CA IDMS™

Utilities Guide

Release 18.5.00, 2nd Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA ADS™
- CA Common Services for z/OS (CCS)
- CA Culprit™
- CA IDMS™
- CA IDMS™/DB
- CA IDMS™/DC (DC)
- CA IDMS™/DC or CA IDMS™ UCF (DC/UCF)
- CA IDMS™ UCF (UCF)
- CA OLQ™ Online Query for CA IDMS™ (CA OLQ)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates were made for the 18.5.00 release of this documentation:

- MAINTAIN INDEX (see page 157)—The record length for SYS005 was increased to 32 and the sorting range for SORT4 was altered.
- JCL Considerations for RESTRUCTURE CONNECT (see page 308), JCL Considerations for RESTRUCTURE (see page 303), RESTRUCTURE (see page 467), RESTRUCTURE CONNECT (see page 468)—The spill file size must now be a multiple of 40 (previously it needed to be a multiple of 32).
- RESTRUCTURE (see page 467), RESTRUCTURE (see page 512), RESTRUCTURE (see page 559)—These utility statements for z/OS, z/VSE and CMS commands were renamed, the old name was RESTRUCTURE SEGMENT.
- JCL Considerations for IDMSDBAN2 (see page 379)—Added information about SYSLST.
- ARCHIVE JOURNAL (see page 440)—Updated the JCL example and definition.
- Sample Output (see page 188)—Added an explanation of how the number of entries in the top level SR8 is expressed in the report output generated by the PRINT INDEX utility.
- REORG Syntax (see page 248)—Corrected the syntax for the AREA parameter.
- Considerations for Running REORG on VSE (see page 279), JCL Considerations (see page 282), REORG (z/OS) (see page 465), REORG (z/VSE) (see page 510)—Updated to reflect that users must allocate a block size of 8192 to the REORG control file (RORGCTL).
- PRINT TRACE parameters (see page 223)— Added a note that print trace from multiple archive files is available only for z/OS.
- ARCHIVE TRACE (z/OS) (see page 441), ARCHIVE TRACE (z/VSE) (see page 488)—Added a reference to the sample JCL for the ARCHIVE TRACE utility.
- PRINT TRACE (z/OS) (see page 460), PRINT TRACE (z/VSE) (see page 506)—Added a reference to the sample JCL for the PRINT TRACE utility.
- Sample JCL (see page 23)—Added this section which explains where to find sample JCL.
- IDMSLOOK (see page 389)—Added the RHDCPINT parameter.
- =Copy Facility (see page 483)—Updated syntax to describe the new MEM=keyword.

The following documentation updates were made for the 18.5.00 release of this documentation:

- EXTRACT JOURNAL (see page 87)—Added UTC keyword.

- IDMSLOOK (see page 389)—Added new display options for DMCL, SUBSCHEMA, BIND SUBSCHEMA, BINDSQL, and AM commands. Updated sample reports to display all fields.
- MAINTAIN INDEX (see page 157)—Added DBNAME support.
- PRINT INDEX (see page 181)—Updated to include new syntax to specify a DBNAME instead of a SEGMENT name, and indexed sets are now an option.
- PRINT JOURNAL (see page 197)—Added UTC keyword.
- PRINT LOG (see page 201)—Added UTC keyword.
- PRINT TRACE (see page 222)—Added UTC keyword and new examples.
- REORG (see page 247)—Added support for DBNAME. Changed description of cross-segment-dependencies and mixed page groups.
- RESTRUCTURE CONNECT (see page 306)—Changed from RESTRUCTURE CONNECT SEGMENT and added the db-name and segment-name parameters.
- RESTRUCTURE (see page 298)—Changed from RESTRUCTURE SEGMENT and added the db-name and segment-name parameters.
- ROLLBACK (see page 310)—Added UTC keyword.
- ROLLFORWARD (see page 319)—Added UTC keyword.
- UPDATE STATISTICS (see page 355)—Updated the descriptions of the AREA and SCHEMA parameters, and the UPDATE STATISTICS for a non-SQL-defined schema section.
- ARCHIVE LOG (see page 50)—Added UTC keyword.
- ARCHIVE TRACE (see page 53)—Added UTC keyword.

Contents

Chapter 1: Introduction 21

Who Should Use This Guide 21
Using This Guide 21
Syntax Diagram Conventions 21
Sample JCL 23

Chapter 2: CA IDMS/DB Utilities 25

Overview 25
Utility Statements 25
Utility Programs 27
Securing Utility Commands 28
JCL Considerations 28

Chapter 3: Utility Operations 31

Overview 31
Using the Utility Statements 31
    Utility Descriptions 32
    SQL and Non-SQL 36
Central Version Considerations 37

Chapter 4: Command Facility Considerations 39

Overview 39
Statement Coding Considerations 39

Chapter 5: Utility Statements 43

ARCHIVE JOURNAL 44
    ARCHIVE JOURNAL Syntax 44
    ARCHIVE JOURNAL Parameter 44
    Usage 47
    JCL Considerations 48
    Example 48
    Sample Output 49
    More Information 49
ARCHIVE LOG 50
    Syntax 50

Parameter 50
Usage 50
JCL Considerations 51
Example 51
Sample Output 52
More Information 53
ARCHIVE TRACE 53
ARCHIVE TRACE Syntax 53
ARCHIVE TRACE Parameter 53
Example: Archive the Contents of a Trace Area 54
BACKUP 54
BACKUP Syntax 54
BACKUP Parameters 55
Usage 56
JCL Considerations 57
Examples 57
Sample Output 58
More Information 59
BUILD 59
BUILD Syntax 60
Build Parameter 61
Usage 63
JCL Considerations 64
Examples 65
Sample Output 66
CLEANUP 67
CLEANUP Syntax 67
CLEANUP Parameter 67
Usage 68
JCL Considerations 69
Example 69
Sample Output 69
CONVERT CATALOG 70
CONVERT CATALOG Syntax 70
Usage 70
CONVERT PAGE 71
CONVERT PAGE Syntax 71
CONVERT PAGE Parameter 72
Usage 73
JCL Considerations 74
Examples 75
Sample Output 79

More Information....81
CREATE DSMODEL....81
CREATE DSMODEL Syntax....81
CREATE DSMODEL Parameter....82
Usage....83
Example....84
EXPAND PAGE....84
EXPAND PAGE Syntax....85
EXPAND PAGE Parameter....85
Usage....85
JCL Considerations....86
Example....86
Sample Output....87
EXTRACT JOURNAL....87
EXTRACT JOURNAL Syntax....88
EXTRACT JOURNAL Parameter....88
Usage....91
JCL Considerations....95
Example....95
Sample Output....95
FASTLOAD....97
FASTLOAD Syntax....97
FASTLOAD Parameter....97
Usage....98
JCL Considerations....107
Example....109
Sample Output....110
FIX ARCHIVE....111
FIX ARCHIVE Syntax....111
FIX ARCHIVE Parameter....111
Usage....112
JCL Considerations....113
Examples....114
Sample Output....114
FIX PAGE....115
FIX PAGE Syntax....115
FIX PAGE Parameter....115
Usage....116
JCL Considerations....117
Examples....117
Sample Output....119
FORMAT....119

