IDMS Database Administration*

**When to use sequential mode:** Use sequential mode if the journal file is very small or very large. If the journal file is small, you get no advantage from sorting. If the journal file is very large, it could require more sort space than you have available.

**When to use sorted mode:** Use sorted mode unless you have a strong reason not to. Be sure you have enough sorting space available. If there is not enough sort space available during ROLLBACK processing, the operation will terminate with an error. The database will not be affected.

**When to use VERIFY:** Specify VERIFY only when rolling back an intact database. If you specify VERIFY when recovering from an abended job or an abended central version, the ROLLBACK utility will terminate with an error.

VERIFY is ignored if you specify SORTED.

**Recovering large numbers of files:** For OS/390 users, the maximum number of files that can be accessed by a central version is greater than the number that can be accessed by a local mode batch job. This has implications for manual recovery. If more than 3,273 files must be recovered, it will be necessary to execute the ROLLBACK utility multiple times in separate job steps, recovering a subset of the areas or segments in each execution.

**When to use HOTBACKUP:** Specify HOTBACKUP when restoring a database from a hot backup. Refer to *CA-IDMS Database Administration* for more information about hot backup.

### 4.27.3 JCL considerations

When you submit a ROLLBACK statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The tape or journal file
- All files that map to areas being restored
- Any file that includes space management pages of areas mapped to by files being restored

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.27.4 Examples

The following example directs the ROLLBACK utility to apply all the before images on the journal file for the EMPDEMO segment and to sort the before images by database key.

```
rollback segment empdemo all sorted;
```

## 4.27.5 Sample output

When the ROLLBACK operation is completed, the following listing is generated.

```
IDMSBCF                                IDMS Batch Command Facility                                09/18/99  PAGE 1
ROLLBACK SEGMENT EMPDEMO ALL SORTED;

RECORDS RESTORED TO AREA EMPDEMO.EMP-DEMO-REGION                734
RECORDS RESTORED TO AREA EMPDEMO.INS-DEMO-REGION                169
RECORDS RESTORED TO AREA EMPDEMO.ORG-DEMO-REGION                438

TOTAL RECORDS RESTORED                1341

BLOCK COUNT          0 BACKWARD          702
RECORD COUNT         0 BACKWARD          5512
Status = 1           Extended Reason Code = 2367      Messages follow:
DB002367 C1M353: Warning: area EMPDEMO.EMP-DEMO-REGION was not locked.
DB002367 C1M353: Warning: area EMPDEMO.INS-DEMO-REGION was not locked.
DB002367 C1M353: Warning: area EMPDEMO.ORG-DEMO-REGION was not locked.
```

## 4.27.6 For more information

- **On journal files**, see *CA-IDMS Database Administration*.
- **On database recovery**, see *CA-IDMS Database Administration*.





**rollforward-options**

Specifies options used to rollforward from a standard archive journal tape.

Expansion of rollforward-options immediately follows the main statement syntax.

**FROM EXTRACT**

Specifies that an extract journal will be used for input instead of a standard archive journal tape.

This option excludes use of the SORTED/SEQUENTIAL, VERIFY, ALL/COMPLETED, and STOP AT options because these options have no effect on the already created extract file.

**ddname**

Specifies the external name for the input extract file.

**COMPLETED**

- If you do not specify a STOP time, after images are applied only for transactions that have been completed before the end of the journal file.
- If you do specify a STOP time, after images are applied for all transactions that have been completed up to the first checkpoint record after the STOP time.

**ALL**

- If you do not specify a STOP time, after images are applied for all transactions in the journal file.
- If you do specify a STOP time, after images are applied until a point at or after the specified time where there are no active transactions. If no such point is found, processing stops at the end of the journal file.

**SEQUENTIAL**

After images are applied in the sequence in which they occur in the journal file. SEQUENTIAL is the default.

**Note:** Backward file reads are not supported under VM/ESA. If an aborted transaction is encountered, a backward read may be attempted, in which case it will fail. For this reason, it is safer for VM/ESA users to specify SORTED.

**SORTED**

Specifies that after images are to be sorted in the journal file by database key and only the latest after image for any database key is applied.

All the database keys on a given database page will be copied back together. Thus, each page will be accessed once.

By default, if you do not specify SORTED, the after images will be copied back sequentially.

**Note:** You cannot use the SORTED option with VSAM KSDS files. A native VSAM KSDS file must be recovered using the SEQUENTIAL option.

**VERIFY**

Indicates that the validity of the specified conditions should be checked. If the VERIFY option is not specified, no checking is performed. If VERIFY is specified without DATABASE or QUIESCE, DATABASE is the default.

- **DATABASE** — Specifies that before images are to be verified in the journal file before the after images are applied. If a before image in the journal file does not match the contents of the database, the roll forward operation will terminate with an error.

By default, if database validity checking is not in effect, after images are copied back without first verifying the before images.

- **QUIESCE** — Specifies the level of quiesce point verification that will be performed.
  - **LIMITED** — Specifies that limited quiesce point verification be performed. This is the default.
  - **FULL** — Specifies that the strictest form of quiesce point verification be performed.

**START AT**

Specifies that the operation should start only after reaching a specified date and time in the journal file. Only images for transactions that begin after the specified start time will be applied. By default, processing starts at the beginning of the journal file.

**STOP AT**

Specifies that the operation should stop as soon as possible after reaching a specified date and time in the journal file.

By default, processing stops when the end of the journal file is reached.

**date**

Specifies the date, in one of the following formats:

- *yyyy-mm-dd*
- *mm/dd/yyyy*
- *dd.mm.yyyy*

In these formats, the following rules apply:

- **Yyyy** specifies the year. *Yyyy* must be an integer in the range 0001 through 9999. Leading zeros are optional.
- **Mm** specifies the month within the year. *Mm* must be an integer in the range 01 through 12. Leading zeros are optional.
- **Dd** specifies the day within the month. *Dd* must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of *yyyy*, *mm*, and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

**date-time**

Specifies the date and time, where:

- The format for specifying the DATE-TIME is:

*yyyy-mm-dd-hh.mm.ss.ffffff*

- The rules for specifying the DATE component of DATE-TIME are the same as for DATE described above. The rules for specifying the TIME component of DATE-TIME are:
  - *Hh* specifies the hour on a 24-hour clock. *Hh* must be an integer in the range 00 through 23. Leading zeros are optional.
  - *Mm* specifies the number of minutes past the hour. *Mm* must be an integer in the range 00 through 59. Leading zeros are optional.
  - *Ss* specifies the number of seconds past the minute. *Ss* must be an integer in the range 00 through 59. Leading zeros are optional.
  - *Fffffff* specifies the number of millionths of a second past the specified second. *Fffffff* is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

**time**

Specifies the time in one of the following formats:

- *hh.mm.ss*
- *hh:mm:ss*

The rules for specifying TIME are the same as those listed for DATE-TIME above.

When TIME is specified, the date defaults to the current date.

## 4.28.2 Usage

**How to submit ROLLFORWARD:** You submit the ROLLFORWARD statement to CA-IDMS/DB only through the batch command facility. When submitting ROLLFORWARD statements, you must run the batch command facility in local mode.

Vary any area being rolled forward offline to all central versions.

**When to use ROLLFORWARD:** The ROLLFORWARD utility is most commonly used to restore a database to its condition before some hardware problem corrupted the database. In this case, use a backup copy of the database that was created before the problem occurred.

You must have a backup of the database from a time before the problem, and all relevant journals.

►► For more information on when and how to use the ROLLBACK and ROLLFORWARD utilities, see *CA-IDMS Database Administration*.

**How ROLLFORWARD works:** The ROLLFORWARD operation starts with a former state of the database, as captured by a backup. It uses after images from a tape or journal file to restore progressively later states of the database, file, segment or area.

**ROLLFORWARD uses one journal file:** The journal file is a sequential file which may reside on disk or tape depending on which options are specified. If you specify SEQUENTIAL, the journal file must reside on tape; if you specify SORTED, the journal file may reside on disk or tape. If you have multiple journal files, you must consolidate them, in the order in which they were created, to a single file; and use the single file with the ROLLFORWARD utility.

If the file requires more than one tape volume, the volumes must be mounted in order.

**ROLLFORWARD and VSAM files:** You can run the ROLLFORWARD utility against native VSAM files.

**When to use sorted mode:** Use sorted mode unless you have a strong reason not to. Be sure you have enough sort space available before executing the ROLLFORWARD utility. If, however, you run out of sort space, the ROLLFORWARD operation will terminate with an error and the database will not be affected.

**Note:** You cannot use sorted mode when applying the journal images created while a hot backup was in progress. For more information on recovering from a hot backup, refer to *CA-IDMS Database Administration*.

### ROLLFORWARD FROM EXTRACT

- **Purpose** — Much of the work of recovery can be performed by EXTRACT JOURNAL during a noncritical time. The actual recovery done by ROLLFORWARD will be relatively fast because the extract file will contain only the records needed for recovery, in correct sort order.
- **Extract data** — Data in the extract file is selected and sorted before the actual recovery is done by ROLLFORWARD. A second sort is performed by ROLLFORWARD so that only the most recent image for each db-key is applied to the database.
- **Multiple input files** — Multiple input files can be processed as standard concatenated files; they do not need to be merged into one file before processing.
- **Tape file** — Since the extract file does not need to be read backwards, it does not need to be on tape.

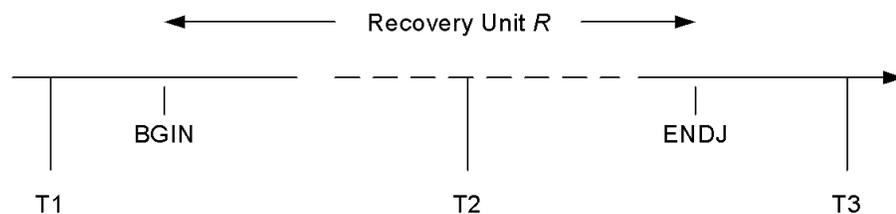
**Controlling the starting point of the rollforward operation:** If no START AT parameter is specified, the rollforward operation starts with the first journal image on the input file(s). If START AT is specified, the rollforward operation starts with the first checkpoint record (BGIN, COMT, ENDJ, ABRT, or CKPT) whose timestamp is on or after the specified start time.

Specifying a start time may save recovery time and also circumvent issues associated with aborted recovery units that span the start of the input file. It further enables the rollforward operation to begin processing at a quiesce point that does not correspond with a journal file boundary.

**Quiesce point verification:** A quiesce point is a point in time during which there is no update activity for some portion of a database. In order to perform a correct recovery, you must begin the recovery operation at a quiesce point for the portion of the database being recovered. In order to assist in this effort, the rollforward utility provides two levels of quiesce point verification: limited and full.

Quiesce point verification is always performed during a rollforward operation. This is done by checking for the existence of spanned recovery units which are recovery units that are active at the start of the rollforward operation. A recovery unit is represented by journal images starting with a BGIN or COMT checkpoint and ending with a COMT, ENDJ, or ABRT checkpoint. If a spanned recovery unit updates the specified portion of the database, then the rollforward operation is not starting at a quiesce point. If this situation is detected, the rollforward operation will terminate with an error.

However, it is not always possible to know whether a spanned recovery unit affects the specified portion of the database or not. If the initial BGIN or COMT checkpoint record for a recovery unit is not contained on the archive file being processed, then it is not possible to determine whether it updated the specified portion of the database. Such a recovery unit is referred to as an indoubt recovery unit.



The above time line illustrates what is meant by an indoubt recovery unit. The journal images for recovery unit *R* are written to the journal file at the times shown. If the archive file includes images starting at time *T1*, then *R* is not an indoubt recovery unit because the archive file contains all journal images written for *R* since its inception. Similarly, if the archive file starts at time *T3*, *R* is not an indoubt recovery unit, because the archive file contains no images for *R* whatsoever. However, if the archive file starts at time *T2*, then *R* is an indoubt recovery unit, since the archive file does not contain all journal images written since its inception.

If an indoubt recovery unit does not span the start of the rollforward operation, its existence doesn't matter. But if an indoubt recovery unit is also a spanned recovery unit then the rollforward operation may not be starting at a quiesce point.

The action taken if an indoubt spanned recovery unit is encountered depends on whether limited or full quiesce point verification is in effect. Under full verification, the rollforward operation will terminate with an error. Under limited verification, a

warning message will be issued identifying the recovery unit, but processing will continue.

Warning messages produced under limited verification should be examined to ensure that the identified recovery units in fact did not affect the specified portion of the database. If there is any doubt, the PRINT JOURNAL utility statement should be used to gain more information about the indoubt recovery units. If after researching the situation, it is found that an indoubt recovery unit did update the specified portion of the database, the affected portion of the database must be restored and the rollforward operation repeated. You must locate a quiesce point corresponding to a backup of the specified portion of the database and begin the rollforward operation from that point.

**When to use full or limited verification:** Full quiesce point verification should only be used if you expect that no indoubt spanned recovery units are active at the starting point of the rollforward operation. The only way to guarantee this is to process archive files that were created immediately following a quiesce of update activity across all areas. One way to establish such a quiesce point is to shutdown a central version. Another way is to use the DCMT QUIESCE command and specify a DBNAME that includes every area in the DMCL.

Limited quiesce point verification can be used when processing the archive files produced immediately following a quiesce operation for the portion of the database being recovered. One way to do this is to use the DCMT QUIESCE SEGMENT command to quiesce a segment and then use limited quiesce point verification when recovering all or a portion of that segment.

**When to use VERIFY DATABASE:** Specify VERIFY DATABASE only when rolling forward from an intact database.

**Recovering large numbers of files:** For OS/390 users, the maximum number of files that can be accessed by a central version is greater than the number that can be accessed by a local mode batch job. This has implications for manual recovery. If more than 3,273 files must be recovered, it will be necessary to execute the ROLLFORWARD utility multiple times in separate job steps, recovering a subset of the areas or segments in each execution.

### 4.28.3 JCL considerations

When you submit a ROLLFORWARD statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The tape or journal file, or journal extract file
- All files that map to areas being restored
- Any file that includes space management pages of areas mapped to by files being restored
- Sort work and message files if the sort option is selected or if recovering from a journal extract file

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.28.4 Examples

The following example directs the ROLLFORWARD utility to restore the EMPDEMO segment using the SORTED option.

```
rollforward segment empdemo all sorted;
```

The following example directs the ROLLFORWARD utility to read an extract file from SYS005 and apply only the images that belong to file USERDB.EMPF1:

```
rollforward file userdb.empf1 from extract sys005;
```

## 4.28.5 Sample output

The ROLLFORWARD utility generates the following listing after executing the statement in the first example above.

IDMSBCF	IDMS Batch Command Facility	09/18/99	PAGE 1
ROLLFORWARD SEGMENT EMPDEMO ALL SORTED;			
RECORDS RESTORED TO AREA EMPDEMO.EMP-DEMO-REGION		1,030	
RECORDS RESTORED TO AREA EMPDEMO.INS-DEMO-REGION		199	
RECORDS RESTORED TO AREA EMPDEMO.ORG-DEMO-REGION		581	
TOTAL RECORDS RESTORED	1810		
BLOCK COUNT	702 BACKWARD	0	
RECORD COUNT	5512 BACKWARD	0	
Status = 1	Extended Reason Code = 2367	Messages follow:	
DB002367 C1M353:	Warning: area EMPDEMO.EMP-DEMO-REGION was not locked.		
DB002367 C1M353:	Warning: area EMPDEMO.INS-DEMO-REGION was not locked.		
DB002367 C1M353:	Warning: area EMPDEMO.ORG-DEMO-REGION was not locked.		

## 4.28.6 For more information

- **On database recovery**, see *CA-IDMS Database Administration*.
- **When to use the ROLLFORWARD utility**, see *CA-IDMS Database Administration*.



**table-name**

Identifies the table that is constrained by an indexed constraint.

**con-specification**

Identifies constraints on the current table that is being tuned.

If omitted, all indexed constraints on the current table are processed.

**sql-segment.area-name**

Identifies an sql-defined area to be selected for processing.

All tables with indexed constraints in the area are processed.

**dbname**

Identifies the dbname to be used when binding the subschema.

**ss-name**

Identifies the subschema to be used for processing a non-SQL database.

**set-name**

Identifies the indexed sets within the subschema that are to be processed.

If omitted, all linked indexed sets defined in the subschema are processed.  
(Linked indexed sets are indexed sets with index pointers.)

**cmt-interval**

Specifies the commit interval. After every cmt-interval record read, a commit is issued. If omitted, the default interval is 100. If specified as zero (0), no commits are issued.

**not-interval**

Specifies the notify interval. After every not-interval record read, a message is issued to the console stating how far the job has progressed. If omitted, the default interval is 10,000 records. If specified as zero (0), no notify message is issued.

**cons-name**

Identifies an indexed constraint on the current table that is to be tuned.

## 4.29.2 Usage

The Tune Index Utility has the following usage considerations:

- In order to use the feature, you must execute the utility specifying one of the following:
  - One or more tables whose indexed constraints are to be tuned
  - One or more areas containing tables whose indexed constraints are to be tuned
  - A subschema and DBNAME and optionally a list of indexed sets to be tuned
- If multiple indexes and/or multiple tables are processed in the same area, increasing the number of buffers will further improve performance. Alternatively, restrict processing to a single constraint or indexed set per statement.

**Orphan elimination:** This utility eliminates orphans only at the bottom level of the index. After execution, an index may still contain orphans at intermediate levels. There may also be a small number of orphans left at the bottom level.

**When to use:** Orphan adoption only occurs for index sets with index pointers (referred to as linked indexes). The Tune Index Utility has no affect on unlinked indexes. Because all SQL-defined indexes are unlinked, there is no benefit in running this utility on such indexes, and only indexed constraints are processed.

**Operating modes:** You can execute this utility both online (through the online command facility) and in batch through central version or batch local.

**Commit interval:** You can specify a commit interval that determines the frequency with which the utility will commit. The interval specifies the number of possible (not actual) updates that may take place before a commit is issued. If a deadlock occurs, the utility automatically restarts execution at the last commit point. You can disable committing and automatic restart by specifying a 0-commit interval.

**Notify interval:** You can specify a notify interval that determines the frequency with which the utility will write a notification message. The interval specifies the number of records/rows read before a message is written. The message is written to the job log and the operator's console. You can disable notification by specifying a 0-notify interval.

### 4.29.3 JCL considerations

When you submit a TUNE INDEX utility to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the files containing the areas to be processed.

### 4.29.4 Example

The following example directs the TUNE INDEX utility to adopt orphaned index records in the EMPDEMO dbname using subschema EMPSS01:

```
tune index for dbname emdemo subschema empss01;
```

### 4.29.5 Sample output

The following is a sample of a report produced by the Tune Index Utility:

```
IDMSBCF 15.0                      CA-IDMS Batch Command Facility

TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01;
Status = 0      SQLSTATE = 00000      Messages follow:
DB002994 C0M333: IDMSTUNE - processing started
DB002994 C0M333: IDMSTUNE - 342 records read for area EMP-DEMO-REGION
DB002994 C0M333: IDMSTUNE - 20 SR8s adopted for area EMP-DEMO-REGION
DB002994 C0M333: IDMSTUNE - 123 records read for area ORG-DEMO-REGION
DB002994 C0M333: IDMSTUNE - 35 SR8s adopted for area ORG-DEMO-REGION
DB002994 C0M333: IDMSTUNE - 465 total records read
DB002994 C0M333: IDMSTUNE - 55 total SR8s adopted
DB002994 C0M333: IDMSTUNE - 5 indexes/sets processed
DB002994 C0M333: IDMSTUNE - processing completed

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

### 4.29.6 For more information

- **On indexed constraints**, see *CA-IDMS SCL Reference*.
- **On dbnames, subschemas, and sets**, see *CA-IDMS Database Administration*.



**subschema-name**

The name of the subschema.

By default, if a *subschema-name* is not specified, it indicates that an SQL-type segment is to be unloaded. In this case, the UNLOAD utility builds the required subschema for unload processing.

**AREA**

Specifies an area to unload.

By default, if you do not specify any areas, all areas defined by the specified subschema will be unloaded.

**area-name**

The name of the area.

**RELOAD INTO**

Specifies the segment in which the data will be reloaded.

This parameter is passed in a control record at the beginning of the RELDCTL file to be used by the RELOAD utility.

**segment-name**

The name of the segment.

If you unload an SQL-defined database, *segment-name* must be the same as the one specified in the SEGMENT clause.

**USING**

Specifies the subschema that describes the segment in which the data will be reloaded.

This parameter is passed in a control record at the beginning of the RELDCTL file to be used by the RELOAD utility statement.

By default, if you do not specify a reload subschema, the subschema that was used to unload the data is used to reload the data.

**subschema-name**

The name of the subschema.

**DMCL**

Specifies the DMCL that defines both the segment whose data is being unloaded and the segment the data will be reloaded into. See "Usage, The unload/reload DMCL" for additional information.

This parameter is passed in a control record at the beginning of the RELDCTL file to be used by the RELOAD utility statement.

By default, if you do not specify a DMCL, the DMCL used during the unload process, as specified in the SYSIDMS parameter file, is used. If you did not specify a DMCL in the SYSIDMS parameter file, the IDMSDMCL name is used.

**dmcl-name**

The name of the DMCL.

## 4.30.2 Usage

**How to submit the UNLOAD statement:** You submit the UNLOAD utility to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**Using UNLOAD on an SQL-defined database:** You can use the UNLOAD utility statement to unload and reload an SQL-defined database to make these changes:

- Change area page ranges
- Change page size
  - ▶▶See 4.7, “EXPAND PAGE” for complete information on changing a page size.
- Change the maximum number of records per page

The page changes are specified in the DMCL definition.

**Note:** The modified DMCL must be available during the unload operation as well as during the reload operation.

You cannot make changes to table definitions in an SQL-defined database between the unload and reload steps.

**Using UNLOAD on a non-SQL defined database:** You unload and reload a non-SQL defined database in order to modify the database based on changes made to the schema or segment definition. These changes must be reflected in the subschema and/or DMCL used for the reload operation. An SQL-defined database must be reloaded into the same segment. You cannot reload it into another segment.

**Note:** The modified DMCL and/or subschema must be available during the unload operation as well as during the reload operation. You can make the following changes by using UNLOAD and RELOAD:

- Change area page ranges
- Change page size
  - ▶▶See 4.7, “EXPAND PAGE.” for complete information on changing a page size.
- Change page ranges for record types
- Reassign records to different areas
- Change record placements within an area
- Change record location modes
- Store VIA records by means of a different set (as long as they already participate in the set)
- Change compression and INDEX BLOCK CONTAINS options for indexes
- Change area and page-range assignments for system-owned indexes
- Change the maximum number of records per page

**UNLOAD and ASF databases:** If the UNLOAD utility is to be run against an ASF data or definition area, refer to the *CA-IDMS ASF User Guide* for more information.

**When not to use UNLOAD:** The areas processed by the UNLOAD utility cannot contain any logically deleted records. Therefore, do not use it until you have removed all logically deleted records with the CLEANUP utility statement.

During UNLOAD and RELOAD processing, record formats are preserved that is, the layout of data within a record and record positions within sets. This limits the modifications that you can make during an unload/reload operation. For example, you cannot use an unload/reload operation to:

- Remove a set
- Remove a record type
- Insert new data fields into a record
- Change the order of a sorted set
- Connect records to new sets
- Change or delete record IDs
- Change the size, location, or data type of sort or index keys
- Change a set from unordered to ordered

These type of changes require other utilities, such as RESTRUCTURE, and possibly user-written programs.

**Using UNLOAD and Mixed Page Groups:** UNLOAD cannot process mixed page groups and will issue an error message if mixed page groups are encountered. You must use multiple invocations of the utility to process different page groups.

**UNLOAD and the CLEANUP utility:** If the area being unloaded contains logically deleted records, run the CLEANUP utility before running unload.

►► For further information on making these kinds of changes, see *CA-IDMS Database Administration*.

### 4.30.2.1 How UNLOAD works

During UNLOAD processing, an area sweep is performed, unloading all CALC, DIRECT, and unowned VIA records. At the same time, the area sweep extracts information for all indexes regardless of whether or not the index has been specified for the data area being unloaded.

After unloading the owner of a VIA set, the area sweep is interrupted to unload all the members of the set.

When unloading a CALC record for which duplicates are allowed, the CALC set is walked in the prior direction to determine the relative position of the record with respect to other record occurrences with duplicate keys.

**Non-SQL defined sets can be partially unloaded:** Records in an area being unloaded or reloaded can have set connections (as owners or members) to records in areas not being processed in the same run.

For example, a CALC record in an area being unloaded can own a VIA member in an area that is not being unloaded.

The UNLOAD utility will keep track of the set connections, and the connections will be rebuilt when the records are reloaded.

**Note:** UNLOAD cannot process mixed page groups and will issue an error message if mixed page groups are encountered. You must use multiple invocations of the utility to process different page groups.

**SQL-defined segments:** Area and table stamps are unloaded during the UNLOAD steps of an SQL-defined database and are reloaded during RELOAD processing. For this reason, you must format the affected areas using the FILE option between the unload and reload operations.

The one exception to this is when unloading and reloading the DDLCATLOD area in a segment defined for SQL. In this case, either format by area or use the INSTALL STAMPS utility before reloading the data.

**DIRECT records and their VIA clusters:** Since DIRECT records have been placed on a specific page by a user, it may be difficult for the UNLOAD utility to determine where they should be placed in the new area. If a DIRECT record's old page exists in the new area, the RELOAD attempts to place the occurrence on the same page. However, if the old page does not exist in the new page range, the occurrence is targeted to the low page of the new area.

If a DIRECT record owns a VIA record, the members are stored in their new area proportional to their owner's position in its old area. If the owners and members reside in the same area and the page range has been extended or completely changed, the VIA cluster may not be on the same page as the owner. If this is not acceptable, the user may want to consider using a user-written program to unload the database and the FASTLOAD utility to reload the area.

### 4.30.2.2 The unload/reload subschemas

The UNLOAD utility uses information in the dictionary when processing an SQL-defined database. When processing a non-SQL defined database, the unload and reload subschemas must include the items listed below.

**The unload subschema:** The subschema used in an unload operation must:

- Include the areas being unloaded

- Include all areas containing records with set connections to records in the areas being unloaded
- Define all record types in the areas being unloaded
- Define all sets in which the record types participate
- Define all record types which can participate in the defined sets

**The reload subschema:** The subschema used in a reload operation (specified by RELOAD INTO..USING) must:

- Define the same record types and sets as the unload subschema
- Include the areas to be reloaded and all areas containing records with set connections to those areas.
- Allow all included areas to be readied in exclusive update mode

**Unloading a dictionary:** When unloading a dictionary, the IDMSNWKU subschema should be used for all areas, except DDLCAT, DDLCATX, and DDLCATLOD.

The DDLCAT and DDLCATX areas should be unloaded and reloaded using the IDMSCATZ subschema.

The DDLCATLOD area should be unloaded and reloaded using the IDMSCATL subschema. Before reloading the DDLCATLOD area in a segment defined for SQL, you must install stamps either by formatting by area or using the INSTALL STAMPS utility.

### 4.30.2.3 The unload/reload SEGMENT

The SEGMENT used in the unload and reload operation must include:

- The area(s) being unloaded
- All areas with physical connections to the area(s) being unloaded. This includes:
  - Areas containing indexes on records or tables being unloaded
  - Areas with set connections to areas being unloaded
  - Areas containing tables related through a linked constraint to tables being unloaded

### 4.30.2.4 The unload/reload DMCL

The DMCL used in the unload and reload operations must include the segment whose areas are being unloaded.

### 4.30.2.5 General procedure for UNLOAD and RELOAD

Take the following steps to unload and reload a database:

**Steps to follow**

1. Create the new subschema or DMCL reflecting the changes to be made
2. Backup all areas to be unloaded and also areas linked to those areas through sets or linked constraints
3. Execute the CLEANUP utility if required
4. Execute the UNLOAD utility
5. Format the areas into which the data will be reloaded
6. Execute the RELOAD utility
7. Backup the same areas as in step 2

The above procedure may vary slightly depending on the type of change being made.

**Output files:** The following output files and sort parameters are generated by the UNLOAD utility for use by the RELOAD utility statement:

File	Contents	Size
SYS002	A record occurrence descriptor for each unloaded record occurrence.	For each occurrence descriptor: 24 bytes plus the size of the data portion of the record occurrence.
	A set descriptor for each unloaded record occurrence and for each record occurrence that is not unloaded but has a set connection to an unloaded record occurrence.	For each set descriptor: 24 bytes plus 12 times the number of pointers associated with the record, up to 32 pointers. If the record has more than 32 pointers, it will get an additional set descriptor.
SYS003	An index key occurrence descriptor for each system owned index on a record occurrence.	For each index on each record occurrence: 32 bytes plus the size of the symbolic keys in the index.
SYSPCH	Sort parameters	
RELDCTL	A control record with information about the subschema, DMCL, and segment to be used by RELOAD.	60 bytes
	A set descriptor for each set in the subschema.	60 bytes

### 4.30.3 JCL considerations

When you submit an UNLOAD utility to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The files containing the areas to be processed
- The journal file(s), which can be dummied
- The output files

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.30.4 Example

The following example directs the UNLOAD utility to unload an area in an SQL-defined segment named USERDB.

```
unload segment userdb
area "emp-area"
reload into userdb dmcl newdmcl;
```

### 4.30.5 Sample output

The UNLOAD utility generates the following listing after successful completion of the example above.

```

IDMSBCF                                IDMS Batch Command Facility                                09/12/99  PAGE 1

SET BATCH WIDTH PAGE 80 ;
Status = 0
  UNLOAD SEGMENT USERDB
    AREA "EMP_AREA"
  RELOAD INTO USERDB DMCL NEWDMCL ;
UT001002 UNLOAD WILL USE SS '*****' DMCL IDMSDMCL DBNAME USERDB
UT001003 RELOAD WILL USE SS '*****' DMCL NEWDMCL DBNAME USERDB
  SUBSCHEMA NAME ----- UNLD$SQL
  COMPILE DATE ----- 99-09-12
  COMPILE TIME ----- 09.54.26
  SUBSCHEMA VERSION --- 1200

IDMSUNL1 AREA RECORD DEPENDENCY TABLE FOLLOWS:
AREA-NAME      SET-NAME      RECORD-NAME:
EMPIX_AREA     MANAGES          SR7             - OWNER
EMPIX_AREA     EMPS             SR7             - OWNER
EMPIX_AREA     DEPS             SR7             - OWNER
ORG_AREA       DEPT_EML         DEPT            - OWNER
ORG_AREA       JOB_POS          JOB             - OWNER
UT004001 IDMSDBL1 RELEÅSE 15.0 TAPE CAGJF0 PROCESSING STARTED
  SUBSCHEMA NAME ----- UNLD$SQL
  COMPILE DATE ----- 99-09-12
  COMPILE TIME ----- 09.54.26
  SUBSCHEMA VERSION --- 1200

UT003002      67 INTERMEDIATE RECORDS WERE WRITTEN TO SYS003
UT004005     105 INTERMEDIATE RECORDS WERE WRITTEN TO SYS002
UT004006      36 CHARACTERS IN SMALLEST INTERMEDIATE RECORD
UT004007      96 CHARACTERS IN LARGEST INTERMEDIATE RECORD
UT004002 IDMSDBL1 RELEASE 15.0 PROCESSING COMPLETED
UT004003      NO ERRORS WERE ENCOUNTERED
UT001001 RECORDS UNLOADED: 74
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings

```

### 4.30.6 For more information

- On unloading and reloading a database, see *CA-IDMS Database Administration*.

## 4.31 UNLOCK

### 4.31.1 Description

**Purpose:** The UNLOCK utility removes the external lock on an area.

**Authorization:**

To	You need this privilege	On
Unlock an area	DBAWRITE	The area

#### Syntax

```

▶▶ UNLOCK [ AREA segment-name.area-name | SEGMENT segment-name ]

```

#### Parameters

##### area

Directs the UNLOCK utility to remove the lock, if present, on the specified area.

##### segment-name

The name of the segment associated with the area.

##### area-name

The name of the area.

##### SEGMENT segment-name

The name of the segment to be unlocked.

### 4.31.2 Usage

**How to submit the UNLOCK statement:** You submit the UNLOCK statement to CA-IDMS/DB only through the batch command facility. When submitting UNLOCK statements, you must run the batch command facility in local mode.

### 4.31.3 JCL considerations

When you submit an UNLOCK statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the file containing the first page of the area to be processed.

▶▶ Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

## 4.31.4 Examples

**Unlock by area:** The following example directs the UNLOCK utility to remove the external lock on the EMPDEMO.EMP-DEMO-REGION area.

```
unlock area empdemo.emp-demo-region;
```

**Unlock by segment:** The following example directs the UNLOCK utility to remove the external lock on the EMPDEMO.EMP-DEMO-REGION segment.

```
unlock segment empdemo;
```

## 4.31.5 Sample output

**Unlock by area:** When UNLOCK processing is completed, the following listing is generated.

```
IDMSBCF                                IDMS Batch Command Facility                09/18/99  PAGE 1
UNLOCK AREA EMPDEMO.EMP-DEMO-REGION;
Status = 1      Extended Reason Code = 2367      Messages follow:
DB002367 C1M353: Warning: area EMPDEMO.EMP-DEMO-REGION was not locked.

AutoCommit will COMMIT transaction

Command Facility ended with warnings
```

**Unlock by segment:** When UNLOCK processing is completed, the following listing is generated.

```
IDMSBCF 15.0      CA-IDMS Batch Command Facility    10/16/99  PAGE 1
UNLOCK SEGMENT EMPDEMO;
Area EMPDEMO.EMP-DEMO-REGION      was not locked
Area EMPDEMO.INS-DEMO-REGION      was not locked
Area EMPDEMO.ORG-DEMO-REGION      was not locked
Status = 0      SQLSTATE = 00000
```

## 4.31.6 For more information

- On area locks, see *CA-IDMS Database Administration*.

## 4.32 UPDATE STATISTICS

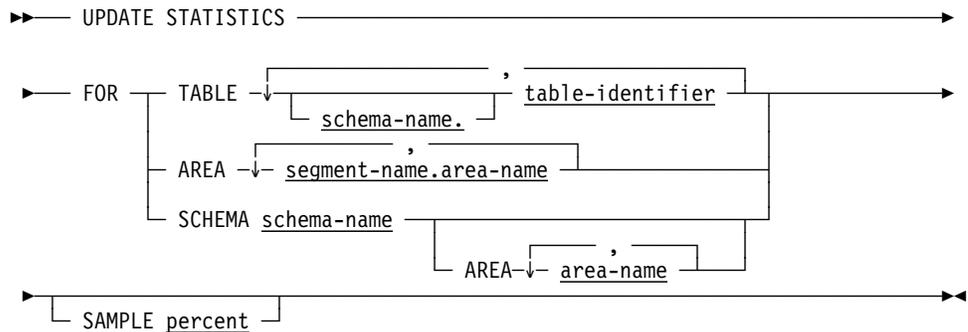
### 4.32.1 Description

**Purpose:** The UPDATE STATISTICS utility updates statistical information maintained in the dictionary for one or more tables. CA-IDMS/DB uses this information when determining the optimal access strategy for processing SQL statements.

**Authorization:**

To update statistics for	You need this privilege	For
A table	ALTER	The table
All tables in an area	DBAREAD	The area
Non-SQL schema	ALTER	All tables processed in the schema

**Syntax**



**Parameters**

**FOR**

Identifies the tables or areas for which the UPDATE STATISTICS utility is to update statistics.

**TABLE**

Specifies one or more SQL-defined tables and non-SQL defined tables for which the UPDATE STATISTICS utility is to update statistics.

**schema-name**

The name of the schema associated with the named table.

**table-identifier**

The identifier of a base table defined in the dictionary.

**AREA**

Updates statistics for all the tables in one or more specified areas.

**segment-name**

The name of the segment associated with the area.

**area-name**

The name of the area.

**SAMPLE**

Specifies the percentage of the pages in an area the UPDATE STATISTICS utility is to examine when calculating statistical information about one or more tables in the area.

**percent**

An integer in the range 1 through 100.

By default, if you do not specify a percentage, the UPDATE STATISTICS utility will examine all the pages in the specified area.

**SCHEMA schema-name**

Identifies that SQL-defined schema that references a nonSQL defined-schema.

**AREA area-name**

Identifies areas of the nonSQL schema from which to collect statistics.

## 4.32.2 Usage

**How to submit the UPDATE STATISTICS statement:** You submit the UPDATE STATISTICS statement to CA-IDMS/DB either online or through the batch command facility. If you submit it through the batch command facility, you should execute it through the central version, because the dictionary is updated as part of processing the statement.

**Journal in local mode:** If you are running CA-IDMS/DB in local mode, you can journal while updating statistics.

**These statistics are updated:** When updating statistics for a table, the UPDATE STATISTICS utility also updates statistics about:

- The indexes defined on the table
- The referential constraints in which the table is the referenced table
- The area associated with the table

**Which pages are examined:** When you specify a percentage of less than 100 in the SAMPLE parameter, the UPDATE STATISTICS utility selects the pages to be examined from across the entire area. For example, if you specify SAMPLE 50, the UPDATE STATISTICS utility examines every other page throughout the entire area, rather than every page in the first half of the area).

**UPDATE STATISTICS extrapolates the statistics:** When calculating statistics based on less than 100 percent of the pages in an area, the UPDATE STATISTICS utility extrapolates the values for the entire area from the values based on the percentage examined. For example, if you specify SAMPLE 50, and the 50 percent of the pages that are examined in the area contain 80 rows of the table, the UPDATE STATISTICS utility assumes the table has 160 rows.

The statistics stored in the dictionary reflect the extrapolated values, not the raw data.

### 4.32.3 JCL considerations

When you submit an UPDATE STATISTICS statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include either:

- A SYSCTL file to direct execution to the central version
- or
- Statements to define:
  - The areas containing the table(s) being examined
  - The dictionary containing the table definitions
  - The journal files of the DMCL you are using

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.32.4 Example

**Updating statistics for specified tables:** The UPDATE STATISTICS statement below updates statistics for the MONTHLY\_BUDGET and PROPOSED\_BUDGET tables in the PROD schema. Since no percentage has been specified, the UPDATE STATISTICS utility will examine all the pages in the area associated with each table when calculating the statistics.

```
update statistics
  for table prod.monthly_budget, prod.proposed_budget;
```

**Updating statistics for all the tables in an area:** The UPDATE STATISTICS statement below updates statistics for all the tables in the SQLDEMO.EMPLAREA area.

```
update statistics
  for area sqldemo.emplarea;
```

### 4.32.5 Sample output

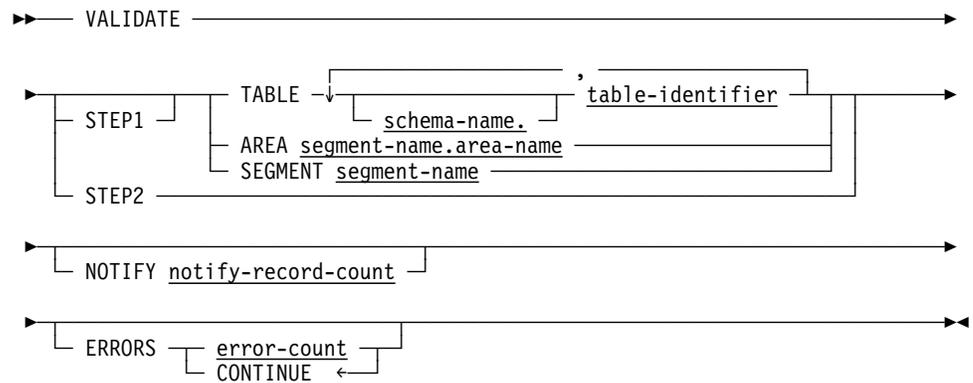
After successful completion of the UPDATE STATISTICS statement on the SQLDEMO.EMPLAREA area above, the following listing is produced.

```
IDMSBCF                                IDMS Batch Command Facility                09/18/99  PAGE 1
      UPDATE STATISTICS FOR AREA SQLDEMO.EMPLAREA;
      Status = 0
      AutoCommit will COMMIT transaction
      Command Facility ended with no errors or warnings
```

### 4.32.6 For more information

**On updating statistics**, refer to *CA-IDMS Database Administration*.



**Parameters****TABLE**

Identifies the referencing table to validate.

**schema-name**

The name of the schema that defines the table.

**table-identifier**

The identifier of the table.

**AREA**

Identifies the area containing tables to be validated. All referencing tables in the area will be validated.

**segment-name**

The name of the segment containing the area.

**area-name**

The name of the area.

**SEGMENT**

Identifies the segment containing tables to be validated. All referencing tables in the segment will be validated.

**segment-name**

The name of the segment.

**NOTIFY**

Directs the VALIDATE utility to send a message to the operator whenever a specified number of rows are processed.

The message states the phase and step currently being executed and the number of records processed.

**notify-record-count**

The number of rows to validate before sending a message.

**ERRORS**

Directs the VALIDATE utility to either continue the validation when errors are detected or stop after a specified number of errors are detected.

By default, processing will not stop when errors are detected.

Detected errors are listed in the report generated by the VALIDATE utility.

**error-count**

The number of errors to detect before terminating.

**CONTINUE**

Directs the VALIDATE utility to continue processing regardless of the number of errors detected.

CONTINUE is the default.

**STEP1**

Validates only linked referential constraints and unlinked index-to-index referential constraints.

If other unlinked referential constraints are detected, VALIDATE STEP1 produces an intermediate work file to be used as input to VALIDATE STEP2. If no such file is produced, you do not need to run VALIDATE STEP2.

If you do not specify a STEP number, the VALIDATE utility will validate all linked and unlinked referential constraints, and is considered a complete VALIDATE.

**STEP2**

Validates all unlinked referential constraints except index to index referential constraints.

## 4.33.2 Usage

**How to submit the VALIDATE statement:** You submit the VALIDATE statement to CA-IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**When to use VALIDATE:** If you have loaded a group of tables using a phased or stepped LOAD and built the indexes and relationships of the tables specifying NO VALIDATE in the BUILD statement, use VALIDATE to ensure that referencing tables have valid references.

You can also use the VALIDATE utility at any time to validate the referential constraints of a table.

**VALIDATE utility uses intermediate work files:** STEP1 of the VALIDATE utility produces an intermediate work file to be used by STEP2. If you run a complete VALIDATE without separating STEP1 from STEP2, data is sorted in the intermediate file between the steps automatically. If you run a stepped VALIDATE, you must run the intermediate sorts.

**Note:** When running a complete VALIDATE, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to *different* intermediate files. The data that is output in SYS003 by STEP1 is input to STEP2 in SYS002.

**Sorting intermediate work files:** If you run the validate process in steps or phases, use the sort parameters in the SYSPCH file to sort the intermediate files.

The following table shows the output of the steps of the VALIDATE process:

Step	Output	Size
STEP1	SYS003	For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24
	SYSPCH contains sort parameters	80 bytes
STEP2	SYS003	For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24
	SYSPCH contains sort parameters	80 bytes

**When not to use VALIDATE:** If you loaded the tables with a complete LOAD, or if you did not specify NO VALIDATE in the BUILD statement, then the validation has already been done. There is no need to run the VALIDATE utility.

If the tables have no referential constraints, there is no need to run the VALIDATE utility.

►► For more information about referential constraints, refer to *CA-IDMS Database Administration*.

### 4.33.3 JCL considerations

When you submit a VALIDATE statement to CA-IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The dictionary containing table definitions
- The files containing the areas associated with the referencing tables to be processed
- The intermediate work files
- Sort work files are needed if doing a complete VALIDATE

►► Refer to the chapter pertaining to your operating system for generic JCL to execute the batch command facility.

### 4.33.4 Example

The example below instructs the VALIDATE utility to perform a validation check against sample tables M and M2. The validation was not performed when the BUILD utility was run against them.

```
validate table load.m,
           load.m2
           errors continue;
```

### 4.33.5 Sample output

The listing below was generated after validating sample tables M and M2 in the above example.

```
IDMSBCF                                IDMS Batch Command Facility

*DEBUG IDMS OFF
CONNECT TO SYSCAT;
Status = 0
SET BATCH
  HEADINGS OFF WIDTH PAGE 79 UNDERLINE '-'
  SQLCODE ERROR
  COMPRESS ON;

UNLOCK AREA SYSSQL.DDL CAT;
Status = 1      Extended Reason Code = 2367      Messages follow:
DB002367 C1M353: Area SYSSQL.DDL CAT was not locked.
UNLOCK AREA SYSSQL.DDL CATX;
Status = 1      Extended Reason Code = 2367      Messages follow:
DB002367 C1M353: Area SYSSQL.DDL CATX was not locked.

-- **** Load data into Tables ****
*DEBUG IDMS ON

VALIDATE TABLE LOAD.M,
           LOAD.M2
           ERRORS CONTINUE;
IDMSLOAD - CAGJF0  VALIDATE INDEXES STEP 1  99-11-07-12.55.26
IDMSLOAD - 0 records processed for table LOAD.M
IDMSLOAD - 0 records processed for table LOAD.M2
IDMSLOAD - 3 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - VALIDATE INDEXES STEP 1  processing completed

AutoCommit will COMMIT transaction

Command Facility ended with warnings
```

### 4.33.6 For more information

- **On designing linked and unlinked referential constraints**, see *CA-IDMS Database Design*.
- **On defining linked and unlinked referential constraints**, see *CA-IDMS Database Administration*.

# Chapter 5. Utility Programs

---

5.1	IDMSCALC	5-3
5.1.1	Description	5-3
5.1.2	Usage	5-3
5.1.3	Calling the IDMSCALC routine	5-3
5.1.4	For more information	5-4
5.2	IDMSDBAN	5-5
5.2.1	Description	5-5
5.2.2	Syntax	5-5
5.2.3	Input parameter statements	5-6
5.2.4	Usage	5-11
5.2.5	JCL Considerations	5-15
5.2.6	Example	5-16
5.2.7	Sample output	5-16
5.2.8	For more information	5-21
5.3	IDMSDIRL	5-22
5.3.1	Description	5-22
5.3.2	Syntax	5-23
5.3.3	Input parameter statements	5-23
5.3.4	Usage	5-24
5.3.5	JCL considerations	5-25
5.3.6	Examples	5-25
5.3.7	Sample output	5-25
5.3.8	For more information	5-25
5.4	IDMSLOOK	5-27
5.4.1	Description	5-27
5.4.2	Syntax	5-27
5.4.3	Input parameter statements	5-28
5.4.4	Usage	5-35
5.4.5	JCL considerations	5-36
5.4.6	Examples	5-36
5.4.7	Sample output	5-36
5.4.8	For more information	5-47
5.5	IDMSRPTS	5-48
5.5.1	Description	5-48
5.5.2	Syntax	5-51
5.5.3	Statements	5-52
5.5.4	Usage	5-60
5.5.5	JCL considerations	5-61
5.5.6	Examples	5-61
5.5.7	Sample output	5-62
5.5.8	For more information	5-68
5.6	IDMSRSTC	5-69
5.6.1	Description	5-69
5.6.2	Syntax	5-69
5.6.3	Input parameter statements	5-70
5.6.4	Usage	5-72
5.6.5	Example	5-74

---

5.6.6 Sample output . . . . .	5-74
5.6.7 For more information . . . . .	5-74

---

## 5.1 IDMSCALC

### 5.1.1 Description

**Purpose:** The IDMSCALC utility is a subroutine which can be called from a user written subroutine to determine the target page of a record, based on a user-supplied CALC key.

It is typically used to optimize the loading of data by allowing you to presort input in target page sequence.

### 5.1.2 Usage

**How IDMSCALC works:** IDMSCALC is implemented as a called subroutine. The utility returns to a user-written program a target page number for storage of a CALC record, based on a page range and CALC key value supplied by the program. The user program, which can be written in any language supporting a call statement, must build a single five-field fullword-aligned argument as outlined in the table below, then call IDMSCALC, passing the argument. IDMSCALC must be link edited with the calling program.

### 5.1.3 Calling the IDMSCALC routine

The example below shows how to call the IDMSCALC routine from a user-written program.

```
01  CALC-PARMS.  
   05  CALC-PAGE-TARGET      PIC S9(9) COMP.  
   05  CALC-PAGE-RANGE-HIGH  PIC S9(9) COMP.  
   05  CALC-PAGE-RANGE-LOW   PIC S9(9) COMP.  
   05  CALC-KEY-LENGTH       PIC S9(4) COMP.  
   05  CALC-KEY               PIC X(16).  
  
   MOVE 75001      TO CALC-PAGE-RANGE-LOW.  
   MOVE 75101      TO CALC-PAGE-RANGE-HIGH.  
   MOVE 16         TO CALC-KEY-LENGTH.  
   MOVE 'SMITH'    TO CALC-KEY.  
   CALL 'IDMSCALC' USING CALC-PARMS.  
   DISPLAY 'TARGET PAGE IS ' CALC-PAGE-TARGET.
```

**The IDMSCALC argument:** The following table outlines the five-field argument that a calling program must pass to IDMSCALC.

Field	Usage	Size	COBOL Picture	Description of Field
1 (Output)	Binary	4 bytes	PIC 9(9) COMP	Target page number for storage of the record.
2 (Input)	Binary	4 bytes	PIC 9(9) COMP	Number of the highest page on which the record can be stored.
3 (Input)	Binary	4 bytes	PIC 9(9) COMP	Number of the lowest page on which the record can be stored.
4 (Input)	Binary	2 bytes	PIC 9(4) COMP	Length, in bytes, of the CALC key value.
5 (Input)	Character	1-256 bytes	PIC X(nnn)	Value of the CALC key.

**Note:** The information in fields 2 and 3 of IDMSCALC must match the database definition for the record type as specified in the schema.

**Input:** Input to the IDMSCALC utility consists of the IDMSCALC argument with fields 2-5 initialized by the calling program.

**Output:** The IDMSCALC utility returns a target page number for storage of a CALC record.

#### 5.1.4 For more information

- **On how the CALC location mode works**, see *CA-IDMS Database Administration*.

## 5.2 IDMSDBAN

### 5.2.1 Description

**Purpose:** The database analysis utility, IDMSDBAN, analyzes the characteristics and structure of a CA-IDMS/DB database (both non-SQL and SQL-defined). The utility provides information useful for system tuning, database structuring, and capacity planning.

IDMSDBAN also verifies the integrity of:

- Page structures
- Line indexes
- Record lengths
- Record locations
- Set connections for:
  - Chained sets
  - Constraints
  - Indexed sets
  - The CALC set
  - Variable-length-record fragment chains

**Authorization:**

To	You need this privilege	On
Analyze an area	DBAREAD	The area
Analyze a set, constraint, or table	DBAREAD	The area where the set, constraint, or table resides

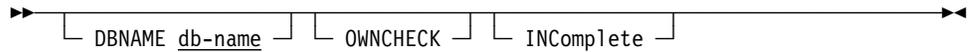
### 5.2.2 Syntax

**PROCESS statement**

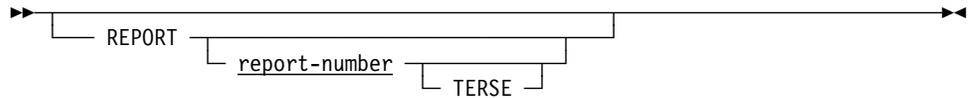
Code on one line only.

```

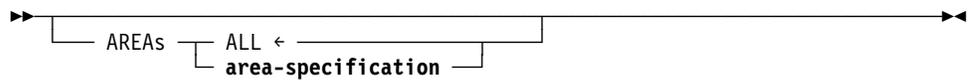
▶▶PROCESS [ SUBSCHEMA subschema-name subschema-specification ]
          [ SQL ]
          [ LOCKed ← ]
          [ UNLOCKed ]
  
```

*Expansion of subschema-specification***REPORT statement**

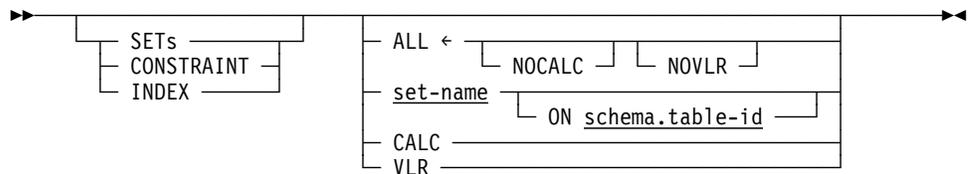
Code on one line only.

**AREA statement**

Code on one line only.

*Expansion of area-specification***SET statement**

Code on one line only.

**5.2.3 Input parameter statements**

IDMSDBAN processing is controlled by the following input parameter statements. If coded, the three types of statements must be coded in the order shown. If not coded, all reports are provided for all areas and sets for the named subschema or SQL-defined database.

**Statement descriptions:**

Statement	Required/ Optional	Description
PROCESS	Required	<p>For non SQL-defined database:</p> <ul style="list-style-type: none"> <li>▪ Identifies subschema and DMCL module</li> <li>▪ Specifies whether areas are to be locked before processing</li> </ul> <p>For SQL-defined database:</p> <ul style="list-style-type: none"> <li>▪ Identifies that an SQL-defined database will be processed</li> <li>▪ Specifies whether areas are to be locked before processing</li> </ul>
REPORT	Optional	Controls generation of reports 2, 3, 4, and 5
AREA	Optional	Specifies area(s) to be processed
SET	Optional	Specifies set(s) or constraints and indexes to be processed

#### PROCESS statement:

**Purpose:** The PROCESS statement identifies the database to be processed and specifies whether areas are to be locked before processing begins. One PROCESS statement is required for each IDMSDBAN run.

#### Syntax:

Code on one line only.