FORMAT Syntax 120
FORMAT Parameter 121
Usage 124
JCL Considerations 129
Examples 130
Sample Output 132
INSTALL STAMPS 133
INSTALL STAMPS Syntax 133
INSTALL STAMPS Parameter 133
Usage 134
JCL Considerations 135
Examples 135
Sample Output 135
LOAD 136
LOAD Syntax 137
LOAD Parameter 138
Usage 144
JCL Considerations 145
Examples 147
Sample Output 149
More Information 154
LOCK 154
LOCK Syntax 154
LOCK Parameter 154
Usage 155
JCL Considerations 156
Examples 156
Sample Output 156
MAINTAIN INDEX 157
MAINTAIN INDEX Syntax 157
MAINTAIN INDEX Parameter 158
Usage 161
JCL Considerations 173
Examples 174
Sample Output 175
More Information 175
MERGE ARCHIVE 176
MERGE ARCHIVE Syntax 176
MERGE ARCHIVE Parameter 177
Usage 178
JCL Considerations 179
Example 179

Sample Output 180
More Information 180
PRINT INDEX 181
Syntax: PRINT INDEX 181
PRINT INDEX Parameter 182
Usage 186
JCL Considerations 186
Examples 187
Sample Output 188
More Information 197
PRINT JOURNAL 197
PRINT JOURNAL Syntax 197
PRINT JOURNAL Parameter 198
Usage 199
JCL Considerations 200
Example 200
Sample Output 201
PRINT LOG 201
PRINT LOG Syntax 202
PRINT LOG Parameter 202
Usage 205
JCL Considerations 205
Examples 205
Sample Output 207
More Information 209
PRINT PAGE 209
PRINT PAGE Syntax 210
PRINT PAGE Parameter 210
Usage 213
JCL Considerations 214
Example 214
Sample Output 215
More Information 215
PRINT SPACE 216
PRINT SPACE Syntax 216
PRINT SPACE Parameter 216
Usage 218
JCL Considerations 218
Examples 218
Sample Output 219
PRINT TRACE 222
PRINT TRACE Syntax 222

PRINT TRACE Parameters 223
PRINT TRACE Usage 225
JCL Considerations 226
Examples 226
Sample Output 228
PUNCH 229
PUNCH Syntax 229
PUNCH Parameter 229
Usage 230
JCL Considerations 230
Example 230
Output 230
More Information 231
RELOAD 231
RELOAD Syntax 231
RELOAD Parameter 232
Usage 233
JCL Considerations 243
Example 244
Sample Output 245
REORG 247
REORG Syntax 248
REORG Parameter 249
Usage 254
JCL Considerations 282
Examples 283
Sample Output 285
DBKEYS File Layout 293
RESTORE 294
RESTORE Syntax 294
RESTORE Parameter 294
Usage 295
JCL Considerations 296
Examples 296
Sample Output 297
RESTRUCTURE 298
RESTRUCTURE Syntax 299
RESTRUCTURE Parameter 299
Usage 300
JCL Considerations 303
Examples and Sample Output 303
More Information 304

Callable Restructure Utility....304
RESTRUCTURE CONNECT....306
RESTRUCTURE CONNECT Syntax....306
RESTRUCTURE CONNECT Parameter....307
Usage....308
JCL Considerations....308
Examples and Sample Output....309
More Information....310
ROLLBACK....310
ROLLBACK Syntax....311
ROLLBACK Parameter....311
Usage....314
JCL Considerations....318
Examples....318
Sample Output....319
More Information....319
ROLLFORWARD....319
ROLLFORWARD Syntax....320
ROLLFORWARD Parameter....321
Usage....325
JCL Considerations....328
Examples....329
Sample Output....329
More Information....330
SYNCHRONIZE STAMPS....330
SYNCHRONIZE STAMPS Syntax....330
SYNCHRONIZE STAMPS Parameter....331
Usage....331
JCL Considerations....332
Examples....332
Sample Output....333
TUNE INDEX....333
TUNE INDEX Syntax....334
TUNE INDEX Parameter....335
Usage....338
JCL Considerations....339
Examples....339
Sample Output....340
More Information....341
UNLOAD....342
UNLOAD Syntax....342
UNLOAD Parameter....343

Usage 344
JCL Considerations 351
Example 351
Sample Output 352
UNLOCK 352
UNLOCK Syntax 353
UNLOCK Parameter 353
Usage 353
JCL Considerations 353
Examples 354
Sample Output 354
UPDATE STATISTICS 355
UPDATE STATISTICS Syntax 355
UPDATE STATISTICS Parameter 355
Usage 356
JCL Considerations 358
Example 358
Sample Output 359
VALIDATE 359
VALIDATE Syntax 360
VALIDATE Parameter 360
Usage 362
JCL Considerations 363
Example 364
Sample Output 364
More Information 364

Chapter 6: Utility Programs 365

IDMSCALC 365
Usage 365
Calling the IDMSCALC Routine 366
IDMSDBAN 367
Syntax 368
Input Parameter Statements 368
Usage 374
JCL Considerations 379
Example 379
Sample Output 380
More Information 385
IDMSDIRL 385
Syntax 386

Input Parameter Statements 386
Usage 387
JCL Considerations 388
Examples 389
Sample Output 389
More Information 389
IDMSLOOK 389
Syntax 390
Input Parameter Statements 391
Usage 395
JCL Considerations 396
Examples 396
Sample Output 396
More Information 403
IDMSRPTS 403
Syntax 407
Statements 408
Usage 418
JCL Considerations 418
Examples 419
Sample Output 420
IDMSRSTC 428
Syntax 428
Input Parameter Statements 429
Usage 432
Example 434
Sample Output 434
More Information 435

Chapter 7: z/OS JCL 437

Overview 437
Batch Command Facility 437
Utility Statements 440
ARCHIVE JOURNAL 440
ARCHIVE LOG 441
ARCHIVE TRACE 441
BACKUP 442
BUILD 442
CLEANUP 443
CONVERT CATALOG 444
CONVERT PAGE 445

EXPAND PAGE 446
EXTRACT JOURNAL 447
FASTLOAD 448
FIX ARCHIVE 450
FIX PAGE 450
FORMAT 451
INSTALL STAMPS 452
LOAD 453
LOCK 454
MAINTAIN INDEX 455
MERGE ARCHIVE 457
PRINT INDEX 457
PRINT JOURNAL 458
PRINT LOG 458
PRINT PAGE 459
PRINT SPACE 459
PRINT TRACE 460
PUNCH 460
RELOAD 462
REORG 465
RESTORE 467
RESTRUCTURE 467
RESTRUCTURE CONNECT 468
ROLLBACK 468
ROLLFORWARD 469
SYNCHRONIZE STAMPS 469
TUNE INDEX 470
UNLOCK 471
UNLOAD 471
UPDATE STATISTICS 472
VALIDATE 473
Utility Programs 475
IDMSDBAN 475
IDMSDIRL 476
IDMSLOOK 478
IDMSRPTS 479
IDMSRSTC 480
IDMSRSTT 482

Chapter 8: z/VSE JCL 483

Overview 483

=COPY Facility 483
IDMSLBLS Procedure 484
SYSIDMS Parameter File 485
Batch Command Facility 485
Utility Statements 487
ARCHIVE JOURNAL 487
ARCHIVE LOG 488
ARCHIVE TRACE 488
BACKUP 488
BUILD 489
CLEANUP 490
CONVERT CATALOG 491
CONVERT PAGE 492
EXPAND PAGE 492
EXTRACT JOURNAL 493
FASTLOAD 494
FIX ARCHIVE 495
FIX PAGE 496
FORMAT 496
INSTALL STAMPS 499
LOAD 500
LOCK 501
MAINTAIN INDEX 502
PRINT INDEX 503
PRINT JOURNAL 503
PRINT LOG 504
PRINT PAGE 504
PRINT SPACE 505
PRINT TRACE 506
PUNCH 507
RELOAD 508
REORG 510
RESTORE 512
RESTRUCTURE 512
RESTRUCTURE CONNECT 513
ROLLBACK 513
ROLLFORWARD 514
SYNCHRONIZE STAMPS 515
TUNE INDEX 516
UNLOCK 516
UNLOAD 517
UPDATE STATISTICS 518