```

▶▶ PROCESS [ SUBSCHEMA subschema-name subschema-specification ]
           [ SQL _____ ]
           [ LOCKed ← ]
           [ UNLOCKed ]

```

#### Expansion of subschema-specification

```

▶▶ [ DBNAME db-name ] [ OWNCHECK ] [ INComplete ]

```

#### Parameters

##### SUBSCHEMA subschema-name

Identifies the subschema containing the areas, sets, and records to be processed. The subschema must contain complete descriptions for all of its sets and records.

**Note:** When processing the DDLCAT and DDLCATX areas, you must use the SUBSCHEMA parameter and the IDMSCATZ subschema name.

When processing the DDLCATLOD area you must use the SUBSCHEMA parameter and the IDMSCATL subschema name.

**DBNAME db-name:**

Identifies the name of the database to bind to at run time. If no DBNAME is specified, the default is the DBNAME specified in the SYSIDMS parameter file.

**OWNCHECK**

Indicates that checks for ownerless loops in chained sets is to be performed. If this parameter is omitted, IDMSDBAN will not check for chained sets that form a loop but do not include an owner record occurrence.

**Note:** This check will increase the run time of the utility and may require that the allocation for the SYS002 file in the IDMSDBN2 step be increased.

**INComplete**

Specifies that the named subschema does not include all record types. INCOMPLETE suppresses messages that would otherwise appear for record types not included in the subschema.

**SQL**

Specifies that an SQL-defined database is to be processed.

Based upon the segments identified on the AREA statement and the sets or constraints identified on the SET statement, IDMSDBAN will build a subschema to process an SQL-defined database.

**Note:** When processing the DDLCAT and DDLCATX areas, you must use the SUBSCHEMA parameter and the IDMSCATZ subschema name.

When processing the DDLCATLOD area you must use the SUBSCHEMA parameter and the IDMSCATL subschema name.

**LOCKed/UNLocked**

Specifies whether IDMSDBAN is to lock the areas to be processed:

- **LOCKED** (default) directs IDMSDBAN to lock the areas during IDMSDBN1 processing. When you specify LOCKED, the areas involved cannot be updated by another application during IDMSDBAN execution.
- **UNLOCKED** directs IDMSDBAN not to lock the areas to be processed. When UNLOCKED is specified, the areas involved can be updated during IDMSDBAN execution.

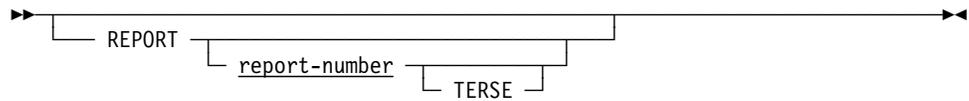
**REPORT statement:**

**Purpose:** The optional REPORT statement controls the generation of reports 2, 3, 4, and 5. IDMSDBAN always generates reports 1 and 1A.

By default, if you do not supply a REPORT statement or if you supply a REPORT statement without any parameters, IDMSDBAN will generate all reports.

**Syntax**

Code on one line only.



### Parameters

#### **report-number:**

Specifies the number of the report to be generated. Valid report numbers are 1, 1A, 2, 3, 4, and 5.

If no REPORTS statement is coded or if a REPORTS statement is provided without any report numbers, *all* reports are generated.

#### **TERSE**

Applies to reports 3, 4, and 5 only, as follows:

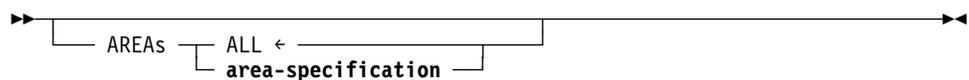
- TERSE causes **report 3** to be produced only for sets that have at least one non-empty set occurrence.
- TERSE causes **report 4** to be produced only for record types with at least one occurrence in the database.
- TERSE causes **report 5** to be produced only for sets with at least one occurrence in the database.

### AREA statement:

**Purpose:** The AREA statement specifies the areas to be processed. If no AREA statement is coded, all areas in the named subschema or segment are processed.

### Syntax

Code on one line only.



#### *Expansion of area-specification*



### Parameters

#### **AREAs**

Specifies the area(s) to process.

#### **ALL**

Specifies that all areas included in the specified subschema are to be processed.

ALL is the default.

You cannot use the ALL option if SQL was selected on the PROCESS statement.



- All chained and indexed sets defined in the subschema specified in the PROCESS statement
- All constraints within the specified segment
- The CALC set
- Sets of variable-length-record fragments located within the areas specified in the AREA statements

ALL is the default.

**NOCALC**

Removes the CALC set from the list of sets specified by ALL.

**NOVLR**

Removes the set of variable length record chains from the list of sets specified by ALL.

**set-name:**

Identifies the name of a set to be processed.

**ON schema.table-id:**

If SQL was specified on the PROCESS statement, you must identify the name of the SQL schema and table identifier the constraint or index is associated with.

You do not specify a *schema.table-id* for non-SQL defined databases. *Schema* is a 1 through 18-character value and *table-name:* is a 1 through 18-character value.

**CALC**

Directs IDMSDBAN to process the CALC set within the specified area(s).

**VLR**

Directs IDMSDBAN to process all sets of variable-length-record fragment chains located within the specified area(s).

## 5.2.4 Usage

**Input:** Input to the IDMSDBAN utility consists of statements to control the utility processing.

**Output:** The IDMSDBAN utility generates message listings.

**Execution mode:** You can execute the IDMSDBAN utility in local mode only.

**Usage considerations:**

**Specifying dictionary name for SQL-defined databases:** When processing an SQL-defined database, you must specify the dictionary name or catalog segment that IDMSDBAN connects to using the SYSIDMS DBNAME parameter (*not* the DICTNAME parameter).

For more information on the SYSIDMS parameter file, see *CA-IDMS Database Administration*.

**Subschema:** The subschema that describes the database areas to be processed must include all member record types for all sets included in the subschema.

**Processing DDLCAT, DDLCATX, and DDLCATLOD areas:** When processing the DDLCAT and DDLCATX areas, you must use the PROCESS statement SUBSCHEMA parameter and the IDMSCATZ subschema name.

When processing the DDLCATLOD area you must use the PROCESS statement SUBSCHEMA parameter and the IDMSCATL subschema name.

**Areas:** By default, IDMSDBAN locks the areas to be processed. If IDMSDBAN attempts to lock an area that is already locked, the utility terminates. To prevent this, perform one of the following actions:

- Either vary offline or vary to retrieval the areas to be processed before running IDMSDBAN.
- Specify UNLOCKED in the PROCESS parameter statement.

**Note:** If you specify UNLOCKED and the database is updated by another application during IDMSDBAN processing, the statistics in reports 2, 3, 4, and 5 may be inaccurate, and the messages in reports 1 and 1A may be misleading.

**Page ranges:** The page ranges selected for processing must include all database pages that can contain owner or member records of any set(s), constraints, or indexes to be processed.

If you specify a set, constraint, or index with an owner and/or member record outside the specified pages, IDMSDBAN will terminate with an error.

**Indexed sets:** If an indexed set is specified for processing, processing must also be requested for all other indexed sets for which the SR8 records can occur on the same pages as the SR8 records for the specified set.

**IDMSDBAN has two parts:** IDMSDBN1 and IDMSDBN2.

**Intermediate work file size:** The size of the intermediate work files will vary depending on the size and complexity of the database being analyzed.

- IDMSDBN1 will generate one output record for each pointer position in each record for each set being processed. The file can be written to tape or disk; however, to ensure adequate space, tape is recommended.
- IDMSDBN2 uses an intermediate work file to hold records it is working on. For every record it receives from IDMSDBN1, it will produce up to two intermediate work records. If the OWNCHECK option has been specified, it is necessary for the utility to produce these records for each occurrence of a record participating in a chained set.

**Tape management systems and IDMSDBAN work files:** The SYS002 work file used by IDMSDBN2 is written to and read from repeatedly. If this file is placed on a tape, then your tape management system may prevent IDMSDBN2 from overwriting the file after it has written to the file the first time. If the tape management system installation defaults allow, specify a zero retention period for the work file and/or specify DISP=(NEW,DELETE) for the work file.

**If problems are detected:** If the messages issued by IDMSDBAN indicate that problems exist in the database, the database should not be updated in between the time of the IDMSDBAN run and the time that the problems are corrected.

### **IDMSDBN1**

**What it does:** IDMSDBN1 sweeps each specified area. For each area, IDMSDBM1:

- Collects statistics
  - Detailed statistics for each area and record type.
  - Summary statistics for user-defined chained and indexed sets.
- Generates an intermediate work file

IDMSDBN1 generates input to IDMSDBN2.

Each record in the file represents a pointer in a set connection between two records. For example, one record might represent the next pointer between two member records in a user-defined set; another might represent the connection between the root of a variable-length record and the first fragment of the record.

Each record includes the database keys of the two connected records. The database key of the record that contains the pointer to the other record is the FROM database key. The database key of the record to which the first record points is the TO database key.

### **Output**

Report	Required/ Optional	Description
#1: Messages	Required	<ul style="list-style-type: none"> <li>■ Lists input parameters used in the run.</li> <li>■ Lists all messages issued by IDMSDBN1. The messages: <ul style="list-style-type: none"> <li>– Report errors in the parameter input</li> <li>– Trace the processing of areas</li> <li>– Define any unexpected conditions detected by the program</li> </ul> </li> </ul>
#2: Area Information	Optional	Provides detailed area statistics. The report includes histograms of space availability and of data records per page.
#3: Set Statistics	Optional	Presents summary set statistics.
#4: Record Information	Optional	Provides detailed statistics for each record type in each area being analyzed. The report includes a histogram of data records per page.

### IDMSDBN2:

#### What it does: IDMSDBN2:

- Verifies set integrity. This is done in three steps.
  1. Reading the intermediate work file from IDMSDBN1
  2. Iteratively massaging and sorting the records
  3. Creating chains

A **chain** is a path that originates at a record located at a FROM database key and terminates at a record located at a TO database key.

IDMSDBN2 concatenates chains until a closed loop is created. A loop is created when the record at the last TO database key matches the record at the first FROM database key.

Each chain created by IDMSDBN2 is associated with a set. IDMSDBN2 verifies the integrity of sets by ensuring that each chain is complete and contains one and only one owner.

- Collects detailed set statistics.

#### Output

Report	Required/ Optional	Description
#1A: Messages	Required	Lists all messages issued by IDMSDBN2. The messages: <ul style="list-style-type: none"> <li>■ Trace the processing of the chain file.</li> <li>■ Report inconsistencies detected during chain processing.</li> </ul>
#5: Set Analysis Information	Optional	Provides detailed statistics for each set type processed. <ul style="list-style-type: none"> <li>■ Chained sets — The report includes histograms of chain length, of page changes, and of pages used to store the set.</li> <li>■ Sets of variable-length-record fragments — The report includes a histogram of pages used to store the set.</li> <li>■ Indexed sets owned by a user record — The report includes histograms of SR8 usage in set occurrences, of members in set occurrences, and of SR8 levels in set occurrences. (SR8 records are internal index records.)</li> </ul>

## 5.2.5 JCL Considerations

The JCL to execute the IDMSDBAN utility program must include statements to define:

- For IDMSDBN1:
  - For SQL databases, you must specify the name of the catalog to be processed on the SYSIDMS DBNAME parameter and *not* the DICTNAME parameter.
  - The files that map to the areas to be processed.
  - The SYSIPT file containing input parameters.
  - SYS002 contains output for use by IDMSDBN2 (this file is known to IDMSDBN2 as SYS001).
- For IDMSDBN2:
  - SYS001 containing the output from IDMSDBN1 (this file was known to IDMSDBN1 as SYS002).
  - SYS002 is a temporary storage file.
  - SORTWK $nn$  are sort work files. The number and size depends on the sort package you use.

- SORTMSG containing sort output messages.

►► Refer to the chapter pertaining to your operating system for JCL to execute IDMSDBAN.

## 5.2.6 Example

**No REPORT parameter specified:** If you run IDMSDBAN with the input parameters shown below, all areas and all sets in the EMPSS01 subschema will be processed and all reports will be produced. All reports are produced because a REPORT parameter is not specified.

```
process subschema empss01 dbname empdemo unlocked;
```

**With REPORT parameters:** If you run IDMSDBAN with the input parameters shown below, only the named areas and sets will be processed as follows:

- The entire EMP-DEMO-REGION area
- Pages in the range 5007102 through 5007149 in the ORG-DEMO-REGION
- Reports 1, 1A, 3 and 5 will be generated
- Report 3 will be produced for all of the named sets
- Report 5 will be produced only for the named sets that have at least one occurrence in the database

```
process subschema empss01 incomplete
report 3
report 5 terse
area emp-demo-region
area org-demo-region page 5007102 to 5007149
set dept-employee
set office-employee
set emp-name-ndx;
```

## 5.2.7 Sample output

The following reports are generated when no REPORT parameters are specified as in the first example above.

### Report 1: Messages Phase I

IDMSDBAN - DATA BASE ANALYSIS	REPORT 1: MESSAGES	DATE 09/17/99	TIME 20234050	PAGE 1
PARAMETER CARD: PROCESS SUBSCHEMA EMPSS01 DBNAME EMPDEMO UNLOCKED				
PARAMETER CARD: AREA EMP-DEMO-REGION				
IDMSDBAN - DATA BASE ANALYSIS	REPORT 1: MESSAGES	DATE 09/17/99	TIME 20234050	PAGE 2



**Report 2: Area**

IDMSDBAN - DATA BASE ANALYSIS		REPORT 2: AREA DATA		DATE	TIME	PAGE
				09/17/99	20234050	2
0	AREA: EMPDEMO.EMP-DEMO-REGION	(CONTINUED)				
SUBSCHEMA: EMPSS01						
DATA RECORDS PER PAGE HISTOGRAM						
DATA RECORDS PER PAGE	PAGES	PERCENT OF TOTAL PAGES	GRAPH	...10...20...30...40...50...60...70...80...90...100		
0	57	58	XXXXXXXXXXXXXXXXXXXXXXXXXXXX			
1 - 25	40	40	XXXXXXXXXXXXXXXXXXXXXXXXXXXX			
26 - 51	2	2	X			
52 - 76	0	0				
77 -102	0	0				
103 -127	0	0				
128 -152	0	0				
153 -178	0	0				
179 -203	0	0				
204 -229	0	0				
230 -254	0	0				
255	0	0				
AVERAGE DATA RECORDS PER PAGE (ALL PAGES)				3		
AVERAGE DATA RECORDS PER PAGE (NON-EMPTY PAGES)				8		

**Report 3: Set Statistics**

IDMSDBAN - DATA BASE ANALYSIS		REPORT 3: SET STATISTICS		DATE	TIME	PAGE
				09/17/99	20234050	1
0	SET: EMP-EMPOSITION	MODE: CHAIN	ORDER: FIRST			
SUBSCHEMA: EMPSS01						
	NAME	SET MEMBERSHIP TYPE	NUMBER OF RECORDS	EMPTY SETS	UNCONNECTED MEMBERS	LOCATION MODE
OWNER:	EMPLOYEE		56	0		CALC
MEMBERS:	EMPOSITION	MA	68		0	VIA EMP-EMPOSITION
			68 MEMBER RECORDS		0 UNCONNECTED MEMBERS	
AVERAGE MEMBERS PER SET FOR NON-EMPTY SETS				1		



### Report 1A: Messages Phase II

IDMSDBAN PHASE II - SET ANALYSIS	REPORT 1A: MESSAGES	DATE	TIME	PAGE
		09/17/99	20240900	1
599001 - PHASE II PROCESSING BEGUN				
599803 - END PASS	0			
599801 - FROM-RECORDS WRITTEN TO SORT	630			
599802 - TO-RECORDS WRITTEN TO SORT	630			
599806 - INDEX SR8 DESCRIPTORS WRITTEN	4			
599807 - INDEX DOWN DESCRIPTORS WRITTEN	58			
599808 - INDEX UP DESCRIPTORS WRITTEN	58			
599809 - INDEX NO-UP MEM DESCRIPTORS WRITTEN	0			
599810 - INDEX NO-UP DOWN DESCRIPTORS WRITTEN	0			
599803 - END PASS	1			
599801 - FROM-RECORDS WRITTEN TO SORT	0			
599802 - TO-RECORDS WRITTEN TO SORT	0			
599805 - INDEX PATHS WRITTEN TO SORT	0			
599806 - INDEX SR8 DESCRIPTORS WRITTEN	1			
599807 - INDEX DOWN DESCRIPTORS WRITTEN	19			
599808 - INDEX UP DESCRIPTORS WRITTEN	19			
599803 - END PASS	2			
599801 - FROM-RECORDS WRITTEN TO SORT	0			
599802 - TO-RECORDS WRITTEN TO SORT	0			
599805 - INDEX PATHS WRITTEN TO SORT	0			
599806 - INDEX SR8 DESCRIPTORS WRITTEN	0			
599807 - INDEX DOWN DESCRIPTORS WRITTEN	0			
599808 - INDEX UP DESCRIPTORS WRITTEN	0			

### Report 5: Set Analysis

IDMSDBAN PHASE II - SET ANALYSIS	REPORT 5: SET ANALYSIS DATA	DATE	TIME	PAGE			
		09/17/99	20240900	1			
SET:	EMP-EMPOSITION	MODE:	CHAIN	ORDER:	FIRST	SUBSCHEMA:	EMPSS01
OWNER:	EMPLOYEE						
MEMBERS:	EMPOSITION						
CHAIN LENGTH HISTOGRAM							
CHAIN LENGTH	NUMBER OF SETS	PERCENT OF TOTAL SETS	GRAPH				
			...10...20...30...40...50...60...70...80...90...100				
0	0	0					
2	47	84	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX				
3	6	11	XXXXX/				
4	3	5	XX/				
5 - 8	0	0					
9 - 16	0	0					
17 - 32	0	0					
33 - 64	0	0					
65 - 128	0	0					
129 - 256	0	0					
257 - 512	0	0					
OVER 512	0	0					
AVERAGE CHAIN LENGTH (ALL SETS):		2					
AVERAGE CHAIN LENGTH (NON-EMPTY SETS):		2					
MAXIMUM CHAIN LENGTH:		4					

## 5.2.8 For more information

- **On database structures analyzed by IDMSDBAN**, refer to *CA-IDMS Database Administration*.
- **On database tuning**, see *CA-IDMS Database Design*.

## 5.3 IDMSDIRL

### 5.3.1 Description

**Purpose:** The dictionary load utility, IDMSDIRL, loads into a dictionary the components required to describe the dictionary itself as well as the components which describe the security information stored in the dictionary.

The CA-supplied internal schema components include:

- Element and record definitions required by the schema components
- IDMSNTWK schema which describes the dictionary itself and two associated subschemas; IDMSNWKA and IDMSNWKG
- IDMSSECS and IDMSSECU schemas which describe the security information stored in the dictionary and the IDMSSECS and IDMSSECU subschemas used for security processing
- Additionally, IDMSDIRL can optionally remove these components from a dictionary without loading new definitions

Specifically, IDMSDIRL performs the following functions when loading the components:

- Establishes the default quotation character for the dictionary
- Adds to the dictionary the element and record definitions used by the schema components
- Adds to the dictionary the definitions for the IDMSNTWK schema and its associated subschemas: IDMSNWKA and IDMSNWKG
- Adds to the dictionary the definitions for the IDMSSECS schema and its associated subschema: IDMSSECS
- Adds to the dictionary the definitions for the IDMSSECU schema and its associated subschema: IDMSSECU
- Optionally connects the S-010 record occurrences just created for schemas IDMSNTWK, IDMSSECS and IDMSSECU to the OOAK-S set so that data dictionary reports include these definitions

**Note:** If the dictionary to be loaded is empty (formatted), you must first input the dictionary into DDDL which will populate the necessary CA-internal definitions into the dictionary.

## 5.3.2 Syntax

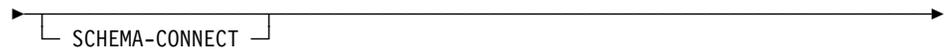
### DBL-QUOTE statement

Code on one line only.

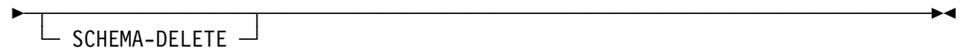


### SCHEMA-CONNECT statement

Code on one line only.



### SCHEMA-DELETE statement



## 5.3.3 Input parameter statements

**Parameter statement descriptions:** IDMSDIRL utility processing can be controlled by optional input parameter statements:

Statement	Description
DBL-QUOTE	Used to override the default quotation character for the data dictionary.
SCHEMA-CONNECT	Used to connect the CA-supplied internal schemas to the OOAK-S set.
SCHEMA-DELETE	Used to delete any CA-supplied internal schemas in the dictionary. Does not perform a dictionary load.

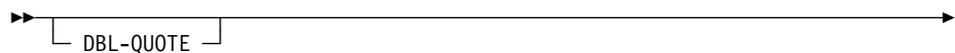
### DBL-QUOTE statement

**Purpose:** Makes the double quotation mark (") the default quotation character for the data dictionary.

By default, if you do not specify DBL-QUOTE, the default quotation character for the dictionary is the single quotation mark (').

#### Syntax

Code on one line only.



### SCHEMA-CONNECT statement



- System generation reports (CReports)
- IDD reports (DReports)

**Performance consideration for SCHEMA-CONNECT:** When reporting on the dictionary, the IDMSRPTS utility walks the OOAK-S set, if you specify SCHEMA=ALL. If the CA-supplied internal schemas are connected to the OOAK-S set, the reports will include all components of these schemas. Excluding the CA-supplied internal schemas from the OOAK-S set allows IDMSRPTS to bypass these components.

### 5.3.5 JCL considerations

The JCL to execute the IDMSDIRL utility program must include statements to define:

- The file containing the input definitions
- The file containing the dictionary into which the definitions are to be loaded

►► Refer to the chapter pertaining to your operating system for JCL to execute IDMSDIRL.

### 5.3.6 Examples

The following example directs the IDMSDIRL utility to add the definitions for the IDMSSECU schema and its associated subschema IDMSSECU to the dictionary and to connect the CA-supplied internal schemas to the OOAK-S set. Additionally, by default, the single quotation mark is established as the quotation character.

```
schema-connect
```

### 5.3.7 Sample output

The IDMSDIRL utility program generates the following listing after processing the above input.

```
IDMSDIRL - DATA DIRECTORY LOAD UTILITY   RELEASE 15.0 CAGJF0
OOAK ALREADY EXISTS
END OF IDMS DIRECTORY LOAD - IDMSDIRL
```

### 5.3.8 For more information

- **On data dictionaries and the use of IDMSDIRL**, refer to *CA-IDMS Database Administration*.
- **On the records and sets included in the IDMSNTWK schema**, refer to *CA-IDMS Dictionary Structure Reference*.

- **On the IDMSDIRL input file**, refer to *CA-IDMS Installation and Maintenance Guide OS/390*.

## 5.4 IDMSLOOK

### 5.4.1 Description

**Purpose:** The load module print utility, IDMSLOOK, reports on the contents of selected load modules. Using IDMSLOOK, you can report on:

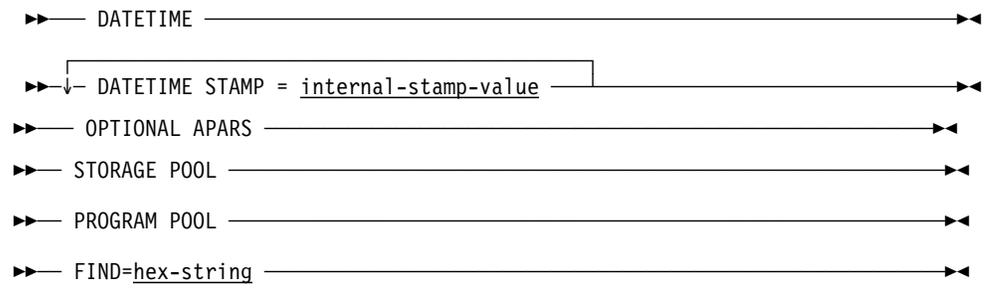
- The contents of a database name table module
- Area, record, and set information in a subschema load module
- Area, record, set, file, and physical characteristics for a subschema bound to a specific database
- File, area, journal, and buffer information in a DMCL load module together with its associated database name table information
- The names of the RCM modules characteristics of the tables referenced by the access module
- A hexadecimal dump of a specified load module in a format suitable for display at a 3270-type terminal
- The date/time stamps of the component modules in a specified load module
- The external value of internal date/time stamps

### 5.4.2 Syntax

```

▶▶ HELP _____▶▶
▶▶ LOAD=OPSYS _____▶▶
▶▶ SUBSCHEMA = subschema-name _____▶▶
▶▶ BIND SUBSCHEMA = subschema-name _____▶▶
   |_____|
   |_____| ,DBNAME = database-name |_____|
▶▶ DBTABLE = _____▶▶
   |_____|
   |_____| dbtable-name |_____|
▶▶ DMCL _____▶▶
   |_____| ALL |_____| |_____| SORTED _____▶▶
   |_____| |_____| |_____| SORTED PAGES |_____|
▶▶ AM=access-module-name _____▶▶
▶▶ AM PROGRAM = access-module-name _____▶▶
▶▶ PROGRAM = load-module-name _____▶▶
▶▶ RCM PROGRAM = RCM-module-name _____▶▶
▶▶ DATES = load-module-name _____▶▶

```



### 5.4.3 Input parameter statements

**Parameter statement descriptions:** IDMSLOOK utility processing is controlled by the following input parameter statements:

---

<b>Statement</b>	<b>Description</b>
HELP	Lists the parameters supported by IDMSLOOK
LOAD=OPSYS	Forces load modules to be loaded from an operating system load library rather than the dictionary load area
SUBSCHEMA	Reports area and record information for a subschema
BIND SUBSCHEMA	Reports area, record, and file information for a subschema bound to a database
DBTABLE	Reports the contents of a database name table
DMCL	Reports area, file, journal, and buffer information for a DMCL load module
AM	Reports the contents of the RCMs included in an access module and the tables referenced by the access module
AM PROGRAM	Produces a hexadecimal dump of an access module
PROGRAM	Reports the date/time stamp for all components of the load module and produces a hexadecimal dump of the load module
RCM PROGRAM	Produces a hexadecimal dump of an RCM module
DATES	Reports the date/time stamp for all components of the load module
DATETIME	Reports the current date and time
DATETIME STAMP	Reports the external value of an internal date/time stamp
OPTIONAL APARS	Displays optional apars activated in the current RHDCOPTF module.
STORAGE POOL	This displays the contents of the STORAGE POOL showing the storage address, storage size, task number that acquired the storage, owner of the storage, and the storage type.
PROGRAM POOL	This displays the contents of the PROGRAM POOL showing the program name, entry point address, load address, use count, and the size of the program.
FIND=xxxxxxxx	This displays the program name and offset into the program where the address was found. The xxxxxxxx are the 8 hexadecimal digits of the address to be searched for. The address must reside in one of the programs that reside in the PROGRAM POOL.

---

**HELP**

**Purpose:** Lists the parameters supported by the IDMSLOOK utility.

**Syntax**

▶▶ — HELP —————▶▶

**LOAD=OPSYS**

**Purpose:** Forces load modules to be loaded from an operating system load library rather than from a dictionary load area. This statement affects the loading technique used by SUBSCHEMA, DATES, PROGRAM, and RCM PROGRAM statements, and, when coded, must precede any of these statements.

**Syntax**

▶▶ — LOAD=OPSYS —————▶▶

**SUBSCHEMA**

**Purpose:** Reports the following information for a specified subschema load module:

- The date and time the subschema was last generated
- The names of included areas
- For each record type:
  - Location mode
  - Record ID
  - Record name
  - Data length
  - Prefix length
  - Database procedures associated with the record
- For each set
  - Set name
  - Set mode
  - Type of pointers used
  - Owner record name and set pointer positions
  - Member record names and set pointer positions
  - Access key information

**Syntax**

▶▶ — SUBSCHEMA = subschema-name —————▶▶

**Parameter****SUBSCHEMA**

Specifies the subschema load module.

**subschema-name**

The name of a subschema load module.

**BIND SUBSCHEMA**

**Purpose:** Reports the following information for a specified subschema load module bound to a database:

- All the information you get from the SUBSCHEMA statement, plus:
- The page group and page range for each area
- For files:
  - File name
  - External name
  - Page size
  - Page range of the associated area to which the file maps

**Syntax**

```

▶▶ ↓ BIND SUBSCHEMA = subschema-name [ ,DBNAME = database-name ]

```

**Parameters****subschema-name**

Specifies the subschema to report on.

**DBNAME**

Specifies the database name to which the subschema will be bound.

**database-name**

A database name in the database name table, or segment name in the DMCL, to which the subschema will be bound.

If you do not specify a *database-name*, the subschema must be a Release 10.x subschema that is to be converted to a Release 12.0 subschema.

**DBTABLE**

**Purpose:** Reports the contents of a database name table.

**Syntax**

```

▶▶ ↓ DBTABLE = [ dtable-name ]

```

**Parameters****DBTABLE**

Reports the contents of a database name table.

**dbtable-name**

Optionally, identifies a specific database name table. If omitted, the default database name table associated with the current DMCL is used.

**DMCL**

**Purpose:** Reports the following information for a DMCL load module:

1. The date and time the DMCL was generated
2. The page range, page size, and file mappings for each area
3. The page size and number of pages in each journal file
4. The page size, number of pages, and total size of each buffer
5. The date each area definition was last updated \*
6. A history of the last date and time that an area was affected by a DCMT VARY DMCL command \*

\* This information is only produced, along with items 1 through 4 in the above list, when you specify the ALL parameter of the DMCL statement.

**Syntax****Parameters****DMCL**

Reports the contents of the current DMCL module.

**ALL**

Optionally, produces the following information in addition to the standard information (items 1 through 4 in the list above) provided on the DMCL report:

- The date each area definition was last updated
- A history of the last date and time that an area was affected by a DCMT VARY DMCL command

**SORTED**

Sorts DMCL information by area name.

**SORTED PAGES**

Sorts DMCL information by page range.

**AM**

**Purpose:** Reports on the contents of the RCMs included in an access module and the tables referenced by the access module.

**Syntax**

▶— AM=access-module-name —▶▶

**Parameter****access-module-name**

Specifies the access module to report on.

**AM PROGRAM**

**Purpose:** Produces a hexadecimal dump of the contents of a specified access module.

**Syntax**

▶— AM PROGRAM = access-module-name —▶▶

**Parameter****access-module-name**

Specifies the access module to report on.

**PROGRAM**

**Purpose:** Produces a hexadecimal dump of the contents of a specified load module, along with date/time information for the object modules included in the load module.

**Syntax**

▶— PROGRAM = load-module-name —▶▶

**Parameter****load-module-name**

Specifies the load module to report on.

**RCM PROGRAM**

**Purpose:** Produces a hexadecimal dump of the contents of a specified RCM module.

**Syntax**

▶— RCM PROGRAM = RCM-module-name —▶▶

**Parameter****rcm-module-name**

Specifies the RCM module to report on.

**DATES**

**Purpose:** Reports on the date/time stamps of the component modules included in a specified load module.

### Syntax

▶▶ DATES = load-module-name ▶▶

### Parameter

#### load-module-name

Specifies the load module to report on.

### DATETIME

**Purpose:** Report the current date and time.

### Syntax

▶▶ DATETIME ▶▶

### DATETIME STAMP

**Purpose:** Displays the external value of an internal date/time stamp.

### Syntax

▶▶ DATETIME STAMP = internal-stamp-value ▶▶

### Parameters

#### internal-stamp-value

The 16 hexadecimal digits that make up the internal representation of the date/time stamp.

### OPTIONAL APARS

**Purpose:** Displays all the active optional apars in the current RHDCOPTF module.

### Syntax

▶▶ OPTIONAL APARS ▶▶

### STORAGE POOL

**Purpose:** Reports the following information for the STORAGE POOL:

- The address of each storage allocation
- The size of each storage allocation
- The task number that acquired each storage allocation
- The owner of each storage allocation

- The storage type of each storage allocation

### Syntax

►— STORAGE POOL —————►

### PROGRAM POOL

**Purpose:** Reports the following information for the PROGRAM POOL:

- The program name of each program in the program pool
- The entry point address of each program in the program pool
- The load address of each program in the program pool
- The number of times each program has been used
- The size of each program in the program pool

### Syntax

►— PROGRAM POOL —————►

### FIND=xxxxxxxx

**Purpose:** This displays the program name and offset into the program where the address was found. The xxxxxxxx are the 8 hexadecimal digits of the address to be searched for. The address must reside in one of the programs that reside in the PROGRAM POOL.

### Syntax

►— FIND=xxxxxxxx —————►

## 5.4.4 Usage

**Input:** Input to the IDMSLOOK utility consists only of statements to control the utility processing.

**Output:** The IDMSLOOK utility generates a printout listing that includes the requested reports.

**Batch operating mode:** You can execute the IDMSLOOK utility in local mode only.

**Coding considerations:** You can include multiple input statements in a single run of IDMSLOOK. However each input statement must be on a separate line.

None of the statements is required.

**Online processing:** Most of the functions of IDMSLOOK may be executed online through the DC task code LOOK.

►► For more information on using DC tasks, see *CA-IDMS System Tasks and Operator Commands*.

### 5.4.5 JCL considerations

Refer to the chapter pertaining to your operating system for the JCL to execute IDMSLOOK.

### 5.4.6 Examples

**Requesting subschema information:** The SUBSCHEMA statement below directs IDMSLOOK to return a report on the subschema load module EMPSS01. The report will include information on the logical and physical attributes of the EMPSS01 subschema when it is bound to database name EMPDEMO.

```
bind subschema=empss01,dbname=empdemo
```

**Requesting DMCL information:** The statement below directs IDMSLOOK to report all information on a DMCL (page range, size, file mappings, etc. as well as date and time history) and to sort the information by area name.

```
dmcl all sorted
```

**Requesting a hexadecimal dump:** The PROGRAM statement below directs IDMSLOOK to return a hexadecimal dump of the load module RTPRG001, along with the date/time stamps of the object modules included in RTPRG001.

```
program=rtprgr001
```

### 5.4.7 Sample output

IDMSLOOK starts a new page in the message listing for each input parameter statement processed in the run. At the top of the page, IDMSLOOK prints the parameter statement. Then IDMSLOOK prints the report requested by the statement.

**Report requested by subschema**

IDMSLOOK - Selection Parameter Follows:  
 SUBSCHEMA=EMPSS01

EMPSS01 was #LOAded From --> APPLDICT  
 Entry Point Offset +0 - Reentrant - AMODE 31 - RMODE ANY

SUBSCHEMA=EMPSS01  
 Compiled=1999-03-01 15.24.58  
 Subschema Structure is Network and Unbound

Area Name	Segment
EMP-DEMO-REGION	n/a
INS-DEMO-REGION	n/a
ORG-DEMO-REGION	n/a

Record Name	Stored	Rec ID	Area Name	Data Length	Prefix Length	Procedures
COVERAGE	VIA	400	INS-DEMO-REGION	20	20	
DENTAL-CLAIM	VIA	405	INS-DEMO-REGION	936	12	
DEPARTMENT	CALC	410	ORG-DEMO-REGION	56	16	
EMPLOYEE	CALC	415	EMP-DEMO-REGION	120	72	
EMPOSITION	VIA	420	EMP-DEMO-REGION	32	24	
EXPERTISE	VIA	425	EMP-DEMO-REGION	12	20	
HOSPITAL-CLAIM	VIA	430	INS-DEMO-REGION	300	8	
INSURANCE-PLAN	CALC	435	INS-DEMO-REGION	132	8	
JOB	CALC	440	ORG-DEMO-REGION	300	24	IDMSCOMP Before STORE IDMSCOMP Before MODIFY IDMSDCOM After GET
NON-HOSP-CLAIM	VIA	445	INS-DEMO-REGION	1,056	12	
OFFICE	CALC	450	ORG-DEMO-REGION	76	16	
SKILL	CALC	455	ORG-DEMO-REGION	76	20	
SR1	VIA	1	n/a	4	8	
SR6	VIA	6	n/a	0	0	
SR7	CALC	7	n/a	16	16	

5.4 IDMSLOOK

```

Chain Sorted-> CALC                               Next,Prior
Owner -----> SR1                               Next=00 Prior=04
Member -----> SR6                               Next=00 Prior=04
Member -----> SR7                               Next=00 Prior=04
Member -----> SKILL                             Ckey Offset=16 Length=16 Data Type=Character
Member -----> OFFICE                           Ckey Offset=20 Length=4  Data Type=Numeric (Unsigned)
Member -----> OFFICE                           Ckey Offset=16 Length=3  Data Type=Character
Member -----> JOB                               Ckey Offset=28 Length=4  Data Type=Numeric (Unsigned)
Member -----> INSURANCE-PLAN                   Ckey Offset=8  Length=3  Data Type=Character
Member -----> EMPLOYEE                         Ckey Offset=72 Length=4  Data Type=Numeric (Unsigned)
Member -----> DEPARTMENT                       Ckey Offset=16 Length=4  Data Type=Numeric (Unsigned)

Chain Last --> COVERAGE-CLAIMS                  Next,Prior
Owner -----> COVERAGE                         Next=12 Prior=16
Via Member --> NON-HOSP-CLAIM                   Next=00 Prior=04
Via Member --> HOSPITAL-CLAIM                  Next=00 Prior=04
Via Member --> DENTAL-CLAIM                   Next=00 Prior=04

Chain Sorted-> DEPT-EMPLOYEE                     Next,Prior,Owner
Owner -----> DEPARTMENT                       Next=08 Prior=12
Member -----> EMPLOYEE                         Next=08 Prior=12 Owner=16
Member -----> EMPLOYEE                         Ckey Offset=86 Length=15 Data Type=Character
Member -----> EMPLOYEE                         Ckey Offset=76 Length=10 Data Type=Character

Chain First -> EMP-COVERAGE                      Next,Prior,Owner
Owner -----> EMPLOYEE                         Next=32 Prior=36
Via Member --> COVERAGE                       Next=00 Prior=04 Owner=08

Chain First -> EMP-EMPOSITION                    Next,Prior,Owner
Owner -----> EMPLOYEE                         Next=40 Prior=44
Via Member --> EMPOSITION                      Next=00 Prior=04 Owner=08

Chain Sorted-> EMP-EXPERTISE                     Next,Prior,Owner
Owner -----> EMPLOYEE                         Next=48 Prior=52
Via Member --> EXPERTISE                       Ckey Offset=20 Length=2  Data Type=Character
Via Member --> EXPERTISE                       Next=00 Prior=04 Owner=08

Index Sorted-> EMP-NAME-NDX                      SR8Next,SR8Prior
Owner -----> SR7                               SR8Next=08 SR8Prior=12
Member -----> EMPLOYEE                         SR8Next=20
Member -----> EMPLOYEE                         Ckey Offset=86 Length=15 Data Type=Character
Member -----> EMPLOYEE                         Ckey Offset=76 Length=10 Data Type=Character

```

File Name	DDNAME	Data- Shared		Buffer Name	Data Set Name (DSN) - Cache Name
		Type	Space Cache		
APPLDICT.DICTDB	DICTDB	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.APPLDICT.DDLML
APPLDICT.DLODDB	DLODDB	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.APPLDICT.DDLCLD
ASFDICT.ASFDATA	ASFDATA	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDATA
ASFDICT.ASFDEFN	ASFDEFN	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDEFN
ASFDICT.ASFML	ASFML	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDEFN.DDLML
ASFDICT.ASFLOD	ASFLOD	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDEFN.ASFLOD
CATSYS.DCCAT	DCCAT	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.CATSYS.DCCAT
CATSYS.DCCATL	DCCATL	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.CATSYS.DCCATL
CATSYS.DCCATX	DCCATX	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.CATSYS.DCCATX
EMPDEMO.EMPDEMO	EMPDEMO	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.EMPDEMO.EMPDEMO
EMPDEMO.INSDEMO	INSDEMO	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.EMPDEMO.INSDEMO
EMPDEMO.ORGDEMO	ORGDEMO	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.EMPDEMO.ORGDEMO
EMPDEMO2.EMPDEMO	EMPDEMO2	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.R141.EMPDEMO2.EMPDEMO
EMPDEMO2.INSDEMO	INSDEMO2	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.R141.EMPDEMO2.INSDEMO
EMPDEMO2.ORGDEMO	ORGDEMO2	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.R141.EMPDEMO2.ORGDEMO
EMPD2001.EMPDEMO	EMPD2001	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.PG2001.EMPDEMO
EMPD2002.INSDEMO	INS2002	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.PG2002.INSDEMO
EMPD2003.ORGDEMO	ORG2003	BDAM	No No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.PG2003.ORGDEMO
INGDBA.IIDBAF1	IIDBAF1	BDAM	No No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGDBA.DINDEX
INGDBA.Q7DBAF2	Q7DBAF2	BDAM	No No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGDBA.DDATA
INGRES.IICATF1	IICATF1	BDAM	No No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.CINDEX
INGRES.IICATF2	IICATF2	BDAM	No No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.CATALOG
INGRES.IICATF3	IICATF3	BDAM	No No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.EINDEX
INGRES.IICATF4	IICATF4	BDAM	No No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.EXTENDED

DMCL Journals	Page Size	# of Pages
J1JRNL	2,004	5,000
J4JRNL	2,004	5,000
J3JRNL	2,004	5,000
J2JRNL	2,004	5,000
SYSJRNL	8,000	Archive

Journal Buffers	Buffer Size	# of Buffers
JNL_BUFFER	2,004	5

DMCL Buffers	Buffer Size	CV Buffers	CV Type	Total CV Size	Local Buffers	Local Type	Total Local Size
II_BUFFER	4,276	500	OS	2,138,000	200	OS	855,200
LSR_BUFFER	28,672	4	OS	114,688	4	OS	114,688
NSR_BUFFER	28,672	4	OS	114,688	4	OS	114,688
DEFAULT_BUFFER	4,276	30	OS	128,280	20	OS	85,520
LOG_BUFFER	4,276	5	OS	21,380	5	DC	21,380

0 Bytes used for CV buffers in DC storage  
2,517,036 Bytes used for CV buffers in OS storage  
2,517,036 Bytes used for CV DMCL Buffers

21,380 Bytes used for LOCAL buffers in DC storage  
1,170,096 Bytes used for LOCAL buffers in OS storage  
1,191,476 Bytes used for LOCAL DMCL Buffers

## 5.4 IDMSLOOK

```
Dbtable=R141DBTB          Compiled Date=1999-11-15  10.44.19
The DEFAULT Dictionary is APPLDICT

DBNAME is *DEFAULT match on subschema is OPTIONAL
  Subschema IDMSNWK? maps to IDMSNWK? using DBNAME ----> APPLDICT
  Subschema IDMSCAT? maps to IDMSCAT? using DBNAME ----> APPLDICT

DBNAME is APPLDICT match on subschema is OPTIONAL
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSSQL          this is an SQL segemnt

DBNAME is ASFDICT match on subschema is OPTIONAL
  Include SEGMENT ----> ASFDICT
  Include SEGMENT ----> SYSMMSG

DBNAME is EMPDEMO2 match on subschema is OPTIONAL
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> EMPDEMO2      non unique PAGE GROUP
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSSQL          this is an SQL segemnt

DBNAME is EMPDMIXD match on subschema is OPTIONAL Mixed PAGE GROUPS
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> EMPD2001
  Include SEGMENT ----> EMPD2002
  Include SEGMENT ----> EMPD2003
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSSQL          this is an SQL segemnt

DBNAME is INGSQL match on subschema is OPTIONAL
  Include SEGMENT ----> INGSQL          this is an SQL segemnt
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSTEM

DBNAME is MIXDBNME match on subschema is OPTIONAL Mixed PAGE GROUPS
  Include SEGMENT ----> MIXSEGMENT
  Include SEGMENT ----> VSAMSEG

DBNAME is RJWDB match on subschema is OPTIONAL
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> RJWDEMO
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSSQL          this is an SQL segemnt

DBNAME is SYSDIRL match on subschema is OPTIONAL
  Include SEGMENT ----> SYSDIRL
  Include SEGMENT ----> SYSMMSG

DBNAME is SYSTEM match on subschema is OPTIONAL
  Include SEGMENT ----> CATSYS          this is an SQL segemnt
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSTEM
```

## Report requested by DMCL

IDMSLOOK - Selection Parameter Follows:  
DMCL ALL SORTED

DMCL=R141DMCL Runtime Size---> 0B154 (45,396 Bytes)  
This DMCL uses dbtable R141DBTB Compiled Size---> 078C4 (30,916 Bytes)  
Date Last Critical Change=1999-11-15 10.42.50 The Operating System is OS  
Date Created=1999-03-11 14.00.21 Date Last Updated=1999-11-15 10.42.50

Area Name	Page Group	Low Page	High Page	Page Size	DDNAME	File Name	Low Page	High Page
APPLDICT.DDLDCLOD	0	70,001	70,500	4,276	DLODDB	APPLDICT.DLODDB	***	SAME ***
Area definition date last critical change=1999-03-01 13.42.42								
Page Range Symbolic is DDLDCLOD Value is 70,001--->70,500								
Page Range Symbolic is 2ND PAGE OF AREA Value is 70,002--->70,002								
APPLDICT.DDLDMML	0	60,001	62,000	4,276	DICTDB	APPLDICT.DICTDB	***	SAME ***
Area definition date last critical change=1999-03-01 13.42.42								
Page Range Symbolic is DDLDMML Value is 60,001--->62,000								
Page Range Symbolic is 2ND PAGE OF AREA Value is 60,002--->60,002								
ASFDICT.DDLDCLOD	0	88,001	90,000	4,276	ASFLOD	ASFDICT.ASFLOD	***	SAME ***
Area definition date last critical change=1999-03-01 13.42.43								
Page Range Symbolic is DDLDCLOD Value is 88,001--->90,000								
Page Range Symbolic is 2ND PAGE OF AREA Value is 88,002--->88,002								
ASFDICT.DDLDMML	0	80,001	82,000	4,276	ASFDMML	ASFDICT.ASFDMML	***	SAME ***
Area definition date last critical change=1999-03-01 13.42.42								
Page Range Symbolic is DDLDMML Value is 80,001--->82,000								
Page Range Symbolic is 2ND PAGE OF AREA Value is 80,002--->80,002								
ASFDICT.IDMSR-AREA	0	83,001	84,000	4,276	ASFDEFN	ASFDICT.ASFDEFN	***	SAME ***
Area definition date last critical change=1999-03-01 13.42.43								
Page Range Symbolic is IDMSR-AREA Value is 83,001--->84,000								
ASFDICT.IDMSR-AREA2	0	85,001	87,000	4,276	ASFDATA	ASFDICT.ASFDATA	***	SAME ***
Area definition date last critical change=1999-03-01 13.42.43								
Page Range Symbolic is IDMSR-AREA2 Value is 85,001--->87,000								
CATSYS.DDLDCAT	0	1	600	4,276	DCCAT	CATSYS.DCCAT	***	SAME ***
Area definition date last critical change=1999-03-01 13.42.42								
Page Range Symbolic is DDLDCAT Value is 1--->600								
CATSYS.DDLDCATL	0	751	950	4,276	DCCATL	CATSYS.DCCATL	***	SAME ***

5.4 IDMSLOOK

Area definition date last critical change=1999-03-01	13.42.42						
Page Range Symbolic is DDLCATLOD	Value is	751---	950				
Page Range Symbolic is 2ND PAGE OF AREA	Value is	752---	752				
CATSYS.DDLCATX	0	601	700	4,276	DCCATX	CATSYS.DCCATX	*** SAME ***
Area definition date last critical change=1999-03-01	13.42.42						
Page Range Symbolic is DDLCATX	Value is	601---	700				
EMPDEMO.EMP-DEMO-REGION	0	75,001	75,050	4,276	EMPDEMO	EMPDEMO.EMPDEMO	*** SAME ***
Area definition date last critical change=1999-03-01	13.42.42						
Page Range Symbolic is EMP-DEMO-REGION	Value is	75,001---	75,050				
EMPDEMO.INS-DEMO-REGION	0	75,101	75,125	4,276	INSDEMO	EMPDEMO.INSDEMO	*** SAME ***
Area definition date last critical change=1999-03-01	13.42.42						
Page Range Symbolic is INS-DEMO-REGION	Value is	75,101---	75,125				
EMPDEMO.ORG-DEMO-REGION	0	75,151	75,175	4,276	ORGDEMO	EMPDEMO.ORGDEMO	*** SAME ***
Area definition date last critical change=1999-03-01	13.42.42						
Page Range Symbolic is ORG-DEMO-REGION	Value is	75,151---	75,175				
EMPDEMO2.EMP-DEMO-REGION	2	75,001	75,050	4,276	EMPDEMO2	EMPDEMO2.EMPDEMO	*** SAME ***
Area definition date last critical change=1999-08-11	15.16.33						
Page Range Symbolic is EMP-DEMO-REGION	Value is	75,001---	75,050				
EMPDEMO2.INS-DEMO-REGION	2	75,101	75,125	4,276	INSDEMO2	EMPDEMO2.INSDEMO	*** SAME ***
Area definition date last critical change=1999-08-11	15.16.33						
Page Range Symbolic is INS-DEMO-REGION	Value is	75,101---	75,125				
EMPDEMO2.ORG-DEMO-REGION	2	75,151	75,175	4,276	ORGDEMO2	EMPDEMO2.ORGDEMO	*** SAME ***
Area definition date last critical change=1999-08-11	15.16.33						
Page Range Symbolic is ORG-DEMO-REGION	Value is	75,151---	75,175				
EMPD2001.EMP-DEMO-REGION	2001	75,001	75,050	4,276	EMPD2001	EMPD2001.EMPDEMO	*** SAME ***
Area definition date last critical change=1999-11-15	10.40.46						
Page Range Symbolic is EMP-DEMO-REGION	Value is	75,001---	75,050				
EMPD2002.INS-DEMO-REGION	2002	75,101	75,125	4,276	INSDEMO2	EMPD2002.INSDEMO	*** SAME ***
Area definition date last critical change=1999-11-15	10.40.46						
Page Range Symbolic is INS-DEMO-REGION	Value is	75,101---	75,125				
EMPD2003.ORG-DEMO-REGION	2003	75,151	75,175	4,276	ORGDEMO3	EMPD2003.ORGDEMO	*** SAME ***
Area definition date last critical change=1999-11-15	10.40.46						
Page Range Symbolic is ORG-DEMO-REGION	Value is	75,151---	75,175				
INGDBA.II-DDATA-AREA	0	103,201	104,000	4,276	Q7DBAF2	INGDBA.Q7DBAF2	*** SAME ***
Area definition date last critical change=1999-05-19	09.46.18						
Page Range Symbolic is II-DDATA-AREA	Value is	103,201---	104,000				
INGDBA.II-DINDEX-AREA	0	103,001	103,200	4,276	IIDBAF1	INGDBA.IIDBAF1	*** SAME ***
Area definition date last critical change=1999-05-19	09.46.18						
Page Range Symbolic is II-DINDEX-AREA	Value is	103,001---	103,200				

INGRES.II-CATALOG-AREA	0	101,101	101,200	4,276	IICATF2	INGRES.IICATF2	***	SAME	***
Area definition date last critical change=1999-05-19 09.46.18									
Page Range Symbolic is II-CATALOG-AREA Value is 101,101--->101,200									
INGRES.II-CINDEX-AREA	0	101,001	101,100	4,276	IICATF1	INGRES.IICATF1	***	SAME	***
Area definition date last critical change=1999-05-19 09.46.18									
Page Range Symbolic is II-CINDEX-AREA Value is 101,001--->101,100									
INGRES.II-EINDEX-AREA	0	102,001	102,100	4,276	IICATF3	INGRES.IICATF3	***	SAME	***
Area definition date last critical change=1999-05-19 09.46.18									
Page Range Symbolic is II-EINDEX-AREA Value is 102,001--->102,100									
INGRES.II-EXTENDED-AREA	0	102,101	102,600	4,276	IICATF4	INGRES.IICATF4	***	SAME	***
Area definition date last critical change=1999-05-19 09.46.18									
Page Range Symbolic is II-EXTENDED-AREA Value is 102,101--->102,600									
INGSQL.DDLCAT	0	100,001	100,300	4,276	IISQLF1	INGSQL.IISQLF1	***	SAME	***
Area definition date last critical change=1999-05-19 09.46.08									
Page Range Symbolic is DDLCAT Value is 100,001--->100,300									
INGSQL.DDLCATL0D	0	100,401	100,450	4,276	IISQLF3	INGSQL.IISQLF3	***	SAME	***
Area definition date last critical change=1999-05-19 09.46.08									
Page Range Symbolic is DDLCATL0D Value is 100,401--->100,450									
Page Range Symbolic is 2ND PAGE OF AREA Value is 100,402--->100,402									
INGSQL.DDLCATX	0	100,301	100,400	4,276	IISQLF2	INGSQL.IISQLF2	***	SAME	***
Area definition date last critical change=1999-05-19 09.46.08									
Page Range Symbolic is DDLCATX Value is 100,301--->100,400									
MIXSEGMENT.MIXAREA	10	78,001	78,020	4,276	MIXFILE	MIXSEGMENT.MIXFILE	***	SAME	***
Area definition date last critical change=1999-07-02 14.06.38									
Page Range Symbolic is MIXAREA Value is 78,001--->78,020									
PROJSEG.PROJAREA	0	77,401	77,450	4,276	PROJDEMO	PROJSEG.PROJDEMO	***	SAME	***
Area definition date last critical change=1999-03-01 13.42.43									
Page Range Symbolic is PROJAREA Value is 77,401--->77,450									
RJWDEMO.EMP-DEMO-REGION	0	575,001	575,050	4,276	RJWEMP	RJWDEMO.EMPDEMO	***	SAME	***
Area definition date last critical change=1999-04-12 17.42.59									
Page Range Symbolic is EMP-DEMO-REGION Value is 575,001--->575,050									
RJWDEMO.INS-DEMO-REGION	0	575,101	575,125	4,276	RJWINS	RJWDEMO.INSDEMO	***	SAME	***
Area definition date last critical change=1999-04-12 17.42.59									
Page Range Symbolic is INS-DEMO-REGION Value is 575,101--->575,125									
RJWDEMO.ORG-DEMO-REGION	0	575,151	575,175	4,276	RJWORG	RJWDEMO.ORGDEMO	***	SAME	***
Area definition date last critical change=1999-04-12 17.42.59									
Page Range Symbolic is ORG-DEMO-REGION Value is 575,151--->575,175									
SQLDEMO.EMPLAREA	0	77,001	77,100	4,276	EMPLDEMO	SQLDEMO.EMPLDEMO	***	SAME	***
Area definition date last critical change=1999-03-01 13.42.43									
Page Range Symbolic is EMPLAREA Value is 77,001--->77,100									
SQLDEMO.INDXAREA	0	77,301	77,350	4,276	INDXDEMO	SQLDEMO.INDXDEMO	***	SAME	***
Area definition date last critical change=1999-03-01 13.42.43									
Page Range Symbolic is INDXAREA Value is 77,301--->77,350									
SQLDEMO.INFOAREA	0	77,201	77,250	4,276	INFODEMO	SQLDEMO.INFODEMO	***	SAME	***

5.4 IDMSLOOK

Page Range Symbolic is DDLCATL0D	Value is	751---	>950			
Page Range Symbolic is 2ND PAGE OF AREA	Value is	752---	>752			
SYSTEM.DDL0ML	0	1,001	2,000	4,276	DC0ML	SYSTEM.DC0ML *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDL0ML	Value is	1,001---	>2,000			
Page Range Symbolic is 2ND PAGE OF AREA	Value is	1,002---	>1,002			
SYSTEM.DDLDCLOD	0	3,001	3,100	4,276	DCLOD	SYSTEM.DCLOD *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDLDCLOD	Value is	3,001---	>3,100			
Page Range Symbolic is 2ND PAGE OF AREA	Value is	3,002---	>3,002			
SYSDIRL.DDLDCLOD	0	4,001	4,010	4,276	DIRLLOD	SYSDIRL.DIRLLOD *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDLDCLOD	Value is	4,001---	>4,010			
Page Range Symbolic is 2ND PAGE OF AREA	Value is	4,002---	>4,002			
SYSDIRL.DDL0ML	0	5,001	9,000	4,276	DIRL0B	SYSDIRL.DIRL0B *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDL0ML	Value is	5,001---	>9,000			
Page Range Symbolic is 2ND PAGE OF AREA	Value is	5,002---	>5,002			
SYSM0G.DDLDCM0G	0	10,001	14,000	4,276	DCM0G	SYSM0G.DCM0G *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDLDCM0G	Value is	10,001---	>14,000			
SYSSQL.DDL0AT	0	20,001	22,000	4,276	SQLDD	SYSSQL.SQLDD *** SAME ***
Area definition date last critical change=1999-03-01			13.42.43			
Page Range Symbolic is DDL0AT	Value is	20,001---	>22,000			
SYSSQL.DDL0ATL0D	0	25,001	25,500	4,276	SQLL0D	SYSSQL.SQLL0D *** SAME ***
Area definition date last critical change=1999-03-01			13.42.43			
Page Range Symbolic is DDLCATL0D	Value is	25,001---	>25,500			
Page Range Symbolic is 2ND PAGE OF AREA	Value is	25,002---	>25,002			
SYSSQL.DDL0ATX	0	28,001	28,500	4,276	SQLXDD	SYSSQL.SQLXDD *** SAME ***
Area definition date last critical change=1999-03-01			13.42.43			
Page Range Symbolic is DDL0ATX	Value is	28,001---	>28,500			
SYSTEM.DDLDCLOG	0	30,001	34,000	4,276	DCLOG	SYSTEM.DCLOG *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDLDCLOG	Value is	30,001---	>34,000			
SYSTEM.DDLDCRUN	0	40,001	41,000	2,676	DCRUN	SYSTEM.DCRUN *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDLDCRUN	Value is	40,001---	>41,000			
SYSUSER.DDLSEC	0	48,001	48,500	4,276	SECDD	SYSUSER.SECDD *** SAME ***
Area definition date last critical change=1999-03-01			13.42.42			
Page Range Symbolic is DDLSEC	Value is	48,001---	>48,500			

File Name	DDNAME	Type	Space	Cache	Buffer Name	Data Set Name (DSN) - Cache Name
APPLDICT.DICTDB	DICTDB	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.APPLDICT.DDLML
APPLDICT.DLODDB	DLODDB	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.APPLDICT.DDLDCLOD
ASFDICT.ASFDATA	ASFDATA	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDICT.ASFDATA
ASFDICT.ASFDEFN	ASFDEFN	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDICT.ASFDEFN
ASFDICT.ASFDFML	ASFDFML	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDICT.DDLML
ASFDICT.ASFLOD	ASFLOD	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.ASFDICT.ASFLOD
CATSYS.DCCAT	DCCAT	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.CATSYS.DCCAT
CATSYS.DCCATL	DCCATL	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.CATSYS.DCCATL
CATSYS.DCCATX	DCCATX	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.CATSYS.DCCATX
EMPDEMO.EMPDEMO	EMPDEMO	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.EMPDEMO.EMPDEMO
EMPDEMO.INSDEMO	INSDEMO	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.EMPDEMO.INSDEMO
EMPDEMO.ORGDEMO	ORGDEMO	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.EMPDEMO.ORGDEMO
EMPDEMO2.EMPDEMO	EMPDEMO2	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.R141.EMPDEMO2.EMPDEMO
EMPDEMO2.INSDEMO	INSDEMO2	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.R141.EMPDEMO2.INSDEMO
EMPDEMO2.ORGDEMO	ORGDEMO2	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.R141.EMPDEMO2.ORGDEMO
EMPD2001.EMPDEMO	EMPD2001	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.PG2001.EMPDEMO
EMPD2002.INSDEMO	INSD2002	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.PG2002.INSDEMO
EMPD2003.ORGDEMO	ORG2003	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.PG2003.ORGDEMO
INGDBA.I1DBAF1	I1DBAF1	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGDBA.DINDEX
INGDBA.Q7DBAF2	Q7DBAF2	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGDBA.DDATA
INGRES.IICATF1	IICATF1	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.CINDEX
INGRES.IICATF2	IICATF2	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.CATALOG
INGRES.IICATF3	IICATF3	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.EINDEX
INGRES.IICATF4	IICATF4	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGRES.EXTENDED
INGSQL.IISQLF1	IISQLF1	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGSQL.DDLCAT
INGSQL.IISQLF2	IISQLF2	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGSQL.DDLCATX
INGSQL.IISQLF3	IISQLF3	BDAM	No	No	II_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.INGSQL.DDLCATL
MIXSEGMENT.MIXFILE	MIXFILE	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.TECHDC40.R141.MIXSEGMENT.MIXFILE
PROJSEG.PROJDEMO	PROJDEMO	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.PROJSEG.PROJDEMO
RJWDEMO.EMPDEMO	RJWEMP	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.RJW.EMPDEMO.T40
RJWDEMO.INSDEMO	RJWINS	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.RJW.INSDEMO.T40
RJWDEMO.ORGDEMO	RJWORG	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=MEN.C1300.RJW.ORGDEMO.T40
SQLDEMO.EMPLDEMO	EMPLDEMO	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SQLDEMO.EMPLDEMO
SQLDEMO.INDXDEMO	INDXDEMO	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SQLDEMO.INDXDEMO
SQLDEMO.INFODEMO	INFODEMO	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SQLDEMO.INFODEMO
SYSDIRL.DIRLDB	DIRLDB	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSDIRL.DDLML
SYSDIRL.DIRLLOD	DIRLLOD	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSDIRL.DDLDCLOD
SYSLC.DCLSCR	DCLSCR	BDAM	No	No	DEFAULT_BUFFER	
SYMSG.DCMSCG	DCMSG	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYMSG.DDLDCMSG
SYSSQL.SQLDD	SQLDD	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSSQL.DDLCAT
SYSSQL.SQLLOD	SQLLOD	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSSQL.DDLCATL
SYSSQL.SQLXDD	SQLXDD	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSSQL.DDLCATX
SYSTEM.DCDML	DCDML	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSTEM.DDLML
SYSTEM.DCLOD	DCLOD	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSTEM.DDLDCLOD
SYSTEM.DCLOG	DCLOG	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSTEM.DDLDCLOG
SYSTEM.DCRUN	DCRUN	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSTEM.DDLDCRUN
SYSTEM.DCSCR	DCSCR	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSTEM.DDLDCSCR
SYSUSER.SECDD	SECDD	BDAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=DIST.SYSTEM86.SYSUSER.DDLSEC
VSAMSEG.ESDSPATH	ESDSPATH	VSAM	No	No	LSR_BUFFER	DISP=SHR,DSN=IDMSV.TECHDC40.R141.ESDSPATH.PATH
VSAMSEG.ESDSVSAM	ESDSVSAM	VSAM	No	No	LSR_BUFFER	DISP=SHR,DSN=IDMSV.TECHDC40.R141.ESDSVSAM.CLUSTER
VSAMSEG.IDMSVSAM	IDMSVSAM	VSAM	No	No	DEFAULT_BUFFER	DISP=SHR,DSN=IDMSV.TECHDC40.R141.IDMSVSAM.CLUSTER
VSAMSEG.KSDSPATH	KSDSPATH	VSAM	No	No	NSR_BUFFER	DISP=SHR,DSN=IDMSV.TECHDC40.R141.KSDSPATH.PATH
VSAMSEG.KSDSVSAM	KSDSVSAM	VSAM	No	No	NSR_BUFFER	DISP=SHR,DSN=IDMSV.TECHDC40.R141.KSDSVSAM.CLUSTER
VSAMSEG.RRDSVSAM	RRDSVSAM	VSAM	No	No	NSR_BUFFER	DISP=SHR,DSN=IDMSV.TECHDC40.R141.RRDSVSAM.CLUSTER

## 5.4 IDMSLOOK

DMCL Journals		Page Size	# of Pages
-----		-----	-----
J1JRNL		2,004	5,000
J4JRNL		2,004	5,000
J3JRNL		2,004	5,000
J2JRNL		2,004	5,000
SYSJRNL		8,000	Archive

Journal Buffers		Buffer Size	# of Buffers
-----		-----	-----
JNL_BUFFER		2,004	5

DMCL Buffers	Buffer Size	CV Buffers	CV Type	Total CV Size	Local Buffers	Local Type	Total Local Size
-----	-----	-----	--	-----	-----	--	-----
II_BUFFER	4,276	500	OS	2,138,000	200	OS	855,200
LSR_BUFFER	28,672	4	OS	114,688	4	OS	114,688
NSR_BUFFER	28,672	4	OS	114,688	4	OS	114,688
DEFAULT_BUFFER	4,276	30	OS	128,280	20	OS	85,520
LOG_BUFFER	4,276	5	OS	21,380	5	DC	21,380

0 Bytes used for CV buffers in DC storage  
 2,517,036 Bytes used for CV buffers in OS storage  
 2,517,036 Bytes used for CV DMCL Buffers

21,380 Bytes used for LOCAL buffers in DC storage  
 1,170,096 Bytes used for LOCAL buffers in OS storage  
 1,191,476 Bytes used for LOCAL DMCL Buffers

Dbtable=R141DBTB                      Compiled Date=1999-11-15 10.44.19  
 The DEFAULT Dictionary is APPLDICT