VALIDATE 519
Utility Programs....520
IDMSDBAN....520
IDMSDIRL....522
IDMSLOOK....523
IDMSRPTS....525
IDMSRSTC....527
IDMSRSTT....529
IDMSLBLS Procedure....529

Chapter 9: CMS Commands 537

Overview....537
Batch Command Facility....537
Utility Statements....539
ARCHIVE JOURNAL....539
ARCHIVE LOG....540
BACKUP....540
BUILD....541
CLEANUP....542
CONVERT CATALOG....542
CONVERT PAGE....543
EXPAND PAGE....544
EXTRACT JOURNAL....544
FASTLOAD....545
FIX ARCHIVE....546
FIX PAGE....546
FORMAT....547
INSTALL STAMPS....548
LOAD....549
LOCK....550
MAINTAIN INDEX....550
MERGE ARCHIVE....552
PRINT INDEX....552
PRINT JOURNAL....552
PRINT LOG....553
PRINT PAGE....553
PRINT SPACE....554
PUNCH....555
RELOAD....556
RESTORE....558
RESTRUCTURE....559

RESTRUCTURE CONNECT 559
ROLLBACK 560
ROLLFORWARD 560
SYNCHRONIZE STAMPS 561
TUNE INDEX 562
UNLOCK 562
UNLOAD 563
UPDATE STATISTICS 564
VALIDATE 565
Utility Programs 566
IDMSDBAN 566
IDMSDIRL 567
IDMSLOOK 568
IDMSRPTS 569
IDMSRSTC 571
IDMSRSTT 572

Appendix A: FASTLOAD Format Program Sample Listing 575

Appendix B: IDMSRSTT Macro Statements 607

Overview 607
IDMSRSTT BUFSIZE 609
IDMSRSTT BUFSIZE Syntax 609
IDMSRSTT BUFSIZE Parameter 609
IDMSRSTT RECNAME 610
IDMSRSTT RECNAME Syntax 612
IDMSRSTT RECNAME Parameter 612
IDMSRSTT SETPTR 615
IDMSRSTT SETPTR Syntax 615
IDMSRSTT SETPTR Parameter 616
IDMSRSTT FIELD 617
IDMSRSTT FIELD Syntax 617
IDMSRSTT FIELD Parameter 618
IDMSRSTT FIELD Usage 621
IDMSRSTT END 622
IDMSRSTT END Syntax 622
END 622
END Syntax 622

Appendix C: Common Facilities for Distributed Transactions 625

Overview 625
Reporting on Distributed Transactions 625
Manual Recovery Input Control File 627
Manual Recovery Output Control File 629
Execution JCL Changes 629

Chapter 1: Introduction

This section contains the following topics:

Who Should Use This Guide (see page 21)
Using This Guide (see page 21)
Syntax Diagram Conventions (see page 21)
Sample JCL (see page 23)

Who Should Use This Guide

This guide is intended for database administrators (DBAs) and system administrators responsible for maintaining CA IDMS/DB databases.

Using This Guide

This guide describes reference material for using utility statements and utility programs with CA IDMS/DB Database. Information is presented as follows:

- **Chapters 2 through 4** introduce the utility statements and programs, explain how to submit the statements and programs to CA IDMS/DB, and present coding considerations.
- **Chapter 5** presents syntax, usage information, JCL considerations, and examples for each utility statement. The utility statements are in alphabetical order by statement name.
- **Chapter 6** presents syntax, usage information, JCL considerations, and examples for each utility program. The utility programs are in alphabetical order by program name.
- **Chapters 7 through 9** presents generic JCL for the batch command facility and sample operating system-specific JCL.

Syntax Diagram Conventions

The syntax diagrams presented in this guide use the following notation conventions:

`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.

italicized lowercase

Represents a value that you supply.

lowercase bold

Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.

Points to the default in a list of choices.

Indicates the beginning of a complete piece of syntax.

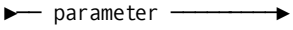
Indicates the end of a complete piece of syntax.

Indicates that the syntax continues on the next line.

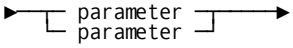
Indicates that the syntax continues on this line.

Indicates that the parameter continues on the next line.

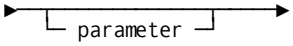
Indicates that a parameter continues on this line.



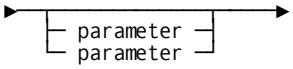
Indicates a required parameter.



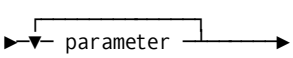
Indicates a choice of required parameters. You must select one.



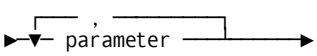
Indicates an optional parameter.



Indicates a choice of optional parameters. Select one or none.



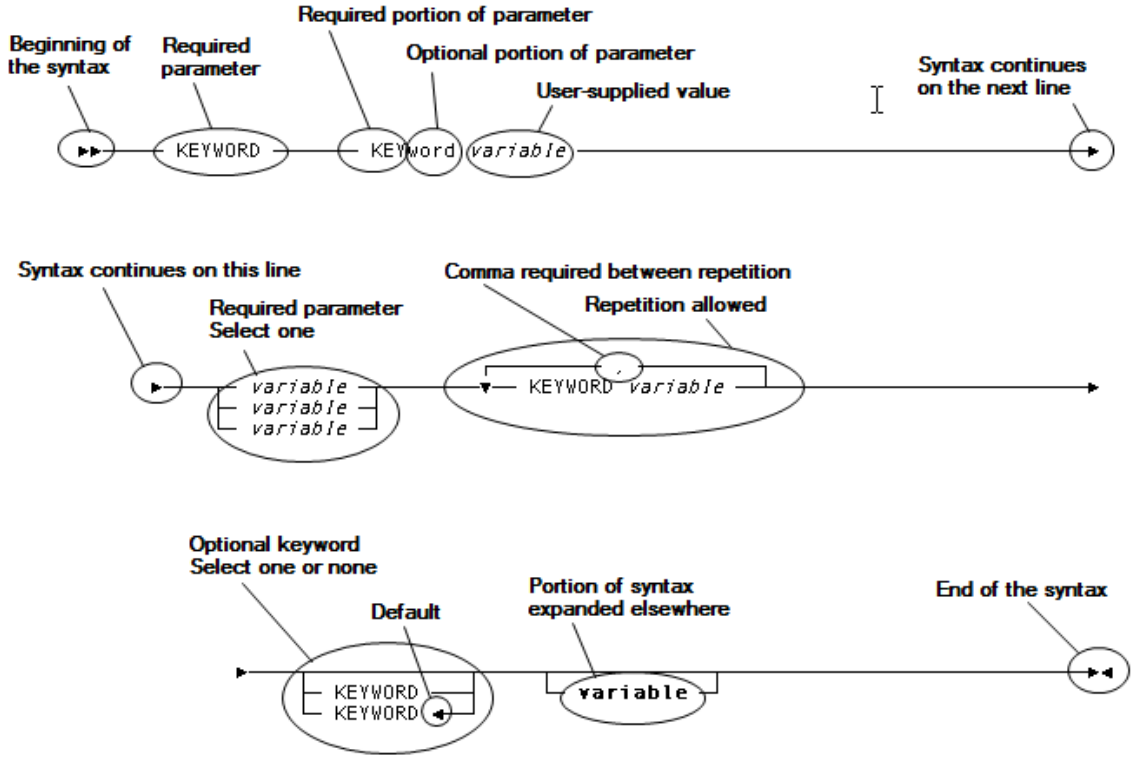
Indicates that you can repeat the parameter or specify more than one parameter.