```

DBNAME is *DEFAULT match on subschema is OPTIONAL
  Subschema IDMSNWK? maps to IDMSNWK? using DBNAME ----> APPLDICT
  Subschema IDMSCAT? maps to IDMSCAT? using DBNAME ----> APPLDICT

DBNAME is APPLDICT match on subschema is OPTIONAL
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSSQL                      this is an SQL segemnt

DBNAME is ASFDICT match on subschema is OPTIONAL
  Include SEGMENT ----> ASFDICT
  Include SEGMENT ----> SYSMMSG

DBNAME is EMPDEMO2 match on subschema is OPTIONAL
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> EMPDEMO2                    non unique PAGE GROUP
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSSQL                      this is an SQL segemnt

DBNAME is EMPDMIXD match on subschema is OPTIONAL Mixed PAGE GROUPS
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> EMPD2001
  Include SEGMENT ----> EMPD2002
  Include SEGMENT ----> EMPD2003
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSSQL                      this is an SQL segemnt

DBNAME is INGSQL match on subschema is OPTIONAL
  Include SEGMENT ----> INGSQL                    this is an SQL segemnt
  Include SEGMENT ----> SYSMMSG
  Include SEGMENT ----> SYSTEM

DBNAME is MIXDBNME match on subschema is OPTIONAL Mixed PAGE GROUPS
  Include SEGMENT ----> MIXSEGMENT
  Include SEGMENT ----> VSAMSEG

DBNAME is RJWDB match on subschema is OPTIONAL
  Include SEGMENT ----> APPLDICT
  Include SEGMENT ----> RJWDEMO
  Include SEGMENT ----> SYSMMSG
  
```

### 5.4.8 For more information

- **On using DCMT DISPLAY facilities for DMCL, DBTABLE, subschema, and program modules**, see *CA-IDMS System Tasks and Operator Commands*.
- **On defining DMCLs, DBTABLEs, and subschemas**, see *CA-IDMS Database Administration*.

## 5.5 IDMSRPTS

### 5.5.1 Description

**Purpose:** The data dictionary reports utility (IDMSRPTS) reports on information stored in a data dictionary. The utility is a useful tool for database administrators, application programmers, operations personnel, and managers. IDMSRPTS generates reports in four categories.

**Non-database reports:** Non-database reports provide information on occurrences of data dictionary entities that are not related to database processing.

You do not need to specify a schema, subschema, DMCL, database name table, or segment to get these reports.

---

<b>Full report name</b>	<b>Syntax name</b>	<b>Description</b>
Global Report Listing	GLBLRPT	Produces all non-database reports
Module/Process/Table Description Listings	MODLST	Produces all module, process, and table reports
Module Description Listing	MODULE	Reports on modules in the dictionary
ADS Process Description Listing	PROCESS	Reports on processes
Protocol Listing	PRTLST	Reports on protocols defined in the data dictionary
Record Copy Description Listing	RECCOPY	Reports on IDD-built records, reports, and transactions
OLQ Qfile Description Listing	QFILE	Reports on q-files defined in the data dictionary
Table Description Listing	TABLE	Reports on edit and code tables
User Listing	USER	Reports on users defined in the data dictionary

---

**Schema reports** Schema reports provide information on specified schemas defined in the data dictionary.

---

<b>Full report name</b>	<b>Syntax name</b>	<b>Description</b>
Schema Report Listing	SCHRPT	Produces all schema reports
Area Listing	AREALST	Reports on areas defined in a specified schema
Program Cross-Reference Listing	PGMLST	Reports on subschemas compiled under a specified schema
Schema Record Description Listing	RECDES	Reports on record types defined in a specified schema
Schema Set Description Listing	SETDES	Reports on sets defined in a specified schema

---

**Subschema reports:** Subschema reports provide information on specified subschemas defined in the data dictionary.

---

<b>Full report name</b>	<b>Syntax name</b>	<b>Description</b>
Subschema Report Listing	SSCRPT	Produces all subschema reports
Subschema Data Directory Listing	DATDIR	Provides general information on record types copied into a specified subschema
Logical Record Activity Descriptions	LRACT	Reports on program activity for logical records defined in a specified subschema
Subschema Logical Record Descriptions	LRDEFS	Reports on logical records defined in a specified subschema
Logical Record Path Descriptions	LRPATH	Reports on the paths defined for logical records in a specified subschema
Subschema Area Description Listing	SUBAREA	Reports on areas copied into a specified subschema
Subschema Record Description Listing	SUBREC	Provides comprehensive information on records copied into a specified subschema
Subschema Set Description Listing	SUBSET	Reports on sets copied into a specified subschema

---

**Physical Database Definition reports:** Physical database definition reports provide information on specified DMCLs, SEGMENTS, and DBTABLES defined in the dictionary.

---

<b>Full report name</b>	<b>Syntax name</b>	<b>Description</b>
Physical Database Report Listing	PDBRPT	Produces all physical database definition reports
DBTABLE Listing	DBTLST	Reports on the database names defined in the specified database names table
DMCL Listing	DMCLST	Reports on files, segments, and areas defined in the specified DMCL
Segment Listing	SEGLST	Reports on files, areas, and symbolics defined in the specified segment

---

### **Authorization**

#### *Physical database definition reports*

---

<b>To</b>	<b>You need this privilege or authority</b>
Run DMCLST report	DBADMIN on the dictionary being processed or DISPLAY on each DMCL on which you want to report
Run SEGLST reports	DBADMIN on the dictionary being processed or DISPLAY on each segment on which you want to report
DBTLST	DBADMIN on the dictionary being processed or DISPLAY on each DBTABLE on which you want to report

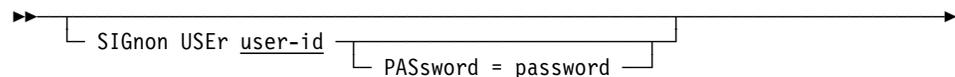
---

#### *Non-database, schema, and subschema reports*

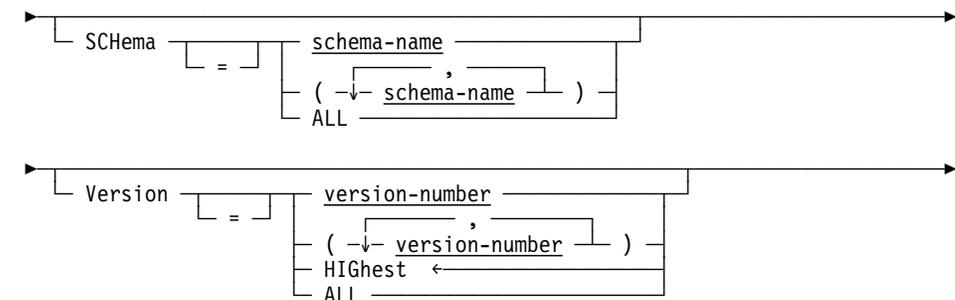
To	You need this privilege	On
Run any reports except physical database definition reports	SIGNON authority	Dictionary being processed if SECURITY FOR IDD SIGNON IS ON
Run all global reports except physical database definition reports	DISPLAY or higher	Basic entity type on which you want to report if SECURITY FOR IDD SIGNON IS ON
Run schema and subschema reports	DISPLAY or higher	Database entity on which you want to report if SECURITY FOR IDMS IS ON
Run all reports except User Listing and physical database definition reports	DISPLAY or higher	Entity occurrence on which you want to report if the PUBLIC ACCESS IS NONE

## 5.5.2 Syntax

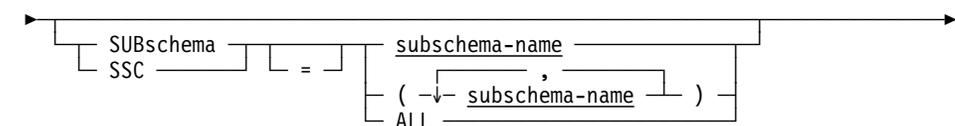
### SIGNON USER statement



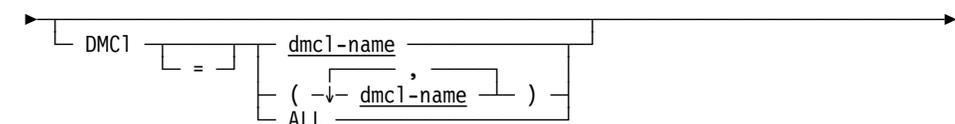
### SCHEMA statement

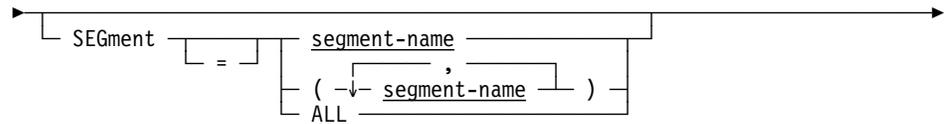
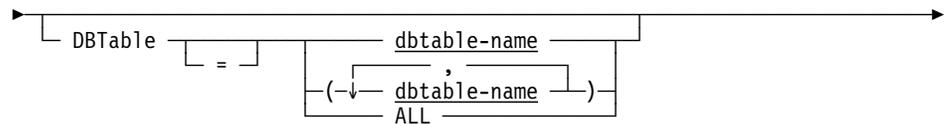
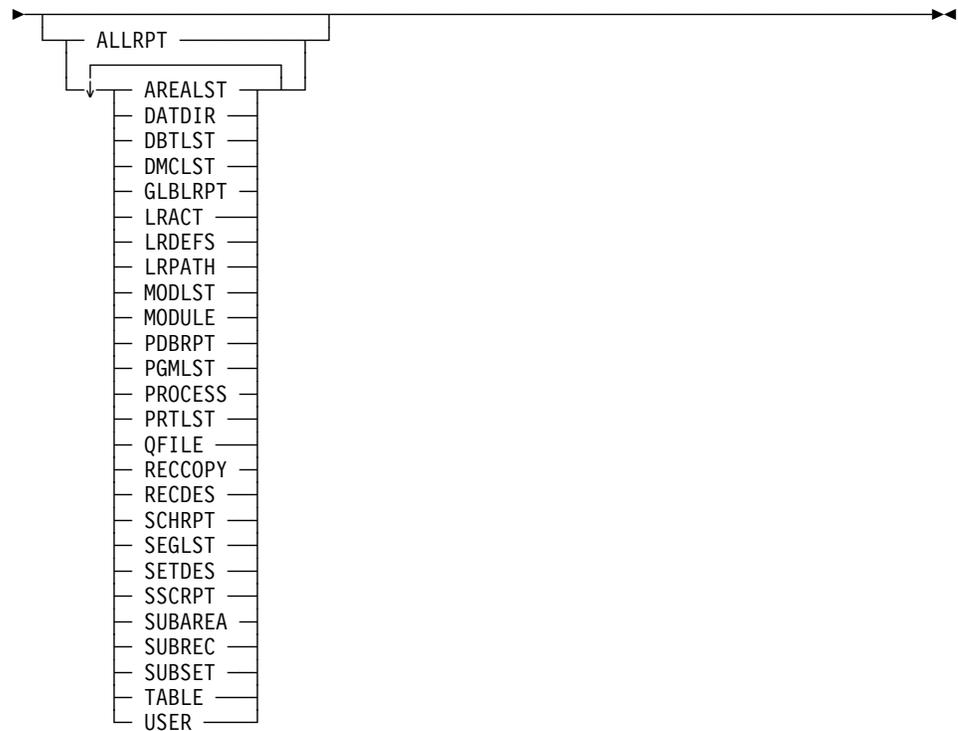


### SUBSCHEMA statement



### DMCL statement



**SEGMENT statement****DBTABLE statement****REPORTS SELECTION statement****5.5.3 Statements**

**Statement descriptions:** IDMSRPTS utility processing is controlled by the input statements listed in the table below.

**Note:** When specifying multiple input statements (other than SIGNON and REPORTS SELECTION), you must specify them in the order listed in the table below.

Statement	Description
SIGNON USER	Identifies the user running IDMSRPTS.
SCHEMA	Specifies the schema(s) to report on.
SUBSCHEMA	Specifies the subschema(s) to report on.
DMCL	Specifies the DMCL(s) to report on.
SEGMENT	Specifies the segment(s) to report on.
DBTABLE	Specifies the database name table(s) to report on.
REPORTS SELECTION	Specifies the report(s) to generate.

### SIGNON USER statement

**Purpose:** Your external user ID (from the job card) is always used for physical database definition security checking. The SIGNON USER statement is provided to override the external user ID for IDD security checking at IDD signon and for entity type and entity occurrence levels for reports other than physical database definition reports.

If you do not specify a SIGNON USER statement, your external user ID is also used for IDD security checking.

Use the SIGNON USER statement, specifying the user ID and password of an authorized user, if your external user ID is:

- Not defined in the dictionary and IDD SIGNON IS ON
- Is defined in the dictionary but does not have authority to signon or access the entity-types and/or occurrences on which you wish to report

If the IDD USER SIGNON OVERRIDE option does not allow signon override, and the external user ID is different from the SIGNON user ID, CA-IDMS ignores the SIGNON user ID for all IDD security checking.

If the dictionary is unsecured, you do not need to specify either a user ID or a password.

### Syntax

```

SIGNON USER user-id [PASSword = password]

```

### Parameters

#### user-id

Identifies a user in the dictionary.

**PASsword = password**

The password assigned to the user in the data dictionary.

A password must be provided if a SIGNON statement has been processed and the specified user ID is not the same as the external user ID and has been assigned a password in the dictionary.

**SCHEMA statement**

**Purpose:** Identifies the schema(s) to report on.

A schema statement is required if the reports selection includes any of the following:

SCHRPT

AREALST

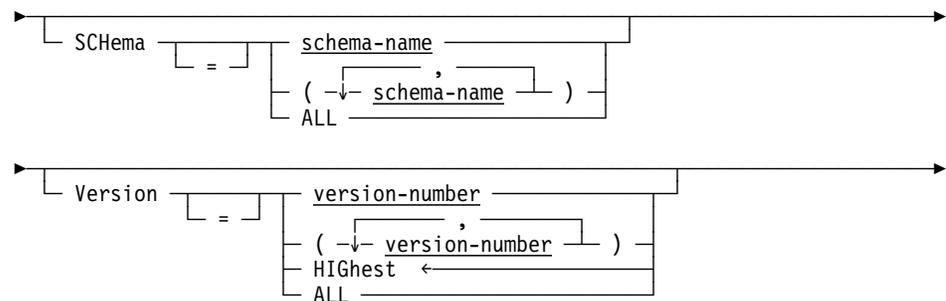
PGMLIST

RECDES

SETDES

**Authorization:** The identified user must hold authority for:

- IDD SIGNON if IDD SIGNON is secured and
- DISPLAY or higher if IDMS is secured and
- DISPLAY or higher on named schema(s) if PUBLIC ACCESS for the occurrence is NONE

**Syntax****Parameters****SCHEMA**

Identifies the schema(s) for which specified schema-level reports are to be produced, and/or the schema(s) to which any specified subschema(s) are related.

**schema-name**

The name of a schema defined in the data dictionary. You can specify multiple occurrences of *schema-name* by enclosing them in parentheses and separating them with commas.

IDMSRPTS reports only on the named schema(s).

You can specify up to 25 schema names.

**ALL**

Directs IDMSRPTS to report on all schemas in the data dictionary.

**Version**

Specifies version numbers for all the schemas named in the SCHEMA parameter.

**version-number**

Directs IDMSRPTS to report on the specified versions of each schema.

You can specify up to 25 version numbers.

**HIGHest**

Directs IDMSRPTS to report on the highest numbered version of each specified schema.

This is the default.

**ALL**

Directs IDMSRPTS to report on all versions of each specified schema.

**SUBSCHEMA statement**

**Purpose:** Identifies the subschema(s) to report on.

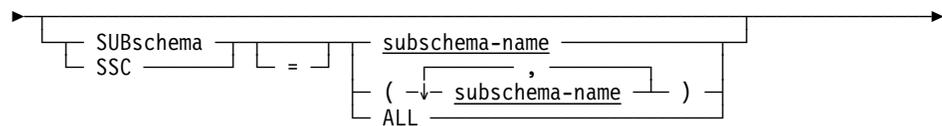
Both a schema and subschema statement are required if the reports selection includes any of the following:

SSCRPT  
 DATDIR  
 LRACT  
 LRDEFS  
 LRPATH  
 SUBREC  
 SUBSET

**Authorization:** The identified user must hold authority for:

- IDD SIGNON if IDD SIGNON is secured and
- DISPLAY or higher if IDMS is secured and
- DISPLAY or higher on requested schema(s) if PUBLIC ACCESS for the occurrence is NONE

**Syntax**



## Parameters

### SUBschema/SSC

Identifies the subschema(s) for which the specified subschema-level reports are to be produced.

SUBSCHEMA and SSC are synonyms, and may be used interchangeably.

### subschemaname

The name of a subschema compiled under a schema identified by the SCHEMA statement.

IDMSRPTS reports only on the named subschema(s).

You can specify up to 25 subschema names.

You can specify multiple occurrences of *subschemaname* by enclosing them in parentheses and separating them with commas.

### ALL

Directs IDMSRPTS to reports on all subschemas compiled under the schema(s) identified by the SCHEMA statement.

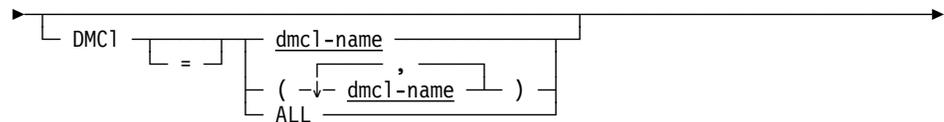
## DMCL statement

**Purpose:** Identifies the DMCL(s) to report on.

A DMCL statement is required if the reports selection includes DMCLST.

**Authorization:** The external user ID must hold either DBADMIN authority for the dictionary or DISPLAY authority on requested DMCL(s).

## Syntax



## Parameters

### DMCL

Identifies the DMCL(s) for which the DMCLST report is to be produced.

### dmcl-name

Identifies the name of a DMCL defined in the catalog component of the dictionary. You can specify multiple occurrences of *dmcl-name* by enclosing them in parentheses and separating them with commas.

IDMSRPTS reports only on the named DMCL(s).

You can specify up to 25 DMCL names.

### ALL

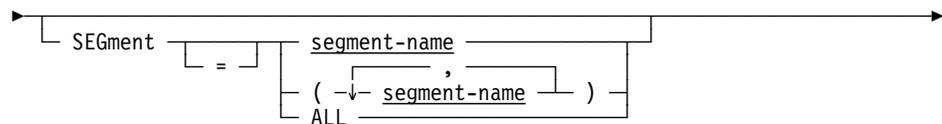
Directs IDMSRPTS to reports on all DMCLs defined in the dictionary.

**SEGMENT statement**

**Purpose:** Identifies the segment(s) to report on.

A SEGMENT statement is required if the reports selection includes SEGLST.

**Authorization:** The external user ID must hold either DBADMIN authority for the dictionary or DISPLAY authority on requested segment(s).

**Syntax****Parameters****SEGment**

Identifies the segment(s) for which the SEGLST report is to be produced.

**segment-name**

The name of a segment defined in the catalog component of the dictionary. You can specify multiple occurrences of *segment-name* by enclosing them in parentheses and separating them with commas.

IDMSRPTS reports only on the named segment(s).

You can specify up to 25 segment names.

**ALL**

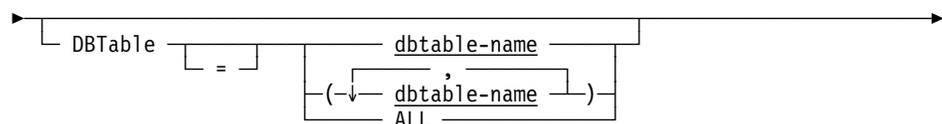
Directs IDMSRPTS to reports on all segments defined in the dictionary.

**DBTABLE statement**

**Purpose:** Identifies the database name table(s) to report on.

A DBTABLE statement is required if the reports selection includes DBTLST.

**Authorization:** The external user ID must hold either DBADMIN authority for the dictionary or DISPLAY authority on requested database names table(s).

**Syntax****Parameters****DBTable**

Identifies the database name table(s) for which specified dbtable-level reports are to be produced.

**dbtable-name**

Identifies the name of a database name table defined in the catalog component of the dictionary. You can specify multiple occurrences of *dbtable-name* by enclosing them in parentheses and separating them with commas.

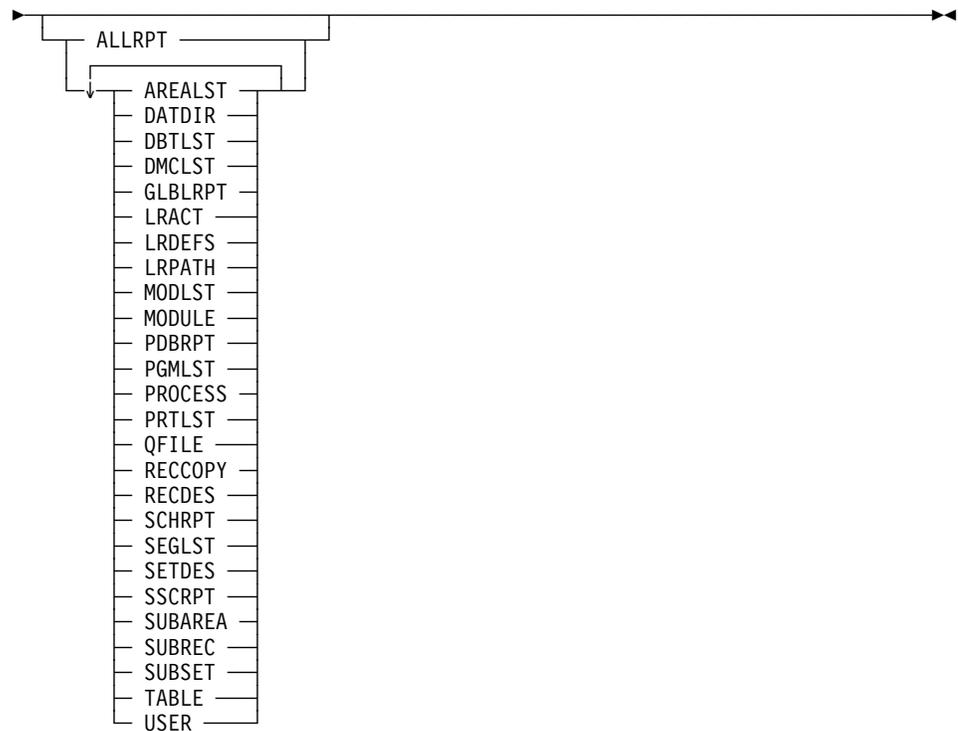
IDMSRPTS reports only on the named database name table(s).

**ALL**

Directs IDMSRPTS to report on all database name tables defined in the catalog segment of the dictionary.

**REPORTS SELECTION statement**

**Purpose** Identifies the reports to generate.

**Syntax****Parameters****ALLRPT**

Generates all non-database reports; all reports except DBTLST, DMCLST, and SEGLST.

**AREALST**

Generates the Area Listing for each specified schema.

**DATDIR**

Generates the Subschema Data Directory Listing for each specified subschema.

**DBTLST**

Generates the DBNAMES Table Report.

**DMCLST**

Generates the DMCL Listing for each specified DMCL.

**GLBLRPT**

Generates all database independent reports: MODULE, PROCESS, TABLE, RECCOPY, PRTLST, QFILE, and USER.

No SCHEMA, SUBSCHEMA, DMCL, or SEGMENT statement is required to generate these reports.

**LRACT**

Generates the Logical Record Activity Descriptions report for each specified subschema.

**LRDEFS**

Generates the Subschema Logical Record Descriptions report for each specified subschema.

**LRPATH**

Generates the Logical Record Path Descriptions report for each specified subschema.

**MODLST**

Generates the module, process, and table reports.

**MODULE**

Generates the Module Description Listing, which lists each module that you are authorized to display. Processes, tables, qfiles, and protocols are not included in this report.

**PDBRPT**

Generates the physical database reports: DBTLST, DMCLST, and SEGLST.

If specified:

- DMCLST will be produced for all DMCLs identified on DMCL statements
- SEGLST will be produced for all SEGMENTS identified on SEGMENT statements
- DBTLST will be produced for all DBTABLES identified on DBTABLE statements

**PGMLST**

Generates the Program Cross-Reference Listing for each specified schema version.

**PROCESS**

Generates the ADS Process Description Listing, which lists each process that you are authorized to display.

**PRTLST**

Generates the Protocol Listing.

**QFILE**

Generates the OLQ Qfile Description Listing.

**RECCOPY**

Generates the Record Copy Description Listing, which lists each IDD-built record, report, and transaction that you are authorized to display.

**RECDES**

Generates the Schema Record Description Listing for each specified schema version.

**SCHRPT**

Generates all schema reports for each specified schema version: AREALST, PGMLST, RECDES, AND SETDES.

The schema reports do not report on unvalidated schemas or schemas that have errors.

**SEGLST**

Generates the Segment Listing for each specified segment.

**SETDES**

Generates the Set Description Listing for each specified schema version.

**SSCRPT**

Generates all subschema reports for each specified subschema, schema, and version: DATDIR, LRACT, LRDEFS, LRPATH, SUBREC, AND SUBSET.

The subschema reports do not report on unvalidated subschemas or subschemas that contain errors.

**SUBAREA**

Generates the Subschema Area Description Listing for each specified subschema.

**SUBREC**

Generates the Subschema Record Description Listing for each specified subschema.

**SUBSET**

Generates the Subschema Set Description Listing for each specified subschema.

**TABLE**

Generates the Table Description Listing, which lists each table that you are authorized to display.

**USER**

Generates the User Listing.

## 5.5.4 Usage

**Input:** Input to the IDMSRPTS utility consists of statements to control the utility processing.

**Note:** When specifying multiple input statements (other than SIGNON and REPORTS SELECTION), you must specify them in this order:

1. SCHEMA
2. SUBSCHEMA
3. DMCL
4. SEGMENT
5. DBTABLE

**Specifying a dictionary name and nodename:** To override the default dictionary name or dictionary nodename the IDMSRPTS utility uses to produce reports, use the DICTNAME and DICTNODE parameters in the SYSIDMS parameter file.

►► For a complete description of the SYSIDMS parameter file, see *CA-IDMS Database Administration*.

**Output:** The IDMSRPTS utility generates reports on information stored in a data dictionary.

**Batch operating mode:** You can execute the IDMSRPTS utility either in local mode or under the central version.

## 5.5.5 JCL considerations

Refer to the chapter pertaining to your operating system for JCL to execute IDMSRPTS.

## 5.5.6 Examples

**Protocol Listing:** The input parameters shown below direct IDMSRPTS to generate the Protocol Listing.

```
prt1st
```

**DMCL, SEGMENT, DBTABLE Listing:** The input parameters shown below direct IDMSRPTS to generate the DMCL Listing for the DMCL IDMSDMCL, a Segment Listing for each segment defined in the dictionary and a DBTABLE Listing for each DBTABLE defined in the dictionary.

```
dmcl=idmsdmc1  
segment=all  
dbtable=all  
pdrprt
```

**Schema Record and Set Description Listing:** The input parameters shown below direct IDMSRPTS to generate the Schema Record Description Listing and the Set Description Listing for versions 1, 99, and 100 of the schema EMPSCHM.

```
schema=empschm version=(1, 99, 100)  
recdes setdes
```

**Subschema Area and Data Directory Listing:** The input parameters shown below direct IDMSRPTS to generate the Subschema Area Description Listing and the Subschema Data Directory Listing for each subschema compiled under the highest existing version of the schema EMPSCHEM.

```
schema=empschm subschema=all
subarea datdir
```

## 5.5.7 Sample output

### Protocol Listing

```

IDMSRPTS 15.0          GLOBAL          PRTLST
PRTLST              ----- PRTLST LISTING -----          DATE    TIME    PAGE
                   DICTIONARY SYSTEM  OF NODE DEFAULT          09/30/99 180854    1

PROTOCOL NAME .. BATCH
PROTOCOL VERSION 0001
DATE CREATED ... 09/18/99
LAST UPDATED ... N/A
LANGUAGE ..... COBOL

                                MOVE @DMLSEQ TO DML-SEQUENCE
                                MOVE @OCCUR TO RECORD-OCCUR
                                MOVE @NODN TO SSC-NODN
                                MOVE @DBN TO SSC-DBN
                                MOVE @DICTNO TO SSC-DNO
                                MOVE @DICTNA TO SSC-DNA
                                MOVE @LNG TO LRC-LRPXELNG
                                MOVE @LRSIZE TO LRC-MAXVXP
                                MOVE '@LRSTAT' TO LR-STATUS
                                MOVE '@VERB' TO LRVERB
                                MOVE '@NAME' TO LRNAME

@)PXELP
                                MOVE @XDE TO @PXE
                                ADD @PXELNG TO LRC-LRPXELNG

@)PXELP
@)MISCLoop
                                @MISC

@)MISCLoop
                                CALL 'IDMS' USING SUBSCHEMA-CTRL @CALLIDMS
                                IDBMSCOM (@FUNC1)
                                @SSCTRL
                                @SSNAME
                                @SSLRCTRL
                                @REC
                                @SET
                                @AREA
                                @DATANAME
                                @RECBUF
                                @OCCUR2
                                IDBMSCOM (@FUNC2)
                                IDBMSCOM (@KEEP)
                                @P1
                                @P2
                                @P3
                                @P4

@)PALoop
                                @PA1
                                @PA2
                                @PA3

@)PALoop
                                @PERIOD

                                IF ERROR-STATUS EQUAL TO '@STATUS'

...

```

## DMCL, SEGMENT, and DBTABLE Listing

IDMSRPTS 15.0 DMCLST	----- DMCL LISTING ----- DICTIONARY SYSTEM OF NODE DEFAULT DMCL CVDML13							DATE 09/30/99	TIME 181203	PAGE 1
	EXTERNAL NAME	LOW BLOCK NR	REL BLOCK NR	HIGH REL BLOCK NR	PAGE GROUP	LOW PAGE NUMBER	CALC HIGH PAGE NR	HIGH PAGE NUMBER	BLK/PAGE SIZE	PAGE RESV
SEGMENT .....	APPLCAT					0				
FILE .....	APPLCAT	APPLCAT	1	1000					4276	
AREA .....	DDL CAT		1	1000	0	304001	305000	305000	4276	0
FILE .....	APPLCATX	APPLCATX	1	500					4276	
AREA .....	DDL CATX		1	500	0	306001	306500	306500	4276	0
FILE .....	APPLCATL	APPLCATL	1	500					4276	
AREA .....	DDL CATLOD		1	500	0	308001	308500	308500	4276	0
SEGMENT .....	APPLNWK					0				
FILE .....	APPLDML	APPLDML	1	1000					4276	
AREA .....	DDL DML		1	1000	0	300001	301000	301000	4276	0
FILE .....	APPLLOD	APPLLOD	1	500					4276	
AREA .....	DDL DLOD		1	500	0	302001	302500	302500	4276	0
SEGMENT .....	ASFNWK					0				
FILE .....	ASFDML	ASFDML	1	1000					4276	
AREA .....	DDL DML		1	1000	0	400001	401000	401000	4276	0
FILE .....	ASFLOD	ASFLOD	1	500					4276	
AREA .....	DDL DLOD		1	500	0	402001	402500	402500	4276	0
FILE .....	ASFDEFN	ASFDEFN	1	500					4276	
AREA .....	IDMSR-AREA		1	500	0	404001	404500	404500	4276	0
FILE .....	ASFDATA	ASFDATA	1	500					4276	
AREA .....	IDMSR-AREA2		1	500	0	406001	406500	406500	4276	0
SEGMENT .....	CATSYS					0				
FILE .....	DCCAT	DCCAT	1	100					4276	
	...									

5.5 IDMSRPTS

```

IDMSRPTS 15.0          ----- SEGMENT LISTING -----          DATE    TIME    PAGE
SEGLST                DICTIONARY SYSTEM  OF NODE DEFAULT    09/30/99 181203  1
                      SEGMENT APPLCAT  PAGE GROUP    0
                                EXTERNAL  LOW REL  HIGH REL  PAGE  LOW PAGE  CALC HIGH  HIGH PAGE  BLK/PAGE  PAGE
                                NAME      BLOCK NR  BLOCK NR  GROUP  NUMBER  PAGE NR   NUMBER   SIZE      RESV
SEGMENT ..... APPLCAT                0
AREA ..... DDLCAT                0  304001  305000  305000  4276  0
  WITHIN FILE .... APPLCAT  APPLCAT      1  1000      304001      305000  4276
AREA ..... DDLCATX                0  306001  306500  306500  4276  0
  WITHIN FILE .... APPLCATX  APPLCATX    1  500      306001      306500  4276
AREA ..... DDLCATL0D                0  308001  308500  308500  4276  0
  WITHIN FILE .... APPLCATL  APPLCATL    1  500      308001      308500  4276
  ...
  
```

```

IDMSRPTS 15.0          ----- DBTABLE LISTING -----          DATE    TIME    PAGE
DBTLST                DICTIONARY SYSTEM  OF NODE DEFAULT    09/30/99 181203  1
                      DBTABLE CVDBTB13
DBTABLE ..... CVDBTB13
DBTABLE MAPPINGS
  SUBSCHEMA IDMSCAT?  MAPS TO SUBSCHEMA IDMSCAT?  USING DBNAME SYSTEM
  SUBSCHEMA IDMSNWK?  MAPS TO SUBSCHEMA IDMSNWK?  USING DBNAME SYSTEM
  SUBSCHEMA IDMSRSSA  MAPS TO SUBSCHEMA IDMSRSSA  USING DBNAME ASFDICT
  SUBSCHEMA RC??????  MAPS TO SUBSCHEMA RC??????  USING DBNAME ASFDICT
  SUBSCHEMA RU??????  MAPS TO SUBSCHEMA RU??????  USING DBNAME ASFDICT
  SUBSCHEMA CITSALLP  MAPS TO SUBSCHEMA CITSALLP  USING DBNAME INFODB
  SUBSCHEMA MTSSINFP  MAPS TO SUBSCHEMA MTSSINFP  USING DBNAME INFODB
DBNAME ..... APPLDICT  MATCH ON SUBSCHEMA OPTIONAL
  SEGMENT  APPLCAT
  SEGMENT  APPLNWK
  SEGMENT  SYSMG
DBNAME ..... ASFDICT  MATCH ON SUBSCHEMA OPTIONAL
  SEGMENT  ASFNWK
  SEGMENT  SYSMG
DBNAME ..... DIRLDICT  MATCH ON SUBSCHEMA OPTIONAL
  SEGMENT  DIRLNWK
  SEGMENT  SYSMG
DBNAME ..... INFODB   MATCH ON SUBSCHEMA OPTIONAL
  SEGMENT  INFOTSIS
  SEGMENT  TSIS
DBNAME ..... INFODICT  MATCH ON SUBSCHEMA OPTIONAL
  SEGMENT  INFONWK
DBNAME ..... SYSTEM   MATCH ON SUBSCHEMA OPTIONAL
  SEGMENT  CATSYS
  SEGMENT  SYSMG
  SEGMENT  SYSTEM
  ...
  
```

**Schema Record and Set Description Listing**

IDMSRPTS 15.0		-- SCHEMA RECORD DESCRIPTION LISTING ---			DATE	TIME	PAGE
RECDES		DICTIONARY APPLDCT OF NODE DEFAULT			09/30/99	181426	1
		SCHEMA EMPSCHM VERSION 1					
RECORD NAME.....	COVERAGE					RLGTH=	36
RECORD VERSION.....	0001					DLGTH=	16
RECORD ID.....	0400					KLGTH=	20
RECORD LENGTH.....	FIXED					DSTRT=	20
LOCATION MODE.....	VIA SET	EMP-COVERAGE	DISPLACEMENT 0000	PAGES			
WITHIN.....	INS-DEMO-REGION	OFFSET	5 PGS FOR	45 PGS			
DBKEY POSITIONS.....	SET.....	TYPE.....	NEXT	PRIOR OWNER			
	EMP-COVERAGE	MEMBER	1	2	3		
	COVERAGE-CLAIMS	OWNER	4	5			
DATA ITEM.....	REDEFINES.....	USAGE.....	VALUE.....	PICTURE.....		STRT	LGTH
02	SELECTION-DATE-0400	DISPLAY				1	6
03	SELECTION-YEAR-0400	DISPLAY			9(2)	1	2
03	SELECTION-MONTH-0400	DISPLAY			9(2)	3	2
03	SELECTION-DAY-0400	DISPLAY			9(2)	5	2
02	TERMINATION-DATE-0400	DISPLAY				7	6
03	TERMINATION-YEAR-0400	DISPLAY			9(2)	7	2
03	TERMINATION-MONTH-0400	DISPLAY			9(2)	9	2
03	TERMINATION-DAY-0400	DISPLAY			9(2)	11	2
02	TYPE-0400	DISPLAY			X	13	1
88	MASTER-0400	COND	'M'			13	
88	FAMILY-0400	COND	'F'			13	
88	DEPENDENT-0400	COND	'D'			13	
02	INS-PLAN-CODE-0400	DISPLAY			X(3)	14	3
88	GROUP-LIFE-0400	COND	'001'			14	
88	HMO-0400	COND	'002'			14	
88	GROUP-HEALTH-0400	COND	'003'			14	
88	GROUP-DENTAL-0400	COND	'004'			14	
*****							*****
REC SYNONYM NAME...	COVERAGE					RLGTH=	36
REC SYNONYM VER....	0001					DLGTH=	16
LANGUAGE(S).....	ASSEMBLER						
DATA ITEM.....	REDEFINES.....	USAGE.....	VALUE.....	PICTURE.....		STRT	LGTH
02	COVSELDT	DISPLAY				1	6
03	COVSELYR	DISPLAY			9(2)	1	2
03	COVSELMO	DISPLAY			9(2)	3	2
03	COVSELDA	DISPLAY			9(2)	5	2
02	COVTRMDT	DISPLAY				7	6
03	COVTRMYR	DISPLAY			9(2)	7	2
03	COVTRMMO	DISPLAY			9(2)	9	2
03	COVTRMDA	DISPLAY			9(2)	11	2
02	COVTYPE	DISPLAY			X	13	1
88	COVMASTR	COND	'M'			13	
88	COVFAMLY	COND	'F'			13	
88	COVDPNDT	COND	'D'			13	
.							
.							
.							

5.5 IDMSRPTS

IDMSRPTS 15.0	----- SET DESCRIPTION LISTING -----			DATE	TIME	PAGE
SETDES	DICTIONARY APPLDICT OF NODE DEFAULT			09/30/99	181426	1
	SCHEMA	EMPSCHM	VERSION			
SET..... CALC	MODE CHAIN		ORDER SORTED			
OWNER.... SRI	0001	NEXT	PRIOR			
MEMBER... SYSTEM	0007	NEXT	PRIOR	MANDATORY	AUTO	
MEMBER... DEPARTMENT	0410	NEXT	PRIOR	MANDATORY	AUTO	
IN AREA ORG-DEMO-REGION				CALC KEY	DEPT-ID-0410	ASC NAT DUP NOT ALLOW
MEMBER... EMPLOYEE	0415	NEXT	PRIOR	MANDATORY	AUTO	NAT DUP NOT ALLOW
IN AREA EMP-DEMO-REGION				CALC KEY	EMP-ID-0415	ASC NAT DUP NOT ALLOW
MEMBER... INSURANCE-PLAN	0435	NEXT	PRIOR	MANDATORY	AUTO	NAT DUP NOT ALLOW
IN AREA INS-DEMO-REGION				CALC KEY	INS-PLAN-CODE-0435	ASC NAT DUP NOT ALLOW
MEMBER... JOB	0440	NEXT	PRIOR	MANDATORY	AUTO	NAT DUP NOT ALLOW
IN AREA ORG-DEMO-REGION				CALC KEY	JOB-ID-0440	ASC NAT DUP NOT ALLOW
MEMBER... OFFICE	0450	NEXT	PRIOR	MANDATORY	AUTO	NAT DUP NOT ALLOW
IN AREA ORG-DEMO-REGION				CALC KEY	OFFICE-CODE-0450	ASC NAT DUP NOT ALLOW
MEMBER... SKILL	0455	NEXT	PRIOR	MANDATORY	AUTO	NAT DUP NOT ALLOW
IN AREA ORG-DEMO-REGION				CALC KEY	SKILL-ID-0455	ASC NAT DUP NOT ALLOW
SET..... COVERAGE-CLAIMS	MODE CHAIN		ORDER LAST			
OWNER.... COVERAGE	0400	NEXT	PRIOR			
IN AREA INS-DEMO-REGION						
MEMBER... HOSPITAL-CLAIM	0430	NEXT	PRIOR	MANDATORY	AUTO	
IN AREA INS-DEMO-REGION						
MEMBER... NON-HOSP-CLAIM	0445	NEXT	PRIOR	MANDATORY	AUTO	
IN AREA INS-DEMO-REGION						
MEMBER... DENTAL-CLAIM	0405	NEXT	PRIOR	MANDATORY	AUTO	
IN AREA INS-DEMO-REGION						
SET..... DEPT-EMPLOYEE	MODE CHAIN		ORDER SORTED			
OWNER.... DEPARTMENT	0410	NEXT	PRIOR			
IN AREA ORG-DEMO-REGION						
MEMBER... EMPLOYEE	0415	NEXT	PRIOR	OWNER	OPTIONAL AUTO	NAT DUP LAST
IN AREA EMP-DEMO-REGION					SORT KEY	ASC
					EMP-LAST-NAME-0415	ASC
					EMP-FIRST-NAME-0415	ASC
SET..... EMP-COVERAGE	MODE CHAIN		ORDER FIRST			
OWNER.... EMPLOYEE	0415	NEXT	PRIOR			
IN AREA EMP-DEMO-REGION						
MEMBER... COVERAGE	0400	NEXT	PRIOR	OWNER	MANDATORY AUTO	
IN AREA INS-DEMO-REGION						
SET..... EMP-EMPOSITION	MODE CHAIN		ORDER FIRST			
OWNER.... EMPLOYEE	0415	NEXT	PRIOR			
IN AREA EMP-DEMO-REGION						
MEMBER... EMPOSITION	0420	NEXT	PRIOR	OWNER	MANDATORY AUTO	
.						
.						
.						

## Subschema Area and Data Directory Listing

IDMSRPTS 15.0	-- SUBSCHEMA DATA DIRECTORY LISTING --	DATE	TIME	PAGE		
DATDIR	DICTIONARY APPLDICT OF NODE DEFAULT	09/30/99	181529	1		
	SUBSCHEMA EMPHTLI OF SCHEMA EMPSCHM VERSION 100					
RECORD:	COVERAGE	ID: 0400	VER: 100	TYPE: I LEN: 16		
	DATA NAME	LEVEL	STRT	LENGTH	TYPE	PICTURE
	SELECTION-DATE-0400	02	1	6	GROUP	
	SELECTION-YEAR-0400	03	1	2	DISPLAY	9(2)
	SELECTION-MONTH-0400	03	3	2	DISPLAY	9(2)
	SELECTION-DAY-0400	03	5	2	DISPLAY	9(2)
	TERMINATION-DATE-0400	02	7	6	GROUP	
	TERMINATION-YEAR-0400	03	7	2	DISPLAY	9(2)
	TERMINATION-MONTH-0400	03	9	2	DISPLAY	9(2)
	TERMINATION-DAY-0400	03	11	2	DISPLAY	9(2)
	TYPE-0400	02	13	1	A/N	X
	MASTER-0400	88			COND	
		VALUE 'M'				
	FAMILY-0400	88			COND	
		VALUE 'F'				
	DEPENDENT-0400	88			COND	
		VALUE 'D'				
	INS-PLAN-CODE-0400	02	14	3	A/N	X(3)
	GROUP-LIFE-0400	88			COND	
		VALUE '001'				
	HMO-0400	88			COND	
		VALUE '002'				
	GROUP-HEALTH-0400	88			COND	
		VALUE '003'				
	GROUP-DENTAL-0400	88			COND	
		VALUE '004'				
	.					
	.					
	.					

IDMSRPTS 15.0	-- SUBSCHEMA DATA DIRECTORY LISTING --	DATE	TIME	PAGE		
DATDIR	DICTIONARY APPLDICT OF NODE DEFAULT	09/30/99	181529	2		
	SUBSCHEMA EMPHTLI OF SCHEMA EMPSCHM VERSION 100					
RECORD:	COVERAGE	ID: 0400	VER: 100	TYPE: I LEN: 16		
	SYNONYM OF: COVERAGE					
	LANGUAGE(S): ASSEMBLER					
	DATA NAME	LEVEL	STRT	LENGTH	TYPE	PICTURE
	COVSELDT	02	1	6	GROUP	
	COVSELYR	03	1	2	DISPLAY	9(2)
	COVSELMO	03	3	2	DISPLAY	9(2)
	COVSELDA	03	5	2	DISPLAY	9(2)
	COVTRMDT	02	7	6	GROUP	
	COVTRMYR	03	7	2	DISPLAY	9(2)
	COVTRMMO	03	9	2	DISPLAY	9(2)
	COVTRMDA	03	11	2	DISPLAY	9(2)
	COVTYPE	02	13	1	A/N	X
	COVMASTR	88			COND	
		VALUE 'M'				
	COVFAMILY	88			COND	
		VALUE 'F'				
	COVDPNDT	88			COND	
		VALUE 'D'				
	COVPLNCD	02	14	3	A/N	X(3)
	GROUP-LIFE	88			COND	
		VALUE '001'				
	HMO	88			COND	
		VALUE '002'				
	GROUP-HEALTH	88			COND	
		VALUE '003'				
	GROUP-DENTAL	88			COND	
		VALUE '004'				
	.					
	.					
	.					

### 5.5.8 For more information

- On CA reports produced by IDMSRPTS, see *CA-IDMS Reports*.

## 5.6 IDMSRSTC

### 5.6.1 Description

The schema compare utility, IDMSRSTC, generates IDMSRSTT macro statements for use in a database restructure operation. IDMSRSTC generates the statements by comparing two schemas:

- **An old schema** that describes the database before restructuring
- **A new schema** that describes the database after restructuring

The utility reads the schema definitions from the data dictionary.

### 5.6.2 Syntax

#### SIGNON statement

```

SIGNON
  USER name [ is ] user-id  PASSWORD [ is ] password
.

```

#### SCHEMA statement

```

OLD SCHEMA name is old-schema-name
  Version is [ version-number ]
             [ HIGhest ]
             [ LOWest ]
NEW SCHEMA name is new-schema-name
  Version is [ version-number ]
             [ HIGhest ]
             [ LOWest ]
.

```

#### SIGNOFF statement

```

SIGNOFF [ BYE ] [ LOGOFF ] .

```

### 5.6.3 Input parameter statements

**Parameter statement descriptions:** IDMSRSTC utility processing is controlled by the following input parameter statements:

Statement	Description
SIGNON	Initiates IDMSRSTC processing
SCHEMA	Identifies the old and new schemas to be compared
SIGNOFF	Terminates IDMSRSTC processing

**Coding considerations:** You must code the IDMSRSTC input parameter statements in uppercase between columns 1 and 72, inclusive. Each statement must end with a period.

**SIGNON statement** Must be the first parameter statement you submit to IDMSRSTC.

**SCHEMA statement** You can include any number of SCHEMA statements.:

**SIGNOFF statement** Must be the last parameter statement.

#### SIGNON statement

**Purpose:** Initiates IDMSRSTC processing and optionally specifies a user identifier and password.

#### Syntax

```

SIGNON _____
|
|  USER name  [ is ]  user-id  —  PASsword  [ is ]  password
|
|_____
. _____

```

#### Parameters

##### USER name is/= user-id

Identifies a user defined in the user catalog. The specified user must have the authority to access the schemas named in the IDMSRSTC run.

If no USER ID is specified, USER ID is the user known to the execution environment. If SIGNON OVERRIDE is not allowed in the dictionary, USER ID, if specified, must be the same as that known to the execution environment.

IS and = are synonyms and can be used interchangeably.

**PASsword is/= password**

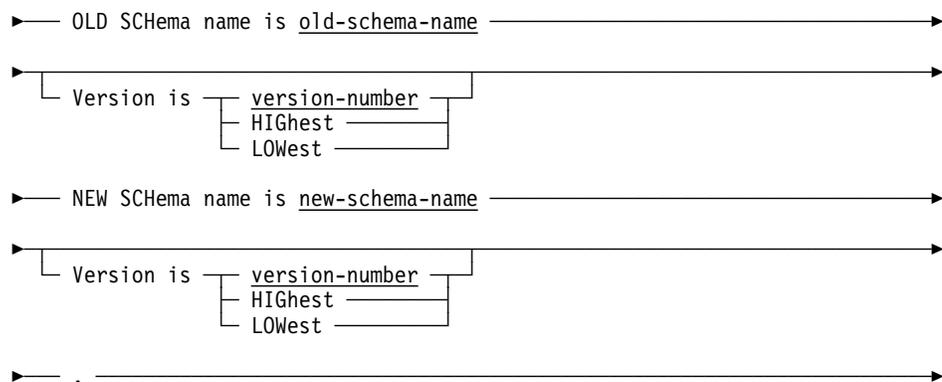
Specifies the password associated with the user identified in the USER parameter. If the password contains embedded blanks, you must enclose it in site-standard quotation marks.

You must include the PASSWORD parameter if the user specified in the USER parameter is associated with a password in the dictionary, unless the USER ID is the same as the user known to the execution environment. In this case, the password is ignored.

IS and = are synonyms and can be used interchangeably.

**SCHEMA statement**

**Purpose:** Identifies an old schema and a new schema to be compared by IDMSRSTC. For each SCHEMA statement you submit, IDMSRSTC generates a set of IDMSRSTT macro statements. Each set of macro statements begins with IDMSRSTT BUFSIZE and ends with END.

**Syntax****Parameters****OLD SCHEMA name is old-schema-name**

Specifies the name of a schema that describes the database before restructuring.

*Old-schema-name* must identify a schema defined in the data dictionary.

**Version is**

Qualifies the old or new schema name with a version number.

If you omit the VERSION parameter for the old or new schema, IDMSRSTC uses the dictionary default version number for existing entities.

**version-number**

Explicitly specifies the schema version number.

**HIGhest**

Directs IDMSRSTC to use the highest version number assigned to the named schema.



Circumstance	Modification
A database procedure not included in the new schema is to be executed during the run of the RESTRUCTURE SEGMENT utility.	Name the procedure in the NUPROCS parameter of the appropriate IDMSRSTT RECNAME statement.
A record being restructured includes one or more redefined elements in the new schema.	Delete the IDMSRSTT FIELD statements for the redefining elements.  Alternatively, if only the prefix portion of the record is being restructured, and the record is fixed length and uncompressed, replace the IDMSRSTT FIELD statements for the record with a single IDMSRSTT FIELD statement that specifies ALL.
A record being restructured has a field that is being expanded by increasing the number of positions to the right of the decimal point.	Move the IDMSRSTT FIELD statement for the new byte(s) after the statement for the original field if the field is an unsigned zone. The new position in each statement should be changed to reflect the proper position in the record's new layout.  Signed numeric or packed fields cannot be restructured by a single run of the RESTRUCTURE utility. To properly handle the sign nibble may require that the new bytes be placed to the left of the original field. You should run a user program to modify each affected record by multiplying the expanded field by the proper factor to realign the decimal position.

►►For a description of the IDMSRSTT macro statements, see Appendix B, “IDMSRSTT Macro Statements.”

**Assemble the IDMSRSTT macro statements:** After reviewing and modifying the IDMSRSTT macro statements, you must assemble the statements into a base restructuring table for use by the RESTRUCTURE SEGMENT and RESTRUCTURE CONNECT utilities.

**Multiple sets of IDMSRSTT macro statements:** IDMSRSTC can generate multiple sets of IDMSRSTT macro statements; one set for each SCHEMA statement you submit. RESTRUCTURE SEGMENT and RESTRUCTURE CONNECT, however, can use only one base restructuring table at a time. Therefore, you must assemble separately each set of IDMSRSTT macro statements generated by IDMSRSTC.

## 5.6.5 Example

If you execute IDMSRSTC with the input statements shown below, the utility generates IDMSRSTT macro statements that describe the differences between the LRDKSCHM and LRDKSCH2 schemas with the dictionary default version number.

```
signon usage mode is retrieval.
old schema name is lrdkschm
  new schema name is lrdksch2.
signoff.
```

## 5.6.6 Sample output

The IDMSRSTC utility generates the following report after executing the parameters in the example above.

```

IDMSRSTC 15.0          COMPUTER ASSOCIATES INTERNATIONAL      DATE      TIME      PAGE
CAGJF0          RESTRUCTURE SCHEMA COMPARE ACTIVITY LIST      09/18/99   16411366   0001

000001          SIGNON USAGE MODE IS RETRIEVAL.
000002          OLD SCHEMA NAME IS LRDKSCHM
000003          NEW SCHEMA NAME IS LRDKSCH2.
IDMSRSTT BUFSIZE=(500,500)    OLD BB          00000001
*              NEW BB          00000002
IDMSRSTT RECNAME=AA          00000003
IDMSRSTT SETPTR=(1,1)        COPY OWNER NEXT A-B 00000004
IDMSRSTT SETPTR=(*,2,A-B)    ADD OWNER PRIOR A-B 00000005
IDMSRSTT FIELD=ALL          00000006
IDMSRSTT RECNAME=BB,MINLEN=(16,40,452),DCT=BUILTIN 00000007
IDMSRSTT SETPTR=(1,1)        COPY MEMBER NEXT A-B 00000008
IDMSRSTT SETPTR=(,2,A-B)    ADD MEMBER PRIOR A-B 00000009
IDMSRSTT SETPTR=(,3,A-B)    ADD MEMBER OWNER A-B 00000010
IDMSRSTT SETPTR=(2,4)        COPY MEMBER INDEX IX-BB 00000011
IDMSRSTT FIELD=(1,1,16)     00000012
IDMSRSTT FIELD=(17,17,436)  00000013
IDMSRSTT END                00000014
END                          00000015
000004          SIGNOFF.
000004** I DC601073 SIGNOFF ACCEPTED
IDMSRSTC 15.0          COMPUTER ASSOCIATES INTERNATIONAL      DATE      TIME      PAGE
CAGJF0          RESTRUCTURE SCHEMA COMPARE ACTIVITY LIST      09/18/99   16411366   0002

** TRANSACTION SUMMARY **
OENTITY          ADD MODIFY REPLACE DELETE DISPLAY
.....
SCHEMA           0    0    0    0    1
NO ERRORS OR WARNINGS ISSUED FOR THIS COMPILE

```

## 5.6.7 For more information

- **On the IDMSRSTT macro statements**, see Appendix B, “IDMSRSTT Macro Statements.”
- **On restructuring a database**, see 4.26, “RESTRUCTURE SEGMENT” and 4.25, “RESTRUCTURE CONNECT.”

## Chapter 6. OS/390 JCL

---

6.1	About this chapter	6-3
6.2	CA-IDMS Batch Command Facility	6-4
6.3	Utility statements	6-6
6.3.1	ARCHIVE JOURNAL	6-6
6.3.2	ARCHIVE LOG	6-6
6.3.3	BACKUP	6-7
6.3.4	BUILD	6-7
6.3.5	CLEANUP	6-8
6.3.6	CONVERT PAGE	6-9
6.3.7	EXPAND PAGE	6-9
6.3.8	EXTRACT JOURNAL	6-10
6.3.9	FASTLOAD	6-10
6.3.10	FIX ARCHIVE	6-12
6.3.11	FIX PAGE	6-12
6.3.12	FORMAT	6-13
6.3.13	INSTALL STAMPS	6-13
6.3.14	LOAD	6-14
6.3.15	MAINTAIN INDEX	6-15
6.3.16	MERGE ARCHIVE	6-16
6.3.17	PRINT INDEX	6-17
6.3.18	PRINT JOURNAL	6-17
6.3.19	PRINT LOG	6-18
6.3.20	PRINT PAGE	6-18
6.3.21	PRINT SPACE	6-18
6.3.22	PUNCH	6-19
6.3.23	RELOAD	6-20
6.3.24	RESTORE	6-23
6.3.25	RESTRUCTURE CONNECT	6-23
6.3.26	RESTRUCTURE SEGMENT	6-24
6.3.27	ROLLBACK	6-24
6.3.28	ROLLFORWARD	6-25
6.3.29	TUNE INDEX	6-25
6.3.30	UNLOCK	6-25
6.3.31	UNLOAD	6-26
6.3.32	UPDATE STATISTICS	6-27
6.3.33	VALIDATE	6-27
6.4	Utility programs	6-29
6.4.1	IDMSDBAN	6-29
6.4.2	IDMSDIRL	6-30
6.4.3	IDMSLOOK	6-31
6.4.4	IDMSRPTS	6-31
6.4.5	IDMSRSTC	6-32
6.4.6	IDMSRSTT	6-34



## 6.1 About this chapter

This chapter presents sample OS/390 JCL used to run CA-IDMS utility statements and programs.

Statements common to utilities that use the CA-IDMS Batch Command Facility are presented first. Additional required statements to run each utility statement and program are presented next, alphabetically, by utility.

## 6.2 CA-IDMS Batch Command Facility

Listed below is sample OS/390 JCL to execute the CA-IDMS Batch Command Facility (IDMSBCF).

When using the IDMSBCF program to execute a utility statement, code these statements along with the required statements for each of the utilities.

The file assignments for each utility are presented on subsequent pages in this chapter.

For more information on the CA-IDMS Command Facility, see *CA-IDMS Command Facility*.

### Local mode IDMSBCF (OS/390)

---

```
//          EXEC PGM=IDMSBCF,REGION=2048K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//dcmsg   DD DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrn1 DD DSN=idms.tapejrn1,DISP=
//SYSLST DD SYSOUT=A
```

Insert file assignments required by the utility statements

```
//SYSIDMS DD *
```

Insert SYSIDMS parameters if applicable

```
//SYSIPT DD *
```

Insert utility statements

```
/*
```

---

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

**Note:** DD statements for database and native VSAM files may be omitted if dataset name information is specified in the file definitions.

---

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing CA-IDMS system software modules
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysjrn1	DDname of the tape journal file, if one is defined in the DMCL

---

---

idms.tapejrn1	Data set name of the tape journal file, if one is defined in the DMCL
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.  ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

### Central version IDMSBCF (OS/390)

---

```
//          EXEC PGM=IDMSBCF,REGION=2048K
//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
//sysjrn1 DD DUMMY
//SYSLST  DD SYSOUT=A
```

Insert file assignments required by the utility statements

```
//SYSIDMS DD *
```

Insert SYSIDMS parameters as appropriate

```
//SYSIPT  DD *
```

Insert utility statements

```
/*
```

---

**Note:** The DD statement for the system message area (SYSMSG.DDLDCMSG) can be omitted if its dataset name is specified in the DMCL.

---

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the CA-IDMS/DB load library
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysjrn1	DDname of the tape journal file, if one is defined in the DMCL
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.  ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

## 6.3 Utility statements

### 6.3.1 ARCHIVE JOURNAL

#### DD statements for the batch command facility (OS/390)

```
//j1jrn1 DD DSN=idms.j1jrn1,DISP=SHR
//j2jrn1 DD DSN=idms.j2jrn1,DISP=SHR
```

Additional journal file assignments, as required

```
//archjrn1 DD DSN=idms.archive,DISP=(NEW,KEEP),
//          UNIT=tape,VOL=SER=nnnnnn,
//          DCB=(RECFM=F,LRECL=rrrr,BLKSIZE=bbbb)
```

Additional archive journal file assignments, as required

j1jrn1	DDname of the first disk journal file, as defined in the DMCL
idms.j1jrn1	Data set name of the first disk journal file
j2jrn1	DDname of the second disk journal file, as defined in the DMCL
idms.j2jrn1	Data set name of the second disk journal file
archjrn1	DDname of the tape archive file, as defined in the DMCL
idms.archive	Data set name of the tape archive file
tape	Symbolic device name of the tape archive file
nnnnnn	Volume serial number of the tape archive file
rrrr	Length of the longest record in the tape archive file
bbbb	Block size of the tape archive file, as defined in the DMCL

### 6.3.2 ARCHIVE LOG

#### DD statements for the batch command facility (OS/390)

```
//dlogdb DD DSN=idms.dlogdb,DISP=SHR
//dmsgdb DD DSN=idms.dmsgdb,DISP=SHR
//sysjrn1 DD DUMMY
//SYS002 DD DSN=idms.archive,DISP=(NEW,CATLG)
//          DCB=(RECFM=VB,LRECL=280,BLKSIZE=bbbb)
```

dlogdb	DDname of the system log area
--------	-------------------------------

---

idms.dlogdb	Data set name of the system log area
dmsgdb	DDname of the system message area
idms.dmsgdb	Data set name of the system message area
sysjrnl	DDname of the tape journal file defined in the DMCL module
idms.archive	Data set name of the archive log file
bbbb	Block size of the archive log file; must be greater than or equal to 284 (typically, equal to 4 plus a multiple of 280)

---

### 6.3.3 BACKUP

#### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional file assignments, as required

```
//SYS001 DD DSN=user.bkpfile,DISP=(NEW,PASS),
// UNIT=tapeout,VOL=SER=nnnnnn,
// DCB=(RECFM=VB,BLKSIZE=bbbb)
```

---

userdb	DDname of the database file
user.userdb	Data set name of the database file
user.bkpfile	Data set name of the tape backup file
tapeout	Symbolic device name of the tape backup file
nnnnnn	Volume serial number of the tape backup file
bbbb	Block size of the tape backup file. Must be as large as the smaller of: <ul style="list-style-type: none"> <li>▪ The size of the largest page being backed up, plus 8</li> <li>▪ 32,760</li> </ul>

---

### 6.3.4 BUILD

#### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=disp
//SYS002 DD DSN=user.load,DISP=OLD
//SYS003 DD DSN=user.build,DISP=(NEW,PASS),UNIT=tape,
// DCB=(RECFM=VB,LRECL=rrr,BLKSIZE=bbb)
//SYSPCH DD DSN=&&sortbuild,DISP=(NEW,PASS),
// UNIT=disk,SPACE=(TRK,1),DCB=BLKSIZE=80
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

The SORTMSG and SORTWKnn files are only needed when performing a complete BUILD.

Add additional SORTWKnn files as necessary.

userdb	DDname of the database file
user.userdb	Data set name of the database file
user.load *	Data set name of the input (SYS002) file; for sizing information see discussion of SYS003 in 4.14, "LOAD"
user.build *	Data set name of the output (SYS003) file; for sizing information see 4.4, "BUILD"
tape	Symbolic device name of the SYS003 file
rrr	Record size of the SYS003 file
bbb	Block size of the SYS003 file
&&sortbuild	Data set name of the SYSPCH file
disk	Symbolic device name of the SYSPCH file

**Note:** \* When running a complete BUILD, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped BUILD, SYS002 and SYS003 must point to a *different* intermediate file.

### 6.3.5 CLEANUP

#### DD statements for the batch command facility (OS/390)

```
//sysjrn1 DD DSN=idms.tapejrn1,DISP=SHR
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional database file assignments, as required

sysjrn1	DDname of the tape journal file
idms.tapejrn1	Data set name of the tape journal file
userdb	DDname of the user database file

---

user.userdb	Data set name of the user database file
-------------	-----------------------------------------

---

### 6.3.6 CONVERT PAGE

#### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional database file DD statements as needed.

```
//newdb DD DSN=user.newdb,DISP=SHR
```

Additional new converted database file DD statements as needed.

---

userdb	DDname of the input database file
user.userdb	Data set name of the input database file
newdb	DDname of the output converted database file
user.newdb	Data set name of the output converted database file

---

### 6.3.7 EXPAND PAGE

#### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=OLD
```

Additional existing database file assignments, as required

```
//xfile DD DSN=user.xbase,DISP=(NEW,CATLG),UNIT=disk,  
// VOL=SER=nnnnnn,SPACE=(expanded-database-size)
```

Additional expanded database file assignments, as required

---

userdb	DDname of the existing database file (as specified by the FILE parameter)
user.userdb	Data set name of the existing database file
xfile	DDname of the expanded database file (as specified by the INTO parameter)
user.xbase	Data set name of the expanded database file
disk	Symbolic device name of the expanded database file
nnnnnn	Volume serial number of the expanded database file
expanded-database-size	Space allocation for the expanded database file

---

## 6.3.8 EXTRACT JOURNAL

### DD statements for the batch command facility (OS/390)

```
//SYS001 DD DSN=&&archive,DISP=(OLD)
//extract DD DSN=jrnl.extract,DISP=(NEW,CATLG),
//          DCB=(RECFM=VB,BLKSIZE=bbbb),
//          UNIT=nnnn,VOL=SER=nnnnn
//SORTMSG DD SYSOUT=A
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

&&archive	Data set name of the complete archive or journal file. It may be on tape or disk and concatenated.
extract	DDname of the extract journal file. If you don't specify, it defaults to SYS002.
jrnl.extract	Data set name for the extract journal file
bbbb	Block size of the extract journal file. Specify a size at least as large as the largest block size on the journal or archive files being processed.

## 6.3.9 FASTLOAD

### DD statements for the batch command facility (OS/390)

```
//SYS001 DD DSN=&&sortunld,DISP=SHR
//SYS002 DD DSN=user.db1001,DISP=(OLD,DELETE)
//SYS004 DD DSN=user.db1004,DISP=(NEW,PASS),UNIT=tape004,
//          DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004)
//SYS005 DD DSN=user.db1005,DISP=(NEW,PASS),UNIT=tape005,
//          DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS009 DD DSN=user.db1009,DISP=(NEW,PASS),UNIT=tape009,
//          DCB=(RECFM=VB,LRECL=rrr009,BLKSIZE=bbb009)
//SYS010 DD DSN=user.db1010,DISP=(NEW,PASS),UNIT=tape010,
//          DCB=(RECFM=VB,LRECL=rrr010,BLKSIZE=bbb010)
//SYS011 DD DSN=user.db1011,DISP=(NEW,PASS),UNIT=tape011,
//          DCB=(RECFM=VB,LRECL=rrr011,BLKSIZE=bbb011)
//SYSPCH DD DSN=&&sortld,DISP=(NEW,PASS),UNIT=disk,
//          SPACE=(TRK,1),DCB=BLKSIZE=80
//RELDCTL DD DSN=user.reldctl,DISP=SHR,UNIT=nnnn,
//          VOL=SER=nnnnn,
//          DCB=(RECFM=FB,LRECL=60,BLKSIZE=bbbct1)
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

```
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional existing database files assignments, as required

---

&&sortunld	Data set name of the sort parameters created by IDMSTBLU
user.dbl001	Data set name of the file put out by the format program
user.dbl004	Data set name of the intermediate work file containing the output from SORT1
tape004	Symbolic device name of the SYS004 file
rrr004	Record size of the SYS004 file; should be the same as the record size for SYS001
bbb004	Block size of the SYS004 intermediate work file; should be the same as the block size for SYS001
user.dbl005	Data set name of the intermediate work file containing a control record and set membership information from IDMSDBL2
tape005	Symbolic device name of the SYS005 file
rrr005	Record size of the SYS005 file; for sizing information, see 4.9, "FASTLOAD"
bbb005	Block size of the SYS005 intermediate work file; must be at least rrr005 plus four bytes
user.dbl009	Data set name of the intermediate work file containing the sorted contents of SYS005
tape009	Symbolic device name of the SYS009 file
rrr009	Record size of the SYS009 file; should be the same as rrr005
bbb009	Block size of the SYS009 intermediate work file; should be the same as bbb005
user.dbl010	Data set name of the intermediate work file containing pointer information from IDMSDBL3
tape010	Symbolic device name of the SYS010 file
rrr010	Record size of the SYS010 file; for sizing information, see 4.9, "FASTLOAD"
bbb010	Block size of the SYS010 intermediate work file; must be at least 60 bytes (the size of the control record plus four bytes)
user.dbl011	Data set name of the intermediate work file containing sorted pointer information from SORT4
tape011	Symbolic device name of the SYS011 file
rrr011	Record size of the SYS011 file; should be the same as rrr010

---

---

bbb011	Block size of the SYS011 intermediate work file; should be the same as bbb010
&&sortld	Data set name of the SYSPCH file
disk	Symbolic device name of the SYSPCH file
user.reldctl	Data set name of the reload control file containing control and set information
bbbctl	Blocksize of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760.
userdb	DDname of the database file
user.userdb	Data set name of the database file

---

### 6.3.10 FIX ARCHIVE

#### DD statements for the batch command facility (OS/390)

```
//SYS001 DD DSN=idms.tjrnlold,DISP=(OLD,PASS),UNIT=tapein
//SYS002 DD DSN=idms.tjrnlfix,DISP=(NEW,PASS),UNIT=tapeout,
//          DCB=BLKSIZE=bbbb
```

---

idms.tjrnlold	Data set name of the input tape journal file
tapein	Symbolic device name of the input tape journal file
idms.tjrnlfix	Data set name of the output tape journal file
tapeout	Symbolic device name of the output tape journal file
bbbb	Block size of the output tape journal file, as defined in the DMCL

---

### 6.3.11 FIX PAGE

#### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=OLD
```

Additional file assignments, as required

---

userdb	DDname of the user file
user.userdb	Data set name of the user file

---

## 6.3.12 FORMAT

### DD statements for the batch command facility (OS/390)

#### To format a new database file:

```
//userdb DD DSN=user.userdb,DISP=(NEW,disp),
//          UNIT=disk,VOL=SER=nnnnnn,
//          SPACE=(space)
```

#### To reformat an existing database file:

```
//userdb DD DSN=user.userdb,DISP=(OLD,PASS)
```

#### To format a new disk journal file:

```
//j1jrn1 DD DSN=idms.j1jrn1,DISP=(NEW,disp),
//          UNIT=disk,VOL=SER=nnnnnn,
//          SPACE=(space)
```

#### To reformat an existing disk journal file:

```
//j1jrn1 DD DSN=idms.j1jrn1,DISP=(OLD,PASS)
```

Additional database and journal file assignments, as required

<u>userdb</u>	DDname of the database file
<u>user.userdb</u>	Data set name of the database file
<u>disp</u>	Disposition of the new file (CATLG, KEEP, or PASS)
<u>disk</u>	Symbolic device name of the file being formatted
<u>nnnnnn</u>	Volume serial number of the file being formatted
<u>space</u>	Space allocation for the file being formatted
<u>j1jrn1</u>	DDname of the disk journal file
<u>idms.j1jrn1</u>	Data set name of the disk journal file

## 6.3.13 INSTALL STAMPS

### Local mode DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=OLD
//userdict DD DSN=user.userdict,DISP=SHR
//sysjrn1 DD DSN=idms.sysjrn1,DISP=OLD
```

<u>userdb</u>	DDname of the database file
<u>user.userdb</u>	Data set name of the database file
<u>userdict</u>	DDname of the database file containing the dictionary with the table definitions

---

sysjrn1	DDname of the tape journal file
idms.sysjrn1	Data set name of the tape journal file

---

**Central version:** To execute INSTALL STAMPS under the central version, modify the JCL shown above, as follows:

- Remove the USERDICT and SYSJRNL DD statements.
- Insert the following statement after the USERDB DD statement:

```
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
```

---

idms.sysctl	Data set name of the SYSCTL file
-------------	----------------------------------

---

## 6.3.14 LOAD

### DD statements for the batch command facility (OS/390)

```
//userdict DD DSN=user.userdict,DISP=disp
//userdb DD DSN=user.userdb,DISP=disp
//SYS001 DD DSN=user.input,DISP=OLD
//SYS002 DD DSN=user.loadin,DISP=(NEW,PASS),UNIT=tapein,
// DCB=(RECFM=VB,LRECL=rrrin,BLKSIZE=bbbin)
//SYS003 DD DSN=user.loadout,DISP=(NEW,PASS),UNIT=tapeout,
// DCB=(RECFM=VB,LRECL=rrrout,BLKSIZE=bbbout)
//SYSPCH DD DSN=&&sortload,DISP=(NEW,PASS),
// UNIT=disk,SPACE=(TRK,1),DCB=BLKSIZE=80
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

The SORTMSG and SORTWKnn files are only needed when performing a complete LOAD. Add additional SORTWKnn files as necessary

---

userdict	DDname of the database file containing the dictionary with the table definitions
user.userdict	Data set name of the database file containing the dictionary with the table definitions
userdb	DDname of the database file being loaded
user.userdb	Data set name of the database file being loaded
user.input	Data set name of the input file
user.loadin *	Data set name of the input SYS002 file; for sizing information see 4.14, "LOAD"
tapein	Symbolic device name of the SYS002 file
rrrin	Record size of the SYS002 file
bbbin	Block size of the SYS002 file

---

---

user.loadout *	Data set name of the output SYS003 file; for sizing information see 4.14, "LOAD"
tapeout	Symbolic device name of the SYS003 file
rrrout	Record size of the SYS003 file
bbbout	Block size of the SYS003 file
&&sortload	Data set name of the SYSPCH file
disk	Symbolic device name of the SYSPCH file

---

**Note:** \* When running a complete LOAD, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped LOAD, SYS002 and SYS003 must point to a *different* intermediate file.

**Note:** When running a complete LOAD, you must preallocate the file referenced by SYS002 and SYS003. Do not use a temporary data set for these files when running a complete LOAD.

## 6.3.15 MAINTAIN INDEX

### DD statements for the batch command facility (OS/390)

---

```
//userdb DD DSN=user.olddb,DISP=SHR

Additional existing database files assignments, as required

//SYS003 DD DSN=user.db1003,DISP=(NEW,PASS),UNIT=tape003,
// DCB=(RECFM=VB,LRECL=rrr003,BLKSIZE=bbb003)
//SYS004 DD DSN=user.db1004,DISP=(NEW,PASS),UNIT=tape004,
// DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004)
//SYS005 DD DSN=user.db1005,DISP=(NEW,PASS),UNIT=tape005,
// DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS006 DD DSN=user.db1006,DISP=(NEW,PASS),UNIT=tape006,
// DCB=(RECFM=VB,LRECL=rrr006,BLKSIZE=bbb006)
//SYSPCH DD DSN=&&sort,DISP=(NEW,PASS),UNIT=disk,
// SPACE=(TRK,1),DCB=BLKSIZE=80
//RELDCTL DD DSN=user.reldctl,DISP=(NEW,CATLG),UNIT=nnnn,
// VOL=SER=nnnnnn,
// DCB=(RECFM=FB,LRECL=60,BLKSIZE=bbbctl)
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))

Add additional SORTWKnn files as necessary
```

---



---

userdb	DDname of the existing database file
user.olddb	Data set name of the existing database file
user.db1003	Data set name of the intermediate work file containing index descriptors from IDMSTABX
tape003	Symbolic device name of the SYS003 file

---

---

rrr003	Record size of the SYS003 file; see Chapter 4, "MAINTAIN INDEX," for sizing information.
bbb003	Block size of the SYS003 intermediate work file
user.db1004	Data set name of the intermediate work file containing the output from SORT3
tape004	Symbolic device name of the SYS004 file
rrr004	Record size of the SYS004 file; should be the same as rrr003
bbb004	Block size of the SYS004 intermediate work file; should be the same as bbb003
user.db1005	Data set name of the intermediate work file containing pointers for user owned index sets from IDMSDBL3
tape005	Symbolic device name of the SYS005 file
rrr005	Record size of the SYS005 file; see Chapter 4, "MAINTAIN INDEX," for sizing information
bbb005	Block size of the SYS005 intermediate work file
user.db1006	Data set name of the intermediate work file containing sorted pointers for user owned index sets from SORT4
tape006	Symbolic device name of the SYS006 file
rrr006	Record size of the SYS006 file; should be the same as rrr005
bbb006	Block size of the SYS006 intermediate work file; should be the same as bbb005
&&sort	Data set name of the SYSPCH file containing sort parameters from IDMSTABX and IDMSDBL3
disk	Symbolic device name of the SYSPCH file
user.reldctl	Data set name of the reload control file containing control and set information
bbbctl	Blocksize of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760.

---

### 6.3.16 MERGE ARCHIVE

**DD statements for the batch command facility (OS/390)**

```
//SYS001 DD DSN=idms.tjrnlold,DISP=(OLD,PASS),UNIT=tapein
//SYS002 DD DSN=idms.tjrnlfix,DISP=(NEW,PASS),UNIT=tapein,
//          DCB=BLKSIZE=bbbb
//JRNMO1 DD DSN=idms.tmrngold,DISP=(OLD,PASS),UNIT=tapein
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

idms.tjrnlold	Data set name of the input tape journal file
tapein	Symbolic device name of the input tape journal file
idms.tjrnlfix	Data set name of the output tape journal
tapeout	Symbolic device name of the output tape journal file
bbbb	Block size of the output tape journal file, as defined in the DMCL
idms.tmrngold	Data set name of the input tape merged file. If none exists yet, specify //JRNMO1 DD DUMMY

**Note:** If the concatenated files have different block sizes, you must specify the following DCB parameter on the first file in the concatenation list:

```
DCB=(RECFM=VB,BLKSIZE=nnnn)
```

where nnnn is greater than or equal to (4 + the largest block size of any file in the concatenation list).

### 6.3.17 PRINT INDEX

#### DD statements for the batch command facility (OS/390)

```
//sysjrn1 DD DUMMY
//userdb  DD DSN=user.userdb,DISP=SHR
```

sysjrn1	DDname of the dummy journal file
userdb	DDname of the database file
user.userdb	Data set name of the database file

### 6.3.18 PRINT JOURNAL

#### DD statements for the batch command facility (OS/390)

```
//SYS001 DD DSN=idms.archjrn1,DISP=SHR
```

idms.archjrn1	Data set name of the archive journal file
---------------	-------------------------------------------

## 6.3.19 PRINT LOG

### DD statements for the batch command facility (OS/390)

To print from the DDLDCLOG area:

```
//dlogdb DD DSN=idms.dlogdb,DISP=SHR
```

To print from the archive log file:

```
//SYS001 DD DSN=idms.archive,DISP=OLD
//dmsgdb DD DSN=idms.dmsgdb, DISP=SHR
//sysjrn1 DD DSN=DUMMY
```

dlogdb	DDname of the data dictionary log area
idms.dlogdb	Data set name of the data dictionary log area
idms.archive	Data set name of the archive log file
dmsgdb	DDname of the data dictionary message area
idms.dmsgdb	Data set name of the data dictionary message area
sysjrn1	DDname of the tape journal file defined in the DMCL module

## 6.3.20 PRINT PAGE

### DD statements for the batch command facility (OS/390)

```
//userdb1 DD DSN=user.userdb1,DISP=SHR
//userdb2 DD DSN=user.userdb2,DISP=SHR
```

Additional database file assignments, as required

userdb1	DDname of the first database file
user.userdb1	Data set name of the first database file
userdb2	DDname of the second database file
user.userdb2	Data set name of the second database file

## 6.3.21 PRINT SPACE

### DD statements for the batch command facility (OS/390)

```
//userdb1 DD DSN=user.userdb1,DISP=SHR
//userdb2 DD DSN=user.userdb2,DISP=SHR
```

Additional database file assignments, as required

---

userdb1	DDname of the first database file
user.userdb1	Data set name of the first database file
userdb2	DDname of the second database file
user.userdb2	Data set name of the second database file

---

## 6.3.22 PUNCH

### Local mode DD statements for the batch command facility (OS/390)

```
//usercat DD DSN=user.ddlcat,DISP=SHR
//usercatx DD DSN=user.ddlcatx,DISP=SHR
//usercatl DD DSN=user.ddlcatl,DISP=SHR
//SYSPCH DD DSN=&&pch,DISP=(NEW,KEEP,DELETE),
//          DCB=(RECFM=FB,BLKSIZE=nnnn,LRECL=80)
//          SPACE=space-specification,UNIT=unit
//          VOL=SERnnnnnn
//sysjrn1 DD DSN=DUMMY
```

---

usercat	DDname of the database file containing the DDLCAT area of the dictionary
user.ddlcat	Dataset name of the database file containing the DDLCAT area of the dictionary
usercatx	DDname of the database file containing the DDLCATX area of the dictionary
user.ddlcatx	Dataset name of the database file containing the DDLCATX area of the dictionary
usercatl	DDname of the database file containing the DDLCATLOD area of the dictionary
user.ddlcatl	Dataset name of the database file containing the DDLCATLOD area of the dictionary

---

**Central version:** To execute PUNCH under the central version, modify the JCL shown above, as follows:

- Remove the USERCAT, USERCATX, USERCATL, and SYSJRNL DD statements.
- Insert the following statement:

```
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
```

---

idms.sysctl	Data set name of the SYSCTL file
-------------	----------------------------------

---

### 6.3.23 RELOAD

#### DD statements for the batch command facility (OS/390)

---

```
//SYS001 DD DSN=&&sortunld,DISP=SHR
//SYS002 DD DSN=user.db1002,DISP=(OLD,DELETE)
//SYS003 DD DSN=user.db1003,DISP=(OLD,DELETE)
//SYS004 DD DSN=user.db1004,DISP=(NEW,PASS),UNIT=tape004,
//          DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004,)
//SYS005 DD DSN=user.db1005,DISP=(NEW,PASS),UNIT=tape005,
//          DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS006 DD DSN=user.db1006,DISP=(NEW,PASS),UNIT=tape006,
//          DCB=(RECFM=VB,LRECL=rrr006,BLKSIZE=bbb006)
//SYS007 DD DSN=user.db1007,DISP=(NEW,PASS),UNIT=tape007,
//          DCB=(RECFM=VB,LRECL=rrr007,BLKSIZE=bbb007)
//SYS008 DD DSN=user.db1008,DISP=(NEW,PASS),UNIT=tape008,
//          DCB=(RECFM=VB,LRECL=rrr008,BLKSIZE=bbb008)
//SYS009 DD DSN=user.db1009,DISP=(NEW,PASS),UNIT=tape009,
//          DCB=(RECFM=VB,LRECL=rrr009,BLKSIZE=bbb009)
//SYS010 DD DSN=user.db1010,DISP=(NEW,PASS),UNIT=tape010,
//          DCB=(RECFM=VB,LRECL=rrr010,BLKSIZE=bbb010)
//SYS011 DD DSN=user.db1011,DISP=(NEW,PASS),UNIT=tape011,
//          DCB=(RECFM=VB,LRECL=rrr011,BLKSIZE=bbb011)
//SYSPCH DD DSN=&&sortrelld,DISP=(NEW,PASS),UNIT=disk,
//          SPACE=(TRK,1),DCB=BLKSIZE=80
//RELDCTL DD DSN=user.reldctl,DISP=SHR
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

```
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional existing database files assignments, as required

---

<i>&amp;&amp;sortunld</i>	Data set name of the sort parameters created by UNLOAD
<i>user.db1002</i>	Data set name of the file put out by UNLOAD as SYS002
<i>user.db1003</i>	Data set name of the file put out by UNLOAD as SYS003
<i>user.db1004</i>	Data set name of the intermediate work file containing the output from SORT1
<i>tape004</i>	Symbolic device name of the SYS004 file
<i>rrr004</i>	Record size of the SYS004 file; should be the same as the record size for SYS002 used by UNLOAD
<i>bbb004</i>	Block size of the SYS004 intermediate work file; should be the same as the block size for SYS002 used by UNLOAD

---

---

<i>user.db1005</i>	Data set name of the intermediate work file containing member descriptors for chained sets from IDMSDBL2
<i>tape005</i>	Symbolic device name of the SYS005 file
<i>rrr005</i>	Record size of the SYS005 file; for sizing information, see 4.23, “RELOAD”
<i>bbb005</i>	Block size of the SYS005 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
<i>user.db1006</i>	Data set name of the intermediate work file containing member descriptors for user owned index sets from IDMSDBL2
<i>tape006</i>	Symbolic device name of the SYS006 file
<i>rrr006</i>	Record size of the SYS006 file; for sizing information, see 4.23, “RELOAD”
<i>bbb006</i>	Block size of the SYS006 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
<i>user.db1007</i>	Data set name of the intermediate work file containing the output of SORT2
<i>tape007</i>	Symbolic device name of the SYS007 file
<i>rrr007</i>	Record size of the SYS007 file; should be the same as the larger of: <ul style="list-style-type: none"> <li>■ the record size for SYS003 used by UNLOAD</li> <li>■ the record size for SYS006</li> </ul>
<i>bbb007</i>	Block size of the SYS007 intermediate work file; must be at least as large as the larger of: <ul style="list-style-type: none"> <li>■ the block size for SYS003 used by UNLOAD</li> <li>■ the block size for SYS006</li> </ul>
<i>user.db1008</i>	Data set name of the intermediate work file containing the reformatted index information from IDMSDBLX
<i>tape008</i>	Symbolic device name of the SYS008 file
<i>rrr008</i>	Record size of the SYS008 file; should be the same as rrr007

---

---

<i>bbb008</i>	Block size of the SYS008 intermediate work file; should be the same as <i>bbb007</i>
<i>user.dbl009</i>	Data set name of the intermediate work file containing the sorted index set descriptors from SORT3
<i>tape009</i>	Symbolic device name of the SYS009 file
<i>rrr009</i>	Record size of the SYS009 file; should be the same as the larger of: <ul style="list-style-type: none"> <li>▪ the record size for SYS005</li> <li>▪ the record size for SYS008</li> </ul>
<i>bbb009</i>	Block size of the SYS009 intermediate work file; must be at least as large as the larger of: <ul style="list-style-type: none"> <li>▪ the block size for SYS003 used by UNLOAD</li> <li>▪ the block size for SYS006</li> </ul>
<i>user.dbl010</i>	Data set name of the intermediate work file containing prefix pointer information from IDMSDBL3
<i>tape010</i>	Symbolic device name of the SYS010 file
<i>rrr010</i>	Record size of the SYS010 file; for sizing information, see 4.23, "RELOAD"
<i>bbb010</i>	Block size of the SYS010 intermediate work file; must be at least 44 bytes (the size of a pointer descriptor plus four bytes)
<i>user.dbl011</i>	Data set name of the intermediate work file containing sorted prefix pointer information from SORT4
<i>tape011</i>	Symbolic device name of the SYS011 file
<i>rrr011</i>	Record size of the SYS011 file; should be the same as <i>rrr010</i>
<i>bbb011</i>	Block size of the SYS011 intermediate work file; should be the same as <i>bbb010</i>
<i>&amp;&amp;sortreld</i>	Data set name of the SYSPCH file
<i>disk</i>	Symbolic device name of the SYSPCH file
<i>user.reldctl</i>	Data set name of the reload control file containing control and set information
<i>bbbctl</i>	Blocksize of the RELDCTL file. It must be a multiple of 60 with a maximum size of 32,760.

---

---

<i>userdb</i>	DDname of the database file
<i>user.userdb</i>	Data set name of the database file

---

### 6.3.24 RESTORE

#### DD statements for the batch command facility (OS/390)

```
//userdb1 DD DSN=user.userdb1,DISP=OLD
//userdb2 DD DSN=user.userdb2,DISP=OLD
//SYS001 DD DSN=user.bkp,DISP=SHR
```

Additional database file assignments, as required

---

<i>userdb1</i>	DDname of the first database file
<i>user.userdb1</i>	Data set name of the first database file
<i>userdb2</i>	DDname of the second database file
<i>user.userdb2</i>	Data set name of the second database file
<i>user.bkp</i>	Data set name of the backup file

---

### 6.3.25 RESTRUCTURE CONNECT

#### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=SHR
//SYS001 DD DSN=idms.spill,DISP=OLD
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

---

<i>userdb</i>	DDname of the database file
<i>user.userdb</i>	Data set name of the database file
<i>idms.spill</i>	Data set name of the spill file.
<i>bbbb</i>	Size of the spill file should be a multiple of 32 with a maximum size of 32,736.

---

## 6.3.26 RESTRUCTURE SEGMENT

### DD statements for the batch command facility (OS/390)

```
//userdb1 DD DSN=user.userdb1,DISP=SHR
//userdb2 DD DSN=user.userdb2,DISP=SHR
//SYS001 DD DSN=idms.spill,DISP=(NEW,PASS)
//          DCB=(RECFM=FB,BLKSIZE=bbbb)
```

Additional database file assignments, as required

userdb1	DDname of the first database file
user.userdb1	Data set name of the first database file
userdb2	DDname of the second database file
user.userdb2	Data set name of the second database file
idms.spill	Data set name of the spill file
bbbb	Block size of the spill file; it should be a multiple of 32 with a maximum size of 32,736.

## 6.3.27 ROLLBACK

### DD statements for the batch command facility (OS/390)

```
//SYS001 DD DSN=&&archive,DISP=OLD,UNIT=tape
//userdb DD DSN=user.userdb,DISP=OLD
```

Additional database file assignments, as required

If using the SORT option add these statements:

```
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

&&archive	Data set name of the complete archive or tape journal file
tape	Symbolic device name of the archive or tape journal file
userdb	DDname of the user database file
user.userdb	Data set name of the user database file

## 6.3.28 ROLLFORWARD

### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=OLD
```

Add database file assignments, as required

If recovering from a standard journal file:

```
//SYS001 DD DSN=&&archive,DISP=(OLD),UNIT=tape
```

If recovering from a journal extract file:

```
//extract DD DSN=jrn1.extract,DISP=OLD
```

If using the SORT option or processing a journal extract file, add these statements:

```
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb  
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add SORTWKnn files as necessary

<u>userdb</u>	DDname of the user database file.
<u>user.userdb</u>	Data set name of the user database file.
<u>&amp;&amp;archive</u>	Data set name of the complete archive or tape journal file.
<u>tape</u>	Symbolic device name of the archive or tape journal file.
<u>extract</u>	DDname of the extract journal file. If not specified in the command it will default to SYS002.
<u>jrn1.extract</u>	Data set name of the extract journal file.

## 6.3.29 TUNE INDEX

### DD statement for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=SHR
```

<u>userdb</u>	DDname of the database file
<u>user.userdb</u>	Data set name of the database file

## 6.3.30 UNLOCK

### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=SHR
```

---

userdb	DDname of the database file
user.userdb	Data set name of the database file

---

### 6.3.31 UNLOAD

#### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.olddb,DISP=SHR
```

Additional existing database files assignments, as required

```
//sysjrn1 DD DUMMY
//SYS002 DD DSN=user.db1002,DISP=(NEW,PASS),UNIT=tape002,
//          DCB=(RECFM=VB,BLKSIZE=bbb002)
//SYS003 DD DSN=user.db1003,DISP=(NEW,PASS),UNIT=tape003,
//          DCB=(RECFM=VB,BLKSIZE=bbb003)
//SYSPCH DD DSN=&&sortunld,DISP=(NEW,PASS),UNIT=disk,
//          SPACE=(TRK,1),DCB=BLKSIZE=80
//RELDCTL DD DSN=user.reldctl,DISP=(NEW,CATLG),UNIT=nnnn,
//          VOL=SER=nnnnnn,
//          DCB=(RECFM=FB,LRECL=60,BLKSIZE=bbbctl)
```

---

userdb	DDname of the existing database file
user.olddb	Data set name of the existing database file
sysjrn1	DDname of the dummy journal file
user.db1002	Data set name of the SYS002 output file
tape002	Symbolic device name of the SYS002 output file
bbb002	Block size of the SYS002 output file; must be at least the size of the largest database record plus 4 bytes
user.db1003	Data set name of the SYS003 output file
tape003	Symbolic device name of the SYS003 output file
bbb003	Block size of the SYS003 output file
&&sortunld	Data set name of the sort parameters created by UNLOAD
disk	Symbolic device name of the sort parameter file
user.reldctl	Data set name of the reload control file containing control and set information
bbbctl	Size of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760.

---

## 6.3.32 UPDATE STATISTICS

### DD statements for the batch command facility (OS/390)

```
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional database file assignments, as required

```
//ddlcat DD DSN=sysdict.ddlcat,DISP=OLD
//ddlxcat DD DSN=sysdict.ddlxcat,DISP=OLD
//jljrn1 DD DSN=tape.jljrn1,DISP=(NEW,disp)
```

Additional journal file assignments, as required

userdb	DDname of the database file
user.userdb	Data set name of the database file
ddlcat	DDname of the database file containing the area of the system dictionary with the table definitions
sysdict.ddlcat	Data set name of the database file containing the area of the system dictionary with the table definitions
ddlxcat	DDname of the database file containing the area of the system dictionary with indexes
sysdict.ddlxcat	Data set name of the database file containing the area of the system dictionary with indexes
jljrn1	DDname of the first journal file, as defined in the DMCL
tape.jljrn1	Data set name of the first journal file
disp	The disposition of the first journal file

## 6.3.33 VALIDATE

### DD statements for the batch command facility (OS/390)

```

//userdb DD DSN=user.userdb,DISP=disp
//SYS002 DD DSN=user.valin,DISP=(NEW,PASS),UNIT=tapein,
          DCB=(RECFM=VB,LRECL=rrrin,BLKSIZE=bbbin)
//SYS003 DD DSN=user.valout,DISP=(NEW,PASS),
          UNIT=tapeout,
          DCB=(RECFM=VB,LRECL=rrrout,
          BLKSIZE=bbbout)
//SYSPCH DD DSN=&&sortval,DISP=(NEW,PASS)
          UNIT=disk,SPACE=(TRK,1),DCB=BLKSIZE=80
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))

```

The SORTMSG and SORTWKnn files are only needed when performing a complete VALIDATE. Add additional SORTWKnn files as necessary

userdb	DDname of the database file
user.userdb	Data set name of the database file
user.valin *	Data set name of the input SYS002 file; for sizing information see 4.33, "VALIDATE"
tapein	Symbolic device name of the SYS002 file
rrrin	Record size of the SYS002 file
bbbin	Block size of the SYS002 file
user.valout *	Data set name of the output SYS003 file; for sizing information see 4.33, "VALIDATE"
tapeout	Symbolic device name of the SYS003 file
rrrout	Record size of the SYS003 file
bbbout	Block size of the SYS003 file
&&sortval	Data set name of the SYSPCH file

**Note:** \* When running a complete VALIDATE, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to a *different* intermediate file.

## 6.4 Utility programs

### 6.4.1 IDMSDBAN

#### Local mode: IDMSDBAN (OS/390)

---

```
//DBN1      EXEC PGM=IDMSDBN1,REGION=region-size
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//userdb   DD DSN=user.userdb,DISP=SHR
```

Additional database file assignments, as required

```
//SYS002 DD DSN=&&chain,DISP=(NEW,PASS),UNIT=tape,
//          DCB=(RECFM=VB,BLKSIZE=9000)
//SYSLST DD SYSOUT=A
//SYSIPT DD *
```

Insert SYSIDMS parameters, as required

/\*

IDMSDBAN input parameters

```
/*
//DBN2      EXEC PGM=IDMSDBN2,REGION=1500K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//SYS001 DD DSN=&&chain,DISP=(OLD),UNIT=tape
//SYS002 DD UNIT=SYSDA,SPACE=(TRK,(400,50)),
//          DCB=(RECFM=VB,BLKSIZE=9000)
//SORTWK01 DD UNIT=SYSDA,SPACE=(TRK,(nnn,nn))
//SORTWK02 DD UNIT=SYSDA,SPACE=(TRK,(nnn,nn))
//SORTMSG DD SYSOUT=A
//SYSLST DD SYSOUT=A
```

Insert SYSIDMS parameters, as required

/\*

---

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

region-size	The size of the region. This depends on the size of the database. A larger region size enables efficient execution of the internal sort.
idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the CA-IDMS/DB load library containing the subschema and DMCL load modules
userdb	DDname of the user file
user.userdb	Data set name of the user file

---

---

&&chain	Data set name of the chain file generated by IDMSDBN1
tape	Symbolic device name of the chain file

---

## 6.4.2 IDMSDIRL

### Local mode IDMSDIRL (OS/390)

```
//DIRL      EXEC PGM=IDMSDIRL,REGION=512K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//sysjrn1 DD DSN=idms.tapejrn1,DISP=(NEW,KEEP),
//          UNIT=tape
//dictdb   DD DSN=idms.dictdb,DISP=SHR
//SYSLST DD SYSOUT=A
//SYS001 DD DSN=idms.dirldata,DISP=(OLD,PASS),UNIT=tapein
//SYSIPT DD *
```

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the CA-IDMS/DB load library
sysjrn1	DDname of the tape journal file
idms.tapejrn1	Data set name of the tape journal file
tape	Symbolic device name of the tape journal file
dictdb	DDname of the data dictionary file
idms.dictdb	Data set name of the data dictionary file
idms.dirldata	Data set name of the IDMSDIRL input file (on the installation tape)
tapein	Symbolic device name of the IDMSDIRL input file

---

**Central version:** To execute IDMSDIRL under the central version, modify the JCL shown above, as follows:

- Remove the SYSJRNL and DICTDB DD statements.
- Insert the following statement after the STEPLIB DD statement:

```
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
```

---

idms.sysctl	Data set name of the SYSCTL file
-------------	----------------------------------

---

**Note:** An IDMSOPTI module link edited with IDMSDIRL can be used in place of or in addition to the SYSCTL file.

### 6.4.3 IDMSLOOK

#### Local mode IDMSLOOK (OS/390)

```
//EXEC      PGM=IDMSLOOK,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//dcmmsg    DD DSN=idms.sysmsg.ddldcmmsg,DISP=SHR
//sysjrn1   DD DSN=idms.tapejrn1,DISP=
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
```

Insert SYSIDMS parameters, as required

```
/*
//SYSIPT DD *
```

IDMSLOOK input parameters

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
------------------	--------------------------------------------------------------------------------------------

---

idms.loadlib	Data set name of the CA-IDMS/DB load library
--------------	----------------------------------------------

---

### 6.4.4 IDMSRPTS

#### Local mode IDMSRPTS (OS/390)

```
//RPTS      EXEC PGM=IDMSRPTS,REGION=256K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//sysjrn1   DD DSN=idms.tapejrn1,DISP=SHR
//dictdb    DD DSN=idms.dictdb,DISP=SHR
//SYSOUT DD SYSOUT=A
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
```

DICTNAME=dictionary-name  
 DICTNODE=node-name This is optional

Insert other SYSIDMS parameters as appropriate

```
//SYSIPT DD *
```

Insert utility statements

```
/*
```

---

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the CA-IDMS/DB load library
sysjrn1	DDname of the tape journal file
idms.tapejrn1	Data set name of the tape journal file
dictdb	DDname of the data dictionary file
idms.dictdb	Data set name of the data dictionary file
dictionary-name	Name of the dictionary against which the reports are to be produced
nodename	Name of the DC/UCF system where <i>dictionary-name</i> resides

**Central version:** To execute IDMSRPTS under the central version, modify the JCL shown above, as follows:

- Remove the SYSJRNL and DICTDB DD statements.
- Insert the following statement after the STEPLIB DD statement:

```
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

**Note:** An IDMSOPTI module link edited with IDMSRPTS can be used in place of or in addition to the SYSCTL file.

## 6.4.5 IDMSRSTC

### Local mode IDMSRSTC (OS/390)

---

```
//RSTC      EXEC PGM=IDMSRSTC,REGION=512K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//sysjrn1 DD DSN=idms.tapejrn1,DISP=SHR
//dictdb   DD DSN=idms.dictdb,DISP=SHR
//SYSLST DD SYSOUT=A
//SYSPCH DD SYSOUT=B
//SYSIDMS DD *
```

DICTNAME=dictionary-name  
 DICTNODE=node-name

Insert other SYSIDMS parameters as appropriate

```
//SYSIPT DD *
```

IDMSRSTC input parameters

/\*

---

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the CA-IDMS/DB load library
sysjrn1	DDname of the tape journal file
idms.tapejrn1	Data set name of the tape journal file
dictdb	DDname of the data dictionary file
idms.dictdb	Data set name of the data dictionary file
dictionary-name	Name of the dictionary containing the schemas identified in the SCHEMA statement
nodename	Name of the DC/UCF system where <i>dictionary-name</i> resides

---

**Central version:** To execute IDMSRSTC under the central version, modify the JCL shown above, as follows:

- Remove the SYSJRNL and DICTDB DD statements.
- Insert the following statement after the STEPLIB DD statement:

```
//sysctl DD DSN=idms.sysctl,DISP=SHR
```

---

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

---

**Note:** An IDMSOPTI module link edited with IDMSRSTC can be used in place of or in addition to the SYSCTL file.

## 6.4.6 IDMSRSTT

### Assemble and link an IDMSRSTT module

```
// ASMFCL      EXEC PGM=ASMFCL
// ASM.SYSIN   DD  *

      Put IDMSRSTT macro statements here

// LKED.SYSLMOD DD  DSN=idms.loadlib,DISP=SHR
// LKED.SYSIN   DD  *
      NAME idmsrstt(R)
```

---

idms.loadlib	Data set name of the CA-IDMS/DB load library
--------------	----------------------------------------------

---

idmsrstt	module name of the IDMSRSTT table
----------	-----------------------------------

---

# Chapter 7. VSE/ESA JCL

---

7.1	About this chapter	7-3
7.2	=COPY Facility	7-4
7.3	IDMSLBLS Procedure	7-5
7.4	SYSIDMS parameter file	7-6
7.5	CA-IDMS Batch Command Facility	7-7
7.6	Utility statements	7-9
7.6.1	ARCHIVE JOURNAL	7-9
7.6.2	ARCHIVE LOG	7-9
7.6.3	BACKUP	7-9
7.6.4	BUILD	7-10
7.6.5	CLEANUP	7-10
7.6.6	CONVERT PAGE	7-11
7.6.7	EXPAND PAGE	7-11
7.6.8	EXTRACT JOURNAL	7-11
7.6.9	FASTLOAD	7-12
7.6.10	FIX ARCHIVE	7-14
7.6.11	FIX PAGE	7-14
7.6.12	FORMAT	7-14
7.6.13	INSTALL STAMPS	7-16
7.6.14	LOAD	7-16
7.6.15	MAINTAIN INDEX	7-17
7.6.16	MERGE ARCHIVE	7-19
7.6.17	PRINT INDEX	7-19
7.6.18	PRINT JOURNAL	7-19
7.6.19	PRINT LOG	7-20
7.6.20	PRINT PAGE	7-20
7.6.21	PRINT SPACE	7-20
7.6.22	PUNCH	7-21
7.6.23	RELOAD	7-21
7.6.24	RESTORE	7-23
7.6.25	RESTRUCTURE CONNECT	7-24
7.6.26	RESTRUCTURE SEGMENT	7-24
7.6.27	ROLLBACK	7-25
7.6.28	ROLLFORWARD	7-25
7.6.29	TUNE INDEX	7-26
7.6.30	UNLOCK	7-26
7.6.31	UNLOAD	7-27
7.6.32	UPDATE STATISTICS	7-27
7.6.33	VALIDATE	7-28
7.7	Utility programs	7-29
7.7.1	IDMSDBAN	7-29
7.7.2	IDMSDIRL	7-30
7.7.3	IDMSLOOK	7-32
7.7.4	IDMSRPTS	7-33
7.7.5	IDMSRSTC	7-34
7.7.6	IDMSRSTT	7-36
7.8	IDMSLBLS Procedure	7-37



## 7.1 About this chapter

This chapter includes sample VSE/ESA JCL to execute CA-IDMS utilities. Additionally, a description of the =COPY facility, IDMSLBS procedure, and the SYSIDMS parameter file, as they relate to the utilities, is included.

## 7.2 =COPY Facility

**Purpose:** Under VSE/ESA, some or all of the utility statements to be submitted to the CA-IDMS Batch Command Facility (IDMSBCF) can be stored as a member in a source statement library. To copy the library member into the job stream, you use the =COPY IDMS statement.

The =COPY IDMS statement identifies the library member and is coded in the JCL along with other input parameter statements to be submitted to IDMSBCF. Multiple =COPY statements can be submitted.

=COPY IDMS statements and input parameter statements can be intermixed in the JCL. The input parameters are submitted to the compiler in the order in which they occur, whether they are coded directly in the JCL or copied in through the =COPY facility.