Indicates that you must enter a comma between repetitions of the parameter.

Sample Syntax Diagram

The following sample explains how the notation conventions are used:



Sample JCL

Sample JCL, when provided, can be found at the following locations:

- z/OS—Sample JCL is installed into its own library with a lowest level qualifier of CAGJSAMP. The higher level qualifiers are determined at install time.
- z/VSE—Sample JCL is installed into the base IDMS sublibrary and the type is defined as SAMPJ. The name of the library and sublibrary is determined at install time.

Chapter 2: CA IDMS/DB Utilities

This section contains the following topics:

Overview (see page 25)
Utility Statements (see page 25)
Utility Programs (see page 27)
Securing Utility Commands (see page 28)
JCL Considerations (see page 28)

Overview

The CA IDMS/DB 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/DB 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.

Utility Statements

The following utilities can be submitted as statements to the CA IDMS/DB Batch Command Facility (IDMSBCF).

- ARCHIVE JOURNAL
- ARCHIVE LOG
- ARCHIVE TRACE
- BACKUP
- BUILD
- CLEANUP
- CONVERT CATALOG
- CONVERT PAGE
- CREATE DSMODEL
- EXPAND PAGE

- EXTRACT JOURNAL
- FASTLOAD
- FIX ARCHIVE
- FIX PAGE
- FORMAT
- INSTALL STAMPS
- LOAD
- LOCK
- MAINTAIN INDEX
- MERGE ARCHIVE
- PRINT INDEX
- PRINT JOURNAL
- PRINT LOG
- PRINT PAGE
- PRINT SPACE
- PUNCH
- RELOAD
- REORG
- RESTORE
- RESTRUCTURE CONNECT
- RESTRUCTURE
- ROLLBACK
- ROLLFORWARD
- SYNCHRONIZE STAMPS
- TUNE INDEX
- UNLOAD
- UNLOCK
- UPDATE STATISTICS
- VALIDATE

The following utility statements can be submitted through the online command facility and through the batch command facility while operating in local mode or under the central version.

- CLEANUP
- CONVERT CATALOG
- FIX PAGE
- FORMAT AREA
- FORMAT SEGMENT
- INSTALL STAMPS
- LOCK AREA
- PRINT INDEX
- PRINT PAGE
- PRINT SPACE FOR AREA
- PRINT SPACE FOR SEGMENT
- SYNCHRONIZE STAMPS
- TUNE INDEX
- UPDATE STATISTICS

Note: For considerations when executing a utility statement under central version, see Central Version Considerations (see page 37).

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

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

- IDMSDIRL
- IDMSRPTS
- IDMSRSTC

Securing Utility Commands

Individual utility commands can be secured whether they run in batch or online. See the *CA IDMS Security Administration Guide* for details.

JCL Considerations

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

Additionally, z/OS JCL (see page 437) through CMS Commands (see page 537) 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.

Note: For more information about using the CA IDMS Batch Command Facility, see the *CA IDMS Common Facilities Guide*.

Operating system-specific JCL

Sample JCL for submitting utility statements and programs is presented in the following chapters:

- z/OS JCL (see page 437)
- z/VSE JCL (see page 483)
- CMS Commands (see page 537)

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 z/VSE 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 Common Facilities Guide*.

IDMSLBLS procedure for z/VSE 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 z/VSE environment is provided during the CA IDMS installation. A copy of this procedure appears in z/VSE JCL.

Only the file definitions for work files, SYSCTL files, and tape and archive journal files appear in the sample JCL in z/VSE JCL.

For more information and considerations when executing a utility statement under central version, see Central Version Considerations (see page 37).

Chapter 3: Utility Operations

This section contains the following topics:

Overview (see page 31)
Using the Utility Statements (see page 31)
Central Version Considerations (see page 37)

Overview

This chapter presents utility statements and programs by database and system operation functions. It also presents considerations when executing utilities under central version.

Using the Utility Statements

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

Utility Descriptions

Backup and recovery utilities

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

Utility	Purpose
ARCHIVE JOURNAL	Offload disk journal files to archive files
BACKUP	Back up database areas
EXTRACT JOURNAL	Extract AFTR images from archived journal file and write 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 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

Note: For more information about backup and recovery operations, see the sections on the individual utilities in this document or see the *CA IDMS Database Administration Guide*.

Log maintenance utilities

CA IDMS/DB provides the following utilities for maintaining the DC/UCF system log:

Utility	Purpose
ARCHIVE LOG	Offload system log to archive file
ARCHIVE TRACE	Offloads the contents of the DC/UCF trace area to an archive file.

Utility	Purpose
PRINT LOG	Print all or part of a system log or archive log

Note: For more information about maintaining the system log, see the sections on the individual utilities in this document or see the *CA IDMS System Operations Guide*.

Area maintenance utilities

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

Utility	Purpose
CLEANUP	Erase logically deleted records
EXPAND PAGE	Increase page size for a database file
FIX PAGE	Verify 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
LOCK	Lock an area or segment in a batch job
PRINT PAGE	Print the contents of database pages
PRINT SPACE	Report on space utilization in areas
SYNCHRONIZE STAMPS	Display the stamps in the catalog and the data area(s) and update the stamps
TUNE INDEX	Walk a sorted index 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

Note: For more information about database area maintenance, see the chapters on the individual utilities in this document or see the *CA IDMS Database Administration Guide*.

Database loading and restructuring utilities

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

Utility	Purpose
BUILD	Build or rebuild indexes and build referential constraints for an SQL-defined database
CONVERT CATALOG	Convert the SYSTEM schema within a catalog to reflect the definition specified by the SYSTEM schema DDL delivered with the current release of CA IDMS
CONVERT PAGE	Change the page range for an area or change the maximum number of records that can be stored on a page of an area
CREATE DSMODEL	Create a temporary model of data set attributes to use for dynamic file allocation in conjunction with the REORG utility
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
REORG	Unload and reload all or part of a database
RESTRUCTURE CONNECT	Connect new prior and owner pointers in existing sets in a non-SQL-defined database
RESTRUCTURE	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

Note: For more information about loading data into the database, see the chapters on the individual utilities in this document or see the *CA IDMS Database Administration Guide*.

Integrity checking utilities

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

Utility	Purpose
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

Note: For more information about checking database integrity, see the chapters on the individual utilities in this document or see the *CA IDMS Database Administration Guide*.

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

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
- SYNCHRONIZE STAMPS
- VALIDATE

Non-SQL database only

- CLEANUP
- FASTLOAD
- MAINTAIN INDEX
- RESTRUCTURE CONNECT
- RESTRUCTURE
- IDMSRSTC

For both SQL and non-SQL databases

- ARCHIVE JOURNAL
- ARCHIVE LOG
- ARCHIVE TRACE
- BACKUP
- CONVERT CATALOG
- CONVERT PAGE
- CREATE DSMODEL
- EXPAND PAGE
- EXTRACT JOURNAL
- FIX ARCHIVE
- FIX PAGE
- FORMAT
- LOCK
- MERGE ARCHIVE

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

Central Version Considerations

The following considerations apply when executing a utility under a central version through the online or batch command facility.

Area usage mode

To execute a utility under a central version, the affected areas must be available to the central version in the appropriate mode. For utilities that perform updates, the affected areas must be in update mode to the central version. For utilities that perform only retrievals, the affected areas must be in retrieval or update mode. If the previous requirement is not met, you receive a DB002352 error message indicating that the required lock mode is not available.

Committing prior work

Before executing certain utilities through central version, you must commit any previous work that has been done within the current session. This requirement applies to the following utilities:

- FIX PAGE
- FORMAT AREA
- FORMAT SEGMENT
- LOCK AREA
- LOCK SEGMENT

The following sequence of statements illustrates how to commit prior work before issuing a FORMAT AREA statement:

```
SELECT * FROM SYSTEM.TABLE;
COMMIT;
FORMAT AREA VSAMT.KSDS2;
```

If you omit the COMMIT, you receive a DB002043 error message: Command not allowed with an open transaction.

Log messages

If you run a FORMAT statement or FIX PAGE statement under central version, an informational message is written to the log identifying the area name being updated and the time of the update.

Batch-only utilities

If you attempt to execute a utility under central version that is supported only in batch local mode, such as UNLOCK or FORMAT FILE, you receive a DB002990 error message indicating that the statement is not supported in central version.

Chapter 4: Command Facility Considerations

This section contains the following topics:

Overview (see page 39)
Statement Coding Considerations (see page 39)

Overview

This chapter presents general coding considerations when using the CA IDMS Command Facility to execute utility statements.

For a complete description of the Command Facility, see the *CA IDMS Common Facilities Guide*.

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 command facility to submit utility statements, you must terminate each statement with a statement delimiter, which by default is a semicolon (;). You can enter the delimiter either on the same line as the rest of the statement or on a separate line. For example, the two statements shown next 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".

More Information

- For more information about the use of quotation marks with identifiers, see the *CA IDMS SQL Reference Guide*.
- For more information about database entity identifiers, see the *CA IDMS Database Administration Guide*.

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 5: Utility Statements

This section contains the following topics:

ARCHIVE JOURNAL (see page 44)
ARCHIVE LOG (see page 50)
ARCHIVE TRACE (see page 53)
BACKUP (see page 54)
BUILD (see page 59)
CLEANUP (see page 67)
CONVERT CATALOG (see page 70)
CONVERT PAGE (see page 71)
CREATE DSMODEL (see page 81)
EXPAND PAGE (see page 84)
EXTRACT JOURNAL (see page 87)
FASTLOAD (see page 97)
FIX ARCHIVE (see page 111)
FIX PAGE (see page 115)
FORMAT (see page 119)
INSTALL STAMPS (see page 133)
LOAD (see page 136)
LOCK (see page 154)
MAINTAIN INDEX (see page 157)
MERGE ARCHIVE (see page 176)
PRINT INDEX (see page 181)
PRINT JOURNAL (see page 197)
PRINT LOG (see page 201)
PRINT PAGE (see page 209)
PRINT SPACE (see page 216)
PRINT TRACE (see page 222)
PUNCH (see page 229)
RELOAD (see page 231)
REORG (see page 247)
RESTORE (see page 294)
RESTRUCTURE (see page 298)
RESTRUCTURE CONNECT (see page 306)
ROLLBACK (see page 310)
ROLLFORWARD (see page 319)
SYNCHRONIZE STAMPS (see page 330)
TUNE INDEX (see page 333)
UNLOAD (see page 342)
UNLOCK (see page 352)
UPDATE STATISTICS (see page 355)
VALIDATE (see page 359)

ARCHIVE JOURNAL

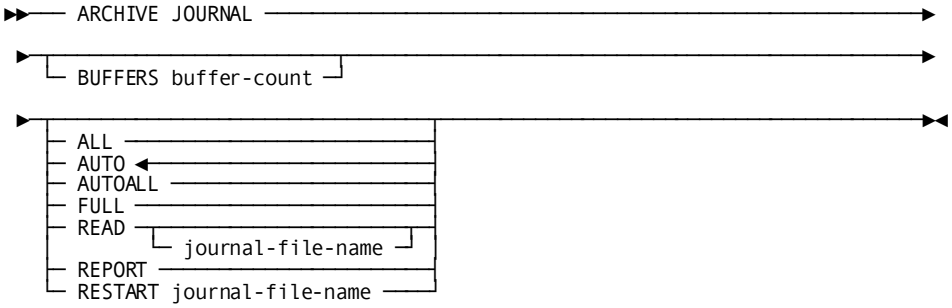
The ARCHIVE JOURNAL utility offloads, to one or more archive journal files, the entries in one or more disk journal files.

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

ARCHIVE JOURNAL Syntax



ARCHIVE JOURNAL Parameter

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: This option is not allowed while the DC/UCF system is active or after an abnormal system shutdown, 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 systems 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: This option is not allowed 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.

REPORT

Displays information about all disk journal files. No archiving or condensing of the journal files is performed.

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.

Usage

Summary of offload parameter options

In summary, if the DC/UCF system is *not* active, one or more disk journals may be processed, and after a file is offloaded, it is 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). After the file has been offloaded it is condensed and a new journal segment is created.

Parameter	While DC/UCF Active or After Abnormal Termination	After Normal Shutdown
ALL	Not Allowed	All non-empty disk journal files, beginning with the oldest file, are offloaded. Offloaded files are marked as empty.
AUTOALL	Oldest full disk journal file is offloaded. Condenses offloaded file, 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.
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 and marked as empty.
FULL	Not Allowed	All full disk journal files, beginning with the oldest file, are offloaded.

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.

Use with two-phase commit feature

ARCHIVE JOURNAL automatically preserves journal records for incomplete or unforgotten distributed transaction records when condensing a journal file.

Obtaining a status report

ARCHIVE JOURNAL always produces a report showing the status of disk journal files. Specify the REPORT option if you only want to obtain the report and you do not want to archive or condense journal information.

Use with change tracking

If the CV whose journals are being archived uses change tracking to record changes to its database environment, the JCL used to archive its journal files should reference the CV's SYSTRK files and should not include file assignments for the journal files. This ensures that any changes made to the journal files in use by the CV are known to the ARCHIVE JOURNAL utility so that it operates on the correct set of files.

Note: For more information about change tracking and the use of SYSTRK files, see Change Tracking in the *CA IDMS System Operations Guide*.

Performance consideration

A reduction in run-time for the ARCHIVE JOURNAL utility might occur if you process the journals using the QSAM access method. To invoke QSAM against the journals specify QSAMAREA=ARCHIVE.JOURNAL in the SYSIDMS file used by the ARCHIVE JOURNAL utility.

JCL Considerations

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

- The disk journal files or the CV's SYSTRK files if change tracking is in use by the CV
- The archive journal files.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

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                    mm/dd/yy   PAGE 1

ARCHIVE JOURNAL ;


    JOURNAL DISK FILES STATUS REPORT

FILENAME          SEGMENT  LORBN HIRBN   FULL  ACTIVE    STATUS    CV ACTIVE

SYSJRNL2              77     10  1000    NO     YES    NON-AJNL     YES
                      75      8     9

SYSJRNL1              76     10  1000    YES    NO     NON-AJNL     YES
                      74      8     9


WILL SELECT SYSJRNL1          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     yyyy-mm-dd-hh.ss.mm.ffffff
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

More Information

- For more information about journaling procedures, see the *CA IDMS Database Administration Guide*.
- For more information about system generation journal parameters, see the *CA IDMS System Generation Guide*.