### Syntax

```

▶▶ — =COPY IDMS [ A, ← _____ ] member-name
                  [ sublibrary-id. ]

```

### Parameters

#### A/sublibrary-id

Identifies the source statement sublibrary that includes the member identified by *member-name*. The default is A.

#### member-name

Identifies the source statement library member that contains the input parameter statements to be submitted to IDMSBCF.

**Note:** If the input parameter statements are stored as a member in a private source statement library, the DLBL file type for the library must be specified as DA.

## 7.3 IDMSLBLE Procedure

IDMSLBLE is a procedure provided during a CA-IDMS VSE/ESA installation. It contains file assignments for CA-IDMS dictionaries, sample databases, disk journal files, and the SYSIDMS parameter file.

A copy of the IDMSLBLE procedure appears at the end of this chapter.

File assignments for the CA-IDMS files in the IDMSLBLE procedure are not included in the sample JCL for each utility in this chapter.

Tape and archive journal files, SYSCTL, user database, and work file assignments are included in the sample JCL as well as an EXEC statement for the IDMSLBLE procedure. The sample JCL assumes you will use either the IDMSLBLE procedure or another procedure that you create.

The files required to run each utility are identified in individual chapters under, "JCL considerations".

## 7.4 SYSIDMS parameter file

SYSIDMS is a parameter file used in the execution of CA-IDMS batch jobs running in either local mode or under the central version.

SYSIDMS parameters allow you to specify physical requirements of the environment such as, DBNAME and DICTNAME and runtime directives such as activating IDMSQSAM. Additionally, there are SYSIDMS parameters for use specifically in a VSE/ESA environment. For example, there is a BLKSIZE parameter that allows you to override the blocksize for a file and a DEVADDR=SYS*mmn* parameter to specify a device address for a tape file.

The SYSIDMS file can be defined as a sequential disk file or SYSIDMS parameters can be passed through the SYSIPT file. The SYSIDMS file is defined in the IDMSLBS procedure so that parameters can be passed through SYSIPT.

SYSIDMS is referenced only in the generic JCL for the CA-IDMS Batch Command Facility. You should include SYSIDMS parameters, as appropriate, in the JCL stream for each utility.

## 7.5 CA-IDMS Batch Command Facility

Listed below is sample VSE/ESA JCL to execute the CA-IDMS Batch Command Facility (IDMSBCF).

When using the IDMSBCF program to execute a utility statement, include these file assignments along with the required statements for each of the utilities. The file assignments for each utility are presented on subsequent pages in this chapter.

### Local mode IDMSBCF (VSE/ESA)

---

```
// EXEC PROC=IDMSLBL5
// DLBL idms150,'idms150.library'
// EXTENT SYSnnn,nnnnnn,,ssss,tttt
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL SYSnnn,'idms.sysjrn1',,nnnnnn,,f
// ASSGN SYSnnn,x'ccc'
// LIBDEF *,SEARCH=CA-IDMS 15.0 libraries
```

Insert file assignments required by the utility statements

```
// EXEC IDMSBCF,SIZE=64K
```

Insert SYSIDMS parameters, as required

```
/*
```

Insert utility statements

```
/*
```

---

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

IDMSLBL5	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, and SYSIDMS parameter file.
----------	-------------------------------------------------------------------------------------------------------------------------------------------------

►►For a complete listing of IDMSLBL5, see "IDMSLBL5 Procedure" later in this section.

---

idms150	Dtfname of the CA-IDMS library
---------	--------------------------------

---

idms150.library	Data set name of CA-IDMS 15.0 libraries, as established during installation
-----------------	-----------------------------------------------------------------------------

---

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
------	----------------------------------------------------

---

tttt	Number of tracks
------	------------------

---

---

vvvvvv	Volume serial number
CA-IDMS 15.0 libraries	The CA-IDMS 15.0 libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file ▶▶For a complete listing of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

**Central version:** To run IDMSBCF under the central version, add a SYSCTL file, and remove the tape journal file assignment.

---

```
// EXEC PROC=IDMSLBLE
// DLBL   idms150,'idms150.library'
// EXTENT SYSnnn,nnnnnn,,ssss,ttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   sysctl,'idms.sysctl',1999/365,SD
// EXTENT SYSnnn,nnnnnn,,ssss,2
// ASSGN  SYSnnn,DISK,VOL=nnnnnn,SHR
// LIBDEF *,SEARCH=CA-IDMS 15.0 libraries
```

Insert file assignments required by the utility statements

```
// EXEC IDMSBCF,SIZE=64K
```

Insert SYSIDMS parameters, as required

```
/*
```

Insert utility statements

```
/*
```

---

sysctl	Filename of the SYSCTL file
idms.sysctl	File-ID of the SYSCTL file

---

## 7.6 Utility statements

### 7.6.1 ARCHIVE JOURNAL

#### File assignments for the batch command facility (VSE/ESA)

```
// TLBL   SYSnnn,'idms.archive',,nnnnn,,f
// ASSGN  SYSnnn,x'ccc'
```

Additional archive journal file assignments, as required

---

idms.archive	File-ID of the tape archive file as defined in the DMCL
--------------	---------------------------------------------------------

---

### 7.6.2 ARCHIVE LOG

#### File assignments for the batch command facility (VSE/ESA)

```
// DLBL   dclog,'idms.system.ddldclog',,DA
// EXTENT SYSnnn,nnnnn,,ssss,
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL   SYS002,'idms.archive',,nnnnn,,f
// ASSGN  SYS002,x'ccc'
```

---

dclog	File-ID of the database log file
-------	----------------------------------

---

idms.archive	File-ID of the archive log file
--------------	---------------------------------

---

### 7.6.3 BACKUP

#### File assignments for the batch command facility (VSE/ESA)

```
// DLBL   userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional file assignments, as required

```
// TLBL   SYS001,'user.bkpfile',,nnnnn,,f
// ASSGN  SYS001,x'ccc'
```

---

userdb	Filename of the user file
--------	---------------------------

---

user.userdb	File-ID of the user file
-------------	--------------------------

---

user.bkpfile	File-ID of the tape backup file
--------------	---------------------------------

---

## 7.6.4 BUILD

### File assignments for the batch command facility (VSE/ESA)

---

```
// DLBL   userdb, 'user.userdb', ,DA
// EXTENT SYSnnn, nnnnnn
// ASSGN  SYSnnn, DISK, VOL=vvvvvv, SHR
// DLBL   userload, 'user.load', ,SD
// EXTENT SYS002, nnnnnn
// ASSGN  SYS002, DISK, VOL=vvvvvv, SHR
// DLBL   userbuild, 'user.build', ,SD
// EXTENT SYS003, nnnnnn
// ASSGN  SYS003, DISK, VOL=vvvvvv, SHR
```

Sort files are only needed when performing a complete BUILD

```
// DLBL   SORTWK1, 'sort.work.file'
// EXTENT SORTWK1, SYSWK0, ,1, 30
// ASSGN  SORTWK1, SYS003, OUTPUT
```

Additional sort files, as required

---

userdb	Filename of the user file
user.userdb	File-ID of the user file
user.load	File-ID of the input (SYS002) file; for sizing information see discussion of SYS003 in 4.14, "LOAD."
user.build	File-ID of the output (SYS003) file; for sizing information see 4.4, "BUILD."

---

**Note:** When running a complete BUILD, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped BUILD, SYS002 and SYS003 must point to *different* intermediate files.

## 7.6.5 CLEANUP

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL   userdb, 'user.userdb', ,DA
// EXTENT SYSnnn, nnnnnn
// ASSGN  SYSnnn, DISK, VOL=vvvvvv, SHR
```

Additional database file assignments, as required

---

userdb	Filename of the user file
user.userdb	File-ID of the user file

---

## 7.6.6 CONVERT PAGE

### File assignments for the batch command facility (VSE/ESA)

```
//DLBL      userdb,'user.userdb',,SD
//EXTENT    SYSnnn,nnnnnn
//ASSIGN    SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional database file assignments as needed.

```
//DLBL      newdb,'user.newdb',,SD
//EXTENT    SYSnnn,nnnnnn
//ASSIGN    SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional new converted database file assignments as needed.

userdb	Filename of the input database file
user.userdb	File-ID of the input database file
newdb	Filename of the output converted database file
user.newdb	File-ID of the output converted database file

## 7.6.7 EXPAND PAGE

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL      userdb,'user.userdb',,DA
// EXTENT    SYSnnn,nnnnnn
// ASSGN     SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional database file assignments, as required

```
// DLBL      userxb,'user.xbase',,DA
// EXTENT    SYSnnn,nnnnnn
// ASSGN     SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional expanded database file assignments, as required

user.userdb	File-ID of the existing database file
user.xbase	File-ID of the expanded database file

## 7.6.8 EXTRACT JOURNAL

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL  SYS001,'archive',,SD
// EXTENT SYS001,nnnnn
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
// DLBL  extract,'jrn1.extract',,SD
// EXTENT sysnnn,nnnnn
// ASSGN  sysnnn,DISK,VOL=vvvvvv,SHR
```

Add sort work files

Add SYSIDMS parameters

FILENAME=SYSnnn,BLKSIZE=bbbb

---

archive	File name of the complete archive or journal file. It can be on tape or disk as specified through the SYSIDMS parameter file.
extract	DDname of the extract journal file. If not specified, it defaults to SYS002. It can be on tape or disk as specified through the SYSIDMS parameter file.
jrn1.extract	Data set name of the extract journal file.
bbbb	Block size of the extract journal file. Specify a size as large as the largest block size on the journal or archive files being processed.

---

## 7.6.9 FASTLOAD

**File assignments for the batch command facility (VSE/ESA)**

---

```
// DLBL SORTWK1,'sort.work.file'
// EXTENT SORTWK1,SYSWK0,,1,30
// ASSGN SORTWK1,SYS003,OUTPUT
```

Additional sort files, as required

```
// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional database file assignments, as required

```
// DLBL RELDCTL,'user.reldctl',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL SYS001,'sort.unload',,SD
// EXTENT SYS001,nnnnnn,,ssss,1111
// ASSGN SYS001,DISK,VOL=vvvvvv,SHR
// DLBL SYS002,'user.db1001',,SD
// EXTENT SYS002,nnnnnn,,ssss,1111
// ASSGN SYS002,DISK,VOL=vvvvvv,SHR
// DLBL SYS004,'db1004.name',,SD
// EXTENT SYS004,nnnnnn,,ssss,1111
// ASSGN SYS004,DISK,VOL=vvvvvv,SHR
// DLBL SYS005,'user.db1005',,SD
// EXTENT SYS005,nnnnnn,,ssss,1111
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// TLBL SYS009,'user.db1009',,nnnnnn,,f
// ASSGN SYS009,x'ccc'
// DLBL SYS010,'user.db1010',,SD
// EXTENT SYS010,nnnnnn,,ssss,1111
// ASSGN SYS010,DISK,VOL=vvvvvv,SHR
// DLBL SYS011,'user.db1011',,SD
// EXTENT SYS011,nnnnnn,,ssss,1111
// ASSGN SYS011,DISK,VOL=vvvvvv,SHR
```

---

sort.unload	File-ID of the sort parameters created by IDMSTBLU
user.reldctl	Data set name of the reload control file containing control and set information. Blocksize of the RELDCTL file should be a multiple of 60 with a maximum size of 32,760. Blocksize is controlled through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see <i>CA-IDMS Database Administration</i> .
user.db1001	File-ID of the file put out by the format program
user.db1004	File-ID of the intermediate work file containing the output from SORT1. Record and block size should be the same as SYS001.
user.db1005	File-ID of the intermediate work file containing a control record and set membership information from IDMSDBL2

---

---

user.db1009	File-ID of the intermediate work file containing the sorted contents of SYS005
user.db1010	File-ID of the intermediate work file containing pointer information from IDMSDBL3
user.db1011	File-ID of the intermediate work file containing sorted pointer information from SORT4
userdb	Filename of the database file
user.userdb	File-ID of the database file

---

### 7.6.10 FIX ARCHIVE

#### File assignments for the batch command facility (VSE/ESA)

```
// TLBL   SYS001,'idms.tjrnlold',,nnnnnn,,f
// ASSGN  SYS001,x'cuu'
// TLBL   SYS002,'idms.tjrnlfix',,nnnnnn,,f
// ASSGN  SYS002,x'cuu'
```

---

idms.tjrnlold	File-ID of the input tape journal file
<i>f</i>	File number of the tape journal file
idms.tjrnlfix	File-ID of the output tape journal file

---

### 7.6.11 FIX PAGE

#### File assignments for the batch command facility (VSE/ESA)

```
// DLBL   userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional file assignments, as required

---

userdb	Filename of the user file
user.userdb	File-ID of the user file

---

### 7.6.12 FORMAT

#### File assignments for the batch command facility (VSE/ESA)

**To format a new database file:**

```
// DLBL   userdb,'user.userdb',,tt
// EXTENT SYSnnn,nnnnn,,ssss,rrrr
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

**To reformat an existing database file:**

```
// DLBL   userdb,'user.userdb',,tt
// EXTENT SYSnnn,nnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

**To format a new disk journal file:**

```
// DLBL   jljrn1,'idms.jljrn1',,SD
// EXTENT SYSnnn,nnnnn,,ssss,rrrr
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

**To reformat an existing disk journal file:**

```
// DLBL   jljrn1,'idms.jljrn1',,SD
// EXTENT SYSnnn,nnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional database and journal file assignments, as required

userdb	Filename of the database file
user.userdb	File-ID of the database file
idms.jljrn1	File-ID of the disk journal file
jljrn1	Filename of the disk journal file
rrrr	Number of tracks (CKD) or blocks (FBA) in the disk extent; must be at least one more than the number of BDAM blocks or VSAM control intervals in the database
tt	File type of the file being formatted: <ul style="list-style-type: none"> <li>■ SD (sequential) if formatting by file or segment (unless it is a VSAM file)</li> <li>■ DA (direct access) if reformatting by area</li> <li>■ VSAM if formatting or reformatting a CA-IDMS/DB VSAM file</li> <li>■ Omit if reformatting by file</li> </ul>

**Multiple extents:** You can format a CA-IDMS/DB VSAM file with multiple extents under VSE/ESA. However, you cannot format a BDAM file with multiple extents.

**The SIZE parameter on the EXEC statement:** The SIZE parameter on the EXEC statement enables the IBM rotational sensing (RPS) feature. When formatting CA-IDMS/DB VSAM files, you should specify SIZE=AUTO on the EXEC statement for IDMSBCF.

Do not use the SIZE parameter at all when formatting BDAM files or files allocated on 3344 disk devices with RPS implemented within the hardware. When you use the SIZE parameter in these cases, CA-IDMS/DB cannot access the required current updated copy of the DTF.

### 7.6.13 INSTALL STAMPS

**Local mode: File assignments for the batch command facility (VSE/ESA)**

```
// DLBL    userdb,'user.userdb',,tt
// EXTENT SYSnnn,nnnnnn,,sss,rrrr
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    userdict,'user.userdict',,tt
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

---

userdb	Filename of the user file
user.userdb	File-ID of the user file
user.userdict	File-ID of the dictionary containing the table definitions

---

**Central version:** To execute INSTALL STAMPS under the central version, remove the USERDICT statement.

### 7.6.14 LOAD

**File assignments for the batch command facility (VSE/ESA)**

---

```
// DLBL   userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   userdict,'user.userdict',,tt
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   SYS001,'user.input',,SD
// EXTENT SYS001,nnnnnn,,ssss,1111
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
// DLBL   SYS002,'user.loadin',,SD
// EXTENT SYS002,nnnnnn,,ssss,1111
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// DLBL   SYS003,'user.loadout',,SD
// EXTENT SYS003,nnnnnn,,ssss,1111
// ASSGN  SYS003,DISK,VOL=vvvvvv,SHR
// DLBL   SYSnnn,'sort.load',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Sort files are only needed when performing a complete LOAD

```
// DLBL   SYSnnn,'sort.work',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional sort files, as required

---

userdb	Filename of the user file
user.userdb	File-ID of the user file
user.userdict	File-ID of the dictionary containing the table definitions
user.input	File-ID of the input file
user.loadin	File-ID of the input SYS002 file; for sizing information see 4.14, "LOAD."
user.loadout	File-ID of the output SYS003 file; for sizing information see 4.14, "LOAD."
sort.load	File-ID of the SYSPCH file

**Note:** When performing a complete LOAD, you must preallocate the file referenced by SYS002 and SYS003 (the same file is used for both assigns in a complete LOAD). Do not use a temporary data set for these files.

## 7.6.15 MAINTAIN INDEX

**File assignments for the batch command facility (VSE/ESA)**

---

```
// DLBL    SORTWK1,'sort.work.file'
// EXTENT  SYSnnn,SYSWK0,,1,30
// ASSGN   SYSnnn,SYS001,OUTPUT
```

Additional sort files, as required

```
// DLBL    userdb,'user.olddb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional existing database file assignments, as required

```
// DLBL    RELDCTL,'user.reldctl',,SD
// EXTENT  SYSnnn,nnnnnn,,ssss,1111
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    SYS003,'user.db1003',,SD
// EXTENT  SYS003,nnnnnn,,ssss,1111
// ASSGN   SYS003,DISK,VOL=vvvvvv,SHR
// DLBL    SYS004,'user.db1004',,SD
// EXTENT  SYS004,nnnnnn,,ssss,1111
// ASSGN   SYS004,DISK,VOL=vvvvvv,SHR
// DLBL    SYS005,'user.db1005',,SD
// EXTENT  SYS005,nnnnnn,,ssss,1111
// ASSGN   SYS005,DISK,VOL=vvvvvv,SHR
// DLBL    SYS006,'user.db1006',,SD
// EXTENT  SYS006,nnnnnn,,ssss,1111
// ASSGN   SYS006,DISK,VOL=vvvvvv,SHR
```

---

user.olddb	File-ID of the existing database file
user.reldctl	Data set name of the reload control file containing control and set information. Blocksize of the RELDCTL file should be a multiple of 60 with a maximum size of 32,760. Blocksize is controlled through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see <i>CA-IDMS Database Administration</i> .
user.db1003	File-ID of the intermediate work file containing index descriptors from IDMSTABX
user.db1004	File-ID of the intermediate work file containing the output from SORT3
user.db1005	File-ID of the intermediate work file containing pointers for user-owned index sets from IDMSDBL3
user.db1006	File-ID of the intermediate work file containing sorted pointers for user-owned index sets from SORT4

---

## 7.6.16 MERGE ARCHIVE

### File assignments for the batch command facility (VSE/ESA)

```
// TLBL   SYS001,'idms.tjrnlold',,nnnnn,,f
// ASSGN  SYS001,x'cuu'
// TLBL   SYS002,'idms.tjrnlfix',,nnnnn,,f
// ASSGN  SYS002,x'cuu'
// TLBL   JRNMO1,'idms.tmrngold',,nnnnn,,f
// ASSGN  SYSnnn,x'cuu'
// DLBL   SYSnnn,'sort.work',,SD
// EXTENT SYSnnn,nnnnn,,ssss,1111
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional sort files, as required

idms.tjrnlold	File-ID of the input tape journal file
f	File number of the tape journal file
idms.tjrnlfix	File-ID of the output tape journal file
idms.tmrngold	File-ID of the input tape merged file
SYSnnn	Logical unit name for merged journal file. This must be assigned to JRNMO1 through SYSIDMS (e.g., FILENAME=JRNMO1,DEVADDR=SYS003).

## 7.6.17 PRINT INDEX

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL   userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional file assignments, as required

userdb	Filename of the user file
user.userdb	File-ID of the user file

## 7.6.18 PRINT JOURNAL

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL   SYS001,'idms.archjrn1',,SD
// EXTENT SYSnnn,nnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

---

SYS001	Filename of the archive journal file
idms.archjrn1	File-ID of the archive journal file

---

## 7.6.19 PRINT LOG

### File assignments for the batch command facility (VSE/ESA)

#### To print from the DDLDCLOG area

```
// DLBL    dcllog,'idms.system.ddldclog',,DA
// EXTENT  SYSnnn,nnnnnn,,sss,
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

#### To print from the archive log file:

```
// TLBL    SYS001,'idms.archive',,nnnnnn,,f
// ASSGN   SYS001,x'ccc'
```

---

idms.archive	File-ID of the archive log file
--------------	---------------------------------

---

## 7.6.20 PRINT PAGE

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL    userdb1,'user.userdb1',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    userdb2,'user.userdb2',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional file assignments, as required

---

userdb1	Filename of the user file
user.userdb1	File-ID of the user file

---

## 7.6.21 PRINT SPACE

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL    userdb1,'user.userdb1',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    userdb2,'user.userdb2',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional file assignments, as required

---

userdb1	Filename of the user file
user.userdb1	File-ID of the user file

---

## 7.6.22 PUNCH

### Local mode: File assignments for the batch command facility (VSE/ESA)

```
// DLBL   usercat, 'user.ddlcat', ,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   usercat1, 'user.ddlcat1', ,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   usercatx, 'user.ddlcatx', ,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL   SYSnnn, 'idms.sysjrnl', ,nnnnnn, ,f
// ASSGN  SYSnnn,x,ccc'
```

Additional dummied journals, as required

---

user.ddlcat	File-ID of the dictionary file containing the DDLCAT area of the dictionary
user.ddlcat1	File-ID of the dictionary file containing the DDLCATLOD area of the dictionary
user.ddlcatx	File-ID of the dictionary file containing the DDLCATX area of the dictionary
idms.sysjrnl	File-ID of the user file

---

**Central version:** To execute INSTALL STAMPS under the central version, remove the USERCAT, USERCATL, USERCATX, and SYSJRNL statements.

## 7.6.23 RELOAD

### File assignments for the batch command facility (VSE/ESA)

---

```
// DLBL SORTWK1,'sort.work.file'
// EXTENT SYSnnn,SYSWK0,,1,30
// ASSGN SYSnnn,SYS003,OUTPUT
```

Additional sort files, as required

```
// DLBL userdb, '.user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional existing database file assignments, as required

```
// DLBL RELDCTL.'user.reldctl',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL SYS001,'sort.unload',,SD
// EXTENT SYS001,nnnnnn,,ssss,1111
// ASSGN SYS001,DISK,VOL=vvvvvv,SHR
// DLBL SYS002,'user.db1002',,SD
// EXTENT SYS002,nnnnnn,,ssss,1111
// ASSGN SYS002,DISK,VOL=vvvvvv,SHR
// DLBL SYS003,'user.db1003',,SD
// EXTENT SYS003,nnnnnn,,ssss,1111
// ASSGN SYS003,DISK,VOL=vvvvvv,SHR
// DLBL SYS004,'user.db1004',,SD
// EXTENT SYS004,nnnnnn,,ssss,1111
// ASSGN SYS004,DISK,VOL=vvvvvv,SHR
// DLBL SYS005,'user.db1005',,SD
// EXTENT SYS005,nnnnnn,,ssss,1111
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL SYS006,'user.db1006',,SD
// EXTENT SYS006,nnnnnn,,ssss,1111
// ASSGN SYS006,DISK,VOL=vvvvvv,SHR
// DLBL SYS007,'user.db1007',,SD
// EXTENT SYS007,nnnnnn,,ssss,1111
// ASSGN SYS007,DISK,VOL=vvvvvv,SHR
// DLBL SYS008,'user.db1008',,SD
// EXTENT SYS008,nnnnnn,,ssss,1111
// ASSGN SYS008,DISK,VOL=vvvvvv,SHR
// DLBL SYS009,'user.db1009',,SD
// EXTENT SYS009,nnnnnn,,ssss,1111
// ASSGN SYS009,DISK,VOL=vvvvvv,SHR
// DLBL SYS010,'user.db1010',,SD
// EXTENT SYS010,nnnnnn,,ssss,1111
// ASSGN SYS010,DISK,VOL=vvvvvv,SHR
// DLBL SYS011,'user.db1011',,SD
// EXTENT SYS011,nnnnnn,,ssss,1111
// ASSGN SYS011,DISK,VOL=vvvvvv,SHR
```

---

*sort.unload*

File-ID of the sort parameters created by  
UNLOAD

---

---

<i>user.reldctl</i>	Data set name of the reload control file containing control and set information. Blocksize of the RELDCTL file should be a multiple of 60 with a maximum size of 32,760. Blocksize is controlled through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see <i>CA-IDMS Database Administration</i> .
<i>user.dbl002</i>	File-ID of the file put out by UNLOAD as SYS001
<i>user.dbl003</i>	File-ID of the file put out by UNLOAD as SYS003
<i>user.dbl004</i>	File-ID of the intermediate work file containing the output from SORT1
<i>user.dbl005</i>	File-ID of the intermediate work file containing member descriptors for chained sets from IDMSDBL2
<i>user.dbl006</i>	File-ID of the intermediate work file containing member descriptors for user-owned index sets from IDMSDBL2
<i>user.dbl007</i>	File-ID of the intermediate work file containing the output of SORT2
<i>user.dbl008</i>	File-ID of the intermediate work file containing the reformatted index information from IDMSDBLX
<i>user.dbl009</i>	File-ID of the intermediate work file containing the sorted index set descriptors from SORT3
<i>user.dbl010</i>	File-ID of the intermediate work file containing prefix pointer information from IDMSDBL3
<i>user.dbl011</i>	File-ID of the intermediate work file containing sorted prefix pointer information from SORT4
<i>userdb</i>	Filename of the database file
<i>user.userdb</i>	File-ID of the database file

---

## 7.6.24 RESTORE

**File assignments for the batch command facility (VSE/ESA)**

```
// DLBL    userdb1,'user.userdb1',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    userdb2,'user.userdb2',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL    SYS001,'user.bkpfile',,SD
// ASSGN   SYS001,DISK,VOL=vvvvvv,SHR
```

Additional file assignments, as required

userdb1	Filename of the user file
user.userdb1	File-ID of the user file
user.bkpfile	File-ID of the backup file

## 7.6.25 RESTRUCTURE CONNECT

**File assignments for the batch command facility (VSE/ESA)**

```
// DLBL    userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    SYS001,'user.spill',,DA
// EXTENT  SYS001,nnnnnn
// ASSGN   SYS001,DISK,VOL=vvvvvv,SHR
// DLBL    SORTWK1,'sort.work.file'
// EXTENT  SYSnnn,SYSWK0,,1,30
// ASSGN   SYSnnn,SYS003,OUTPUT
```

Additional sort files, as required

user.userdb	File-ID of the user database file
user.spill	File-ID of the spill file; for sizing information see 4.25, "RESTRUCTURE CONNECT."

## 7.6.26 RESTRUCTURE SEGMENT

**File assignments for the batch command facility (VSE/ESA)**

```
// DLBL   userdb1,'user.userdb1',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   userdb2,'user.userdb2',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   SYS001,'user.spill',,DA
// EXTENT SYS001,nnnnnn
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
```

Additional file assignments, as required

userdb1	Filename of the user file
user.userdb1	File-ID of the user file
user.spill	File-ID of the spill file; for sizing information see 4.25, "RESTRUCTURE CONNECT."

## 7.6.27 ROLLBACK

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL   userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional database file assignments, as required

```
// TLBL   SYS001,'archive',,SD
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```

If using the SORT option, add sort files

userdb	Filename of the user file
archive	File-ID of the complete archive or tape journal file

## 7.6.28 ROLLFORWARD

### File assignments for the batch command facility (VSE/ESA)

---

```
// DLBL    userdb, 'user.userdb', ,DA
// EXTENT  SYSnnn, nnnnnn
// ASSGN   SYSnnn, DISK, VOL=vvvvvv, SHR
```

Add database file assignments, as required

If recovering from a standard journal file:

```
// TLBL    SYSnnn, 'archive', ,SD
// ASSGN   SYSnnn, DISK, VOL=vvvvvv, SHR
```

If recovering from a journal extract file:

```
// DLBL    extract, 'jrnل.extract', ,SD
// EXTENT  SYSnnn, nnnnnn
// ASSGN   SYSnnn, DISK, VOL=vvvvvv, SHR
```

If using the SORT option or processing a journal extract file, add sort files as required

---

userdb	Filename of the user file.
archive	File-ID of the complete archive or tape journal file.
extract	File name of the extract journal file. If you do not specify, it defaults to SYS002.
jrnل.extract	File-ID of the extract journal file.

---

## 7.6.29 TUNE INDEX

**File assignments for the batch command facility (VSE/ESA)**

```
//DLBL    userdb, 'user.userdb', ,DA
//EXTENT  SYSnnn, nnnnnn
//ASSIGN  SYSnnn, DISK, VOL=vvvvvv, SHR
```

---

userdb	Filename of the user file
user.userdb	File-ID of the user file

---

## 7.6.30 UNLOCK

**File assignments for the batch command facility (VSE/ESA)**

```
// DLBL    userdb, 'user.userdb', ,DA
// EXTENT  SYSnnn, nnnnnn
// ASSGN   SYSnnn, DISK, VOL=vvvvvv, SHR
```

---

user.userdb	File-ID of the user database file
-------------	-----------------------------------

---

## 7.6.31 UNLOAD

### File assignments for the batch command facility (VSE/ESA)

---

```
// DLBL  SORTWK1,'sort.work.file'
// EXTENT SYSnnn,SYSWK0,,1,30
// ASSGN  SYSnnn,SYSO03,OUTPUT
```

Additional sort files, as required

```
// DLBL  RELDCTL.'user.reldctl',,SD
// EXTENT SYSnnn,nnnnnn,,sss,1111
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  SYS002,'user.db1002',,SD
// EXTENT SYS002,nnnnnn,,sss,1111
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// DLBL  SYS003,'user.db1003',,SD
// EXTENT SYS003,nnnnnn,,sss,1111
// ASSGN  SYS003,DISK,VOL=vvvvvv,SHR
// DLBL  IJSPCH,'sort.unload'
// EXTENT SYSPCH,nnnnnn,,sss,1111
// ASSGN  SYSPCH,DISK,VOL=vvvvvv,SHR
```

---

user.reldctl	File-ID of the reload control file containing control and set information. Blocksize of the RELDCTL file should be a multiple of 60 with a maximum size of 32,760. Blocksize is controlled through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see <i>CA-IDMS Database Administration</i> .
user.db1002	File-ID of the SYS002 output file
user.db1003	File-ID of the SYS003 output file
sort.unload	File-ID of the file containing sort parameters created by UNLOAD

---

## 7.6.32 UPDATE STATISTICS

### File assignments for the batch command facility (VSE/ESA)

```
// DLBL  userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL  SYSnnn,'idms.tapejrn1',,nnnnnn,,f
// ASSGN  SYSnnn,x'cuu'
```

---

userdb	Filename of the user file
--------	---------------------------

---

### 7.6.33 VALIDATE

#### File assignments for the batch command facility (VSE/ESA)

```
// DLBL    userdb, 'user.userdb', ,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    SYS002, 'user.valin', ,DA
// EXTENT  SYS002,nnnnnn
// ASSGN   SYS002,DISK,VOL=vvvvvv,SHR
// DLBL    SYS003, 'user.valout', ,DA
// EXTENT  SYS003,nnnnnn
// ASSGN   SYS003,DISK,VOL=vvvvvv,SHR
// DLBL    SYSnnn, 'sort.load', ,SD
// EXTENT  SYSnnn,nnnnnn, , ,ssss,1111
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

If performing a complete VALIDATE, add sort files as necessary:

```
// DLBL    SYSnnn, 'sort.work', ,SD
// EXTENT  SYSnnn,nnnnnn, , ,ssss,1111
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

user.userdb	File-ID of the database file
user.valin	File-ID of the input SYS002 file; for sizing information see 4.33, "VALIDATE."
user.valout	File-ID of the SYS003 output file; for sizing information see 4.33, "VALIDATE."
sort.load	File-ID of the SYSPCH file

**Note:** When running a complete VALIDATE, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to a *different* intermediate file.

## 7.7 Utility programs

### 7.7.1 IDMSDBAN

#### IDMSDBAN (VSE/ESA)

---

```
// EXEC PROC=IDMSLBLs
// DLBL idms150,'idms150.library'
// EXTENT SYSnnn,nnnnnn,,sss,ttt
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA-IDMS 15.0 libraries
// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
```

#### Additional database file assignments, as required

```
// TLBL SYS002,'chain.file',,nnnnnn,,f
// ASSGN SYS002,x'cuu'
// EXEC IDMSDBN1
```

#### Insert SYSIDMS parameters, as required

```
/*
```

#### IDMSDBAN input parameters

```
// ASSGN SYS004,x'cuu'
// EXTENT SYS004
```

#### Additional sort work file assignments, as required

```
// TLBL SYS001,'chain.file',,nnnnnn,,f
// ASSGN SYS001,x'cuu'
// TLBL SYSnnn,'output.work',,nnnnnn,,f
// ASSGN SYSnnn,x'cuu'
// TLBL SYSnnn,'input.work',,nnnnnn,,f
// ASSGN SYSnnn,SYS002
// EXEC IDMSDBN2
```

#### Insert SYSIDMS parameters, as required

```
/*
```

---

IDMSLBLs	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, and SYSIDMS parameter file.
----------	-------------------------------------------------------------------------------------------------------------------------------------------------

►►For a complete listing of IDMSLBLs, see "IDMSLBLs Procedure" later in this section.

idms150	Dtfname of the CA-IDMS library
---------	--------------------------------

---

---

idms150.library	Data set name of CA-IDMS 15.0 libraries, as established during installation
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
tttt	Number of tracks
vvvvvv	Volume serial number
CA-IDMS 15.0 libraries	The CA-IDMS 15.0 libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file ▶▶For a complete listing of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .
userdb	Filename of the user database file
user.userdb	File-ID of the user database file
chain.file	File-ID of the chain file generated by IDMSDBN1
SYS004	Logical unit assignment of the sort work file
ccc	Physical device assignment of the sort work file
output.work	Output file-ID of the intermediate chain file used by IDMSDBN2
input.work	Input file-ID of the intermediate chain file used by IDMSDBN2

---

## 7.7.2 IDMSDIRL

### Local mode IDMSDIRL (VSE/ESA)

---

```

// EXEC PROC=IDMSLBLS
// DLBL   idms150,'idms150.library'
// EXTENT SYSnnn,nnnnnn,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA-IDMS 15.0 libraries
// TLBL   SYSnnn,'idms.tapejrn1',,nnnnnn,,f
// ASSGN  SYSnnn,x'cuu'
// DLBL   dictdb,'idms.dictdb',,DA
// EXTENT SYS005,nnnnnn
// ASSGN  SYS005,DISK,VOL=nnnnnn,SHR
// TLBL   SYS001,'idms.dir1data',,nnnnnn,,f
// ASSGN  SYS001,x'cuu'
// EXEC   IDMSDIRL

```

Insert SYSIDMS parameters, as required

/\*

IDMSDIRL input parameters

/\*

---

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, and SYSIDMS parameter file.  ▶▶For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
<hr/>	
idms150	Dtfname of the CA-IDMS library
<hr/>	
idms150.library	Data set name of CA-IDMS 15.0 libraries, as established during installation
<hr/>	
SYSnnn	Logical unit of the volume for which the extent is effective
<hr/>	
nnnnnn	Volume serial identifier of appropriate disk volume
<hr/>	
ssss	Starting track (CKD) or block (FBA) of disk extent
<hr/>	
tttt	Number of tracks
<hr/>	
vvvvvv	Volume serial number
<hr/>	
CA-IDMS 15.0 libraries	The CA-IDMS 15.0 libraries, as established during installation
<hr/>	
SYSIDMS	Filename of the SYSIDMS parameter file  ▶▶For a complete listing of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .
<hr/>	
idms.tapejrn1	File-ID of the tape journal file
<hr/>	
f	File number of the tape file
<hr/>	

dictdb	Filename of the data dictionary file
idms.dictdb	File-ID of the data dictionary file
idms.dirldata	File-ID of the IDMSDIRL input file

**Central version:** To execute IDMSDIRL under the central version,

- Remove the TLBL, EXTENT, and ASSGN statements for the tape journal file from the JCL above
- Add a SYSCTL file

### 7.7.3 IDMSLOOK

#### IDMSLOOK (VSE/ESA)

```
// EXEC  PROC=IDMSLBLS
// DLBL  idms150,'idms150.library'
// EXTENT SYSnnn,nnnnnn,,ssss,tttt
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL  SYSnnn,'idms.tapejrn1',,nnnnnn,,f
// ASSGN SYSnnn,x'cuu'
// LIBDEF *,SEARCH=CA-IDMS 15.0 libraries
// EXEC  IDMSLOOK,SIZE=
```

Insert SYSIDMS parameters, as required

/\*

Insert IDMSLOOK parameters

/\*

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, and SYSIDMS parameter file.  ▶▶For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idms150	Dtfname of the CA-IDMS library
idms150.library	Data set name of CA-IDMS 15.0 libraries, as established during installation
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
tttt	Number of tracks

---

vvvvvv	Volume serial number
CA-IDMS 15.0 libraries	The CA-IDMS 15.0 libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file ►►For a complete listing of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

## 7.7.4 IDMSRPTS

### Local mode: IDMSRPTS (VSE/ESA)

---

```
// EXEC PROC=IDMSLBLS
// DLBL idms150,'idms150.library'
// EXTENT SYSnnn,nnnnnn,,ssss,ttt
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA-IDMS 15.0 libraries
// TLBL SYSnnn,'idms.tapejrnl',,nnnnn,,f
// ASSGN SYSnnn,x'cuu'
// DLBL dictdb,'idms.dictdb',,DA
// EXTENT sys005,nnnnnn
// ASSGN sys005,DISK,VOL=nnnnnn,SHR
// EXEC IDMSRPTS
```

Insert SYSIDMS parameters, as required

/\*

IDMSRPTS input parameters

/\*

---

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, and SYSIDMS parameter file. ►►For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idms150	Dtfname of the CA-IDMS library
idms150.library	Data set name of CA-IDMS 15.0 libraries, as established during installation
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

---

tttt	Number of tracks
vvvvvv	Volume serial number
CA-IDMS 15.0 libraries	The CA-IDMS 15.0 libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file  ▶▶For a complete listing of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .
SYS009	Logical unit assignment of the tape journal file
idms.tapejrn1	File-id of the tape journal file
nnnnnn	Volume serial number
f	File number of the tape journal file
dictdb	Filename of the data dictionary file
idms.dictdb	File-id of the data dictionary file
sys005	Logical unit assignment of the data dictionary file

**Central version:** To execute IDMSRPTS under the central version,

- Remove the TLBL, EXTENT, and ASSGN statements for the tape journal file from the JCL above.
- Add a SYSCTL file

## 7.7.5 IDMSRSTC

**Local mode: IDMSRSTC (VSE/ESA)**

```
// EXEC PROC=IDMSLBLE
// DLBL idms150,'idms150.library'
// EXTENT SYSnnn,nnnnnn,,ssss,tttt
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA-IDMS 15.0 libraries
// TLBL SYSnnn,'idms.tapejrn1',,nnnnnn,,f
// ASSGN SYSnnn,x'cuu'
// DLBL dictdb,'idms.dictdb',A
// EXTENT SYS005,nnnnnn
// ASSGN SYS005,DISK,VOL=nnnnnn,,SHR
// DLBL IJSYSPH,'rstc.macros',0
// EXTENT SYSPCH,nnnnnn,,ssss,1111
// ASSGN SYSPCH,x'cuu'
// EXEC IDMSRSTC
```

Insert SYSIDMS parameters

```
/*
IDMSRSTC input parameters
```

```
/*
```

---

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, and SYSIDMS parameter file.  ▶▶For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idms150	Dtfname of the CA-IDMS library
idms150.library	Data set name of CA-IDMS 15.0 libraries, as established during installation
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
tttt	Number of tracks
vvvvvv	Volume serial number
CA-IDMS 15.0 libraries	The CA-IDMS 15.0 libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file  ▶▶For a complete listing of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .
sys009	Logical unit assignment of the tape journal file
idms.tapejrn1	File-id of the tape journal file
nnnnnn	Volume serial number
f	File number of the tape journal file
dictdb	Filename of the data dictionary file
idms.dictdb	File-id of the data dictionary file
sys005	logical unit assignment of the data dictionary file
rstc.macros	File-id of the card-image file containing the output IDMSRSTT macro statements
ssss	Starting track (CKD) or block (FBA) of the disk extent
llll	Number of tracks (CKD) or blocks (FBA) in the disk extent
ccc	Physical device assignment of the card-image file

---

**Central version:** To execute IDMSRSTC under the central version:

- Remove the TLBL, EXTENT, and ASSGN statements for the tape journal from the JCL above
- Add a SYSCTL file

## 7.7.6 IDMSRSTT

### Assemble and link an IDMSRSTT module

```
// DLBL userlib  
// EXTENT ,nnnnnn  
// LIBDEF SOURCE,SEARCH=:(userlib.idmslib)  
// OPTION CATAL  
// PHASE idmsrstt,*  
// EXEC ASMA90
```

Put IDMSRSTT macro statements here

```
/*  
// DLBL userlib  
// EXTENT ,nnnnnn  
// LIBDEF PHASE,T0=(userlib.idmslib)  
// EXEC LNKEDT
```

---

<u>userlib</u>	Filename of the user library
<u>nnnnnn</u>	Volume serial number of the library
<u>userlib.idmslib</u>	File identifier of the CA-IDMS sublibrary
<u>idmsrstt</u>	Phase name of the IDMSRSTT table

---

## 7.8 IDMSLBLE Procedure

**What is the IDMSLBLE procedure:** IDMSLBLE is a procedure provided during a CA-IDMS VSE/ESA 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

**Note:** You can define a SYSCTL procedure that overrides IDMSOPTI specifications for central version operations. For more information, refer to *CA-IDMS Installation and Maintenance Guide — VSE*.

Tailor the IDMSLBLE procedure to reflect the filenames and definitions in use at your site and include this procedure in VSE/ESA JCL job streams.

The sample VSE/ESA JCL provided in this document includes the IDMSLBLE procedure. Therefore, individual file definitions for CA-IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

### **IDMSLBLE procedure listing**

---

```
/* ----- 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=vvvvvv,SHR
// DLBL   dccatl,'idms.system.dccatlod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dccatx,'idms.system.dccatx',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dcdml,'idms.system.ddldml',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,101
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dclod,'idms.system.ddldclod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,21
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dcllog,'idms.system.ddldcllog',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,401
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dcrun,'idms.system.ddldcrun',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,68
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dcscr,'idms.system.ddldcscr',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,135
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dcmsg,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dclscr,'idms.sysloc.ddlocscr',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dirldb,'idms.sysdirl.ddldml',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   dirllod,'idms.sysdirl.ddldclod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,2
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   empdemo,'idms.empdemo1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   insdemo,'idms.insdemo1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   orgdemo,'idms.orgdemo1',1999/365,DA
```

---

---

```

// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL emp1dem,'idms.sqldemo.emp1demo',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL infodem,'idms.sqldemo.infodemo',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL projdem,'idms.projseq.projdemo',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL indxdem,'idms.sqldemo.indxdemo',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL sysctl,'idms.sysctl',1999/365,SD
// EXTENT SYSnnn,nnnnnn,,ssss,2
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL secdd,'idms.sysuser.ddlsec',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,26
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL dictdb,'idms.appldict.ddldm1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,51
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL dloddb,'idms.appldict.ddldclod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,51
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL sqldd,'idms.syssql.ddlcat',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,101
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL sqllod,'idms.syssql.ddlcat1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,51
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL sqlxdd,'idms.syssql.ddlcatx',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,26
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL asfdm1,'idms.asfdict.ddldm1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL asflod,'idms.asfdict.asflod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,401
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL asfdata,'idms.asfdict.asfdata',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL ASFDEFN,'idms.asfdict.asfdefn',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,101
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR

```

---

---



---

```
// DLBL    j1jrn1,'idms.j1jrn1',1999/365,DA
// EXTENT  SYSnnn,nnnnn,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    j2jrn1,'idms.j2jrn1',1999/365,DA
// EXTENT  SYSnnn,nnnnn,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    j3jrn1,'idms.j3jrn1',1999/365,DA
// EXTENT  SYSnnn,nnnnn,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    SYSIDMS,'#SYSIPT',0,SD
/+
/*
```

---



---

<i>idmslib.sublib</i>	Name of the sublibrary within the library containing CA-IDMS modules
<i>user.sublib</i>	Name of the sublibrary within the library containing user modules
<i>idmslib</i>	Filename of the file containing CA-IDMS modules
<i>idms.library</i>	File-ID associated with the file containing CA-IDMS modules
<i>SYSnnn</i>	Logical unit of the volume for which the extent is effective
<i>nnnnn</i>	Volume serial identifier of appropriate disk volume
<i>ssss</i>	Starting track (CKD) or block (FBA) of disk extent
<i>dccat</i>	Filename of the system dictionary catalog (DDL CAT) area
<i>idms.system.dccat</i>	File-ID of the system dictionary catalog (DDL CAT) area
<i>dccatl</i>	Filename of the system dictionary catalog load (DDL CATLOD) area
<i>idms.system.dccatlod</i>	File-ID of the system dictionary catalog load (DDL CATLOD) area
<i>dccatx</i>	Filename of the system dictionary catalog index (DDL CATX) area
<i>idms.system.dccatx</i>	File-ID of the system dictionary catalog index (DDL CATX) area
<i>dcdml</i>	Filename of the system dictionary definition (DDL DML) area
<i>idms.system.ddldml</i>	File-ID of the system dictionary definition (DDL DML) 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 (DDLDCSCR) area
idms.sysloc.ddlocscr	File-ID of the local mode system scratch (DDLDCSCR) area
dirlb	Filename of the IDMSDIRL definition (DDLDCML) area
idms.sysdirl.ddldml	File-ID of the IDMSDIRL definition (DDLDCML) 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 ORGDEMO 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 (DDLML) 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 (DDLML) area
idms.asfdict.ddldml	File-ID of the asf dictionary definition (DDLML) 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



## Chapter 8. CMS Commands

---

8.1	About this chapter	8-3
8.2	CA-IDMS Batch Command Facility	8-4
8.3	Utility statements	8-6
8.3.1	ARCHIVE JOURNAL	8-6
8.3.2	ARCHIVE LOG	8-6
8.3.3	BACKUP	8-7
8.3.4	BUILD	8-7
8.3.5	CLEANUP	8-8
8.3.6	CONVERT PAGE	8-8
8.3.7	EXPAND PAGE	8-9
8.3.8	EXTRACT JOURNAL	8-9
8.3.9	FASTLOAD	8-10
8.3.10	FIX ARCHIVE	8-11
8.3.11	FIX PAGE	8-11
8.3.12	FORMAT	8-12
8.3.13	INSTALL STAMPS	8-12
8.3.14	LOAD	8-13
8.3.15	MAINTAIN INDEX	8-14
8.3.16	MERGE ARCHIVE	8-15
8.3.17	PRINT INDEX	8-15
8.3.18	PRINT JOURNAL	8-16
8.3.19	PRINT LOG	8-16
8.3.20	PRINT PAGE	8-16
8.3.21	PRINT SPACE	8-17
8.3.22	PUNCH	8-17
8.3.23	RELOAD	8-18
8.3.24	RESTORE	8-21
8.3.25	RESTRUCTURE CONNECT	8-21
8.3.26	RESTRUCTURE SEGMENT	8-22
8.3.27	ROLLBACK	8-22
8.3.28	ROLLFORWARD	8-23
8.3.29	TUNE INDEX	8-23
8.3.30	UNLOCK	8-23
8.3.31	UNLOAD	8-24
8.3.32	UPDATE STATISTICS	8-24
8.3.33	VALIDATE	8-25
8.4	Utility programs	8-27
8.4.1	IDMSDBAN	8-27
8.4.2	IDMSDIRL	8-28
8.4.3	IDMSLOOK	8-29
8.4.4	IDMSRPTS	8-30
8.4.5	IDMSRSTC	8-31
8.4.6	IDMSRSTT	8-32



## 8.1 About this chapter

This chapter presents sample CMS commands to run CA-IDMS utility statements and programs.

Common commands used to run the CA-IDMS Batch Command Facility are presented first. Additional commands required to run each utility statement and program are presented next, alphabetically, by utility.

## 8.2 CA-IDMS Batch Command Facility

Listed below are CMS commands to execute the CA-IDMS Batch Command Facility.

When using the IDMSBCF program to execute a utility statement, code these commands along with the required statements for each of the utilities.

The file assignments for each utility statement and program are presented on subsequent pages in this chapter.

### Local mode IDMSBCF (CMS)

```
FILEDEF SYSLST PRINTER
FI dcmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
```

Insert file assignments required by the utility statements

```
FILEDEF sysjrn1 TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb)
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK utility input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSBCF
```

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

idmslib	Filename of the load library containing CA-IDMS executable modules
dbalib	Filename of the load library containing the DMCL and database name table load modules
dcmsg	DDname of the system message (DDLDCMSG) area
idms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
sysjrn1	DDname of the tape journal file
nnnnnn	Volume serial number of the tape journal file
lll	Record length of the tape journal file
rbbbb	Block size of the tape journal file
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.  ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

sysidms input a	File identifier of the file containing SYSIDMS parameters if applicable
utility input a	File identifier of the file containing utility statements

---

**Central version IDMSBCF (CMS)**

```

FI dcmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF SYSCTL DISK sysctl file fm
FILEDEF sysjrn1 DUMMY
FILEDEF SYSLST PRINTER
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib

```

Insert file assignments required by the utility statements

```

FILEDEF SYSIPT DISK utility input a
OSRUN IDMSBCF

```

---

idmslib	Filename of the load library containing CA-IDMS executable modules
utility input a	File identifier of the file containing utility statements

---

## 8.3 Utility statements

### 8.3.1 ARCHIVE JOURNAL

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF j1jrn1 DISK j1jrn1 jrn1file fm (RECFM F LRECL 111 XTENT nnn
FILEDEF j2jrn1 DISK j2jrn1 jrn1file fm (RECFM F LRECL 111 XTENT nnn
```

Additional journal file assignments, as required

```
FILEDEF archjrn1 TAP1 SL VOLID nnnnnn (RECFM VB LRECL rrrr BLKSIZE bbbb
```

<u>j1jrn1</u>	DDname of the first disk journal file, as defined in the DMCL
<u>j1jrn1 jrn1file fm</u>	File identifier of the first disk journal file
<u>j2jrn1</u>	DDname of the second disk journal file
<u>j2jrn1 jrn1file fm</u>	File identifier of the second disk journal file
<u>lll</u>	Page size of the journal file
<u>nnn</u>	Number of pages in the journal file
<u>archjrn1</u>	DDname of the tape archive file, as defined in the DMCL
<u>nnnnnn</u>	Volume serial number of the tape archive file
<u>rrrr</u>	Length of the longest record in the tape archive file
<u>bbbb</u>	Block size of the tape archive file, as defined in the DMCL

### 8.3.2 ARCHIVE LOG

#### FILEDEF commands for the batch command facility (CMS)

```
FI dlogdb DISK idms dlogdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF sysjrn1 DUMMY
FILEDEF SYS002 DISK archive dbfile fm (RECFM VB LRECL 280 BLKSIZE bbbb
```

<u>dlogdb</u>	DDname of the system log area
<u>idms dlogdb fm</u>	File identifier of the system log area
<u>ppp</u>	Page size of the database file

---

nnn	Number of pages in the database file
dcmsg	DDname of the system message (DDLDCMSG) area
idms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
sysjrnl	DDname of the tape journal file defined in the DMCL module named in the IDMSNWKS subschema
archive dbfile fm	File identifier of the archive log file
bbbb	Block size of the archive log file; must be greater than or equal to 284 (typically, equal to 4 plus a multiple of 280)

---

### 8.3.3 BACKUP

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
```

Additional file assignments, as required

```
FILEDEF SYS001 TAP1 SL VOLID nnnnnn (RECFM VB LRECL 1111 BLKSIZE bbbb)
```

---

userdb	DDname of the database file
userdb dbfile fm	File identifier of the database file
nnnnnn	Volume serial number of the tape backup file
1111	Record length of the tape backup file. Must be the size of the largest page being backed up.
bbbb	Block size of the tape backup file. Must be as large as the smaller of: <ul style="list-style-type: none"> <li>■ The size of the largest page being backed up, plus 8 or</li> <li>■ 32,760</li> </ul>

---

### 8.3.4 BUILD

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
```

```
FILEDEF SYS002 DISK user load fm
```

```
FILEDEF SYS003 TAP1 SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbb)
```

```
FILEDEF SYSPCH DISK sort build fm
```

```
FILEDEF SORTMSG PRINTER
```

```
FILEDEF SORTWK01 DISK sortwknn file fm
```

---

userdb	DDname of the database file
userdb dbfile fm	File identifier of the database file
user load fm	File identifier of the input (SYS002) file; for sizing information see discussion of SYS003 in 4.14, "LOAD"
SYS003	DDname of the output (SYS003) file; for sizing information see 4.4, "BUILD"
nnnnnn	Volume serial number of the SYS003 file
lll	Record size of the SYS003 file
bbb	Block size of the SYS003 file
sort build fm	File identifier of the SYSPCH file
sortwknn file fm	File identifier of the SORTWKnn file

---

**Note:** When running a complete BUILD, SYS002 and SYS003 must point to the SAME intermediate file. When running a stepped BUILD, SYS002 and SYS003 must point to a DIFFERENT intermediate file.

### 8.3.5 CLEANUP

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF sysjrn1 DISK tapejrn1 jrn1file fm
FILEDEF userdb DISK userdb dbfile fm
```

Additional database file assignments, as required

---

sysjrn1	DDname of the tape journal file
tapejrn1 jrn1file fm	File identifier of the tape journal file
userdb	DDname of the user database file
userdb dbfile fm	File identifier of the user database file

---

### 8.3.6 CONVERT PAGE

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
```

Additional existing database file assignments, as required

```
FILEDEF newdb DISK newdb dbfile fm
```

Additional converted database file assignments as required

---

userdb	DDname of the input database file
userdb dbfile fm	File identifier of the existing database file
newdb	DDname of the output converted database file
newdb dbfile fm	File identifier of the output converted database file

---

### 8.3.7 EXPAND PAGE

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
```

Additional existing database file assignments, as required

```
FILEDEF xfile DISK xbase dbfile fm (expanded-database-size)
```

---

userdb	DDname of the existing database file (as specified by the FILE parameter)
userdb dbfile fm	File identifier of the existing database file
xfile	DDname of the expanded database file (as specified by the FILEOUT parameter)
xfile dbfile fm	File identifier of the expanded database file
expanded-database-size	DCB information for the expanded database file

---

### 8.3.8 EXTRACT JOURNAL

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 DISK archive ft fm
```

```
FILEDEF extract DISK extj (RECFM VB BLKSIZE bbbbb XTENT nn
```

```
FILEDEF SORTMSG PRINTER
```

```
FILEDEF SORTWK01 DISK fn ft fm
```

Add additional SORTWKnn files as necessary

---

archive ft fm	File identifier of the complete archive or journal file. It can be on tape or disk and can be concatenated.
extract	DDname of the extract journal file. If you do not specify one, it defaults to SYS002.
bbbbbb	Block size of the extract journal file. Specify a size at least as large as the largest block size on the journal or archive files being processed.

---

### 8.3.9 FASTLOAD

#### FILEDEF commands for the batch command facility (CMS)

```

FILEDEF SYS001 DISK tblu sortparm fm
FILEDEF SYS002 DISK fmpgm file fm
FILEDEF SYS004 TAP1 SL VOLID dbl004 (RECFM VB LRECL rr04 BLKSIZE bb04
FILEDEF SYS005 TAP2 SL VOLID dbl005 (RECFM VB LRECL rr05 BLKSIZE bb05
FILEDEF SYS009 TAP3 SL VOLID dbl009 (RECFM VB LRECL rr09 BLKSIZE bb09
FILEDEF SYS010 TAP4 SL VOLID dbl010 (RECFM VB LRECL rr10 BLKSIZE bb10
FILEDEF SYS011 TAP5 SL VOLID dbl011 (RECFM VB LRECL rr11 BLKSIZE bb11
FILEDEF SYSPCH DISK sortld file fm (RECFM F LRECL 80 BLKSIZE 80
FILEDEF RELDCTL DISK reldctl ctl fm (RECFM FB LRECL 60 BLKSIZE bbbctl
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk file fm

```

Additional sort files, as required

<u>tblu sortparm fm</u>	File identifier of the file containing sort parameters created by IDMSTBLU
<u>fmpgm file fm</u>	File identifier of the file put out by the format program
<u>dbl004</u>	Volume serial number of the intermediate work file containing the output from SORT1
<u>rr04</u>	Record size of the SYS004 file; should be the same as the record size for SYS001
<u>bb04</u>	Block size of the SYS004 intermediate work file; should be the same as the block size for SYS001
<u>dbl005</u>	Volume serial number of the intermediate work file containing a control record and set membership information from IDMSDBL2
<u>rr05</u>	Record size of the SYS005 file; for sizing information, see 4.9, "FASTLOAD"
<u>bb05</u>	Block size of the SYS005 intermediate work file; must be at least rr05 plus four bytes
<u>dbl009</u>	Volume serial number of the intermediate work file containing the sorted contents of SYS005
<u>rr09</u>	Record size of the SYS009 file; should be the same as rr05
<u>bb09</u>	Block size of the SYS009 intermediate work file; should be the same as bb05
<u>dbl010</u>	Volume serial number of the intermediate work file containing pointer information from IDMSDBL3
<u>rr10</u>	Record size of the SYS010 file; for sizing information, see 4.9, "FASTLOAD"

---

bb10	Block size of the SYS010 intermediate work file; must be at least 60 bytes (the size of the control record plus four bytes)
dbl011	Volume serial number of the intermediate work file containing sorted pointer information from SORT4
rr11	Record size of the SYS011 file; should be the same as rr10
bb11	Block size of the SYS011 intermediate work file; should be the same as bb10
sortld file fm	File identifier of the SYSPCH file
reldctl ctl fm	File identifier of the RELDCTL file
bbbctl	Block size of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760
sortwk file fm	File identifier of the SORTWK file

---

### 8.3.10 FIX ARCHIVE

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 TAP1 SL VOLID tjrnold (RECFM VB LRECL 111 BLKSIZE bbbb)
FILEDEF SYS002 TAP2 SL VOLID tjrnlfix (RECFM VB LRECL 111 BLKSIZE bbbb)
```

---

tjrnold	Volume serial number of the input tape journal file
tjrnlfix	Volume serial number of the output tape journal file
bbbb	Block size of the tape journal file as defined in the DMCL

---

### 8.3.11 FIX PAGE

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
```

Additional file assignments, as required

---

userdb	DDname of the user database file
userdb dbfile fm	File identifier of the user database file

---

## 8.3.12 FORMAT

### FILEDEF commands for the batch command facility (CMS)

#### To format a new or existing database file

```
FILEDEF userdb DISK userdb dbfile fm (RECFM F LRECL 111 XTENT nnn)
```

Additional file assignments, as required

#### To format a new or existing disk journal file

```
FILEDEF jljrn1 DISK jljrn1 jrn1file fm (RECFM F LRECL 111 XTENT nnn)
```

Additional journal file assignments, as required

userdb	DDname of the user database file
userdb dbfile fm	File identifier of the user database file
jljrn1	DDname of the first disk journal file, as defined in the DMCL
jljrn1 jrn1file fm	DDname of the first disk journal file

## 8.3.13 INSTALL STAMPS

### Local Mode FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
FILEDEF userdict DISK userdict dbfile fm
FILEDEF sysjrn1 DISK sysjrn1 jrn1file fm
```

userdb	DDname of the user database file
userdb dbfile fm	File identifier of the user database file
userdict	DDname of the database file containing the dictionary with the table definitions
userdict dbfile fm	File identifier of the database file containing the dictionary with the table definitions
sysjrn1	DDname of the tape journal file
sysjrn1 jrn1file fm	File identifier of the tape journal file

**Central Version:** To execute INSTALL STAMPS under the central version, modify the JCL shown above, as follows:

- Remove the USERDICT and SYSJRNL filedef statements
- Insert the following statement after the USERDB filedef statement:

```
FILEDEF SYSCTL DISK sysctl file fm
```

---

sysctl file fm	File identifier of the SYSCTL file
----------------	------------------------------------

---

## 8.3.14 LOAD

### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdict DISK userdict dbfile fm
FILEDEF userdb DISK userdb dbfile fm
FILEDEF SYS001 DISK user input fm
FILEDEF SYS002 TAP1 SL VOLID loadin (RECFM VB LRECL rrin BLKSIZE bbin
FILEDEF SYS003 TAP2 SL VOLID lodout (RECFM VB LRECL rrout BLKSIZE bbout
FILEDEF SYSPCH DISK sortload file fm (RECFM F LRECL 80 BLKSIZE 80
```

The SORTMSG and SORTWKnn files are only needed when performing a complete LOAD.