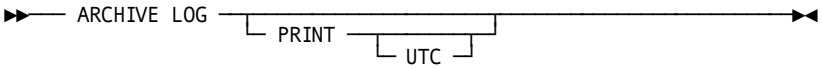
ARCHIVE LOG

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



Parameter

PRINT

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.

UTC

Displays the time stamps of each print line as UTC time. If not specified, times are displayed in local time.

Usage

How to submit the ARCHIVE LOG statement

You submit an ARCHIVE LOG statement 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 z/OS systems or DITTO for z/VSE 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.

JCL Considerations

When you submit an ARCHIVE LOG statement through the batch command facility, the JCL to execute the facility must include statements to define the following:

- Log area (DDLDCLOG)
- Message area (DDLDCMSG)
- Dummied journal file
- Archive log file being created

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Example

The following ARCHIVE LOG statement 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;
```

Sample Output

The ARCHIVE LOG utility produces the following standard listing:

```
IDMSBCF                                        IDMS Batch Command Facility                          mm/dd/77   PAGE 1

ARCHIVE LOG;
CA IDMS-DB/DC  Print Log Utility             CA IDMS-DB/DC is a Proprietary Software Product         DATE       TIME    PAGE
volser         Release nn.n                  Licensed from CA                                      mm/dd/yy   hh:mm:ss      1

*** PAGE 000030002 STATUS 0000 mmddyy  19.14.59  6 S
*** PAGE 000031001 STATUS 0000 mmddyy  14.20.00  6 S
*** PAGE 000031500 STATUS 0000 mmddyy  10.19.58  6 S
*** PAGE 000031750 STATUS 0000 mmddyy  14.30.01  6 S
*** PAGE 000031875 STATUS 0000 mmddyy  16.46.34  6 S
*** PAGE 000031937 STATUS 0000 mmddyy  17.54.59  6 S
*** PAGE 000031968 STATUS 0000 mmddyy  18.34.58  6 S
*** PAGE 000031984 STATUS 0000 mmddyy  18.54.59  6 S
*** PAGE 000031992 STATUS 0000 mmddyy  19.04.58  6 S
*** PAGE 000031996 STATUS 0000 mmddyy  19.10.00  6 S
*** PAGE 000031998 STATUS 0000 mmddyy  19.14.58  6 S
*** PAGE 000031999 STATUS 0000 mmddyy  19.14.59  6 S
*** PAGE 000032000 STATUS 0000 mmddyy  19.14.59  6 S
*** PAGE 000030002 STATUS 0000 mmddyy  19.14.59  6 S
*** PAGE 000032000 STATUS 0000 mmddyy  19.14.59  6 S
*** PAGE 000031001 STATUS 0000 mmddyy  14.20.00  6 S
*** PAGE 000031500 STATUS 0000 mmddyy  10.19.58  6 S
*** PAGE 000031250 STATUS 0000 mmddyy  16.02.43  6 S
*** PAGE 000031375 STATUS 0000 mmddyy  16.24.57  6 S
*** PAGE 000031437 STATUS 0000 mmddyy  16.30.42  6 S
*** PAGE 000031468 STATUS 0000 mmddyy  09.25.12  6 L
*** PAGE 000031484 STATUS 0000 mmddyy  09.59.59  6 S
*** PAGE 000031476 STATUS 0000 mmddyy  09.49.58  6 S
*** PAGE 000031472 STATUS 0000 mmddyy  09.29.58  6 L
*** PAGE 000031474 STATUS 0000 mmddyy  09.44.59  6 S
*** PAGE 000031473 STATUS 0000 mmddyy  09.29.58  6 L
*** PAGE 000031474 STATUS 0000 mmddyy  09.44.59  6 S
*** PAGE 000031473 STATUS 0000 mmddyy  09.29.58  6 L
*** PAGE 000031474 STATUS 0000 mmddyy  09.44.59  6 S
*** PAGE 000030474 STATUS 0000 mmddyy  05.04.59  6 S
*** PAGE 000030973 STATUS 0000 mmddyy  14.17.00  6 S
*** PAGE 000031223 STATUS 0000 mmddyy  15.59.04  6 S
*** PAGE 000031348 STATUS 0000 mmddyy  16.18.18  6 S
*** PAGE 000031410 STATUS 0000 mmddyy  16.30.41  6 S
*** PAGE 000031441 STATUS 0000 mmddyy  16.30.43  6 S
*** PAGE 000031457 STATUS 0000 mmddyy  09.25.05  4 L
*** PAGE 000031449 STATUS 0000 mmddyy  16.30.43  6 S
*** PAGE 000031453 STATUS 0000 mmddyy  16.30.44  6 S
*** PAGE 000031455 STATUS 0000 mmddyy  16.30.44  6 S
*** PAGE 000031456 STATUS 0000 mmddyy  09.23.57  1 L
*** PAGE 000031456 STATUS 0000 mmddyy  09.23.57  1 L
*** PAGE 000031473 STATUS 0000 mmddyy  09.29.58  6 L
*** PAGE 000031473 STATUS 0000 mmddyy  09.29.58  6 L
*** PAGE 000031472 STATUS 0000 mmddyy  09.29.58  6 L
*** DDLDCLOG 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
```

More Information

- For more information about defining the DC/UCF system log, see the *CA IDMS System Generation Guide*.
- For more information about maintaining the DC/UCF system log, see the *CA IDMS System Operations Guide*.
- For more information about statistics written to the DC/UCF system log, see the *CA IDMS Reports Guide*.

ARCHIVE TRACE

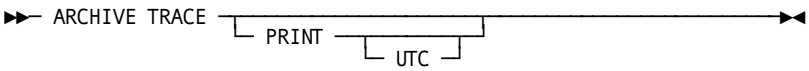
The ARCHIVE TRACE utility offloads the contents of the DC/UCF trace area to an archive file.

Authorization

To	You Need This Privilege	On
To archive trace information	DBAWRITE	SYSTEM.DDLDCTRC area

ARCHIVE TRACE Syntax

The following diagram shows the syntax for the ARCHIVE TRACE utility statement:



ARCHIVE TRACE Parameter

This section describes the parameters for the ARCHIVE TRACE utility:

PRINT

Prints a copy of the contents of the archived trace information. If you do not specify PRINT, a copy of the archived trace information is not printed.

Note: For a description of the output produced by the PRINT option of the ARCHIVE TRACE utility, see the PRINT TRACE Utility (see page 222).

UTC

Displays the time stamps of each print line as UTC time. If not specified, times are displayed in local time.

Example: Archive the Contents of a Trace Area

The following example writes the contents of a system trace area (DDLDCTRC) to an archive file and empties the area.