```
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwknn file fm
```

Additional sort files, as required

---

userdict	DDname of the database file containing the dictionary with the table definitions
userdict dbfile fm	File identifier of the database file containing the dictionary with the table definitions
userdb	DDname of the user database file
userdb dbfile fm	File identifier of the user database file
user input fm	File identifier of the input file
loadin*	Volume serial number of the input SYS002 file; for sizing information see 4.14, "LOAD"
rrin	Record size of the SYS002 file
bbin	Block size of the SYS002 file
lodout*	Volume serial number of the output SYS003 file; for sizing information see 4.14, "LOAD"
rrout	Record size of the SYS003 file
bbout	Block size of the SYS003 file
sortload file fm	File identifier of the SYSPCH file
sortwknn file fm	File identifier of the SORTWKnn file

---

**Note:** \* When running a complete LOAD, SYS002 and SYS003 must point to the SAME intermediate file. When running a stepped LOAD, SYS002 and SYS003 must point to a DIFFERENT intermediate file.

**Note:** When running a complete LOAD, you must preallocate the file referenced by SYS002 and SYS003 and do not use a temporary data set.

### 8.3.15 MAINTAIN INDEX

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK usrolddb dbfile fm
FILEDEF SYS003 TAP1 SL VOLID db1003 (RECFM VB LRECL rr03 BLKSIZE bb03)
FILEDEF SYS004 TAP2 SL VOLID db1004 (RECFM VB LRECL rr04 BLKSIZE bb04)
FILEDEF SYS005 TAP3 SL VOLID db1005 (RECFM VB LRECL rr05 BLKSIZE bb05)
FILEDEF SYS006 TAP4 SL VOLID db1006 (RECFM VB LRECL rr06 BLKSIZE bb06)
FILEDEF SYSPCH DISK sort file fm (RECFM F LRECL 80 BLKSIZE 80)
FILEDEF RELDCTL DISK reldctl ctl fm (RECFM FB LRECL 60 BLKSIZE bbbctl)
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk file fm
```

Additional sort files, as required

userdb	DDname of the existing database file
usrolddb dbfile fm	File identifier of the existing database file
db1003	Volume serial number of the intermediate work file containing index descriptors from IDSMTABX
rr03	Record size of the SYS003 file; see Chapter 4, "MAINTAIN INDEX," for sizing information
bb03	Block size of the SYS003 intermediate work file
db1004	Volume serial number of the intermediate work file containing the output from SORT3
rr04	Record size of the SYS004 file; should be the same as rr03
bb04	Block size of the SYS004 intermediate work file; should be the same as bb03
db1005	Volume serial number of the intermediate work file containing pointers for user owned index sets from IDMSDBL3
rr05	Record size of the SYS005 file; see Chapter 4, "MAINTAIN INDEX," for sizing information
bb05	Block size of the SYS005 intermediate work file
db1006	Volume serial number of the intermediate work file containing sorted pointers for user owned index sets from SORT4
rr06	Record size of the SYS006 file; should be the same as rr05

---

bb06	Block size of the SYS006 intermediate work file; should be the same as bb05
sort file fm	File identifier of the SYSPCH file containing sort parameters from IDMSTABX and IDMSDBL3
reldctl ctl fm	File identifier of the RELDCTL file
bbbctl	Block size of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760
sortwk file fm	File identifier of the SORTWK file

---

### 8.3.16 MERGE ARCHIVE

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 TAP1 SL VOLID tjrnlold (RECFM F BLKSIZE bbbb
FILEDEF SYS002 TAP2 SL VOLID tjrnlfix (RECFM F BLKSIZE bbbb
FILEDEF JRN01 TAP3 SL VOLID tmrgold (RECFM F BLKSIZE bbbb
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk file fm
```

Additional sort files, as required

---

tjrnlold	Volume serial number of the input tape journal file
tjrnlfix	Volume serial number of the output tape journal file
tmrgold	Volume serial number of the input tape merged journal file. If none specify DUMMY.
bbbb	Block size of the tape journal file as defined in the DMCL

---

### 8.3.17 PRINT INDEX

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF sysjrn1 DUMMY
FILEDEF userdb DISK userdb dbfile fm
```

---

sysjrn1	DDname of the dummy journal file
userdb	DDname of the database file
userdb dbfile fm	File identifier of the database file

---

## 8.3.18 PRINT JOURNAL

### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 DISK archjrn1 jnlfile fm
```

---

archjrn1	DDname of the archive journal file
archjrn1 jnlfile fm	File identifier of the archive journal file

---

## 8.3.19 PRINT LOG

### FILEDEF commands for the batch command facility (CMS)

To print from the DDLDCLOG area:

```
FI dlogdb DISK idms dlogdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
```

To print from the archive log file:

```
FI SYS001 DISK archive dbfile fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
FI dcmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
FILEDEF sysjrn1 DUMMY
```

---

dlogdb	DDname of the data dictionary log area
idms dlogdb fm	File identifier of the data dictionary log area
archive dbfile fm	File identifier of the archive log file
ppp	Page size of the database file
nnn	Number of pages in the database file
dcmsg	DDname of the data dictionary message area
idms dmsgdb fm	File identifier of the data dictionary message area
sysjrn1	DDname of the tape journal file defined in the DMCL module named in the IDMSNWKS subschema

---

## 8.3.20 PRINT PAGE

### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
```

Additional database file assignments, as required

---

userdb1	DDname of the first database file
user userdb1 f	File identifier of the first database file
userdb2	DDname of the second database file
user userdb2 f	File identifier of the second database file
pppp	Page size of the database file
nnnn	Number of pages in the database file

---

### 8.3.21 PRINT SPACE

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
```

Additional database file assignments, as required

---

userdb1	DDname of the first database file
user userdb1 f	File identifier of the first database file
userdb2	DDname of the second database file
user userdb2 f	File identifier of the second database file
pppp	Page size of the database file
nnnn	Number of pages in the database file

---

### 8.3.22 PUNCH

#### Local Mode FILEDEF commands for the batch command facility (CMS)

```
FI usercat DISK user ddlcat fm (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FI usercatl DISK user ddlcatl fm (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FI usercatx DISK user ddlcatx fm (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSPCH DISK punch fm
FILEDEF sysjrn1 DUMMY
```

Additional dummied journals as required

---

usercat	DDname of the DDLCAT area of the dictionary
user ddlcat fm	File identifier of the DDLCAT area of the dictionary
usercatl	DDname of the DDLCATLOD area of the dictionary
user ddlcatl fm	File identifier of the DDLCATL area of the dictionary

---

---

usercatx	DDname of the DDLCATX area of the dictionary
user ddlcax fm	File identifier of the DDLCATX area of the dictionary
sysjrn1	DDname of the tape journal file defined in the DMCL module named in the IDMSNWKS subschema
pppp	Page size of the database file
nnnn	Number of pages in the database file
punch fm	File identifier of the punch output

---

**Central Version:** To execute PUNCH under the central version, modify the JCL shown above, as follows:

- Remove the USERCAT, USERCATL, USERCATX, and SYSJRNL statements
- Insert the following statement:

```
FILEDEF SYSCTL DISK sysctl file fm
```

---

sysctl file fm	File identifier of the SYSCTL file
----------------	------------------------------------

---

### 8.3.23 RELOAD

#### FILEDEF commands for the batch command facility (CMS)

---

```
FILEDEF SYS001 DISK sortunld parms a
FILEDEF SYS002 DISK user dbl002 a
FILEDEF SYS003 DISK user dbl003 a
FILEDEF SYS004 tap4 SL VOLID nnnnnn (RECFM VB LRECL rrr004 BLKSIZE bbb004)
FILEDEF SYS005 tap5 SL VOLID nnnnnn (RECFM VB LRECL rrr005 BLKSIZE bbb005)
FILEDEF SYS006 tap6 SL VOLID nnnnnn (RECFM VB LRECL rrr006 BLKSIZE bbb006)
FILEDEF SYS007 tap7 SL VOLID nnnnnn (RECFM VB LRECL rrr007 BLKSIZE bbb007)
FILEDEF SYS008 tap8 SL VOLID nnnnnn (RECFM VB LRECL rrr008 BLKSIZE bbb008)
FILEDEF SYS009 tap9 SL VOLID nnnnnn (RECFM VB LRECL rrr009 BLKSIZE bbb009)
FILEDEF SYS010 tap10 SL VOLID nnnnnn (RECFM VB LRECL rrr010 BLKSIZE bbb010)
FILEDEF SYS011 tap11 SL VOLID nnnnnn (RECFM VB LRECL rrr011 BLKSIZE bbb011)
FILEDEF SYSPCH DISK sortreld parms a
FILEDEF reldctl DISK user reldctl a (RECFM f LRECL 60 BLKSIZE bbbctl)
FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
```

Additional database file assignments, as required

```
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk01 work a3
```

Additional sort files, as required

---

<i>sortunld</i> parms a	File identifier of the sort parameters created by UNLOAD
-------------------------	----------------------------------------------------------

---

---

<i>user dbl002 a</i>	File identifier of the file put out by UNLOAD as SYS001
<i>user dbl003 a</i>	File identifier of the file put out by UNLOAD as SYS003
<i>tap4</i>	Symbolic device name of the SYS004 file containing the output from SORT1
<i>rrr004</i>	Record size of the SYS004 file; should be the same as the record size for SYS002 used by UNLOAD
<i>bbb004</i>	Block size of the SYS004 intermediate work file; should be the same as the block size for SYS002 used by UNLOAD
<i>tap5</i>	Symbolic device name of the SYS005 file containing member descriptors for chained sets from IDMSDBL2
<i>rrr005</i>	Record size of the SYS005 file; for sizing information, see 4.23, "RELOAD"
<i>bbb005</i>	Block size of the SYS005 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
<i>tap6</i>	Symbolic device name of the SYS006 file containing member descriptors for user owned index sets from IDMSDBL2
<i>rrr006</i>	Record size of the SYS006 file; for sizing information, see 4.23, "RELOAD"
<i>bbb006</i>	Block size of the SYS006 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
<i>tap7</i>	Symbolic device name of the SYS007 file containing the output of SORT2
<i>rrr007</i>	Record size of the SYS007 file; should be the same as the larger of: <ul style="list-style-type: none"> <li>■ the record size for SYS003 used by UNLOAD</li> <li>■ the record size for SYS006</li> </ul>

---

---

<i>bbb007</i>	Block size of the SYS007 intermediate work file; must be at least as large as the larger of: <ul style="list-style-type: none"><li>■ the block size for SYS003 used by UNLOAD</li><li>■ the block size for SYS006</li></ul>
<i>tap8</i>	Symbolic device name of the SYS008 file containing the reformatted index information from IDMSDBLX
<i>rrr008</i>	Record size of the SYS008 file; should be the same as <i>rrr007</i>
<i>bbb008</i>	Block size of the SYS008 intermediate work file; should be the same as <i>bbb007</i>
<i>tap9</i>	Symbolic device name of the SYS009 file containing the sorted index set descriptors from SORT3
<i>rrr009</i>	Record size of the SYS009 file; should be the same as the larger of: <ul style="list-style-type: none"><li>■ the record size for SYS005</li><li>■ the record size for SYS008</li></ul>
<i>bbb009</i>	Block size of the SYS009 intermediate work file; must be at least as large as the larger of: <ul style="list-style-type: none"><li>■ the block size for SYS003 used by UNLOAD</li><li>■ the block size for SYS006</li></ul>
<i>tap10</i>	Symbolic device name of the SYS010 file containing prefix pointer information from IDMSDBL3
<i>rrr010</i>	Record size of the SYS010 file; for sizing information, see 4.23, "RELOAD"
<i>bbb010</i>	Block size of the SYS010 intermediate work file; must be at least 44 bytes (the size of a pointer descriptor plus four bytes)
<i>tap11</i>	Symbolic device name of the SYS011 file containing the sorted prefix pointer information from SORT4
<i>rrr011</i>	Record size of the SYS011 file; should be the same as <i>rrr010</i>
<i>bbb011</i>	Block size of the SYS011 intermediate work file; should be the same as <i>bbb010</i>

---

---

<i>user reldctl a</i>	File identifier of the RELDCTL file containing control information created during the UNLOAD
<i>sortreld parms a</i>	File identifier of the SYSPCH file
<i>bbbctl</i>	Blocksize of the RELDCTL file. It must be a multiple of 60 with a maximum size of 32,760.
<i>sortwk01 work a3</i>	File identifier of temporary sort work file (if needed)
<i>userdb1</i>	DDname of the first database file
<i>user userdb1 f</i>	File identifier of the first database file

---

### 8.3.24 RESTORE

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYS001 TAP1 SL VOLID vvvvvv (RECFM VB LRECL llll BLKSIZE bbbb
```

Additional database file assignments, as required

---

<i>userdb1</i>	DDname of the first database file
<i>user userdb1 f</i>	File identifier of the first database file
<i>userdb2</i>	DDname of the second database file
<i>user userdb2 f</i>	File identifier of the second database file
<i>vvvvvv</i>	Volume id of the tape file
<i>llll</i>	Record length of the tape backup file
<i>bbbb</i>	Block size of the tape backup file

---

### 8.3.25 RESTRUCTURE CONNECT

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK user userdb f FILEDEF SYS001 DISK idms spill f FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk01 work a3
```

---

<i>userdb</i>	DDname of the database file
<i>user userdb f</i>	File identifier of the database file
<i>idms spill f</i>	File identifier of the spill file

---

---

bbbb	Size of the spill file should be a multiple of 32 with a maximum size of 32,736.
sortwk01 work a3	File identifier of temporary sort work file (if needed)

---

### 8.3.26 RESTRUCTURE SEGMENT

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYS001 DISK idms spill f (RECFM FB LRECL bbbb BLKSIZE bbbb XTENT nnnn
```

---

userdb1	DDname of the first database file
user userdb1 f	File identifier of the first database file
userdb2	DDname of the second database file
user userdb2 f	File identifier of the second database file
idms spill f	File identifier of the spill file.
bbbb	Size of the spill file should be a multiple of 32 with a maximum size of 32,736.

---

### 8.3.27 ROLLBACK

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 DISK tapn SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbbb
FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
```

Additional database file assignments, as required

If using the SORT option add these statements:

```
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk01 disk a3
```

Additional sort files, as required

---

&sysjrnl	DDname of the complete archive or tape journal file
tapn	Symbolic device name of the archive or tape journal file
userdb	DDname of the user database file
user userdb f	File identifier of the user database file

---

## 8.3.28 ROLLFORWARD

### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 DISK tapn SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbbb
FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
```

Additional database file assignments, as required

If recovering from a journal extract file:

```
FILEDEF extract DISK jrn1.ext ft fm
```

If using the SORT option or processing a journal extract file, add these statements:

```
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk01 disk a3
```

Add additional sort files, as required

sysjrn1	DDname of the complete archive or tape journal file
tapn	Symbolic device name of the archive or tape journal file
userdb	DDname of the user database file
user userdb f	File identifier of the user database file
extract	DDname of the extract journal file. If you don't specify, it defaults to SYS002.
jrn1.ext ft fm	File identifier of the journal extract file

## 8.3.29 TUNE INDEX

### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
```

userdb	DDname of the database file
userdb dbfile fm	File identifier of the database file

## 8.3.30 UNLOCK

### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
```

userdb	DDname of the database file
--------	-----------------------------

---

user userdb f	File identifier of the database file
---------------	--------------------------------------

---

### 8.3.31 UNLOAD

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK user olddb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
```

Additional database file assignments, as required

```
FILEDEF sysjrn1 DUMMY
FILEDEF SYS002 tap2 SL VOLID nnnnnn (RECFM VB LRECL rrr002 BLKSIZE bbb002
FILEDEF SYS003 tap2 SL VOLID nnnnnn (RECFM VB LRECL rrr003 BLKSIZE bbb003
FILEDEF SYSPCH DISK sort unld a (RECFM F BLKSIZE 80
FILEDEF RELDCTL DISK user reldctl a (RECFM FB LRECL 60 BLKSIZE bbbctl]
```

---

userdb	DDname of the existing database file
user olddb a	File identifier of the existing database file
sysjrn1	DDname of the dummy journal file
tap2	Symbolic device name of the SYS002 output file
rrr002	Record size of the SYS002 output file; for sizing information, see 4.30, "UNLOAD"
bbb002	Block size of the SYS002 output file; must be at least the size of the largest database record plus 4 bytes
tap3	Symbolic device name of the SYS003 output file.
rrr003	Record size of the SYS003 output file; for sizing information, see 4.30, "UNLOAD"
bbb003	Block size of the SYS003 output file
sort unload a	Data set name of the sort parameters created by UNLOAD
disk	Symbolic device name of the sort parameter file
bbbctl	Size of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760.

---

### 8.3.32 UPDATE STATISTICS

#### FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK user olddb f
```

Additional database file assignments, as required

```
FILEDEF ddlcat DISK sysdict ddlcat f (RECFM F LRECLppp BLKSIZE ppp XTENT nnn)
FILEDEF ddlxcatDISK sysdict ddlxcat f(RECFM F LRECLppp BLKSIZE ppp XTENT nnn)
FILEDEF j1jrn1DISK j1jrn1 jrn1file e (RECFM F LRECLppp BLKSIZE ppp XTENT nnn)
```

Additional journal file assignments, as required

<u>userdb</u>	DDname of the database file
<u>user userdb f</u>	File identifier of the database file
<u>ddlcat</u>	DDname of the database file containing the area of the system dictionary with the table definitions
<u>sysdict ddlcat f</u>	File identifier of the database file containing the area of the system dictionary with the table definitions
<u>ddlxcat</u>	DDname of the database file containing the area of the system dictionary with indexes
<u>sysdict ddlxcat f</u>	File identifier of the database file containing the area of the system dictionary with indexes
<u>j1jrn1</u>	DDname of the first journal file, as defined in the DMCL
<u>j1jrn1 jrn1file e</u>	File identifier of the first journal file

### 8.3.33 VALIDATE

#### **FILEDEF commands for the batch command facility (CMS)**

```
FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE ppp XTENT nnn)
FILEDEF SYS002 tap2 SL VOLID nnnnnn (RECFM VB LRECL rrrin BLKSIZE bbbin)
FILEDEF SYS003 tap3 SL VOLID nnnnnn (RECFM VB LRECL rrrout BLKSIZE bbbout)
FILEDEF SYSPCH DISK sort build fm
```

The SORTMSG and SORTWKnn files are only needed when performing a complete VALIDATE.

```
FILEDEF SORTMSG PRINTER
FILEDEF sortwknn DISK sort worknn a3
```

Additional sort files, as required

<u>userdb</u>	DDname of the database file
<u>user userdb f</u>	File identifier of the database file
<u>tap2*</u>	Symbolic device name of the input SYS002 file; for sizing information see 4.33, "VALIDATE"

### 8.3 Utility statements

---

rrrin	Record size of the SYS002 file
bbbin	Block size of the SYS002 file
tap3*	Symbolic device name of the output SYS003 file; for sizing information see 4.33, "VALIDATE"
rrrout	Record size of the SYS003 file
bbbout	Block size of the SYS003 file
sort build fm	File identifier of the SYSPCH file
sortwknn file fm	File identifier of the SORTWKnn file

**Note:** \* When running a complete VALIDATE, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to a *different* intermediate file.

## 8.4 Utility programs

### 8.4.1 IDMSDBAN

#### IDMSDBAN (CMS)

---

```
FILEDEF userdb DISK user userdb_b (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
```

Additional database file assignments, as required

```
FILEDEF SYS002 TAP1 SL VOLID nnnnnn (RECFM VB LRECL 512 BLKSIZE 9000)
FILEDEF SYSOUT PRINTER
FILEDEF SYSLST PRINTER
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK dban input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSDBN1
```

```
FILEDEF SYS001 TAP1 SL VOLID nnnnnn (RECFM VB LRECL 512 BLKSIZE 9000)
FILEDEF SYS002 DISK chain work a3, (RECFM VB LRECL 512 BLKSIZE 9000)
FILEDEF SYSOUT PRINTER
FILEDEF SYSLST PRINTER
FILEDEF SYSIDMS DISK sysidms input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSDBN2
```

---

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

<u>userdb</u>	DDname of the user database file
<u>user userdb b</u>	File identifier of the user database file
<u>pppp</u>	Page size of the user database file
<u>nnnn</u>	Number of pages in the user database file
<u>nnnnnn</u>	Volume serial number of the chain file generated by IDMSDBN1
<u>SYSIDMS</u>	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.  ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .
<u>sysidms input a</u>	File identifier of the file containing SYSIDMS parameters if applicable
<u>dban input a</u>	File identifier of the file containing IDMSDBAN input parameters

---

---

dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	Filename of the CA-IDMS/DB load library containing the subschema and DMCL load modules
chain work a3	File identifier of the intermediate chain file used by IDMSDBN2

---

**Note:** IDMSDBAN requires the presence of an external sort package (other than the CMS SORT command) that can be loaded dynamically.

## 8.4.2 IDMSDIRL

### Local mode IDMSDIRL (CMS)

```
FILEDEF sysjrn1 TAP1 SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbbb
FILEDEF dictdb DISK idms dictdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSLST PRINTER
FILEDEF SYS001 TAP2 SL VOLID nnndr1 (RECFM VB LRECL 600 BLKSIZE 5992
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK dir1 input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSDIRL
```

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

sysjrn1	DDname of the tape journal file, as defined in the DMCL module
nnnnnn	Volume serial number of the tape journal file
lll	Record length of the tape journal file
bbbb	Block size of the tape journal file
dictdb	DDname of the data dictionary file
idms dictdb f	File identifier of the data dictionary file
pppp	Page size of the data dictionary file
nnnn	Number of pages in the data dictionary file
nnndr1	Volume serial number of the IDMSDIRL input file (on the installation tape)
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.  ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

---

sysidms input a	File identifier of the file containing SYSIDMS parameters if applicable
dirl input a	File identifier of the file containing the IDMSDIRL input parameters
dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	Filename of the CA-IDMS/DB load library

---

**Central version:** To execute IDMSDIRL under the central version, modify the CMS commands shown above, as follows:

- Remove the SYSJRNL and DICTDB FILEDEF commands.
- Identify the DC/UCF system to be accessed by IDMSDIRL using the CVMACH and CVNUM SYSIDMS parameters.

►►For more information on the SYSIDMS parameter file, see *CA-IDMS Database Administration*.

### 8.4.3 IDMSLOOK

#### IDMSLOOK (CMS)

```
FILEDEF idmslib DISK idmslib loadlib a
FILEDEF SYSLST PRINTER
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT look input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSLOOK
```

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	DDname for statement containing the CA-IDMS load library containing CA-IDMS modules
idmslib loadlib a	File identifier of the CA-IDMS load library containing CA-IDMS modules
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.  ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

---

sysidms input a	File identifier of the file containing SYSIDMS parameters if applicable
look input a	File identifier for the file containing the IDMSLOOK input parameters

---

## 8.4.4 IDMSRPTS

### Local mode IDMSRPTS (CMS)

```

FILEDEF sysjrn1 TAP1 SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbbb
FILEDEF dictdb DISK idms dictdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSOUT PRINTER
FILEDEF SYSLST PRINTER
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK rpts input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSRPTS

```

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

sysjrn1	DDname of the tape journal file, as defined in the DMCL module
nnnnnn	Volume serial number of the tape journal file
lll	Record length of the tape journal file
bbbb	Block size of the tape journal file, as defined in the DMCL module
dictdb	DDname of the data dictionary file
idms dictdb f	File identifier of the data dictionary file
pppp	Page size of the data dictionary file
nnnn	Number of pages in the data dictionary file
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.  ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .
sysidms input a	File identifier of the file containing SYSIDMS parameters if applicable
rpts input a	File identifier of the file containing the IDMSRPTS input parameters
dbalib	Filename of the load library containing the DMCL and database name table load modules

---

---

idmslib	Filename of the CA-IDMS/DB load library
---------	-----------------------------------------

---

**Central version:** To execute IDMSRPTS under the central version, modify the CMS commands shown above, as follows:

- Remove the SYSJRNL and DICTDB FILEDEF commands.
- Identify the DC/UCF system to be accessed by IDMSRPTS using the CVMACH and CVNUM SYSIDMS parameters.

►►For more information on the SYSIDMS parameter file, see *CA-IDMS Database Administration*.

## 8.4.5 IDMSRSTC

### Local mode IDMSRSTC (CMS)

```
FILEDEF sysjrn1 TAP1 SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbbb
FILEDEF dictdb DISK idms dictdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSLST PRINTER
FILEDEF SYSPCH DISK rsttmac ASSEMBLE A
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK rstc input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSRSTC
```

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

sysjrn1	DDname of the tape journal file
nnnnnn	Volume serial number of the tape journal file
111	Record length of the tape journal file
bbbb	Block size of the tape journal file
dictdb	DDname of the data dictionary file
idms dictdb f	File identifier of the data dictionary file
pppp	Page size of the data dictionary file
nnnn	Number of pages in the data dictionary file
rsttmac ASSEMBLE A	File identifier of the card-image file containing the IDMSRSTT macro statements
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters.
	►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

---

---

sysidms input a	File identifier of the file containing SYSIDMS parameters if applicable
rstc input a	File identifier of the file containing the IDMSRSTC input parameters
dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	Filename of the CA-IDMS/DB load library

---

**Central version:** To execute IDMSRSTC under the central version, modify the CMS commands shown above, as follows:

- Remove the SYSJRNL and DICTDB FILEDEF commands.
- Identify the DC/UCF system to be accessed by IDMSRSTC using the CVMACH and CVNUM SYSIDMS parameters.

## 8.4.6 IDMSRSTT

### Assemble and link an IDMSRSTT module

```
GLOBAL MACLIB IDMSLIB
FILEDEF TEXT DISK idmsrstt TEXT A
ASSEMBLE rsttmac
TXTLIB ADD userlib idmsrstt
```

---

idmsrstt	Module name of the IDMSRSTT table
rsttmac	Filename of the card-image file containing the IDMSRSTT macro statement
userlib	Filename of the user text library

---

►►For more information on the SYSIDMS parameter file, see *CA-IDMS Database Administration*.

## Chapter 9. BS2000/OSD JCL

---

9.1	About this chapter	9-3
9.2	General JCL considerations	9-4
9.3	=COPY Facility	9-5
9.4	CA-IDMS Batch Command Facility	9-6
9.5	Utility statements	9-8
9.5.1	ARCHIVE JOURNAL	9-8
9.5.2	ARCHIVE LOG	9-8
9.5.3	BACKUP	9-9
9.5.4	BUILD	9-9
9.5.5	CLEANUP	9-10
9.5.6	CONVERT PAGE	9-10
9.5.7	EXPAND PAGE	9-10
9.5.8	EXTRACT JOURNAL	9-11
9.5.9	FASTLOAD	9-11
9.5.10	FIX ARCHIVE	9-13
9.5.11	FIX PAGE	9-14
9.5.12	FORMAT	9-14
9.5.13	INSTALL STAMPS	9-15
9.5.14	LOAD	9-16
9.5.15	MAINTAIN INDEX	9-17
9.5.16	MERGE ARCHIVE	9-18
9.5.17	PRINT INDEX	9-18
9.5.18	PRINT JOURNAL	9-19
9.5.19	PRINT LOG	9-19
9.5.20	PRINT PAGE	9-19
9.5.21	PRINT SPACE	9-20
9.5.22	PUNCH	9-20
9.5.23	RELOAD	9-21
9.5.24	RESTORE	9-23
9.5.25	RESTRUCTURE CONNECT	9-24
9.5.26	RESTRUCTURE SEGMENT	9-24
9.5.27	ROLLBACK	9-25
9.5.28	ROLLFORWARD	9-25
9.5.29	TUNE INDEX	9-27
9.5.30	UNLOCK	9-27
9.5.31	UNLOAD	9-27
9.5.32	UPDATE STATISTICS	9-28
9.5.33	VALIDATE	9-29
9.6	Utility programs	9-30
9.6.1	IDMSDBAN	9-30
9.6.2	IDMSDIRL	9-31
9.6.3	IDMSLOOK	9-32
9.6.4	IDMSRPTS	9-33
9.6.5	IDMSRSTC	9-34
9.6.6	IDMSRSTT	9-35



## 9.1 About this chapter

This chapter includes sample JCL to run the CA-IDMS utilities. A description of the =COPY facility is included as it can be used to submit utility statements and parameters.

## 9.2 General JCL considerations

In many cases, a utility works with input or output files that can reside either on disk, tape, or cartridge. To avoid needless duplication of JCL and descriptions, these parameters are used in a generic way:

**device-dependent parameters:** these parameters of the CREATE-FILE statement depend on the medium on which the file resides.

For a file on cartridge or tape, specify:

```
SUP=*TAPE(VOL=nnnnnn,DEV-TYPE=devtyp)
```

For a file on a public disk, specify:

```
SUP=*PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
```

For a file on a private disk, specify:

```
SUP=*PRIVATE-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary),-  
VOLUME=diskvol,DEV-TYPE=diskdev)
```

**nnnnnn:** volume serial name for the file.

**tape:** device type on which nnnnnn is to be mounted.

**primary:** primary space allocation for the file.

**secondary:** secondary space allocation for the file.

**diskvol:** volume serial name of the private disk.

**diskdev:** device type of the private disk.

## 9.3 =COPY Facility

**Purpose:** You can copy source code into the CA-IDMS Batch Command Facility (IDMSBCF) from a source statement library by means of the =COPY facility.

### Syntax

▶— =COPY IDMS library-linkname.member-name —▶

### Parameters

#### library-linkname

Specifies the linkname of the source library.

#### member-name

Specifies the name of the library member containing the source to be copied.

## 9.4 CA-IDMS Batch Command Facility

Listed below is sample BS2000/OSD JCL to execute the CA-IDMS Batch Command Facility.

When using the IDMSBCF program to execute a utility statement, code these statements along with the required statements for each of the utilities.

The file assignments for each utility statement and program are presented on subsequent pages in this chapter.

For a complete description of the CA-IDMS Command Facility, see *CA-IDMS Command Facility*.

### Local mode: IDMSBCF (BS2000/OSD)

```
/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
```

Insert file assignments required by the utility statements

```
/ADD-FILE-LINK L-NAME=dcmsg,F-NAME=idms.sysmsg.ddldcmsg,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysjrn1,F-NAME=idms.sysjrn1
/ADD-FILE-LINK L-NAME=SYSIDMS,F-NAME=idms.sysidms
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSBCF,LIB=idms.loadlib,RUN-MODE=*ADV)
```

Insert utility statements here

**Important:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

**Note:** File assignments may be omitted for database files if the file information is specified in the file definition.

idms.dba.loadlib	Filename of the CA-IDMS/DB load library containing DMCL and database name table load modules
idms.loadlib	Filename of the CA-IDMS load library containing executable modules
dcmsg	Linkname of the system message area (DDLDCMSG)
idms.sysmsg.ddldcmsg	Filename of the system message area (DDLDCMSG)
idms.sysidms	Filename of the CA-IDMS SYSIDMS parameter file For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .
sysjrn1	Linkname of the tape journal file
idms.tapejrn1	Filename of the tape journal file

**Central version: IDMSBCF (BS2000/OSD)**

```

/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

```

Insert file assignments required by the utility statements

```

/ADD-FILE-LINK L-NAME=dcmsg,F-NAME=idms.sysmsg.ddldcmsg,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysctl,F-NAME=idms.sysctl,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysjrn1,F-NAME=*DUMMY
/ADD-FILE-LINK L-NAME=SYSIDMS,F-NAME=idms.sysidms
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSBCF,LIB=idms.loadlib,RUN-MODE=*ADV)

```

Insert utility statements here

dcmsg	Linkname of the system message area (DDLDCMSG)
idms.sysmsg.ddldcmsg	Filename of the system message area (DDLDCMSG)
sysctl	Name of the SYSCTL file
idms.sysctl	ID of the SYSCTL file

## 9.5 Utility statements

### 9.5.1 ARCHIVE JOURNAL

**/Additional JCL statements for the batch command facility (BS2000/OSD)**

```
/ADD-FILE-LINK L-NAME=j1jrn1,F-NAME=idms.j1jrn1
/ADD-FILE-LINK L-NAME=j2jrn1,F-NAME=idms.j2jrn1
```

Insert additional journal file assignments as required

```
/CREATE-FILE F-NAME=idms.archive,SUPPRESS-ERRORS=*FILE-EXISTING, -
/      device-dependent-parameters
/ADD-FILE-LINK L-NAME=archjrn1,F-NAME=idms.archive
```

j1jrn1	Linkname of the first disk journal file
idms.j1jrn1	Filename of the first disk journal file
j2jrn1	Linkname of the second disk journal file
idms.j2jrn1	Filename of the second disk journal file
archjrn1	Linkname of the tape archive file as defined in the DMCL
idms.archive	Filename of the tape archive file

### 9.5.2 ARCHIVE LOG

**/Additional JCL statements for the batch command facility (BS2000/OSD)**

```
/ADD-FILE-LINK L-NAME=dlogdb,F-NAME=idms.dlogdb,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=dmsgdb,F-NAME=idms.dmsgdb,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysjrn1,F-NAME=*DUMMY
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=idms.archive
```

dlogdb	Linkname of the data dictionary log area
idms.dlogdb	Filename of the data dictionary log area
dmsgdb	Linkname of the data dictionary message area
idms.dmsgdb	Filename of the data dictionary message area
sysjrn1	Linkname of the tape journal file defined in the DMCL module
idms.archive	Filename of the archive log file

### 9.5.3 BACKUP

#### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional file assignments, as required

```
/CREATE-FILE F-NAME=user.bkpfile,SUPPRESS-ERRORS=*FILE-EXISTING, -  
/ device-dependent-parameters  
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=user.bkpfile
```

userdb	Linkname of the database file
user.userdb	Filename of the database file
user.bkpfile	Filename of the tape backup file

### 9.5.4 BUILD

#### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES  
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=user.load  
/ADD-FILE-LINK L-NAME=SYS003,F-NAME=user.build  
/ASSIGN-SYSOPT TO=&&sortbuild
```

When performing a complete BUILD, add:

```
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -  
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))  
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

userdb	Linkname of the database file
user.userdb	Filename of the database file
user.load *	Filename of the input (SYS002) file; for sizing information see discussion of SYS003 in 4.14, "LOAD"
user.build *	Filename of the output (SYS003) file; for sizing information see 4.4, "BUILD"
&&sortbuild	Filename of the SYSPCH file

**Note:** \* When running a complete BUILD, SYS002 and SYS003 must point to the same intermediate file. When running a stepped BUILD, SYS002 and SYS003 must point to a different intermediate file.

## 9.5.5 CLEANUP

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/ADD-FILE-LINK L-NAME=sysjrn],F-NAME=idms.tapejrn]
```

userdb	Linkname of the user database file
user.userdb	Filename of the user database file
sysjrn]	Linkname of the tape journal file
idms.tapejrn]	Filename of the tape journal file

## 9.5.6 CONVERT PAGE

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=oldfile,F-NAME=user.oldfile
```

```
/ADD-FILE-LINK L-NAME=newfile,F-NAME=user.newfile
```

Additional database file assignments as required

oldfile	Linkname of the database file to convert
user.oldfile	Filename of the database file to convert
newfile	Linkname of the target database file
user.newfile	Filename of the target database file

## 9.5.7 EXPAND PAGE

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional existing database file assignments, as required

```
/CREATE-FILE F-NAME=user.xbase,SUPPRESS-ERRORS=*FILE-EXISTING, -
/SUP=*PUB-DISK(SPACE=RELA(PRIM-ALLOC=expanded-database-size,SEC-ALLOC=0))
/ADD-FILE-LINK L-NAME=xfile,F-NAME=user.xbase
```

Additional expanded database file assignments, as required

---

userdb	Linkname of the existing database file
user.userdb	Filename of the existing database file
xfile	Linkname of the expanded database file
user.xbase	Filename of the expanded database file

---

## 9.5.8 EXTRACT JOURNAL

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/IMPORT-FILE SUP=*TAPE(VOL=nnnnn,DEV-TYPE=tape,FILE-NAME=archive)
```

Specify the `IMPORT-FILE` statement only when the file is a tape file that is no longer cataloged.

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=archive
/CREATE-FILE F-NAME=jrnl.extract,SUPPRESS-ERRORS=*FILE-EXISTING, -
/   device-dependent-parameters
/ADD-FILE-LINK L-NAME=extract,F-NAME=jrnl.extract,BUF-LEN=bbbb -
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/   SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add `SORTWKnn` files as necessary

---

archive	Filename of the complete archive or journal file. It can be on tape or disk and concatenated.
extract	Linkname of the extract journal file. If you do not specify one, it defaults to <code>SYS002</code> .
jrnl.extract	Filename for the extract journal file.
bbbb	Block size of the extract journal file. Specify a size at least as large as the largest block size on the journal or archive files being processed.
temp.sortwork	Filename of the temporary <code>SORT</code> work file.

---

## 9.5.9 FASTLOAD

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=&&sortunld
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=user.db1001
/CREATE-FILE F-NAME=user.db1004,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS004,F-NAME=user.db1004
/CREATE-FILE F-NAME=user.db1005,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS005,F-NAME=user.db1005
/CREATE-FILE F-NAME=user.db1009,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS009,F-NAME=user.db1009
/CREATE-FILE F-NAME=user.db1010,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS010,F-NAME=user.db1010
/CREATE-FILE F-NAME=user.db1011,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS011,F-NAME=user.db1011
/ADD-FILE-LINK L-NAME=RELDCTL,F-NAME=user.reldctl
/ASSIGN-SYSOPT TO=&&sortld
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

**Note:** Specification of the SPACE parameter is for performance only. For sizing information, see 4.9, “FASTLOAD.”

---

&&sortunld	Filename of the sort parameters created by IDMSTBLU
user.dbl001	Data set name of the file put out by the format program
user.dbl004	Data set name of the intermediate work file containing the output from SORT1
user.dbl005	Data set name of the intermediate work file containing a control record and set membership information from IDMSDBL2
user.dbl009	Data set name of the intermediate work file containing the sorted contents of SYS005
user.dbl010	Data set name of the intermediate work file containing pointer information from IDMSDBL3
user.dbl011	Data set name of the intermediate work file containing sorted pointer information from SORT4
user.reldctl	Filename of the reload control file containing control and set information
&&sortld	Data set name of the SYSPCH file
sortlib	Name of the SORT library. The assignment of TASKLIB is not required if FASTLOAD STEP stepname is specified.
temp.sortwork	Name of the sort workfile. This file should be allocated only if TASKLIB is assigned. The space allocation is for performance reasons only.
userdb	Linkname of the database file
user.userdb	Filename of the database file

---

### 9.5.10 FIX ARCHIVE

#### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/IMPORT-FILE SUP=*TAPE(VOL=nnnnnn,DEV-TYPE=tape,FILE-NAME=idms.tjrnld)
```

Specify the IMPORT-FILE statement only when the file is a tape file that is no longer cataloged.

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=idms.tjrnld
/CREATE-FILE F-NAME=jrn1.extract,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ device-dependent-parameters
/ADD-FILE-LINK L-NAME=extract,F-NAME=jrn1.extract,BUF-LEN=bbbb
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKn files as necessary

---

idms.tjrnlold	Filename of the input tape journal file
nnnnnn	Volume serial number of the tape journal file
tapein	Symbolic device name of the input tape journal file
idms.tjrnlfix	Filename of the output tape journal file
bbbb	Block size of the output tape journal file, as defined in the DMCL module
tapeout	Symbolic device name of the output tape journal file
temp.sortwork	Filename of the temporary SORT work file.

---

### 9.5.11 FIX PAGE

#### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

---

userdb	Linkname of the user file
user.userdb	Filename of the user file

---

### 9.5.12 FORMAT

#### Additional JCL statements for the batch command facility (BS2000/OSD)

##### For a new database file, allocate disk space for it:

```
/CREATE-FILE F-NAME=user.userdb,SUP=PUB-DISK(SPACE=RELA -  
/ (PRIM-ALLOC=primary,SEC-ALLOC=secondary))
```

##### To format an already allocated database file:

```
/ADD-FILE-LINK L-NAME=userdb,F-Name=user.userdb
```

##### For a new disk journal file, allocate disk space for it:

```
/CREATE-FILE F-NAME=idms.j1jrn1,SUP=PUB-DISK(SPACE=RELA -  
/ (PRIM-ALLOC=primary,SEC-ALLOC=secondary))
```

##### To format an already allocated disk journal file:

```
/ADD-FILE-LINK L-NAME=j1jrn1,F-NAME=idms.j1jrn1
```

Additional database and journal file assignments, as required

---

---

userdb	Linkname of the database file
user.userdb	Filename of the database file
j1jrnl	Linkname of the disk journal file
idms.j1jrnl	Filename of the disk journal file
disk	Symbolic device name of the file being formatted
nnnnnn	Volume serial number of the file being formatted
primary	Primary space allocation for the file being formatted; should be large enough to ensure contiguous disk blocks for the entire file
secondary	Secondary space allocation for the file being formatted; the suggested value is 0 to avoid dispersion of the file blocks on the disk

---

### 9.5.13 INSTALL STAMPS

**Local mode: Additional JCL statements for the batch command facility (BS2000/OSD)**

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=userdict,F-NAME=user.userdict,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysjrnl,F-NAME=idms.sysjrnl
```

---

userdb	Linkname of the database file
user.userdb	Filename of the database file
userdict	Linkname of the database file containing the dictionary with the table definitions
user.userdict	Filename of the database file containing the dictionary with the table definitions
sysjrnl	Linkname of the tape journal file
idms.sysjrnl	Filename of the tape journal file

---

**Central version:** To run INSTALL STAMPS under the central version, modify the JCL shown above as follows:

Replace the ADD-FILE-LINK statements with:

```
/ADD-FILE-LINK L-NAME=sysctl,F-NAME=idms.sysctl,SHARED-UPD=*YES
```

---

idms.sysctl	Filename of the SYSCTL file
-------------	-----------------------------

---

## 9.5.14 LOAD

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdict,F-NAME=user.userdict,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=user.input
/CREATE-FILE F-NAME=user.loadin,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=user.loadin
/CREATE-FILE F-NAME=user.loadout,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS003,F-NAME=user.loadout
/ASSIGN-SYSOPT TO=&&sortload
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

userdict	Linkname of the database file containing the dictionary with the table definitions
user.userdict	Filename of the database file containing the dictionary with the table definitions
userdb	Linkname of the database file being loaded
user.userdb	Filename of the database file being loaded
user.input	Filename of the input file
user.loadin *	Filename of the input SYS002 file; for sizing information see 4.14, "LOAD"
user.loadout *	Filename of the output SYS003 file; for sizing information see 4.14, "LOAD"
sortlib	Name of the SORT library. The assignment of TASKLIB is only required when performing a complete LOAD.
temp.sortwork	Name of the sort workfile. This file should be allocated only if TASKLIB is assigned. The space allocation is for performance reasons only.

**Note:** \* When running a complete LOAD, SYS002 and SYS003 must point to the same intermediate file. When running a stepped LOAD, SYS002 and SYS003 must point to a different intermediate file.

**Note:** When running a complete LOAD, you must preallocate the file referenced by SYS002 and SYS003 and the data sets cannot be temporary.

## 9.5.15 MAINTAIN INDEX

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/CREATE-FILE F-NAME=user.db1003,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS003,F-NAME=user.db1003
/CREATE-FILE F-NAME=user.db1004,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS004,F-NAME=user.db1004
/CREATE-FILE F-NAME=user.db1005,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS005,F-NAME=user.db1005
/CREATE-FILE F-NAME=user.db1006,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS006,F-NAME=user.db1006
/ADD-FILE-LINK L-NAME=RELDCTL,F-NAME=user.reldctl
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

```
/ASSIGN-SYSOPT TO=&&sort
```

**Note:** Specification of the SPACE parameter is for performance only. For sizing information, see 4.15, “MAINTAIN INDEX.”

userdb	Linkname of the existing database file
user.userdb	Filename of the existing database file
user.db1003	Filename of the intermediate work file containing index descriptors from IDMSTABX
user.db1004	Filename of the intermediate work file containing the output from SORT3
user.db1005	Filename of the intermediate work file containing pointers for user-owned index sets from SORT4
user.db1006	Filename of the intermediate work file containing sorted pointers for user-owned index sets from SORT4
user.reldctl	Filename of the reload control file containing control and set information

## 9.5.16 MERGE ARCHIVE

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/IMPORT-FILE SUP=*TAPE(VOL=nnnnnn,DEV-TYPE=tape,FILE-NAME=idms.tjrnlold)
/IMPORT-FILE SUP=*TAPE(VOL=nnnnnn,DEV-TYPE=tape,FILE-NAME=idms.tmrgold)
```

Specify the IMPORT-FILE statements only when the file is a tape file that is no longer cataloged.

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=idms.tjrnlold
/ADD-FILE-LINK L-NAME=JRN001,F-NAME=idms.tmrgold
/CREATE-FILE F-NAME=idms.tjrnlfix,SUPPRESS-ERRORS=*FILE-EXISTING, -
/   device-dependent-parameters
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=idms.tjrnlfix,BUF-LEN=bbbb
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/   SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

idms.tjrnlold	Filename of the input tape journal file
nnnnnn	Volume serial number of the tape journal file
idms.tmrgold	Filename of the input merged file. If none exists yet, specify *DUMMY.
idms.tjrnlfix	Filename of the output tape journal file
bbbb	Block size of the output tape journal file, as defined in the DMCL module

## 9.5.17 PRINT INDEX

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/ADD-FILE-LINK L-NAME=sysjrn1,F-NAME=*DUMMY
```

userdb	Linkname of the database file
user.userdb	Filename of the database file
sysjrn1	Linkname of the tape backup file

## 9.5.18 PRINT JOURNAL

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=idms.archjrn1
```

SYS001	Linkname of the archive journal file
idms.archjrn1	Filename of the archive journal file

## 9.5.19 PRINT LOG

### Additional JCL statements for the batch command facility (BS2000/OSD)

To print from the DDLDCLOG area:

```
/ADD-FILE-LINK L-NAME=dlogdb,F-NAME=idms.dlogdb,SHARED-UPD=*YES
```

To print from the archive log file:

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=idms.archive  
/ADD-FILE-LINK L-NAME=dmsgdb,F-NAME=idms.dmsgdb,SHARED-UPD=*YES  
/ADD-FILE-LINK L-NAME=sysjrn1,F-NAME=*DUMMY
```

dlogdb	Linkname of the data dictionary log area
idms.dlogdb	Filename of the data dictionary log area
idms.archive	Filename of the archive log file
dmsgdb	Linkname of the data dictionary message area
idms.dmsgdb	Filename of the data dictionary message area
sysjrn1	Linkname of the tape journal file defined in the DMCL module

## 9.5.20 PRINT PAGE

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb1,F-NAME=user.userdb1,SHARED-UPD=*YES  
/ADD-FILE-LINK L-NAME=userdb2,F-NAME=user.userdb2,SHARED-UPD=*YES
```

Additional database file assignments as required

---

userdb1	Linkname of the first database file
user.userdb1	Filename of the first database file
userdb2	Linkname of the second database file
user.userdb2	Filename of the second database file

---

### 9.5.21 PRINT SPACE

**/FILE commands for the batch command facility (BS2000/OSD)**

```
/ FILE LINK=userdb1,user.userdb1,SHARUPD=YES
/ FILE LINK=userdb2,user.userdb2,SHARUPD=YES
```

Additional database file assignments as required

---

userdb1	Linkname of the first database file
user.userdb1	Filename of the first database file
userdb2	Linkname of the second database file
user.userdb2	Filename of the second database file

---

### 9.5.22 PUNCH

**Local mode: Additional JCL statements for the batch command facility (BS2000/OSD)**

```
/ADD-FILE-LINK L-NAME=usercat,F-NAME=user.ddlcat,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=usercat1,F-NAME=user.ddlcat1,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=usercatx,F-NAME=user.ddlcatx,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysjrn1,F-NAME=*DUMMY
/ASSIGN-SYSOPT TO=#punch
```

**Note:** The resulting object needs to be put in the object library using the BS2KOBJM utility. The use of BS2KOBJM is described in *CA-IDMS Usage under BS2000/OSD*.