```
ARCHIVE TRACE
```

BACKUP

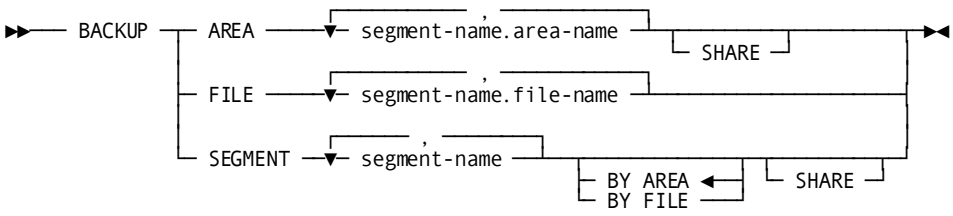
The BACKUP utility copies one or more areas in a database to a backup file. The backup file can be used later as input for a restore operation.

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

BACKUP Syntax



BACKUP 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, for example, 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. Therefore, 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.

Usage

How to submit the BACKUP statement

You submit the BACKUP statement 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, for example, for each BACKUP statement the BACKUP utility (such as, 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.

JCL Considerations

When you submit a BACKUP statement 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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

Back up by area

When the backup by area operation in the previous example is completed, the BACKUP utility provides the following report.

```
IDMSBCF  nn.n                                  CA IDMS Batch Command Facility                        mm/dd/yy   PAGE 1


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

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
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 previous example is completed, the BACKUP utility provides the following report.

```
IDMSBCF  nn.n                                  CA IDMS Batch Command Facility                        mm/dd/yy   PAGE 3


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

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
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 backup by segment

When the backup by segment operation in the previous example is completed, the BACKUP utility provides the following report.

```
IDMSBCF  nn.n                                CA IDMS Batch Command Facility                    mm/dd/yy   PAGE 5


  BACKUP  SEGMENT EMPDEMO;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
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
```

More Information

- For more information about using the SHARE option, see the *CA IDMS Database Administration Guide*.
- For more information about varying areas, see the *CA IDMS System Tasks and Operator Commands Guide*.

BUILD

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

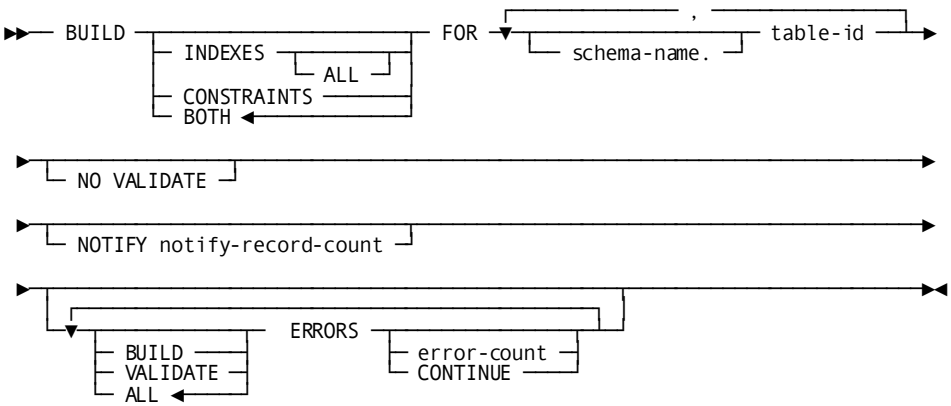
To	You need this privilege	On
Build indexes and/or referential constraints on a table	INSERT	The indexed or referencing table

For more information about designing indexes and referential constraints in SQL-defined databases, see the *Database Design Guide*.

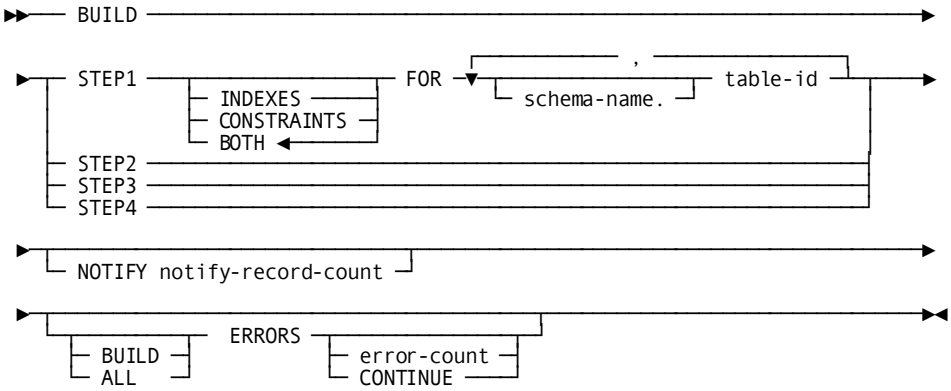
For more information about defining indexes and referential constraints in SQL-defined databases, see the *Database Administration Guide*.

BUILD Syntax

Syntax for complete BUILD



Syntax for stepped BUILD



Note: Only one LOAD, BUILD, or VALIDATE statement can be performed during one execution of the Batch Command Facility (IDMSBCF).

Build Parameter

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 this 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.

Usage

How to submit the BUILD statement

You submit the BUILD statement 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.

Note: For more information and help in deciding which options to specify, see the *Database Administration Guide*.

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.

JCL Considerations

When you submit a BUILD statement 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, if doing a complete BUILD

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 be 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
Step 1	SYS003	For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24

Step	Output	Size
	SYSPCH contains sort parameters	80 bytes
Step 2	SYS003	For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24
	SYSPCH contains sort parameters	80 bytes
Step 3	SYS003	For each prefix: 56 bytes
	SYSPCH contains sort parameters	80 bytes

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter specific to your operating system.

Examples

The following example 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;
```

Sample Output

The following report was generated after executing the BUILD statement in the previous 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.DDLCAT;
Status = 1       Extended Reason Code = 2367     Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCAT was not locked.
UNLOCK AREA SYSSQL.DDLCATX;
Status = 1       Extended Reason Code = 2367     Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCATX was not locked.
--   ****  Load data into Tables      ****
*DEBUG IDMS ON

BUILD FOR LOAD.M,
          LOAD.M2
     ERRORS CONTINUE
     NO VALIDATE;
IDMSLOAD  -   volser     SWEEP DATABASE     yy-mm-dd-hh.mm.ss
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  -   volser     CONNECT UP INDEXES     yy-mm-dd-hh.mm.ss
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
```

CLEANUP

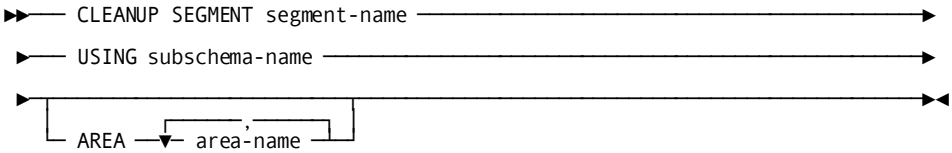
The CLEANUP utility physically erases logically deleted records from all or some areas in a database segment.

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

CLEANUP Syntax



CLEANUP Parameter

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.

Usage

How to submit the CLEANUP statement

You submit the CLEANUP statement by using either the batch command facility or the online command facility. The batch command facility can run in local mode or under central version.

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 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.

LDEL counts

The number of logically deleted records (LDELs) reported in messages UT011020 and UT011021 reflect the number of LDELs encountered by the CLEANUP utility. Due to the architecture of how LDELs are removed, multiple LDELs might get removed for each one counted by the utility. To obtain an accurate count of the number of LDELs processed by the CLEANUP utility, a DBAN or PRINT SPACE report should be created for the requested area(s), both before and after the execution of the CLEANUP utility.

JCL Considerations

When you submit a CLEANUP statement 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.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

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

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   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
```

Note: For more information about preventing logically deleted records, see the *CA IDMS Database Administration Guide*.