---

usercat	Linkname of database file containing the DDLCAT area of the dictionary
user.ddlcat	Filename of database file containing the DDLCAT area of the dictionary
usercatl	Linkname of database file containing the DDLCATLOD area of the dictionary
user.ddlcatl	Filename of the database file containing the DDLCATLOD area of the dictionary
usercatx	Linkname of database file containing the DDLCATX area of the dictionary
user.ddlcatx	Filename of the database file containing the DDLCATX area of the dictionary
sysjrnl	Linkname of the tape journal file defined in the DMCL module

---

**Central version:** To execute PUNCH under the central version, modify the JCL shown above as follows:

Replace the ADD-FILE-LINK statements with:

```
/ADD-FILE-LINK L-NAME=sysctl,F-NAME=idms.sysctl,SHARED-UPD=*YES
```

---

sysctl	Name of the SYSCTL file
idms.sysctl	ID of the SYSCTL file

---

### 9.5.23 RELOAD

**/Additional JCL statements for the batch command facility (BS2000/OSD)**

---

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=&&sortunld
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=user.db1002
/ADD-FILE-LINK L-NAME=SYS003,F-NAME=user.db1003
/CREATE-FILE F-NAME=user.db1004,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS004,F-NAME=user.db1004
/CREATE-FILE F-NAME=user.db1005,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS005,F-NAME=user.db1005
/CREATE-FILE F-NAME=user.db1006,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS006,F-NAME=user.db1006
/CREATE-FILE F-NAME=user.db1007,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS007,F-NAME=user.db1007
/CREATE-FILE F-NAME=user.db1008,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS008,F-NAME=user.db1008
/CREATE-FILE F-NAME=user.db1009,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS009,F-NAME=user.db1009
/CREATE-FILE F-NAME=user.db1010,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS010,F-NAME=user.db1010
/CREATE-FILE F-NAME=user.db1011,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS011,F-NAME=user.db1011
/ADD-FILE-LINK L-NAME=RELDCTL,F-NAME=user.reldctl
/ASSIGN-SYSOPT TO=&&sortreld
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKn files as necessary

---

**Note:** Specification of the SPACE parameter is for performance only. For sizing information, see 4.23, “RELOAD”

---

userdb	Linkname of the database file being loaded
user.userdb	Filename of the database file being loaded
&&sortunld	Filename name of the sort parameters created by UNLOAD
user.dbl002	Filename name of the file put out by UNLOAD as SYS001
user.dbl003	Filename name of the file put out by UNLOAD as SYS003
user.dbl004	Filename of the intermediate work file containing the output from SORT1
user.dbl005	Filename of the intermediate work file containing member descriptors for chained sets from IDMSDBL2
user.dbl006	Filename of the intermediate work file containing member descriptors for user owned index sets from IDMSDBL2
user.dbl007	Filename of the intermediate work file containing the output of SORT2
user.dbl008	Filename of the intermediate work file containing the reformatted index information from IDMSDBLX
user.dbl009	Filename of the intermediate work file containing the sorted index set descriptors from SORT3
user.dbl010	Filename of the intermediate work file containing prefix pointer information from IDMSDBL3
user.dbl011	Filename of the intermediate work file containing sorted prefix pointer information from SORT4
user.reldctl	Filename of the reload control file containing control and set information
&&sortreld	Data set name of the SYSPCH file

---

## 9.5.24 RESTORE

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb1,F-NAME=user.userdb1,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=userdb2,F-NAME=user.userdb2,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=user.bkpfile
```

---

userdb1	Linkname of the first database file
user.userdb1	Filename of the first database file
userdb2	Linkname of the second database file
user.userdb2	Filename of the second database file
user.bkpfile	Filename of the tape backup file

---

### 9.5.25 RESTRUCTURE CONNECT

#### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=user.spill
```

---

userdb	Linkname of the database file
user.userdb	Filename of the database file
user.spill	Filename of the spill file

---

### 9.5.26 RESTRUCTURE SEGMENT

#### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb1,F-NAME=user.userdb1,SHARED-UPD=*YES
```

```
/ADD-FILE-LINK L-NAME=userdb2,F-NAME=user.userdb2,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/CREATE-FILE F-NAME=user.spill,SUPPRESS-ERRORS=*FILE-EXISTING, -  
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))  
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=user.spill
```

**Note:** Specification of the SPACE parameter is for performance only

---

userdb1	Linkname of the first database file
user.userdb1	Filename of the first database file
userdb2	Linkname of the second database file
user.userdb2	Filename of the second database file
user.spill	Filename of the spill file

---

## 9.5.27 ROLLBACK

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/IMPORT-FILE SUP=*TAPE(VOL=nnnnnn,DEV-TYPE=tape,FILE-NAME=&&archive)
```

Specify the IMPORT-FILE statement only when the file is a tape file that is no longer cataloged.

```
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=&&archive
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required.

If using the SORT option, add the following statements:

```
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

**Note:** Specification of space allocations are for performance reasons only.

sysjrnl	Linkname of the tape journal file defined in the DMCL module
&&archive	Filename of the complete archive or tape journal file
userdb	Linkname of the database file
user.userdb	Filename of the database file
sortlib	Name of the SORT library.
temp.sortwork	Name of the sort workfile.
primary	Primary space allocation for the file being rolled forward; should be large enough to ensure contiguous disk blocks for the entire file
secondary	Secondary space allocation for the file being rolled forward; the suggested value is 0 to avoid dispersion of the file blocks on the disk

## 9.5.28 ROLLFORWARD

### Additional JCL statements for the batch command facility (BS2000/OSD)

---

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

If recovering from a standard journal file:

```
/IMPORT-FILE SUP=*TAPE(VOL=nnnnnn,DEV-TYPE=tape,FILE-NAME=&&archive)
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=&&archive
```

If recovering from a journal extract file:

```
/IMPORT-FILE SUP=*TAPE(VOL=nnnnnn,DEV-TYPE=tape,FILE-NAME=jrnl.extract)
```

Specify the IMPORT-FILE statements only when the file is a tape file that is no longer cataloged.

```
/ADD-FILE-LINK L-NAME=extract,F-NAME=jrnl.extract
```

If using the SORT option or processing a journal extract file, add the following statements:

```
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

---

**Note:** Specification of space allocations are for performance reasons only.

sysjrnl	Linkname of the tape journal file defined in the DMCL module.
&&archive	Filename of the complete archive or tape journal file.
userdb	Linkname of the database file.
user.userdb	Filename of the database file.
extract	Linkname of the extract journal file. If you don't specify, it defaults to SYS002.
jrnl.extract	Filename of the extract journal file.
sortlib	Name of the SORT library.
temp.sortwork	Name of the sort workfile. Space allocation is for performance reasons only.
primary	Primary space allocation for the file being rolled forward; should be large enough to ensure contiguous disk blocks for the entire file.
secondary	Secondary space allocation for the file being rolled forward; the suggested value is 0 to avoid dispersion of the file blocks on the disk.

---

## 9.5.29 TUNE INDEX

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

userdb	Linkname of the database file
user.userdb	Filename of the database file

## 9.5.30 UNLOCK

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

userdb	Linkname of the database file
user.userdb	Filename of the database file

## 9.5.31 UNLOAD

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/CREATE-FILE F-NAME=user.db1002,SUPPRESS-ERRORS=*FILE-EXISTING, -  
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))  
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=user.db1002  
/CREATE-FILE F-NAME=user.db1003,SUPPRESS-ERRORS=*FILE-EXISTING, -  
/ SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))  
/ADD-FILE-LINK L-NAME=SYS003,F-NAME=user.db1003  
/ADD-FILE-LINK L-NAME=RELDCTL,F-NAME=user.reldctl  
/ASSIGN-SYSOPT TO=&&sortunld
```

**Note:** Specification of space allocations are for performance reasons only.

userdb	Linkname of the database file
user.userdb	Filename of the database file
sysjrn1	Linkname of the tape journal file defined in the DMCL module
user.dbl002	Linkname of the database file
user.dbl003	Filename of the database file
user.reldctl	Name of the reload control file
&&sortunld	Filename of the sort parameters created by IDMSTBLU

### 9.5.32 UPDATE STATISTICS

#### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/ADD-FILE-LINK L-NAME=ddlcat,F-NAME=sysdict.ddlcat,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=ddlxcat,F-NAME=sysdict.ddlxcat,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=jljrn1,F-NAME=tape.jljrn1
```

Additional journal file assignments, as required

userdb	Linkname of the database file
user.userdb	Filename of the database file
ddlcat	Linkname of the database file containing the area of the system dictionary with the table definitions
sysdict.ddlcat	Filename of the database file containing the area of the system dictionary with the table definitions
ddlxcat	Linkname of the database file containing the area of the system dictionary with indexes
sysdict.ddlxcat	Filename of the database file containing the area of the system dictionary with indexes
jljrn1	Linkname of the first journal file as defined in the DMCL
tape.jljrn1	Filename of the first journal file

## 9.5.33 VALIDATE

### Additional JCL statements for the batch command facility (BS2000/OSD)

```
/ADD-FILE-LINK L-NAME=userdb,F-NAME=user.userdb,SHARED-UPD=*YES
```

Additional database file assignments, as required

```
/CREATE-FILE F-NAME=user.valin,SUPPRESS-ERRORS=*FILE-EXISTING, -
/  SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=user.valin
/CREATE-FILE F-NAME=user.valout,SUPPRESS-ERRORS=*FILE-EXISTING, -
/  SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=user.valout
/ASSIGN-SYSOPT TO=&&sortbuild
```

Following assignments are needed only when performing a complete VALIDATE:

```
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/  SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork
```

Add SORTWKnn files as necessary

**Note:** Specification of space allocations are for performance reasons only.

userdb	Linkname of the database file.
user.userdb	Filename of the database file.
user.valin *	Filename of the input SYS002 file; for sizing information, see 4.33, "VALIDATE."
user.valout *	Filename of the output SYS003 file; for sizing information see 4.33, "VALIDATE."
&&sortbuild	Filename of the SYSPCH file.
sortlib	Name of the SORT library.
temp.sortwork	Name of the sort workfile. The space allocation is for performance reasons only.

**Note:** \*When running a complete VALIDATE, SYS002 and SYS003 must point to the same intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to a different intermediate file.

## 9.6 Utility programs

### 9.6.1 IDMSDBAN

#### IDMSDBAN JCL (BS2000/OSD)

---

```

/.DBN1  REMARK
/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=userdb,F-NAME=user.userdb,SHARED-UPD=*YES

Additional database file assignments, as required

/CREATE-FILE F-NAME=chain.file,SUPPRESS-ERRORS=*FILE-EXISTING, -
/  SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=chain.file
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSDBN1,LIB=idms.loadlib,RUN-MODE=*ADV)

IDMSDBAN input parameters

/.DBN2  REMARK
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=chain.file
/CREATE-FILE F-NAME=chain.work,SUPPRESS-ERRORS=*FILE-EXISTING, -
/  SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SYS002,F-NAME=chain.work
/ADD-FILE-LINK L-NAME=BLSLIB,F-NAME=sortlib
/CREATE-FILE F-NAME=temp.sortwork,SUPPRESS-ERRORS=*FILE-EXISTING, -
/  SUP=PUB-DISK(SPACE=RELA(PRIM-ALLOC=primary,SEC-ALLOC=secondary))
/ADD-FILE-LINK L-NAME=SORTWK,F-NAME=temp.sortwork

Add SORTWKnn files as necessary

/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSDBN2,LIB=idms.loadlib,RUN-MODE=*ADV)

```

---

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

idms.dba.loadlib	Filename of the CA-IDMS load library containing the DMCL and database name table load modules
idms.loadlib	Filename of the CA-IDMS load library containing executable modules
userdb	Linkname of the user database file
user.userdb	Filename of the user database file
chain.file	Filename of the chain file generated by IDMSDBN1
primary,secondary	File space allocation
chain.work	Filename of the intermediate chain file used by IDMSDBN2
sort.work	Filename of the sort work file
sortlib	Filename of the object module library containing all modules required to execute the SORT utility as a subroutine

---

## 9.6.2 IDMSDIRL

### Local mode IDMSDIRL JCL (BS2000/OSD)

```

/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=sysjrn1,F-NAME=idms.sysjrn1
/ADD-FILE-LINK L-NAME=dictdb,F-NAME=idms.dictdb,S?HARED-UPD=*YES
/ADD-FILE-LINK L-NAME=SYS001,F-NAME=idms.dir1data
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSDIRL,LIB=idms.loadlib,RUN-MODE=*ADV)

```

#### IDMSDIRL input parameters

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

---

idms.dba.loadlib	Filename of the CA-IDMS load library containing DMCL and database name table load modules
idms.loadlib	Filename of the CA-IDMS load library containing CA-IDMS executable load modules
sysjrn	Linkname of the tape journal file
idms.tapejrn	Filename of the tape journal file
dictdb	Linkname of the data dictionary file
idms.dictdb	Filename of the data dictionary file
idms.dirldata	Filename of the IDMSDIRL input file (offloaded to disk)

---

**Central version:** To execute IDMSDIRL under the central version, modify the JCL shown above, as follows:

. Replace the ADD-FILE-LINK statements for sysjrn and dictdb with:  
 /ADD-FILE-LINK L-NAME=sysctl,F-NAME=idms.sysctl,SHARED-UPD=\*YES

---

sysctl	Linkname of the SYSCTL file
idms.sysctl	Filename of the SYSCTL file

---

**Note:** Depending on the central version operating environment, an IDMSOPTI module link edited with IDMSDIRL can be used in place of or in addition to the SYSCTL file.

### 9.6.3 IDMSLOOK

#### IDMSLOOK JCL (BS2000/OSD)

```
/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=sysjrn,F-NAME=idms.tapejrn
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSLOOK,LIB=idms.loadlib,RUN-MODE=*ADV)
```

#### IDMSLOOK input parameters

---

idms.dba.loadlib	Filename of the CA-IDMS load library containing DMCL and database name table load modules
idms.loadlib	Filename of the CA-IDMS load library containing executable modules

---

## 9.6.4 IDMSRPTS

### Local mode IDMSRPTS JCL (BS2000/OSD)

```

/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=sysjrn1,F-NAME=idms.tapejrn1
/ADD-FILE-LINK L-NAME=dictdb,F-NAME=idms.dictdb,SHARED-UPD=*YES
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSRPTS,LIB=idms.loadlib,RUN-MODE=*ADV)

```

IDMSRPTS input parameters

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

idms.dba.loadlib	Filename of the CA-IDMS load library containing DMCL and database name table load modules
idms.loadlib	Filename of the CA-IDMS load library containing executable CA-IDMS modules
sysjrn1	Linkname of the tape journal file
idms.tapejrn1	Filename of the tape journal file
dictdb	Linkname of the data dictionary file
idms.dictdb	Filename of the data dictionary file

**Central version:** To execute IDMSRPTS under the central version, modify the JCL shown above, as follows:

Replace the ADD-FILE-LINK statements for sysjrn1 and dictdb with:

```

/ADD-FILE-LINK L-NAME=sysctl,F-NAME=idms.sysctl,SHARED-UPD=*YES

```

sysctl	Linkname of the SYSCTL file
idms.sysctl	Filename of the SYSCTL file

**Note:** Depending on the central version operating environment, an IDMSOPTI module link edited with IDMSRPTS can be used in place of or in addition to the SYSCTL file.

## 9.6.5 IDMSRSTC

### Local mode IDMSRSTC JCL (BS2000/OSD)

```

/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=dictdb,F-NAME=idms.dictdb,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysjrn1,F-NAME=idms.tapejrn1
/ASSIGN-SYSOPT TO=temp.rstc.macros
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG *MOD(ELEM=IDMSRSTC,LIB=idms.loadlib,RUN-MODE=*ADV)

```

#### IDMSRSTC input parameters

**Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

idms.dba.loadlib	Filename of the CA-IDMS load library containing DMCL and database name table load modules
idms.loadlib	Filename of the CA-IDMS load library containing CA-IDMS executable modules
sysjrn1	Linkname of the tape journal file
idms.tapejrn1	Filename of the tape journal file
dictdb	Linkname of the data dictionary file
idms.dictdb	Filename of the data dictionary file
temp.rstc.macros	Filename of the card-image file containing the output IDMSRSTT macro statements

**Central version:** To run IDMSRSTC under the central version, modify the JCL shown above, as follows:

Replace the ADD-FILE-LINK statements for sysjrn1 and dictdb with:

```

/ADD-FILE-LINK L-NAME=sysctl,F-NAME=idms.sysctl,SHARED-UPD=*YES

```

sysctl	Linkname of the SYSCTL file
idms.sysctl	Filename of the SYSCTL file

**Note:** Depending on the central version operating environment, an IDMSOPTI module link edited with IDMSRSTC can be used in place of or in addition to the SYSCTL file.

## 9.6.6 IDMSRSTT

### Assemble and link an IDMSRSTT module

```

/DELETE-SYSTEM-FILE SYSTEM-FILE=*OMF
/ADD-FILE-LINK L-NAME=ALTLIB,F-NAME=idms.maclib
/ASSIGN-SYSDTA TO=*SYSCMD
/START-ASSEMBH
//COMPILE SOURCE=temp.rstc.macros
//      ,MACRO-LIB=*LINK(ALTLIB)
//      ,COPY-LIB=*LINK(ALTLIB,MACRO-ONLY)
//      ,COMP-ACT=MOD-GEN(MODULE-FORMAT=OM)
//      ,MOD-LIB=*OMF
//      ,COMPILER-TERMINATION=(MAX-ERROR-NUMBER=0)
//END
/REM-FILE-LINK ALTLIB
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=rstt-modname
//INC-MOD LIB=*OMF
//SAVE-LLM LIB=idms.dba.loadlib,ELEM=rstt-modname(VER=@),OVER=YES
//END

```

idms.dba.loadlib	Filename of the CA-IDMS/DB load library containing DMCL and database name table load modules
idms.maclib	Filename of the CA-IDMS macro library
rstt-modname	Name to be given to the rstt module. This name has to be specified in the RESTRUCTURE SEGMENT and RESTRUCTURE CONNECT utility statements.
temp.rstc.macros	Filename of the file generated by IDMSRSTC



# Appendix A. Format Program for FASTLOAD

---

A.1 About this appendix ..... A-3



## A.1 About this appendix

This appendix shows a listing of a format program that can be used with IDMSDBLU to load the sample Commonwealth database provided during the CA-IDMS installation. The program is written in COBOL and has been run through the CA-IDMS DML COBOL precompiler.

### Format program for FASTLOAD

```

*RETRIEVAL
*DMLIST
  IDENTIFICATION DIVISION.
  PROGRAM-ID.                EMPFLOAD.

*AUTHOR.                      KGV.
*
*INSTALLATION.                COMPUTER ASSOCIATES INTL INC
*                              8600 BRYN MAWR AVENUE
*                              CHICAGO, IL 60131.
*
*DATE-WRITTEN.                08/20/90.
*UPDATED FOR 15.0.            11/16/00.  LRD.
*
*REMARKS.                      THIS PROGRAM CREATES DATA TO
*                              BE USED AS INPUT TO THE FASTLOAD
*                              UTILITY, TO LOAD THE EMPLOYEE
*                              DEMO DATABASE.  IT USES THE SAME
*                              INPUT AS EMPLOAD.

ENVIRONMENT DIVISION.

IDMS-CONTROL SECTION.
  PROTOCOL.                  MODE IS BATCH
                              DEBUG
                              IDMS-RECORDS WITHIN
                              WORKING-STORAGE SECTION.

DATA DIVISION.

SCHEMA SECTION.
  DB  EMPSS01 WITHIN EMPSCHM  VERSION 100.
  SKIP2
  WORKING-STORAGE SECTION.
*
  01  OWNER-DESCRIPTOR-ONE.
      03  OWNER-ONE-SET          PIC X(16).
      03  OWNER-ONE-SERIAL       PIC S9(8) COMP.
      03  OWNER-ONE-KEY          PIC X(40).
      03  OWNER-ONE-KEY-RDEF REDEFINES OWNER-ONE-KEY.
          05  OWNER-ONE-KEY-SERIAL PIC S9(8) COMP.
          05  FILLER             PIC X(36).

  01  OWNER-DESCRIPTOR-TWO.
      03  OWNER-TWO-SET          PIC X(16).
      03  OWNER-TWO-SERIAL       PIC S9(8) COMP.
      03  OWNER-TWO-KEY          PIC X(40).
      03  OWNER-TWO-KEY-RDEF REDEFINES OWNER-TWO-KEY.
          05  OWNER-TWO-KEY-SERIAL PIC S9(8) COMP.

```

```

05 FILLER PIC X(36).

01 OWNER-DESCRIPTOR-THREE.
03 OWNER-THREE-SET PIC X(16).
03 OWNER-THREE-SERIAL PIC S9(8) COMP.
03 OWNER-THREE-KEY PIC X(40).
03 OWNER-THREE-KEY-RDEF REDEFINES OWNER-THREE-KEY.
05 OWNER-THREE-KEY-SERIAL PIC S9(8) COMP.
05 FILLER PIC X(36).

01 OCCURRENCE-DESCRIPTOR.
03 RECORD-SR-NAME PIC X(18).
03 FILLER PIC X(6) VALUE LOW-VALUES.
03 RECORD-ID PIC S9(8) COMP.
03 RECORD-SUGGESTED-PAGE PIC S9(8) COMP.
03 FILLER PIC S9(8) COMP VALUE ZERO.
03 RECORD-SERIAL PIC S9(8) COMP.
03 RECORD-LOAD-STATUS PIC X(4).
03 RECORD-DATA PIC X(2040).
03 FILLER REDEFINES RECORD-DATA.
05 RECORD-DATA-REDEF PIC X(40).
05 FILLER PIC X(2000).

*
*
*
01 MISCELLANEOUS-FIELDS.
02 END-FLAG PIC XXX VALUE SPACES.
88 END-OF-DATA VALUE 'END'.
02 COUNTS.
03 SUM-CARDS-IN PIC 9(6) VALUE ZERO.
03 SUM-TRANSACTIONS PIC 9(6) VALUE ZERO.
03 CARD-COUNT PIC 9(6) VALUE ZERO.
02 ERROR-MESSAGE PIC X(30) VALUE SPACES.
02 I-CTRL PIC S9(4) COMP SYNC.
02 SAVE-COVERAGE-SERIAL PIC S9(8) COMP SYNC.
SKIP2
01 CARD-IMAGE.
02 CI-DATA-IMAGE.
03 CI-KEYFIELDS.
04 FILLER PIC X.
04 CI-CARD-TYPE PIC XX.
88 CI-END VALUE 'EN'.
04 CI-CARD-TYPE-RD REDEFINES CI-CARD-TYPE.
05 CI-CARD-TYPE-MAJ PIC X.
05 CI-CARD-TYPE-MIN PIC X.
88 CI-FIRST-PART VALUES ARE
'A', 'C', 'E', 'G', 'I', 'M', 'O', 'Q', 'S'.
88 CI-2ND-PART VALUES ARE
'B', 'D', 'F', 'H', 'J', 'L', 'N', 'P', 'R', 'T'.
04 FILLER PIC X.
04 CI-EMP-ID PIC 9(4).
04 CI-INSPLAN REDEFINES CI-EMP-ID.
05 CI-INSPLAN-CODE PIC 9(3).
05 FILLER PIC X.
04 CI-OFFICE REDEFINES CI-EMP-ID.
05 CI-OFFICE-CODE PIC 9(3).
05 FILLER PIC X.
03 CI-DATAFIELDS PIC X(72).

```

## SKIP3

```

*****
*      TRANSACTION-STORAGE:                                *
*      ONE CARD TYPE FOR EACH INPUT RECORD TYPE;          *
*      EACH CARD CONTAINS A CARD-TYPE CODE.              *
*      INPUT CARDS MAY INCLUDE KEYFIELDS USED BY         *
*      THE PROGRAM TO MAKE THE APPROPRIATE OWNER         *
*      RECORDS CURRENT BEFORE A MEMBER IS STORED.        *
*      TRANSACTION-STORAGE IS REDEFINED FOR THE          *
*      FORMAT OF EACH TYPE OF INPUT CARD.                *
*****

```

```

01 TRANSACTION-STORAGE-AREA.
02 TRANSACTION-STORAGE-ALL.
03 TSA-SINGLE-CARD.
04 TSA-KEYFIELDS.
05 FILLER                PIC X.
05 TSA-CARD-TYPE         PIC XX.
88 TSA-DEPARTMENT        VALUE IS 'D '.
88 TSA-EMPLOYEE          VALUE IS 'E1'.
88 TSA-JOB                VALUE IS 'J1'.
88 TSA-EMPOSITION        VALUE IS 'P '.
88 TSA-EXPERTISE        VALUE IS 'T '.
88 TSA-SKILL              VALUE IS 'S '.
88 TSA-OFFICE            VALUE IS 'O1'.
88 TSA-STRUCTURE         VALUE IS 'OG'.
88 TSA-INS-PLAN-CODE     VALUE IS 'I1'.
88 TSA-COVERAGE          VALUE IS 'C '.
88 TSA-DENTAL            VALUE IS 'L1'.
88 TSA-HOSPITAL          VALUE IS 'H1'.
88 TSA-NON-HOSP-CLAIM    VALUE IS 'N1'.
04 FILLER                PIC X(77).
03 TSA-OTHER-CARD-SPACE  PIC X(400).

02 DEPT-STORAGE-AREA    REDEFINES
                        TRANSACTION-STORAGE-ALL.
03 D-CARD.
04 D-KEYFIELDS.
05 FILLER                PIC X(4).
05 D-DEPT-ID             PIC 9(4).
04 D-DATAFIELDS.
05 D-DEPT-NAME          PIC X(45).
05 D-DEPT-HEAD-ID       PIC 9(4).
05 FILLER                PIC X(23).
03 FILLER                PIC X(400).

02 EMPLOYEE-STORAGE-AREA REDEFINES
                        TRANSACTION-STORAGE-ALL.
03 E1-CARD.
04 E1-KEYFIELDS.
05 FILLER                PIC X(4).
05 E1-EMP-ID            PIC 9(4).
04 E1-DATAFIELDS.
05 E1-EMP-NAME          PIC X(25).
05 E1-EMP-DEPT-ID       PIC 9(4).
05 E1-EMP-OFFICE        PIC 9(3).
05 FILLER                PIC X(40).
03 E2-CARD.

```

```

04 E2-KEYFIELDS.
05 FILLER PIC X(4).
05 E2-EMP-ID PIC 9(4).
04 E2-DATAFIELDS.
05 E2-EMP-ADDRESS PIC X(46).
05 E2-EMP-PHONE PIC 9(10).
05 E2-EMP-STATUS PIC 9(2).
05 E2-EMP-SS-NUMBER PIC 9(9).
05 FILLER PIC X(5).
03 E3-CARD.
04 E3-KEYFIELDS.
05 FILLER PIC X(4).
05 E3-EMP-ID PIC 9(4).
04 E3-DATAFIELDS.
05 E3-EMP-START PIC 9(8).
05 E3-EMP-DOB PIC 9(8).
05 E3-EMP-TERM PIC 9(8).
05 FILLER PIC X(48).
03 FILLER PIC X(240).

02 JOB-STORAGE-AREA REDEFINES
TRANSACTION-STORAGE-ALL.

03 J1-CARD.
04 J1-KEYFIELDS.
05 FILLER PIC X(4).
05 J1-JOB-ID PIC 9(4).
04 J1-DATAFIELDS.
05 J1-JOB-TITLE PIC X(20).
05 J1-JOB-MIN-SAL PIC 9(8).
05 J1-JOB-MAX-SAL PIC 9(8).
05 J1-JOB-SAL-GRDS PIC 9(2) OCCURS 4.
05 J1-JOB-NUM-POSTS PIC 9(3).
05 J1-JOB-NUM-OPEN PIC 9(3).
05 FILLER PIC X(22).
03 J2-CARD.
04 J2-KEYFIELDS.
05 FILLER PIC X(4).
04 J2-DATAFIELDS.
05 J2-JOB-DES-LINE PIC X(60).
05 FILLER PIC X(16).
03 J3-CARD.
04 J3-KEYFIELDS.
05 FILLER PIC X(4).
04 J3-DATAFIELDS.
05 J3-JOB-DES-LINE PIC X(60).
05 FILLER PIC X(16).
03 J4-CARD.
04 J4-KEYFIELDS.
05 FILLER PIC X(4).
04 J4-DATAFIELDS.
05 J4-JOB-REQ-LINE PIC X(60).
05 FILLER PIC X(16).
03 J5-CARD.
04 J5-KEYFIELDS.
05 FILLER PIC X(4).
04 J5-DATAFIELDS.
05 J5-JOB-REQ-LINE PIC X(60).
05 FILLER PIC X(16).
03 FILLER PIC X(80).

```

```
02 POSITION-STORAGE-AREA REDEFINES
                           TRANSACTION-STORAGE-ALL.
03 P-CARD.
04 P-KEYFIELDS.
05 FILLER PIC X(4).
05 P-JOB-ID PIC 9(4).
05 P-EMP-ID PIC 9(4).
04 P-DATAFIELDS.
05 P-START-DATE PIC 9(8).
05 P-FINISH-DATE PIC 9(8).
05 P-SALARY-GRADE PIC 9(2).
05 P-SALARY-AMOUNT PIC 9(6)V99.
05 P-BONUS-PERCENT PIC V999.
05 P-COMM-PERCENT PIC V999.
05 P-OVERTIME-RATE PIC 9V99.
04 FILLER PIC X(33).
03 FILLER PIC X(400).

02 EXPERTISE-STORAGE-AREA REDEFINES
                           TRANSACTION-STORAGE-ALL.
03 T-CARD.
04 T-KEYFIELDS.
05 FILLER PIC X(4).
05 T-SKILL-ID PIC 9(4).
05 T-EMP-ID PIC 9(4).
04 T-DATAFIELDS.
05 T-SKILL-LEVEL PIC 9(2).
05 T-EXPERTISE-DATE PIC 9(8).
05 FILLER PIC X(58).
03 FILLER PIC X(400).

02 SKILL-STORAGE-AREA REDEFINES
                       TRANSACTION-STORAGE-ALL.
03 S-CARD.
04 S-KEYFIELDS.
05 FILLER PIC X(4).
05 S-SKILL-ID PIC 9(4).
04 S-DATAFIELDS.
05 S-SKILL-NAME PIC X(12).
05 S-SKILL-DESC PIC X(60).
03 FILLER PIC X(400).

02 OFFICE-STORAGE-AREA REDEFINES
                       TRANSACTION-STORAGE-ALL.
03 01-CARD.
04 01-KEYFIELDS.
05 FILLER PIC X(4).
05 01-OFFICE-CODE PIC 9(3).
04 01-DATAFIELDS.
05 01-OFFICE-ADDRESS PIC X(56).
05 FILLER PIC X(17).
03 02-CARD.
04 02-KEYFIELDS.
05 FILLER PIC X(4).
05 02-OFFICE-CODE PIC 9(3).
04 02-DATAFIELDS.
05 02-OFFICE-PHONE PIC 9(7) OCCURS 3 TIMES.
05 02-OFFICE-AREA PIC 9(3).
```

## A.1 About this appendix

---

```

    05 02-OFFICE-SPEED-DIAL PIC 9(3).
    05 FILLER                PIC X(46).
03  FILLER                  PIC X(320).

02  STRUCTURE-STORAGE-AREA REDEFINES
                                TRANSACTION-STORAGE-ALL.
03  OG-CARD.
04  OG-KEYFIELDS.
    05 FILLER                PIC X(4).
    05 OG-EMP-RPTS-TO       PIC 9(4).
    05 OG-EMP-MANAGES      PIC 9(4).
04  OG-DATAFIELDS.
    05 OG-STRUCT-CODE     PIC X(2).
    05 OG-RELATION-DATE   PIC 9(8).
    05 FILLER             PIC X(58).
03  FILLER                  PIC X(400).

02  INSURANCE-STORAGE-AREA REDEFINES
                                TRANSACTION-STORAGE-ALL.
03  I1-CARD.
04  I1-KEYFIELDS.
    05 FILLER                PIC X(4).
    05 I1-INSPLAN-CODE     PIC X(3).
04  I1-DATAFIELDS.
    05 I1-INSPLAN-CO-NAME  PIC X(45).
    05 FILLER              PIC X(28).
03  I2-CARD.
04  I2-KEYFIELDS.
    05 FILLER                PIC X(4).
    05 I2-INSPLAN-CODE     PIC X(3).
04  I2-DATAFIELDS.
    05 I2-CO-ADDRESS      PIC X(46).
    05 I2-CO-PHONE        PIC 9(10).
    05 FILLER              PIC X(17).
03  I3-CARD.
04  I3-KEYFIELDS.
    05 FILLER                PIC X(4).
    05 I3-INSPLAN-CODE     PIC X(3).
04  I3-DATAFIELDS.
    05 I3-GROUP-NUM       PIC 9(6).
    05 I3-DESCRIPTION.
        06 I3-DEDUCT        PIC 9(6)V99.
        06 I3-MAX-LIFE-COST PIC 9(6)V99.
        06 I3-FAM-COST     PIC 9(6)V99.
        06 I3-DEP-COST     PIC 9(6)V99.
    05 FILLER              PIC X(35).
03  FILLER                  PIC X(240).

02  COVERAGE-STORAGE-AREA REDEFINES
                                TRANSACTION-STORAGE-ALL.
03  C-CARD.
04  C-KEYFIELDS.
    05 FILLER                PIC X(4).
    05 C-INSPLAN-CODE      PIC X(3).
    05 C-EMP-ID            PIC 9(4).
04  C-DATAFIELDS.
    05 C-SELECT-DATE      PIC 9(8).
    05 C-TERMIN-DATE      PIC 9(8).
    05 C-TYPE              PIC X.
```

```

05 C-INS-PLAN-CODE PIC X(3).
05 FILLER PIC X(49).
03 FILLER PIC X(400).

02 DENTAL-STORAGE-AREA REDEFINES
                        TRANSACTION-STORAGE-ALL.
03 L1-CARD.
04 L1-KEYFIELDS.
05 FILLER PIC X(4).
04 L1-DATAFIELDS.
05 L1-DC-CLAIM-DATE PIC 9(8).
05 L1-DC-PATIENT-NAME PIC X(25).
05 L1-DC-PATIENT-DOB PIC 9(8).
05 L1-DC-SEX PIC X.
05 L1-DC-REL-TO-EMP PIC X(10).
05 L1-DC-DENTIST-NAME PIC X(24).
03 L2-CARD.
04 L2-KEYFIELDS.
05 FILLER PIC X(4).
04 L2-DATAFIELDS.
05 L2-DC-DENTIST-ADDRESS PIC X(46).
05 L2-DC-DENTIST-LIC-NUM PIC 9(6).
05 L2-DC-NUM-PROCEDURES PIC 9(2).
05 FILLER PIC X(22).
03 LA-CARD.
04 LA-KEYFIELDS.
05 FILLER PIC X(4).
04 LA-DATAFIELDS.
05 LA-DC-TOOTH-NUM PIC 9(2).
05 LA-DC-SERVICE-DATE PIC 9(8).
05 LA-DC-PROC-CODE PIC 9(4).
05 LA-DC-FEE PIC 9(6)V99.
05 FILLER PIC X(54).
03 LB-CARD.
04 LB-KEYFIELDS.
05 FILLER PIC X(4).
04 LB-DATAFIELDS.
05 LB-DC-DESC-OF-SERVICE PIC X(60).
05 FILLER PIC X(16).
03 FILLER PIC X(160).

02 HOSPITAL-STORAGE-AREA REDEFINES
                        TRANSACTION-STORAGE-ALL.
03 H1-CARD.
04 H1-KEYFIELDS.
05 FILLER PIC X(4).
04 H1-DATAFIELDS.
05 H1-HC-CLAIM-DATE PIC 9(8).
05 H1-HC-PATIENT-NAME PIC X(25).
05 H1-HC-PATIENT-DOB PIC 9(8).
05 H1-HC-SEX PIC X.
05 H1-HC-REL-TO-EMP PIC X(10).
05 H1-HC-HOSP-NAME PIC X(24).
03 H2-CARD.
04 H2-KEYFIELDS.
05 FILLER PIC X(4).
04 H2-DATAFIELDS.
05 H2-HC-HOSP-ADDRESS PIC X(46).
05 H2-HC-ADMIT-DATE PIC 9(8).

```

## A.1 About this appendix

---

```

    05 H2-HC-DISCH-DATE      PIC 9(8).
    05 FILLER                 PIC X(14).
03 H3-CARD.
    04 H3-KEYFIELDS.
    05 FILLER                 PIC X(4).
    04 H3-DATAFIELDS.
    05 H3-HC-DIAGNOSIS      PIC X(60).
    05 FILLER                 PIC X(16).
03 H4-CARD.
    04 H4-KEYFIELDS.
    05 FILLER                 PIC X(4).
    04 H4-DATAFIELDS.
    05 H4-HC-DIAGNOSIS      PIC X(60).
    05 FILLER                 PIC X(16).
03 H5-CARD.
    04 H5-KEYFIELDS.
    05 FILLER                 PIC X(4).
    04 H5-DATAFIELDS.
    05 H5-HOSP-CHARGES.
    06 H5-HC-WARD.
    07 H5-HC-WARD-DAYS      PIC 9(4).
    07 H5-HC-WARD-RATE      PIC 9(6)V99.
    07 H5-HC-WARD-TOTAL     PIC 9(6)V99.
    06 H5-SEMI-PRIVATE.
    07 H5-HC-SEMI-DAYS      PIC 9(4).
    07 H5-HC-SEMI-RATE      PIC 9(6)V99.
    07 H5-HC-SEMI-TOTAL     PIC 9(6)V99.
    06 H5-HC-OTHER.
    07 H5-HC-DEL-COST       PIC 9(6)V99.
    07 H5-HC-ANESTH-COST    PIC 9(6)V99.
    07 H5-HC-LAB-COST       PIC 9(6)V99.
    05 FILLER                 PIC X(12).
03 FILLER                     PIC X(80).

02 NONHOSP-STORAGE-AREA      REDEFINES
                               TRANSACTION-STORAGE-ALL.

03 N1-CARD.
    04 N1-KEYFIELDS.
    05 FILLER                 PIC X(4).
    04 N1-DATAFIELDS.
    05 N1-NC-CLAIM-DATE     PIC 9(8).
    05 N1-NC-PATIENT-NAME   PIC X(25).
    05 N1-NC-PATIENT-DOB    PIC 9(8).
    05 N1-NC-SEX            PIC X.
    05 N1-NC-REL-TO-EMP     PIC X(10).
    05 N1-NC-PHYS-NAME      PIC X(24).
03 N2-CARD.
    04 N2-KEYFIELDS.
    05 FILLER                 PIC X(4).
    04 N2-DATAFIELDS.
    05 N2-NC-PHYS-ADDRESS   PIC X(46).
    05 N2-NC-PHYS-ID        PIC 9(6).
    05 N2-NC-NUM-PROCS      PIC 9(2).
    05 FILLER                 PIC X(22).
03 N3-CARD.
    04 N3-KEYFIELDS.
    05 FILLER                 PIC X(4).
    04 N3-DATAFIELDS.
    05 N3-NC-DIAGNOSIS      PIC X(60).
```

```

      05 FILLER                PIC X(16).
03  N4-CARD.
      04 N4-KEYFIELDS.
      05 FILLER                PIC X(4).
      04 N4-DATAFIELDS.
      05 N4-NC-DIAGNOSIS      PIC X(60).
      05 FILLER                PIC X(16).
03  NA-CARD.
      04 NA-KEYFIELDS.
      05 FILLER                PIC X(4).
      04 NA-DATAFIELDS.
      05 NA-NC-SERVICE-DATE   PIC 9(8).
      05 NA-NC-PROC-CODE      PIC 9(4).
      05 NA-NC-FEE            PIC 9(6)V99.
      05 FILLER                PIC X(56).
03  NB-CARD.
      04 NB-KEYFIELDS.
      05 FILLER                PIC X(4).
      04 NB-DATAFIELDS.
      05 NB-NC-DESC-OF-SERVICE PIC X(62).
      05 FILLER                PIC X(14).

01  NAMES-INFO.
      02 NAMES-SSNAME          PIC X(8)
                                VALUE 'EMPSS01 '.
      02 NAMES-DBNAME          PIC X(8)
                                VALUE 'EMPDEMO '.
      02 NAMES-DMCLNAME        PIC X(8)
                                VALUE 'IDMSDMCL'.

```

## PROCEDURE DIVISION.

```

*****
*      PROCEDURE DIVISION GENERAL STRATEGY:      *
*      1) READ 1 OR MORE CARDS TO FOR A TRANSACTION *
*      2) PERFORM THE APPROPRIATE ROUTINE, BASED UPON THE *
*      TRANSACTION CODE *
*      3) CONTINUE UNTIL ALL CARD INPUT IS EXHAUSTED *
*****

```

## 0000-MAIN-LINE SECTION.

## 0001-SETUP.

```

      DISPLAY '*** BEFORE FIRST CALL ***'.
      CALL 'IDMSDBLU' USING NAMES-INFO.

```

## 0005-ML-START.

```

      ACCEPT CARD-IMAGE.
      DISPLAY '*** AFTER ACCEPT ***'
      PERFORM 0020-MAIN-LOOP THRU 0020-ML-EXIT UNTIL
      END-OF-DATA.
      PERFORM 9999-END.

```

## 0020-MAIN-LOOP.

```

      PERFORM 0510-READ-TRANSACTION THRU 0515-RT-EXIT.
      DISPLAY '*** AFTER PERFORM 510- ***'.

```

```
IF END-OF-DATA
  GO TO 0020-ML-EXIT.

ADD 1 TO SUM-TRANSACTIONS.

IF TSA-DEPARTMENT
  PERFORM 1010-D0-DEPARTMENT THRU 1090-DD-EXIT
  ELSE
IF TSA-EMPLOYEE
  PERFORM 1510-D0-EMPLOYEE THRU 1590-DE-EXIT
  ELSE
IF TSA-JOB
  PERFORM 2010-D0-JOB THRU 2090-DJ-EXIT
  ELSE
IF TSA-EMPOSITION
  PERFORM 2510-D0-EMPOSITION THRU 2590-DEM-EXIT
  ELSE
IF TSA-EXPERTISE
  PERFORM 3010-D0-EXPERTISE THRU 3090-DEX-EXIT
  ELSE
IF TSA-SKILL
  PERFORM 3510-D0-SKILL THRU 3590-DS-EXIT
  ELSE
IF TSA-OFFICE
  PERFORM 4510-D0-OFFICE THRU 4590-D0-EXIT
  ELSE
IF TSA-STRUCTURE
  PERFORM 5010-D0-STRUCTURE THRU 5090-DS-EXIT
  ELSE
IF TSA-INS-PLAN-CODE
  PERFORM 5510-D0-INSURANCE THRU 5590-DI-EXIT
  ELSE
IF TSA-COVERAGE
  PERFORM 6010-D0-COVERAGE THRU 6090-DC-EXIT
  ELSE
IF TSA-DENTAL
  PERFORM 6510-D0-DENTAL THRU 6590-DDN-EXIT
  ELSE
IF TSA-HOSPITAL
  PERFORM 7010-D0-HOSPITAL THRU 7090-DH-EXIT
  ELSE
  PERFORM 7510-D0-NONHOSP THRU 7590-DN-EXIT.
0020-ML-EXIT.
EXIT.
```

```
*****
*   UTILITY ROUTINES FOLLOW   *
*****
```

0500-UTILITY SECTION.

```
*****
*   THIS ROUTINE ASSEMBLES A TRANSACTION FROM ONE OR MORE   *
*   INDIVIDUAL CARDS; NOTE THAT WHEN THIS PROCEDURE IS     *
*   ENTERED, A CARD IS ALWAYS PRESENT IN THE 'CARD IMAGE'   *
*   BUFFER.                                                  *
*                                                           *
*   DEPARTMENT HAS A SINGLE 'D' CARD                        *
*   EMPLOYEE HAS AN 'E1', AN 'E2', AND AN 'E3' CARD        *
*   JOB HAS 'J1' THRU 'J5' CARDS                            *
*****
```

```

*      EMPOSITION HAS A SINGLE 'P ' CARD      *
*      EXPERTISE HAS A SINGLE 'T ' CARD      *
*      SKILL HAS A SINGLE 'S ' CARD          *
*      OFFICE HAS AN '01' AND AN '02' CARD   *
*      STRUCTURE HAS A SINGLE 'OG' CARD     *
*      INSURANCE-PLAN HAS AN 'I1', AN 'I2', AND AN 'I3' CARD *
*      COVERAGE HAS A SINGLE 'C ' CARD     *
*      DENTAL-CLAIM HAS AN 'L1' AND AN 'L2' CARD, FOLLOWED *
*      BY 2 TO 20 'LX' CARDS (WHERE 'X' IS A LETTER *
*      FROM A TO T)                          *
*      HOSPITAL-CLAIM HAS 'H1' THRU 'H5' CARDS *
*      NON-HOSP-CLAIM HAS 'N1' THRU 'N4' CARDS, FOLLOWED *
*      BY 2 TO 20 'NX' CARDS (WHERE 'X' IS A LETTER *
*      FROM A TO T)                          *
*
*****
0510-READ-TRANSACTION.
      MOVE SPACES TO TRANSACTION-STORAGE-AREA.

      IF CI-END
          MOVE 'END' TO END-FLAG
          GO TO 0515-RT-EXIT.

      IF CI-CARD-TYPE = 'D ' OR 'P ' OR 'T ' OR 'S '
          OR 'OG' OR 'C '
          MOVE CI-DATA-IMAGE TO TSA-SINGLE-CARD
          PERFORM 0600-READ-CARD THRU 0615-RC-EXIT
          ELSE
      IF CI-CARD-TYPE = 'E1'
          PERFORM 0520-ASSEM-EMPLOYEE THRU 0528-AE-EXIT
          ELSE
      IF CI-CARD-TYPE = '01'
          PERFORM 0530-ASSEM-OFFICE THRU 0538-AO-EXIT
          ELSE
      IF CI-CARD-TYPE = 'J1'
          PERFORM 0540-ASSEM-JOB THRU 0548-AJ-EXIT
          ELSE
      IF CI-CARD-TYPE = 'I1'
          PERFORM 0550-ASSEM-INS THRU 0558-AI-EXIT
          ELSE
      IF CI-CARD-TYPE = 'L1'
          PERFORM 0560-ASSEM-DENT THRU 0568-AD-EXIT
          ELSE
      IF CI-CARD-TYPE = 'H1'
          PERFORM 0570-ASSEM-HOSP THRU 0578-AH-EXIT
          ELSE
      IF CI-CARD-TYPE = 'N1'
          PERFORM 0580-ASSEM-NONHOSP THRU 0588-AN-EXIT
          ELSE
      MOVE 'INVALID CARD TYPE/SEQ' TO ERROR-MESSAGE
      PERFORM 0620-DISPLAY-CARD-ERROR THRU 0640-DCE-EXIT
      PERFORM 0600-READ-CARD THRU 0615-RC-EXIT
      GO TO 0510-READ-TRANSACTION.
0515-RT-EXIT.
      EXIT.

*****
*      THE FOLLOWING MODULES ASSEMBLE MULTIPLE INPUT CARDS      *

```

```
* INTO THE APPROPRIATE WORK RECORDS. *
*****
0520-ASSEM-EMPLOYEE.
    MOVE CI-DATA-IMAGE      TO E1-CARD.

    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = 'E2'
        AND CI-EMP-ID = E1-EMP-ID
        MOVE CI-DATA-IMAGE TO E2-CARD
        PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = 'E3'
        AND CI-EMP-ID = E1-EMP-ID
        MOVE CI-DATA-IMAGE TO E3-CARD
        PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0528-AE-EXIT.
    EXIT.

0530-ASSEM-OFFICE.
    MOVE CI-DATA-IMAGE      TO 01-CARD.

    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = '02'
        AND CI-OFFICE-CODE = 01-OFFICE-CODE
        MOVE CI-DATA-IMAGE      TO 02-CARD
        PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0538-AO-EXIT.
    EXIT.

0540-ASSEM-JOB.
    MOVE CI-DATA-IMAGE      TO J1-CARD.

    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = 'J2'
        MOVE CI-DATA-IMAGE      TO J2-CARD
        PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = 'J3'
        MOVE CI-DATA-IMAGE      TO J3-CARD
        PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = 'J4'
        MOVE CI-DATA-IMAGE      TO J4-CARD
        PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = 'J5'
        MOVE CI-DATA-IMAGE      TO J5-CARD
        PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0548-AJ-EXIT.
    EXIT.

0550-ASSEM-INS.
```

```
MOVE CI-DATA-IMAGE      TO I1-CARD.

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'I2'
  AND CI-INSPLAN-CODE = I1-INSPLAN-CODE
  MOVE CI-DATA-IMAGE    TO I2-CARD
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'I3'
  AND CI-INSPLAN-CODE = I1-INSPLAN-CODE
  MOVE CI-DATA-IMAGE    TO I3-CARD
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0558-AI-EXIT.
EXIT.

0560-ASSEM-DENT.
MOVE CI-DATA-IMAGE TO L1-CARD.

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'L2'
  MOVE CI-DATA-IMAGE    TO L2-CARD
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

MOVE 0 TO I-CTRL.
PERFORM 0563-GET-CHARGES THRU 0563-GC-EXIT
  UNTIL (CI-CARD-TYPE-MAJ NOT = 'L' OR
         CI-CARD-TYPE-MIN NOT ALPHABETIC).

0568-AD-EXIT.
EXIT.

0563-GET-CHARGES.
IF CI-FIRST-PART
  ADD 1 TO I-CTRL

  MOVE CI-DATA-IMAGE  TO LA-CARD
  MOVE LA-DC-TOOTH-NUM TO TOOTH-NUMBER-0405 (I-CTRL)
  MOVE LA-DC-SERVICE-DATE TO SERVICE-DATE-0405 (I-CTRL)
  MOVE LA-DC-PROC-CODE TO PROCEDURE-CODE-0405 (I-CTRL)
  MOVE LA-DC-FEE      TO FEE-0405 (I-CTRL).

IF CI-2ND-PART
  MOVE CI-DATA-IMAGE    TO LB-CARD
  MOVE LB-DC-DESC-OF-SERVICE TO
    DESCRIPTION-OF-SERVICE-0405 (I-CTRL).

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
0563-GC-EXIT.
EXIT.

0570-ASSEM-HOSP.
MOVE CI-DATA-IMAGE      TO H1-CARD.

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'H2'
```

```
MOVE CI-DATA-IMAGE    TO H2-CARD
PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'H3'
MOVE CI-DATA-IMAGE    TO H3-CARD
PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'H4'
MOVE CI-DATA-IMAGE    TO H4-CARD
PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'H5'
MOVE CI-DATA-IMAGE    TO H5-CARD
PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0578-AH-EXIT.
EXIT.

0580-ASSEM-NONHOSP.
MOVE CI-DATA-IMAGE    TO N1-CARD.

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'N2'
MOVE CI-DATA-IMAGE    TO N2-CARD
PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'N3'
MOVE CI-DATA-IMAGE    TO N3-CARD
PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'N4'
MOVE CI-DATA-IMAGE    TO N4-CARD
PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

MOVE 0 TO I-CTRL.
PERFORM 0583-GET-CHARGES THRU 0583-GC-EXIT
UNTIL (CI-CARD-TYPE-MAJ NOT = 'N' OR
CI-CARD-TYPE-MIN NOT ALPHABETIC).

0588-AN-EXIT.
EXIT.

0583-GET-CHARGES.
IF CI-FIRST-PART
ADD 1 TO I-CTRL

MOVE CI-DATA-IMAGE    TO NA-CARD
MOVE NA-NC-SERVICE-DATE TO SERVICE-DATE-0445 (I-CTRL)
MOVE NA-NC-PROC-CODE    TO PROCEDURE-CODE-0445 (I-CTRL)
MOVE NA-NC-FEE          TO FEE-0445 (I-CTRL).

IF CI-2ND-PART
MOVE CI-DATA-IMAGE    TO NB-CARD
MOVE NB-NC-DESC-OF-SERVICE
TO DESCRIPTION-OF-SERVICE-0445 (I-CTRL).

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
0583-GC-EXIT.
```

```

EXIT.

0600-READ-CARD.
  IF CARD-COUNT = '50'
    MOVE ZERO TO CARD-COUNT.
  DISPLAY CARD-IMAGE.
  ACCEPT CARD-IMAGE.
  ADD 1 TO CARD-COUNT.
  ADD 1 TO SUM-CARDS-IN.

0615-RC-EXIT.
  EXIT.

0620-DISPLAY-CARD-ERROR.
  DISPLAY ERROR-MESSAGE, CARD-IMAGE.

0640-DCE-EXIT.
  EXIT.

1000-PROCESS SECTION.
*
*****
*
*   THIS MAIN PROCESS SECTION HANDLES ALL FORMATTING OF      *
*   OWNER DESCRIPTOR RECORDS AND CALLS TO IDMSDBLU.          *
*
*****

1010-DO-DEPARTMENT.
*****
*
*   THIS ROUTINE STORES A DEPARTMENT RECORD.                  *
*
*****

***BUILD RECORD OCCURRENCE OF DEPARTMENT RECORD*****

      MOVE D-DEPT-ID          TO DEPT-ID-0410.
      MOVE D-DEPT-NAME        TO DEPT-NAME-0410.
      MOVE D-DEPT-HEAD-ID     TO DEPT-HEAD-ID-0410.

1020-STORE-DEPT.
      MOVE 'DEPARTMENT'        TO RECORD-SR-NAME.
      MOVE 410                  TO RECORD-ID.
      MOVE DEPARTMENT           TO RECORD-DATA.
      CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.
      PERFORM DBLU-STATUS.

1090-DD-EXIT.
  EXIT.

1510-DO-EMPLOYEE.
*****
*   THIS ROUTINE STORES THE EMPLOYEE RECORD.  THE OWNERS IN  *
*   THE DEPT-EMPLOYEE AND OFFICE-EMPLOYEE SETS MUST BE     *
*   PRESENT BY THE END OF THE RUN.                          *
*****

```

\*\*\*\*\*BUILD OWNER OCCURRENCE OF SKILL-NAME-NDX SET\*\*\*\*\*

```

MOVE 'EMP-NAME-NDX'      TO OWNER-ONE-SET.
MOVE -1                  TO OWNER-ONE-SERIAL.
MOVE -1                  TO OWNER-ONE-KEY-SERIAL.

```

\*\*\*\*\*BUILD OWNER OCCURRENCE OF DEPT-EMPLOYEE SET\*\*\*\*\*

```

MOVE 'DEPT-EMPLOYEE'    TO OWNER-TWO-SET.
MOVE -1                  TO OWNER-TWO-SERIAL.
MOVE E1-EMP-DEPT-ID     TO OWNER-TWO-KEY.

```

\*\*\*\*\*BUILD OWNER OCCURRENCE OF OFFICE-EMPLOYEE SET\*\*\*\*\*

```

MOVE 'OFFICE-EMPLOYEE'  TO OWNER-THREE-SET.
MOVE -1                  TO OWNER-THREE-SERIAL.
MOVE E1-EMP-OFFICE      TO OWNER-THREE-KEY.

```

\*\*\*\*\*BUILD RECORD OCCURRENCE OF EMPLOYEE RECORD\*\*\*\*\*

```

MOVE E1-EMP-ID          TO EMP-ID-0415.
MOVE E1-EMP-NAME        TO EMP-NAME-0415.
MOVE E2-EMP-ADDRESS     TO EMP-ADDRESS-0415.
MOVE E2-EMP-PHONE       TO EMP-PHONE-0415.
MOVE E2-EMP-STATUS      TO STATUS-0415.
IF STATUS-0415 EQUAL TO '05'
    MOVE ZEROS          TO TERMINATION-DATE-0415
ELSE
    MOVE E3-EMP-TERM    TO TERMINATION-DATE-0415.
MOVE E2-EMP-SS-NUMBER   TO SS-NUMBER-0415.
MOVE E3-EMP-START       TO START-DATE-0415.
MOVE E3-EMP-DOB         TO BIRTH-DATE-0415.

```

1520-STORE-EMP.

```

MOVE 'EMPLOYEE'         TO RECORD-SR-NAME.
MOVE 415                 TO RECORD-ID.
MOVE EMPLOYEE           TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
                      OWNER-DESCRIPTOR-ONE
                      OWNER-DESCRIPTOR-TWO
                      OWNER-DESCRIPTOR-THREE.

```

PERFORM DBLU-STATUS.

1590-DE-EXIT.

EXIT.

2010-DO-JOB.

```

*****
*   THIS ROUTINE STORES THE JOB RECORD   *
*****

```

\*\*\*\*\*BUILD OWNER OCCURRENCE OF JOB-TITLE-NDX SET\*\*\*\*\*

```

MOVE 'JOB-TITLE-NDX'    TO OWNER-ONE-SET.
MOVE -1                  TO OWNER-ONE-SERIAL.
MOVE -1                  TO OWNER-ONE-KEY-SERIAL.

```

\*\*\*\*\*BUILD RECORD OCCURRENCE OF JOB RECORD\*\*\*\*\*

```

MOVE J1-JOB-ID          TO JOB-ID-0440.
MOVE J1-JOB-TITLE       TO TITLE-0440.
MOVE J1-JOB-MIN-SAL     TO MINIMUM-SALARY-0440.
MOVE J1-JOB-MAX-SAL     TO MAXIMUM-SALARY-0440.
MOVE J1-JOB-NUM-POSTS   TO NUMBER-OF-POSITIONS-0440.
MOVE J1-JOB-NUM-OPEN    TO NUMBER-OPEN-0440.
MOVE J2-JOB-DES-LINE    TO DESCRIPTION-LINE-0440 (1).
MOVE J3-JOB-DES-LINE    TO DESCRIPTION-LINE-0440 (2).
MOVE J4-JOB-REQ-LINE    TO REQUIREMENT-LINE-0440 (1).
MOVE J5-JOB-REQ-LINE    TO REQUIREMENT-LINE-0440 (2).

MOVE 0 TO I-CTRL.
PERFORM 2110-DO-SAL-GRDS THRU 2110-DSG-EXIT
    VARYING I-CTRL FROM 1 BY 1 UNTIL I-CTRL = +4.

2020-STORE-JOB.
MOVE 'JOB'              TO RECORD-SR-NAME.
MOVE 440                TO RECORD-ID.
MOVE JOB                TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
    OWNER-DESCRIPTOR-ONE.
PERFORM DBLU-STATUS.

2090-DJ-EXIT.
EXIT.

2110-DO-SAL-GRDS.
MOVE J1-JOB-SAL-GRDS (I-CTRL)
    TO SALARY-GRADES-0440 (I-CTRL).
2110-DSG-EXIT.
EXIT.

2510-DO-EMPOSITION.

*****
*   THIS ROUTINE STORES THE EMPOSITION RECORD.  THE OWNERS   *
*   THE JOB-EMPOSITION AND EMP-EMPOSITION SETS MUST BE     *
*   PRESENT BY THE END OF THE RUN.                          *
*****

*****BUILD OWNER OCCURRENCE FOR THE EMP-EMPOSITION SET*****

MOVE 'EMP-EMPOSITION'   TO OWNER-TWO-SET.
MOVE -1                TO OWNER-TWO-SERIAL.
MOVE P-EMP-ID          TO OWNER-TWO-KEY.

*****BUILD OWNER OCCURRENCE FOR THE JOB-EMPOSITION SET*****

MOVE 'JOB-EMPOSITION'   TO OWNER-ONE-SET.
MOVE -1                TO OWNER-ONE-SERIAL.
MOVE P-JOB-ID          TO OWNER-ONE-KEY.

*****BUILD RECORD OCCURRENCE OF EMPOSITION RECORD*****

MOVE P-START-DATE      TO START-DATE-0420.
MOVE P-FINISH-DATE     TO FINISH-DATE-0420.
MOVE P-SALARY-GRADE    TO SALARY-GRADE-0420.
MOVE P-SALARY-AMOUNT   TO SALARY-AMOUNT-0420.
MOVE P-BONUS-PERCENT   TO BONUS-PERCENT-0420.

```

```

        MOVE P-COMM-PERCENT      TO COMMISSION-PERCENT-0420.
        MOVE P-OVERTIME-RATE    TO OVERTIME-RATE-0420.

2520-STORE-EMPOSITION.
        MOVE 'EMPOSITION'      TO RECORD-SR-NAME.
        MOVE 420                TO RECORD-ID.
        MOVE EMPOSITION         TO RECORD-DATA.
        CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
                               OWNER-DESCRIPTOR-ONE
                               OWNER-DESCRIPTOR-TWO.

2590-DEM-EXIT.
        EXIT.

3010-DO-EXPERTISE.

*****
*   THE NEXT ROUTINE STORES A NEW EXPERTISE RECORD.   THE   *
*   SKILL AND EMPLOYEE OWNER RECORDS MUST BE PRESENT   *
*   BY THE END OF THE RUN.                             *
*****

*****BUILD OWNER OCCURRENCE FOR THE EMP-EXPERTISE SET*****

        MOVE 'EMP-EXPERTISE'    TO OWNER-TWO-SET.
        MOVE -1                 TO OWNER-TWO-SERIAL.
        MOVE T-EMP-ID           TO OWNER-TWO-KEY.

*****BUILD OWNER OCCURRENCE FOR SKILL-EXPERTISE SET*****

        MOVE 'SKILL-EXPERTISE'  TO OWNER-ONE-SET.
        MOVE -1                 TO OWNER-ONE-SERIAL.
        MOVE T-SKILL-ID         TO OWNER-ONE-KEY.

*****BUILD OCCURRENCE OF EXPERTISE RECORD*****

        MOVE T-SKILL-LEVEL      TO SKILL-LEVEL-0425.
        MOVE T-EXPERTISE-DATE   TO EXPERTISE-DATE-0425.

3020-STORE-EXPERTISE.
        MOVE 'EXPERTISE'        TO RECORD-SR-NAME.
        MOVE 425                TO RECORD-ID.
        MOVE EXPERTISE          TO RECORD-DATA.
        CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
                               OWNER-DESCRIPTOR-ONE
                               OWNER-DESCRIPTOR-TWO.

        PERFORM DBLU-STATUS.

3090-DEX-EXIT.
        EXIT.

3510-DO-SKILL.

*****
*   THIS ROUTINE STORES A NEW SKILL RECORD.           *
*****

*****BUILD OWNER OCCURRENCE FOR THE SKILL-NAME-NDX SET*****

```

```
MOVE 'SKILL-NAME-NDX' TO OWNER-ONE-SET.
MOVE -1 TO OWNER-ONE-SERIAL.
MOVE -1 TO OWNER-ONE-KEY-SERIAL.

*****BUILD OCCURRENCE OF SKILL RECORD*****

MOVE S-SKILL-ID TO SKILL-ID-0455.
MOVE S-SKILL-NAME TO SKILL-NAME-0455.
MOVE S-SKILL-DESC TO SKILL-DESCRIPTION-0455.

3520-STORE-SKILL.
MOVE 'SKILL' TO RECORD-SR-NAME.
MOVE 455 TO RECORD-ID.
MOVE SKILL TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE.
PERFORM DBLU-STATUS.

3590-DS-EXIT.
EXIT.

4510-D0-OFFICE.

*****
* THIS ROUTINE STORES A NEW OFFICE RECORD *
*****

*****BUILD OCCURRENCE OF OFFICE RECORD*****

MOVE 01-OFFICE-CODE TO OFFICE-CODE-0450.
MOVE 01-OFFICE-ADDRESS TO OFFICE-ADDRESS-0450.
MOVE 02-OFFICE-AREA TO OFFICE-AREA-CODE-0450.
MOVE 02-OFFICE-SPEED-DIAL TO SPEED-DIAL-0450.

PERFORM 4615-OFFICE-PHONE THRU 4615-OP-EXIT
VARYING I-CTRL FROM 1 BY 1 UNTIL I-CTRL = +4.

4520-STORE-OFFICE.
MOVE 'OFFICE' TO RECORD-SR-NAME.
MOVE 450 TO RECORD-ID.
MOVE OFFICE TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.
PERFORM DBLU-STATUS.

4590-D0-EXIT.
EXIT.

4615-OFFICE-PHONE.
IF 02-OFFICE-PHONE (I-CTRL) IS NOT NUMERIC
MOVE ZEROS TO OFFICE-PHONE-0450 (I-CTRL)
ELSE
MOVE 02-OFFICE-PHONE (I-CTRL)
TO OFFICE-PHONE-0450 (I-CTRL).
4615-OP-EXIT.
EXIT.

5010-D0-STRUCTURE.
```

```

*****
*   THIS ROUTINE STORES A NEW STRUCTURE RECORD.  THE OWNERS   *
*   IN THE MANAGES AND REPORTS-TO SETS MUST BE PRESENT      *
*   BY THE END OF THE RUN.                                   *
*****

*****BUILD OWNER OCCURRENCE FOR THE MANAGES SET*****

      MOVE 'MANAGES'          TO OWNER-ONE-SET.
      MOVE -1                 TO OWNER-ONE-SERIAL.
      MOVE OG-EMP-MANAGES     TO OWNER-ONE-KEY.

*****BUILD OWNER OCCURRENCE FOR SKILL-EXPERTISE SET*****

      MOVE 'REPORTS-TO'      TO OWNER-TWO-SET.
      MOVE -1                TO OWNER-TWO-SERIAL.
      MOVE OG-EMP-RPTS-TO    TO OWNER-TWO-KEY.

*****BUILD OCCURRENCE OF STRUCTURE RECORD*****

      MOVE OG-STRUCT-CODE    TO STRUCTURE-CODE-0460.
      MOVE OG-RELATION-DATE  TO STRUCTURE-DATE-0460.

5020-STORE-STRUCTURE.
      MOVE 'STRUCTURE'       TO RECORD-SR-NAME.
      MOVE 460               TO RECORD-ID.
      MOVE STRUCTURE         TO RECORD-DATA.
      CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
                          OWNER-DESCRIPTOR-ONE
                          OWNER-DESCRIPTOR-TWO.

      PERFORM DBLU-STATUS.

5090-DS-EXIT.
      EXIT.

5510-DO-INSURANCE.

*****
*   THIS ROUTINE STORES A NEW INSURANCE-PLAN RECORD          *
*****

*****BUILD RECORD OCCURRENCE OF INSURANCE-PLAN RECORD*****

      MOVE I1-INSPLAN-CODE   TO INS-PLAN-CODE-0435.
      MOVE I1-INSPLAN-CO-NAME TO INS-CO-NAME-0435.
      MOVE I2-CO-ADDRESS     TO INS-CO-ADDRESS-0435.
      MOVE I2-CO-PHONE       TO INS-CO-PHONE-0435.
      MOVE I3-GROUP-NUM      TO GROUP-NUMBER-0435.
      MOVE I3-DEDUCT         TO DEDUCT-0435.
      MOVE I3-MAX-LIFE-COST  TO MAXIMUM-LIFE-COST-0435.
      MOVE I3-FAM-COST       TO FAMILY-COST-0435.
      MOVE I3-DEP-COST       TO DEP-COST-0435.

5520-STORE-INSURANCE.
      MOVE 'INSURANCE-PLAN'  TO RECORD-SR-NAME.
      MOVE 435               TO RECORD-ID.
      MOVE INSURANCE-PLAN    TO RECORD-DATA.
      CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.
      PERFORM DBLU-STATUS.

```

5590-DI-EXIT.  
EXIT.

6010-DO-COVERAGE.

```
*****
*   THIS MODULE STORES A NEW COVERAGE RECORD.  THE OWNER IN   *
*   THE EMP-COVERAGE SET MUST BE PRESENT BY THE END OF THE   *
*   RUN.  SINCE THIS IS NOT A CALC RECORD THE SERIAL NUMBER  *
*   RETURNED FROM IDMSDBLU MUST BE SAVED, SO MEMBERS OWNED  *
*   BY THIS OCCURRENCE CAN REFER TO IT.                      *
*****
```

\*\*\*\*\*BUILD OWNER OCCURRENCE FOR EMP-COVERAGE SET\*\*\*\*\*

```
MOVE 'EMP-COVERAGE'      TO OWNER-ONE-SET.
MOVE -1                  TO OWNER-ONE-SERIAL.
MOVE C-EMP-ID            TO OWNER-ONE-KEY.
```

\*\*\*\*\*BUILD OCCURRENCE OF COVERAGE RECORD\*\*\*\*\*

```
MOVE C-DATAFIELDS        TO COVERAGE.
```

6020-STORE-COVERAGE.

```
MOVE 'COVERAGE'         TO RECORD-SR-NAME.
MOVE 400                 TO RECORD-ID.
MOVE COVERAGE            TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
                        OWNER-DESCRIPTOR-ONE.
PERFORM DBLU-STATUS.
MOVE RECORD-SERIAL      TO SAVE-COVERAGE-SERIAL.
```

6090-DC-EXIT.  
EXIT.

6510-DO-DENTAL.

```
*****
*   THIS ROUTINE STORES A NEW DENTAL-CLAIM RECORD.           *
*   THE SERIAL NUMBER OF THE OWNER IN THE COVERAGE-CLAIMS  *
*   SET MUST BE OBTAINED AND SAVED PRIOR TO AN ATTEMPT TO  *
*   STORE THIS RECORD.                                       *
*****
```

\*\*\*\*\*BUILD OWNER OCCURRENCE FOR COVERAGE-CLAIMS SET\*\*\*\*\*

```
MOVE 'COVERAGE-CLAIMS'  TO OWNER-ONE-SET.
MOVE -1                  TO OWNER-ONE-SERIAL
MOVE SAVE-COVERAGE-SERIAL TO OWNER-ONE-KEY-SERIAL.
```

\*\*\*\*\*BUILD OCCURRENCE OF DENTAL-CLAIM RECORD\*\*\*\*\*

```
MOVE L1-DC-CLAIM-DATE    TO CLAIM-DATE-0405.
MOVE L1-DC-PATIENT-NAME  TO PATIENT-NAME-0405.
MOVE L1-DC-PATIENT-DOB   TO PATIENT-BIRTH-DATE-0405.
MOVE L1-DC-SEX           TO PATIENT-SEX-0405.
MOVE L1-DC-REL-TO-EMP    TO RELATION-TO-EMPLOYEE-0405.
MOVE L1-DC-DENTIST-NAME  TO DENTIST-NAME-0405.
```

```

MOVE L2-DC-DENTIST-ADDRESS TO DENTIST-ADDRESS-0405.
MOVE L2-DC-DENTIST-LIC-NUM TO DENTIST-LICENSE-NUMBER-0405.
MOVE I-CTRL                TO NUMBER-OF-PROCEDURES-0405.

6520-STORE-DENTAL.
MOVE 'DENTAL-CLAIM'        TO RECORD-SR-NAME.
MOVE 405                   TO RECORD-ID.
MOVE DENTAL-CLAIM         TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE.
PERFORM DBLU-STATUS.

6590-DDN-EXIT.
EXIT.

7010-DO-HOSPITAL.

*****
*   THIS ROUTINE STORES A NEW HOSPITAL-CLAIM RECORD.   *
*   THE SERIAL NUMBER OF THE OWNER IN THE COVERAGE-CLAIMS *
*   SET MUST BE OBTAINED AND SAVED PRIOR TO AN ATTEMPT TO *
*   STORE THIS RECORD.                                   *
*****

*****BUILD OWNER OCCURRENCE FOR COVERAGE-CLAIMS SET*****

MOVE 'COVERAGE-CLAIMS'    TO OWNER-ONE-SET.
MOVE -1                   TO OWNER-ONE-SERIAL
MOVE SAVE-COVERAGE-SERIAL TO OWNER-ONE-KEY-SERIAL.

*****BUILD OCCURRENCE OF HOSPITAL-CLAIM RECORD*****

MOVE H1-HC-CLAIM-DATE     TO CLAIM-DATE-0430.
MOVE H1-HC-PATIENT-NAME   TO PATIENT-NAME-0430.
MOVE H1-HC-PATIENT-DOB    TO PATIENT-BIRTH-DATE-0430.
MOVE H1-HC-SEX            TO PATIENT-SEX-0430.
MOVE H1-HC-REL-TO-EMP     TO RELATION-TO-EMPLOYEE-0430.
MOVE H1-HC-HOSP-NAME      TO HOSPITAL-NAME-0430.
MOVE H2-HC-HOSP-ADDRESS   TO HOSP-ADDRESS-0430.
MOVE H2-HC-ADMIT-DATE     TO ADMIT-DATE-0430.
MOVE H2-HC-DISCH-DATE     TO DISCHARGE-DATE-0430.
MOVE H3-HC-DIAGNOSIS      TO DIAGNOSIS-0430 (1).
MOVE H4-HC-DIAGNOSIS      TO DIAGNOSIS-0430 (2).
MOVE H5-HC-WARD-DAYS      TO WARD-DAYS-0430.
MOVE H5-HC-WARD-RATE      TO WARD-RATE-0430.
MOVE H5-HC-WARD-TOTAL     TO WARD-TOTAL-0430.
MOVE H5-HC-SEMI-DAYS      TO SEMI-DAYS-0430.
MOVE H5-HC-SEMI-RATE      TO SEMI-RATE-0430.
MOVE H5-HC-SEMI-TOTAL     TO SEMI-TOTAL-0430.
MOVE H5-HC-DEL-COST       TO DELIVERY-COST-0430.
MOVE H5-HC-ANESTH-COST    TO ANESTHESIA-COST-0430.
MOVE H5-HC-LAB-COST       TO LAB-COST-0430.

7020-STORE-HOSPITAL.
MOVE 'HOSPITAL-CLAIM'     TO RECORD-SR-NAME.
MOVE 430                   TO RECORD-ID.
MOVE HOSPITAL-CLAIM       TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE.

```



```
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.

GOBACK.

*****
DBLU-STATUS SECTION.
*****
IF RECORD-LOAD-STATUS NOT = '0000'
  DISPLAY 'LOAD STATUS ----- ' RECORD-LOAD-STATUS
  DISPLAY 'RECORD NAME ----- ' RECORD-SR-NAME
  DISPLAY 'RECORD ID ----- ' RECORD-ID
  DISPLAY 'RECORD SERIAL NO.-- ' RECORD-SERIAL
  DISPLAY 'SUGGESTED PAGE ---- ' RECORD-SUGGESTED-PAGE
  DISPLAY 'RECORD DATA ----- ' RECORD-DATA-REDEF
  DISPLAY '***** '
DBLU-STATUS-EXIT.
EXIT.
```

# Appendix B. IDMSRSTT Macro Statements

---

- B.1 Overview . . . . . B-3
- B.2 IDMSRSTT BUFSIZE . . . . . B-5
- B.3 IDMSRSTT RECNAME . . . . . B-6
- B.4 IDMSRSTT SETPTR . . . . . B-9
- B.5 IDMSRSTT FIELD . . . . . B-11
- B.6 IDMSRSTT END . . . . . B-14
- B.7 END . . . . . B-15



## B.1 Overview

**What IDMSRSTT macro statements do:** IDMSRSTT macro statements define the changes to be made to the database during a restructure operation. The statements reflect the information in two schemas:

- **An old schema** that describes the database before restructuring
- **An new schema** that describes the database after restructuring

You can code the IDMSRSTT macro statements manually, or you can use the IDMSRSTC utility to generate the statements automatically. In either case, you must assemble the statements into a base restructuring table, which is then used by the RESTRUCTURE SEGMENT and RESTRUCTURE CONNECT utility statements.

►► For more information on generating the IDMSRSTT macro statements automatically, see 5.6, “IDMSRSTC.” For more information on restructuring a database, see 4.26, “RESTRUCTURE SEGMENT” and 4.25, “RESTRUCTURE CONNECT.”

**Statement descriptions:** You assemble a base restructuring table from the following statements:

Statement	Description
IDMSRSTT BUFSIZE	One for a base restructuring table
IDMSRSTT RECNAME	One for each record type being modified
IDMSRSTT SETPTR	For each record type being modified: <ul style="list-style-type: none"> <li>▪ One for each pointer in the record if any pointers are being changed</li> <li>▪ One that specifies ALL if no pointers are being changed</li> </ul>
IDMSRSTT FIELD	For each record type being modified: <ul style="list-style-type: none"> <li>▪ One for each field or group of contiguous fields if the record is variable length or if the record is fixed length and one or more fields are being changed</li> <li>▪ One that specifies ALL if the record is fixed length and no fields are being changed</li> </ul>
IDMSRSTT END	One for each base restructuring table
END	One for each base restructuring table

**Coding considerations:** Use the following conventions when reviewing, modifying, or manually coding IDMSRSTT macro statements:

- Code each statement in uppercase on a separate line, beginning in column 2 or greater
- Do not code past column 72 on any line
- To continue a statement on another line:
  - Put an X in column 72 of the line to be continued
  - Begin the continuation line in column 16
- Use commas as place-holders for omitted macro parameters
- Begin comment lines with an asterisk (\*) in column 1

You can find the values required by the macro parameters in the old and new schema definitions and in reports generated by the IDMSRPTS utility.

▶▶ For descriptions of the IDMSRPTS reports, see 5.5, “IDMSRPTS.”

---

## B.2 IDMSRSTT BUFSIZE

**Purpose:** Specifies the sizes of the record buffers to be used during IDMSRSTU processing.

### Syntax

►— IDMSRSTT BUFSIZE = ( new-buffer-size ) —►

### Parameters

#### new-buffer-size

Specifies the size, in bytes, of a buffer large enough to hold the largest restructured record, as defined in the new schema.

**Usage:** IDMSRSTT BUFSIZE must be the first macro statement coded for each base restructuring table you assemble.

**IDMSRSTC buffer-size values:** The IDMSRSTC utility calculates the values of *old-buffer-size* and *new-buffer-size* to be the length of the largest record in the applicable schema, rounded up to the nearest multiple of four. The record length equals:

- The length of the prefix
- Plus the length of the data portion
- Plus, for variable-length records, an 8-byte overhead

**Estimating buffer-size values:** When coding the IDMSRSTT BUFSIZE statement manually, make a generous estimate for the *old-buffer-size* and *new-buffer-size* parameters. For example, round the calculated record length up to the nearest multiple of 100.

## B.3 IDMSRSTT RECNAME

**Purpose:** Identifies a record type that is being restructured and provides new format and length information for the record, if applicable. The IDMSRSTT RECNAME statement also names database procedures to be executed during the restructure process.

### Syntax

```

IDMSRSTT RECNAME = record-name
,MINLEN = ( min-root-length,min-fragment-length,max-data-length )
           FIXED
DCT = dctname
,NUPROCS = ( procedure-name )

```

### Parameters

#### RECNAME = record-name

Specifies the name of the record being restructured.

#### MINLEN =

New format and/or length information for the record.

#### min-root-length

Applies only to variable-length records and fixed-length compressed records.

The minimum root length, in bytes, of the record, rounded up to the nearest multiple of four, with the following qualifications:

- This value **does not include** the four-byte variable-length indicator for variable-length records nor the record prefix.
- If the record is being changed from fixed-length uncompressed, *min-root-length* must be at least four.
- If the minimum root length is not being changed, *min-root-length* must be zero.

#### min-fragment-length:

The minimum fragment length, in bytes, for the record, **not including** the fragment prefix. Must be at least four.

#### max-data-length:

The maximum data length, in bytes, for the record, **not including** the four-byte variable-length indicator, rounded up to the nearest multiple of four.

#### FIXED

Specifies that the record is being changed from variable-length or fixed-length compressed to fixed-length uncompressed.

**DCT =**

Specifies a PressPack DCT to use when:

- Converting the record from uncompressed to PressPack compressed
- Converting a PressPack compressed record from another PressPack DCT to the specified PressPack DCT.

**dctname:**

The name of the PressPack DCT.

**NUPROCS =**

Specifies one or more database procedures to be executed **during** the restructure process. The procedures are executed just before IDMSRSTU uses a MODIFY statement to rewrite the restructured record to the database.

**procedure-name:**

The name of the procedure.

**Usage:**

**Account for all changing records:** Code one RECNAME statement for each:

- Record being restructured
- Procedure to be executed before IDMSRSTU uses a MODIFY statement to rewrite the restructured record

**Length parameters:** All lengths are specified in bytes, rounded up to the nearest multiple of four.

**Compressing records with DCT =:** Records to be converted to PressPack compressed must be variable length records, or have a new format specified by the MINLEN parameter. You cannot specify MINLEN = FIXED for such records.

Records being converted to PressPack compressed should not have any Before Modify procedures defined that expect to see uncompressed data. The record will already be compressed before the Before procedures are called.

RESTRUCTURE SEGMENT will issue a warning if it encounters Before Modify procedures, but it will continue processing.

**Uncompressing PressPack compressed records:** To convert records from PressPack compressed records to fixed uncompressed, specify MINLEN = FIXED.

**When to use NUPROCS =:** List all procedures to be executed for a record during the restructure process **before a MODIFY statement**, as follows:

- A procedure specified in the new schema should be included if it is not included in the old schema, as long as the procedure is to be executed before a MODIFY statement. For example, if CALL IDMSCOMP BEFORE MODIFY is specified for the record in the new schema, include IDMSCOMP.

**Note:** If a procedure is specified in the new schema, but is not called before a MODIFY statement, do not include the procedure here. For example, if CALL IDMSDCOM AFTER GET is specified for the record in the new schema, IDMSDCOM should not be included here.

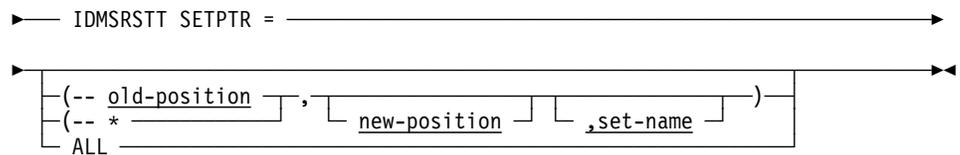
- A procedure that is not specified in the new schema should be included if the procedure is to be executed before a MODIFY statement during the restructure process; such procedures must be added manually to the IDMSRSTT macro statements generated by IDMSRSTC. For example, a procedure that initializes a new data item specified in an IDMSRSTT FIELD statement for the record should be included.

## B.4 IDMSRSTT SETPTR

**Purpose:** Specifies pointer positions for a restructured record. Using the IDMSRSTT SETPTR statement, you can:

- Add new pointers
- Delete existing pointers
- Initialize existing pointers
- Copy existing pointers to new positions

### Syntax



### Parameters

#### old-position:

Specifies the position of the pointer in the original record.

#### \*:

Specifies that the pointer is new and it's being added to the owner record.

**Note:** If a new pointer is being added to the member record, do *not* specify *old-position* or '\*'; omit the parameter entirely.

#### new-position:

Specifies the new position for a new pointer or for a pointer being moved.

If the pointer is being deleted, do not specify a new position.

#### ,set-name:

The name of an **existing** set for which prior pointers are being added.

**Note:** Do not use this parameter if owner pointers are being added to an existing set.

#### **ALL**

Specifies that no pointers in the record are being added, deleted, or changed.

### Usage:

**Accounting for all pointers:** The IDMSRSTT SETPTR statements you code for a record must account for *all* the pointers in the record:

- If you are adding, modifying, or deleting any pointers in the record, code one IDMSRSTT SETPTR statement for each pointer in the record
- If you are not adding, modifying, or deleting any pointers in the record, code a single IDMSRSTT SETPTR=ALL statement

**Calculating pointer positions:** The first user pointer position in a record is 1.

Do not count system-maintained pointers (for example, those for the CALC set) when calculating pointer positions.

**Order of changes:** You can modify pointers in any order and with any sequence of modifications. For example, you can add a new pointer in position 4 before or after deleting an existing pointer in position 2.

**Overwriting existing pointers:** When you add a new pointer or copy an existing pointer to a position that already contains a pointer, the pointer in the position is overwritten.

## B.5 IDMSRSTT FIELD

**Purpose:** Identifies the data fields in a restructured record. Using the IDMSRSTT FIELD statement, you can:

- Add new data fields
- Delete existing data fields
- Move existing data fields to new positions

### Syntax

```

IDMSRSTT FIELD =
( field-specification ) ,OCC = occurrence-count
ALL DEP

```

### Expansion of field-specification:

```

old-displacement
new-field-value
,new-displacement
,field-length
,NEW
,CTRL

```

### Parameters

#### field-specification

Specifies a field or group of contiguous fields.

For the purposes of IDMSRSTT, a group of contiguous fields, which stays together as a group and undergoes changes as a group, can be treated as a single field.

See the expanded parameters below.

#### **,OCC =**

Identifies the field as occurring more than once in the restructured record.

#### **occurrence-count:**

The number of occurrences of the field, as defined by an OCCURS clause for the field in the new schema.

For a contiguous group of fields, they must all be defined by identical OCCURS clauses in the new schema.

**DEP**

Indicates that the field is defined by an OCCURS DEPENDING ON clause in the new schema.

For a contiguous group of fields, they must all be defined by identical OCCURS DEPENDING ON clauses in the new schema.

**ALL**

Specifies that all data items in the original record are duplicated in the new record, with no additions, deletions, or changes and the control length of the record is not being changed. The control length of a record may change if a set is being changed from sorted to unsorted, unsorted to sorted, or if a sorted set, index key, or CALC key is being added or removed from the record.

**old-displacement:**

For previously existing fields only.

Specifies the beginning location of the field in the original record.

**new-field-value:**

For new fields only.

Specifies the initial value of the new field, expressed as an Assembler constant.

See "Usage" below.

**,new-displacement:**

Specifies the beginning location of the field in the restructured record.

**,field-length:**

Specifies the length, in bytes, of the field.

For a field that occurs more than once, *field-length* specifies the length of a single occurrence.

**,NEW**

Specifies that the field is a new field being added to the restructured record.

**,CTRL**

Specifies that the field is a control field (CALC key, sort key, or index key). This parameter can be used for any type of record, but *must* be specified if ALL is not specified, control fields exist within the record, and the record is:

- Fixed-length compressed
- Variable-length (compressed or uncompressed)
- Associated with a database procedure

If specified, CTRL need only be included on the last control field within the record. If you choose, you can specify it on all control fields.

**Usage:**

**Contiguous fields:** If a contiguous group of fields remains contiguous and internally unchanged in all occurrences, it can be treated as if it were a single field. Thus *field-length*, *old-displacement*, and *new-displacement* can refer to the entire group.

If any part of the group is changed, or occurring multiply without maintaining the same relationship with the rest of the group, then it must be treated separately.

**Accounting for all data fields:** The IDMSRSTT FIELD statements you code for a record must account for *all* the data fields in the record:

- If you are adding, modifying, or deleting any data fields in the record, or if the control field of the record is changing, code one IDMSRSTT FIELD statement for each field or group of contiguous fields in the record
- If you are not adding, modifying, or deleting any data fields in the record, and the control length of the record is not changing, code a single IDMSRSTT FIELD=ALL statement

**Calculating data field positions:** The first byte of data in a record is at location 1. When calculating field positions for variable-length records, do not count the 4-byte variable-length indicator maintained at the beginning of the data portion.

**Specifying new field values:** The value for *new-field-value* is expressed as an Assembler constant (for example, CL2' ', indicating a 2-byte field initialized to spaces).

For new fields that have not been assigned an initial value in the new schema, IDMSRSTC supplies an initial value based on the usage specification in the schema, as follows:

Usage	Initial value
DISPLAY, (PIC X), BIT, or POINTER	CL $x$ ' '
DISPLAY, (PIC 9)	$x$ CL1'0'
COMP (BINARY)	XL $x$ '0'
COMP-1 (SHORT-POINT)	F'0'
COMP-2 (LONG-POINT)	D'0'
COMP-3 (PACKED)	PL $x$ '0'

$X$  is the length, in bytes, of the field.

**Expansion to the right of a decimal point:** The restructure utility as a general rule cannot expand a field to the right of a decimal point. To do this the standard procedure would be to first add the needed bytes to the left of the field. After the RESTRUCTURE has been performed, a user-written program must modify each affected record by multiplying the expanded fields by the proper factor to realign the decimal point to the new position.

## B.6 IDMSRSTT END

**Purpose:** Terminates the restructuring information.

IDMSRSTT END must be the next to last macro statement coded for each base restructuring table you assemble.

**Syntax**

▶— IDMSRSTT END —————▶

## B.7 END

**Purpose:** Terminates the assembler input.

END must be the last macro statement coded for each base restructuring table you assemble.

**Syntax**

▶— END —————▶◀



# Index

---

## Special Characters

=COPY facility 7-4

## A

### ARCHIVE JOURNAL

- condense processing 4-7
- JCL (BS2000/OSD) 9-8
- JCL (CMS) 8-6
- JCL (OS/390) 6-6
- JCL (VSE/ESA) 7-9
- JCL considerations 4-10
- offloading a single file 4-8
- offloading multiple files 4-8
- sample input 4-11
- sample output 4-11
- syntax 4-7

### ARCHIVE LOG

- for an active system 4-12
- JCL (BS2000/OSD) 9-8
- JCL (CMS) 8-6
- JCL (OS/390) 6-6
- JCL (VSE/ESA) 7-9
- JCL considerations 4-13
- PRINT option 4-12
- sample input 4-13
- sample output 4-13
- syntax 4-12

area maintenance utilities 2-5

## B

### BACKUP

- by area 4-15
- by file 4-16
- JCL (BS2000/OSD) 9-9
- JCL (CMS) 8-7
- JCL (OS/390) 6-7
- JCL (VSE/ESA) 7-9
- JCL considerations 4-17
- locking areas 4-17
- sample input by area 4-17
- sample input by file 4-17
- sample output 4-18
- syntax 4-15

backup and recovery utilities 2-4

batch command facility

*See* IDMSBCF

### BUILD

- JCL (BS2000/OSD) 9-9
- JCL (CMS) 8-7
- JCL (OS/390) 6-7
- JCL (VSE/ESA) 7-10
- JCL considerations 4-25
- NO VALIDATE option 4-24
- output after each step 4-24
- sample input 4-25
- sample output 4-26
- size of intermediate files 4-24
- syntax for complete BUILD 4-20
- syntax for stepped BUILD 4-21
- use of intermediate work files 4-24

## C

CA-IDMS Batch Command Facility

*See* IDMSBCF

Callable Restructure Utility 4-171

considerations 4-171

parameters 4-172

chains 5-14

### CLEANUP

- JCL (BS2000/OSD) 9-10
- JCL (CMS) 8-8
- JCL (OS/390) 6-8
- JCL (VSE/ESA) 7-10
- JCL considerations 4-28
- journaling with 4-28
- sample input 4-28
- sample output 4-29
- syntax 4-27

### CONVERT PAGE

- JCL (BS2000/OSD) 9-10
- JCL (CMS) 8-8
- JCL (OS/390) 6-9
- JCL (VSE/ESA) 7-11

Convert Page Utility 4-30

examples 4-33

syntax 4-30

## D

database definition reports 5-49—5-50

---

database loading and restructuring utilities 2-6  
database reporting utilities 2-8

## E

### EXPAND PAGE

- area maps to multiple files 4-39
- JCL (BS2000/OSD) 9-10
- JCL (CMS) 8-9
- JCL (OS/390) 6-9
- JCL (VSE/ESA) 7-11
- JCL considerations 4-39
- regenerating DMCL modules 4-39
- sample input 4-39
- sample output 4-40
- syntax 4-38
- usage 4-39

### EXTRACT JOURNAL

- JCL (BS2000/OSD) 9-11
- JCL (CMS) 8-9
- JCL (OS/390) 6-10
- JCL (VSE/ESA) 7-11
- sample input 4-47
- sample output 4-47
- syntax 4-41
- use with ROLLFORWARD 4-185

## F

### FASTLOAD

- example 4-61
- format program 4-51
- IDMSDBL2 4-54
- IDMSDBL3 4-54
- IDMSDBL4 4-54
- IDMSDBLU 4-51
- intermediate work files 4-59
- JCL (BS2000/OSD) 9-11
- JCL (CMS) 8-10
- JCL (OS/390) 6-10
- JCL (VSE/ESA) 7-12
- JCL considerations 4-59
- preparing to run 4-51
- RELDCTL file 4-54
- restarting 4-52
- running in step mode 4-52
- sample format program A-3
- sample output 4-61
- sorting intermediate work files 4-56
- steps 4-54
- summary of input and output steps 4-54

### FASTLOAD (*continued*)

- syntax 4-50
- usage 4-51
- valid step names 4-50

### FASTLOAD format program

- functions 4-56
- IDMSDBLU 4-51
- output 4-56
- owner descriptors 4-58
- purpose 4-52
- record descriptors 4-57
- sample A-3
- specifying DBNAME 4-56
- specifying DMCL 4-56
- specifying subschema 4-56
- subschema 4-52

### FIX ARCHIVE

- archived journal files 4-64
- JCL (BS2000/OSD) 9-13
- JCL (CMS) 8-11
- JCL (OS/390) 6-12
- JCL (VSE/ESA) 7-14
- JCL considerations 4-64
- multivolume journal files 4-64
- process 4-63
- sample input 4-64
- sample output 4-64
- syntax 4-63

### FIX PAGE

- JCL (BS2000/OSD) 9-14
- JCL (CMS) 8-11
- JCL (OS/390) 6-12
- JCL (VSE/ESA) 7-14
- JCL considerations 4-67
- replacing data 4-68
- sample input 4-67
- sample output 4-68
- syntax 4-66
- varying area status 4-67
- VERIFY option 4-67
- verifying page contents 4-67

### FORMAT

- AREA parameter restriction 4-72
- by area 4-70, 4-71
- by file 4-70, 4-71
- by segment 4-71
- JCL (BS2000/OSD) 9-14
- JCL (CMS) 8-12
- JCL (OS/390) 6-13
- JCL (VSE/ESA) 7-14
- JCL considerations 4-73

---

FORMAT (*continued*)

journal files 4-71, 4-72  
native VSAM files 4-73  
reformatting 4-72  
sample input 4-73—4-74  
sample output 4-74  
SQL-defined areas 4-71  
syntax 4-70

I

IDMSBCF

coding considerations 3-4  
copying source code into 7-4  
JCL (BS2000/OSD) 9-4  
JCL (CMS) 8-4  
JCL (OS/390) 6-3  
JCL (VSE/ESA) 7-3  
statement delimiter 3-4  
SYSIDMS parameter file 1-7  
utility statements used with 1-4

IDMSCALC

argument passed 5-4  
calling from a program 5-3  
input 5-4  
output 5-4  
process 5-3  
usage 5-3

IDMSCRSU 4-171

IDMSDBAN

AREA statement 5-6, 5-9—5-10  
chains 5-14  
CMS commands 8-27  
coding requirements 5-6  
IDMSDBN1 5-13—5-14  
IDMSDBN2 5-14—5-15  
indexed sets 5-12  
JCL (BS2000/OSD) 9-30  
JCL (OS/390) 6-29  
JCL (VSE/ESA) 7-29  
locking areas 5-8, 5-12  
PROCESS statement 5-5, 5-7—5-8  
report 1 5-14  
report 1A 5-15  
report 2 5-14  
report 3 5-14  
report 4 5-14  
report 5 5-15  
REPORT statement 5-6, 5-8—5-9  
sample input 5-16  
SET statement 5-6, 5-10—5-11

IDMSDBAN (*continued*)

subschema used 5-12

IDMSDBL2 4-54

IDMSDBL3 4-54, 4-95

IDMSDBL4 4-54, 4-95

IDMSDBLU 4-51

IDMSDIRL

CMS commands 8-28  
DBL-QUOTE parameter 5-23  
JCL (BS2000/OSD) 9-31  
JCL (OS/390) 6-30  
JCL (VSE/ESA) 7-30  
JCL considerations 5-25  
purpose 5-22  
sample input 5-25  
SCHEMA-CONNECT parameter 5-23  
SCHEMA-DELETE parameter 5-24  
summary of parameters 5-23  
syntax 5-23  
usage 5-24

IDMSLBS procedure 7-37

JCL (VSE/ESA) 1-8, 7-37

IDMSLOOK

AM PROGRAM statement 5-33  
AM statement 5-32  
BIND SUBSCHEMA statement 5-31  
coding considerations 5-35  
DATES statement 5-33  
DATETIME STAMP statement 5-34  
DATETIME statement 5-34  
DBTABLE statement 5-31  
DMCL statement 5-32  
FIND statement 5-35  
HELP statement 5-29  
JCL (BS2000/OSD) 9-32  
JCL (CMS) 8-29  
JCL (OS/390) 6-31  
JCL (VSE/ESA) 7-32  
LOAD=OPSYS statement 5-30  
OPTIONAL APARS statement 5-34  
PROGRAM POOL statement 5-35  
PROGRAM statement 5-33  
RCM PROGRAM statement 5-33  
sample input 5-36  
sample output 5-36  
STORAGE POOL statement 5-34  
SUBSCHEMA statement 5-30  
summary of statements 5-28

IDMSRPTS

CMS commands 8-30  
database definition reports 5-49—5-50

---

## IDMSRPTS (continued)

- DBTABLE statement 5-57
- DMCL statement 5-56
- JCL (BS2000/OSD) 9-33
- JCL (VSE/ESA) 7-33
- REPORTS SELECTION statement 5-58
- sample DBTABLE Listing 5-64
- sample DMCL Listing 5-63
- sample input 5-61—5-62
- sample output 5-62—5-67
- sample Protocol Listing 5-62
- sample Schema Record and Set Listing 5-64
- sample SEGMENT Listing 5-63
- sample Subschema Area and Data Directory Listing 5-67
- schema reports 5-48—5-49
- SCHEMA statement 5-54
- SEGMENT statement 5-57
- SIGNON USER statement 5-53
- subschema reports 5-49
- SUBSCHEMA statement 5-55
- summary of reports 5-48
- summary of statements 5-52

## IDMSRSTC

- CMS commands 8-31
- coding considerations 5-70
- input 5-72
- JCL (BS2000/OSD) 9-34
- JCL (OS/390) 6-32
- JCL (VSE/ESA) 7-34
- output 5-72
- reviewing IDMSRSTT macro statements 5-72
- sample input 5-74
- sample output 5-74
- SCHEMA statement 5-69, 5-71
- SIGNOFF statement 5-72
- SIGNON statement 5-69, 5-70

## IDMSRSTT

- See also* IDMSRSTC
- assembly of 5-73
- calculating data field positions B-13
- calculating pointer positions B-10
- CMS commands 8-32
- coding considerations B-4
- END statement B-14
- estimating buffer size B-5
- examples B-15
- FIELD statement B-11
- IDMSRSTT BUFSIZE statement B-5
- JCL (BS2000/OSD) 9-35
- JCL (OS/390) 6-34

## IDMSRSTT (continued)

- JCL (VSE/ESA) 7-36
  - macro statements B-3
  - purpose B-3
  - RECNAME statement B-6
  - reviewing macro statements 5-72
  - SETPRT statement B-9
  - specifying new fields B-13
  - use with IDMSRSTC B-3
- ## IDMSTABX
- See* MAINTAIN INDEX
- ## indexed sets
- IDMSDBAN 5-12
  - maintaining 4-102
- ## INSTALL STAMPS
- and the FORMAT utility 4-76
  - JCL (BS2000/OSD) 9-15
  - JCL (CMS) 8-12
  - JCL (OS/390) 6-13
  - JCL (VSE/ESA) 7-16
  - JCL considerations 4-77
  - sample input 4-77
  - sample output 4-77
  - syntax 4-76
- ## integrity checking utilities 2-7

## J

### JCL (BS2000/OSD)

- ARCHIVE JOURNAL 9-8
- ARCHIVE LOG 9-8
- BACKUP 9-9
- batch command facility (IDMSBCF) 9-4
- BUILD 9-9
- CLEANUP 9-10
- CONVERT PAGE 9-10
- EXPAND PAGE 9-10
- EXTRACT JOURNAL 9-11
- FASTLOAD PAGE 9-11
- FIX ARCHIVE 9-13
- FIX PAGE 9-14
- FORMAT 9-14
- IDMSDBAN 9-30
- IDMSDIRL 9-31
- IDMSLOOK 9-32
- IDMSRPTS 9-33
- IDMSRSTC 9-34
- IDMSRSTT 9-35
- INSTALL STAMPS 9-15
- LOAD 9-16
- MAINTAIN INDEX 9-17

---

JCL (BS2000/OSD) *(continued)*

MERGE ARCHIVE 9-18  
PRINT INDEX 9-18  
PRINT JOURNAL 9-19  
PRINT LOG 9-19  
PRINT PAGE 9-19  
PRINT SPACE 9-20  
PUNCH SPACE 9-20  
RELOAD 9-21  
RESTORE SPACE 9-23  
RESTRUCTURE CONNECT 9-24  
RESTRUCTURE SEGMENT 9-24  
ROLLBACK 9-25  
ROLLFORWARD 9-25  
TUNE INDEX 9-27  
UNLOAD 9-27  
UNLOCK 9-27  
UPDATE STATISTICS 9-28  
VALIDATE 9-29

JCL (CMS)

ARCHIVE JOURNAL 8-6  
ARCHIVE LOG 8-6  
BACKUP 8-7  
BUILD 8-7  
CLEANUP 8-8  
CONVERT PAGE 8-8  
EXPAND PAGE 8-9  
EXTRACT JOURNAL 8-9  
FASTLOAD 8-10  
FIX ARCHIVE 8-11  
FIX PAGE 8-11  
FORMAT 8-12  
IDMSBCF 8-4  
IDMSDBAN 8-27  
IDMSDIRL 8-28  
IDMSLOOK 8-29  
IDMSRPTS 8-30  
IDMSRSTC 8-31  
IDMSRSTT 8-32  
INSTALL STAMPS 8-12  
LOAD 8-13  
MAINTAIN INDEX 8-14  
MERGE ARCHIVE 8-15  
PRINT INDEX 8-15  
PRINT JOURNAL 8-16  
PRINT LOG 8-16  
PRINT PAGE 8-16  
PRINT SPACE 8-17  
PUNCH 8-17  
RELOAD 8-18  
RESTORE 8-21

JCL (CMS) *(continued)*

RESTRUCTURE CONNECT 8-21  
RESTRUCTURE SEGMENT 8-22  
ROLLBACK 8-22  
ROLLFORWARD 8-23  
TUNE INDEX 8-23  
UNLOAD 8-24  
UNLOCK 8-23  
UPDATE STATISTICS 8-24  
VALIDATE 8-25

JCL (OS/390)

ARCHIVE JOURNAL 6-6  
ARCHIVE LOG 6-6  
BACKUP 6-7  
BUILD 6-7  
CLEANUP 6-8  
CONVERT PAGE 6-9  
EXPAND PAGE 6-9  
EXTRACT JOURNAL 6-10  
FASTLOAD 6-10  
FIX ARCHIVE 6-12  
FIX PAGE 6-12  
FORMAT 6-13  
IDMSBCF 6-3  
IDMSDBAN 6-29  
IDMSDIRL 6-30  
IDMSLOOK 6-31  
IDMSRSTC 6-32  
IDMSRSTT 6-34  
INSTALL STAMPS 6-13  
LOAD 6-14  
MAINTAIN INDEX 6-15  
MERGE ARCHIVE 6-16  
PRINT INDEX 6-17  
PRINT JOURNAL 6-17  
PRINT LOG 6-18  
PRINT PAGE 6-18  
PRINT SPACE 6-18  
PUNCH 6-19  
RELOAD 6-20  
RESTORE 6-23  
RESTRUCTURE CONNECT 6-23  
RESTRUCTURE SEGMENT 6-24  
ROLLBACK 6-24  
ROLLFORWARD 6-25  
TUNE INDEX 6-25  
UNLOAD 6-26  
UNLOCK 6-25  
UPDATE STATISTICS 6-27  
VALIDATE 6-27

JCL (VSE/ESA)

- ARCHIVE JOURNAL 7-9
- ARCHIVE LOG 7-9
- BACKUP 7-9
- BUILD 7-10
- CLEANUP 7-10
- CONVERT PAGE 7-11
- EXPAND PAGE 7-11
- EXTRACT JOURNAL 7-11
- FASTLOAD 7-12
- FIX ARCHIVE 7-14
- FIX PAGE 7-14
- FORMAT 7-14
- IDMSBCF 7-3
- IDMSDBAN 7-29
- IDMSDIRL 7-30
- IDMSLBLE procedure 7-37
- IDMSLOOK 7-32
- IDMSRPTS 7-33
- IDMSRSTC 7-34
- IDMSRSTT 7-36
- INSTALL STAMPS 7-16
- LOAD 7-16
- MAINTAIN INDEX 7-17
- MERGE ARCHIVE 7-19
- PRINT INDEX 7-19
- PRINT JOURNAL 7-19
- PRINT LOG 7-20
- PRINT PAGE 7-20
- PRINT SPACE 7-20
- PUNCH 7-21
- RELOAD 7-21
- RESTORE 7-23
- RESTRUCTURE CONNECT 7-24
- RESTRUCTURE SEGMENT 7-24
- ROLLBACK 7-25
- ROLLFORWARD 7-25
- sublibrary, using copied code from 7-4
- TUNE INDEX 7-26
- UNLOAD 7-27
- UNLOCK 7-26
- UPDATE STATISTICS 7-27
- VALIDATE 7-28

## L

LOAD

- intermediate work files 4-86
- JCL (BS2000/OSD) 9-16
- JCL (CMS) 8-13
- JCL (OS/390) 6-14

LOAD (*continued*)

- JCL (VSE/ESA) 7-16
- JCL considerations 4-87
- messages and codes 4-87
- output from stepped load 4-86
- sample input 4-88
- sample output 4-88
- sorting intermediate files 4-87
- syntax for complete load 4-78
- syntax for stepped load 4-83
- log maintenance utilities 2-5
- logically deleted records 4-27, 4-168, 4-195

## M

MAINTAIN INDEX

- area status 4-96
- BUILD parameter 4-95
- database record descriptor 4-106
- DELETE parameter 4-95
- end-of-file descriptor 4-106
- IDMSDBL3 4-95, 4-100
- IDMSDBL4 4-95, 4-100
- IDMSTABX 4-95, 4-100
- JCL (BS2000/OSD) 9-17
- JCL (CMS) 8-14
- JCL (OS/390) 6-15
- JCL (VSE/ESA) 7-17
- JCL considerations 4-106
- member descriptor 4-105
- modifying indexes 4-102
- owner descriptor 4-105
- REBUILD parameter 4-95
- RELDCTL file 4-100
- sample input 4-108
- sample output 4-109
- sorting intermediate work files 4-102
- steps 4-95, 4-100
- subschema descriptor 4-104
- syntax 4-93

MAX AREA

- journal files 4-72

MERGE ARCHIVE

- JCL (BS2000/OSD) 9-18
- JCL (CMS) 8-15
- JCL (OS/390) 6-16
- JCL (VSE/ESA) 7-19
- JCL considerations 4-112
- syntax 4-110
- merging archived journal files 4-110

---

modifying indexes 4-102

## P

### PRINT INDEX

JCL (BS2000/OSD) 9-18  
JCL (CMS) 8-15  
JCL (OS/390) 6-17  
JCL (VSE/ESA) 7-19  
JCL considerations 4-117  
sample input 4-117  
sample output 4-118  
syntax 4-114

### PRINT JOURNAL

date and time formats 4-122  
JCL (BS2000/OSD) 9-19  
JCL (CMS) 8-16  
JCL (OS/390) 6-17  
JCL (VSE/ESA) 7-19  
JCL considerations 4-123  
sample input 4-123  
sample output 4-123  
syntax 4-121

### PRINT LOG

date and time formats 4-126  
JCL (BS2000/OSD) 9-19  
JCL (CMS) 8-16  
JCL (OS/390) 6-18  
JCL (VSE/ESA) 7-20  
JCL considerations 4-128  
sample input 4-128  
sample output 4-129  
syntax 4-125

### PRINT PAGE

JCL (BS2000/OSD) 9-19  
JCL (CMS) 8-16  
JCL (OS/390) 6-18  
JCL (VSE/ESA) 7-20  
JCL considerations 4-134  
sample input 4-135  
sample output 4-135  
syntax 4-131

### PRINT SPACE

JCL (BS2000/OSD) 9-20  
JCL (CMS) 8-17  
JCL (OS/390) 6-18  
JCL (VSE/ESA) 7-20  
JCL considerations 4-139  
sample input 4-139  
sample output 4-139  
syntax 4-137

## PUNCH

JCL (BS2000/OSD) 9-20  
JCL (CMS) 8-17  
JCL (OS/390) 6-19  
JCL (VSE/ESA) 7-21  
JCL considerations 4-143  
sample input 4-143  
sample output 4-143  
syntax 4-142

## R

recovery 4-185

RELDCTL file 4-54, 4-200

### RELOAD

and ASF databases 4-146  
complete 4-146  
IDMSDBL2 4-144, 4-150  
IDMSDBL3 4-144, 4-150  
IDMSDBL4 4-144, 4-150  
IDMSDBLX 4-144, 4-150  
input to 4-147  
input to each step 4-151  
JCL (BS2000/OSD) 9-21  
JCL (CMS) 8-18  
JCL (OS/390) 6-20  
JCL (VSE/ESA) 7-21  
JCL considerations 4-154  
output from each step 4-151  
relationship to UNLOAD 4-147  
restarting 4-146  
sample input 4-155  
sample output 4-156  
sort parameters 4-147  
step mode 4-144, 4-146  
steps 4-150  
syntax 4-144  
usage 4-146

reload control file

*See* RELDCTL file

### RESTORE

area status 4-159  
by area 4-159  
by file 4-159  
JCL (BS2000/OSD) 9-23  
JCL (CMS) 8-21  
JCL (OS/390) 6-23  
JCL (VSE/ESA) 7-23  
JCL considerations 4-159  
locking areas 4-159  
sample input 4-160

---

## RESTORE *(continued)*

sample output 4-160  
syntax 4-158

## RESTRUCTURE CONNECT

JCL (BS2000/OSD) 9-24  
JCL (CMS) 8-21  
JCL (OS/390) 6-23  
JCL (VSE/ESA) 7-24  
JCL considerations 4-164  
sample input 4-164  
sample output 4-165  
syntax 4-162

use with RESTRUCTURE SEGMENT 4-164

varying area status 4-163

## RESTRUCTURE SEGMENT

area status 4-168  
backing up the database 4-168  
base restructuring table 4-167  
CALC and sort keys 4-168  
data compression considerations 4-169  
database keys 4-168  
database restructuring procedures 4-169  
JCL (BS2000/OSD) 9-24  
JCL (CMS) 8-22  
JCL (OS/390) 6-24  
JCL (VSE/ESA) 7-24  
JCL considerations 4-170  
logically deleted records 4-168  
native VSAM 4-168  
pointers 4-168  
relationship to RESTRUCTURE CONNECT 4-170  
sample input 4-170  
sample output 4-170  
spill file 4-168, 4-170  
syntax 4-166

## ROLLBACK

area status 4-176  
JCL (BS2000/OSD) 9-25  
JCL (CMS) 8-22  
JCL (OS/390) 6-24  
JCL (VSE/ESA) 7-25  
JCL considerations 4-178  
process 4-177  
sample input 4-179  
sample output 4-179  
sequential mode 4-178  
sorted mode 4-178  
syntax 4-173  
using VERIFY option 4-178  
VSAM files 4-178

## ROLLFORWARD

area status 4-184  
CMS considerations 4-182  
JCL (BS2000/OSD) 9-25  
JCL (CMS) 8-23  
JCL (OS/390) 6-25  
JCL (VSE/ESA) 7-25  
JCL considerations 4-187  
process 4-185  
sample input 4-188  
sample output 4-188  
sorted mode 4-185  
syntax 4-180  
use with EXTRACT JOURNAL 4-185  
VSAM files 4-185

## S

schema reports 5-48—5-49

## SEGMENT

sample input by segment 4-17

## spill file

definition 4-168  
size 4-170

sublibrary, using copied code from 7-4

subschema reports 5-49

## SYSIDMS parameter file

and IDMSBCF 1-7  
definition 1-7  
VSE/ESA JCL 7-6

## T

### TUNE INDEX

JCL (BS2000/OSD) 9-27  
JCL (CMS) 8-23  
JCL (OS/390) 6-25  
JCL (VSE/ESA) 7-26

### Tune Index Utility 4-189

output/sample report 4-191  
syntax 4-189  
usage 4-190

## U

### UNLOAD

and ASF databases 4-196  
DMCL 4-198  
JCL (BS2000/OSD) 9-27  
JCL (CMS) 8-24  
JCL (OS/390) 6-26  
JCL (VSE/ESA) 7-27

---

UNLOAD (*continued*)

- logically deleted records 4-195
- non-SQL defined database 4-195
- output files 4-199
- procedures 4-199
- process 4-196
- RELDCTL file 4-194, 4-200
- sample input 4-200
- sample output 4-201
- SEGMENT 4-198
- SQL-defined database 4-195
- SQL-type segments 4-197
- subschema 4-197
- syntax 4-193

UNLOCK

- JCL (BS2000/OSD) 9-27
- JCL (CMS) 8-23
- JCL (OS/390) 6-25
- JCL (VSE/ESA) 7-26
- JCL considerations 4-202
- sample output 4-203
- syntax 4-202

UPDATE STATISTICS

- JCL (BS2000/OSD) 9-28
- JCL (CMS) 8-24
- JCL (OS/390) 6-27
- JCL (VSE/ESA) 7-27
- JCL considerations 4-206
- pages examined 4-205
- sample input 4-206
- sample output 4-207
- statistics updated 4-205
- syntax 4-204

## V

VALIDATE

- intermediate work files 4-210
- JCL (BS2000/OSD) 9-29
- JCL (CMS) 8-25
- JCL (OS/390) 6-27
- JCL (VSE/ESA) 7-28
- JCL considerations 4-211
- sample input 4-212
- sample output 4-212
- syntax 4-208
- usage 4-210