CONVERT CATALOG

The CONVERT CATALOG utility converts the SYSTEM schema within a catalog to reflect the definition specified by the SYSTEM schema DDL delivered with the current release of CA IDMS.

Authorization

None

CONVERT CATALOG Syntax

Usage

When to convert a catalog

After installing a new version of CA IDMS, you should convert every catalog that contains a definition of the SYSTEM schema. Typically, deferring the conversion of a catalog will result in the inability to retrieve new information using SQL queries against the catalog. It may also prevent certain CA IDMS Visual DBA functions from executing correctly.

How to convert a catalog

Through either the online or batch command facility, connect to the dictionary containing the catalog to be converted and issue the CONVERT CATALOG command. If executing the batch command facility in local mode, be sure to backup the catalog before converting it.

The CONVERT CATALOG utility can be executed more than once against the same catalog. if the target catalog has previously been converted to the current release format, no updates take place.

Prior release considerations

The CONVERT CATALOG utility can be used to upgrade a catalog from any prior release of CA IDMS starting with 12.0. Should fallback to the earlier release be necessary after a catalog is converted, there is usually no special action needed with regard to the catalog. However, check each intervening *CA IDMS Release Summary* for any special considerations that might apply when upgrading from a specific release.

Reporting successful execution

After successful execution, CA IDMS issues one of the following informational messages to indicate the status of the conversion:

- If a catalog conversion is performed, the message indicates the number of rows of each type that are changed.
- If a catalog conversion is not required, an appropriate message is issued.

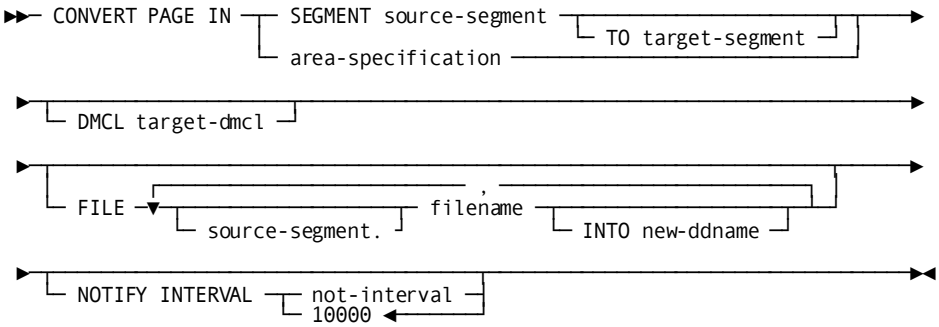
CONVERT PAGE

The CONVERT PAGE utility changes the page range for an area or changes the maximum number of records that can be stored on a page of an area.

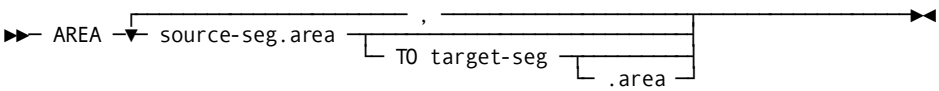
Authorization

To	You Need This Privilege	On
Change the page range for an area or the maximum number of records that can be stored on a page of an area	DBAWRITE	All areas being converted

CONVERT PAGE Syntax



Expansion of *area-specification*



CONVERT PAGE Parameter

source-segment

Names a segment to be processed. All areas in this segment are compared to matching areas in the target segment for page range and maximum records per page changes. Unless restricted by the FILE parameter, all files in this segment are scanned for page range changes and copied to new files.

source-seg.area

Names an area to be processed. All specified areas are compared to the corresponding target areas and checked for page range and maximum records per 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.

Usage

You can 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 permitting more effective use of the space on a page and more flexibility in choosing page sizes.

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.

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.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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.

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.

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.

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.

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.

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 ORGDDX;
```

Note: 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.

Sample Output

```
IDMSBCF  nn.n    CA IDMS Batch Command Facility   mm/dd/yy   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 = 00000
```

```
AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

More Information

- For more information about page ranges, see the *CA IDMS Database Administration Guide*.
- For more information about max records per page, see the *CA IDMS Database Administration Guide*.

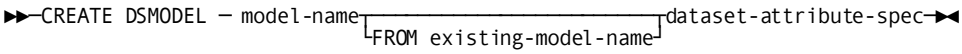
CREATE DSMODEL

The CREATE DSMODEL utility defines a temporary model of data set attributes to use for dynamic file allocation in conjunction with the REORG utility.

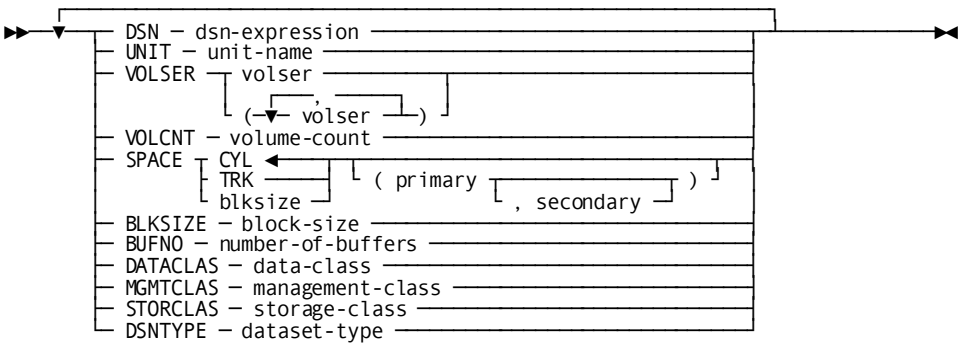
Authorization

None.

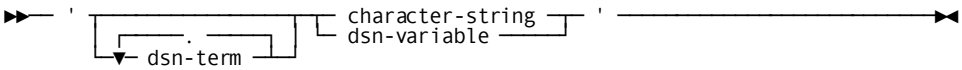
CREATE DSMODEL Syntax



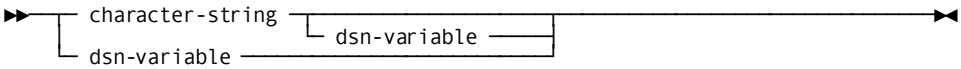
Expansion of *dataset-attribute-spec*



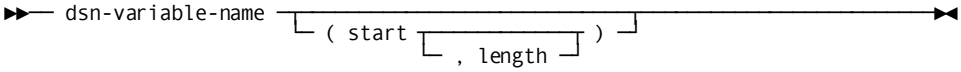
Expansion of *dsn-expression*



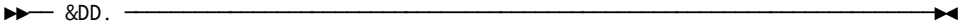
Expansion of *dsn-term*



Expansion of *dsn-variable*



Expansion of *dsn-variable-name*



CREATE DSMODEL Parameter

model-name

Specifies the name of the data set model being created. *Model-name* must be 1 to 18 characters. If the last non-blank character of *model-name* is an asterisk (*), it matches any search string whose value begins with the character string to the left of the asterisk in the model name. A model name consisting of a single asterisk matches all search keys.

existing-model-name

Specifies the name of an existing data set model whose attributes are to be copied to the model being created.

dataset-attribute-spec

Specifies attributes of the data set model.

dsn-expression

Specifies an expression used to construct data set names.

unit-name

Specifies the unit type on which the file resides or will reside.

volser

Specifies the name of a volume on which the file resides or will reside.

volume-count

Specifies the maximum number of volumes on which the file can reside. The default is the operating system default.

CYL/TRK/*blksize*

Specifies the allocation unit for a disk data set. CYL allocates the file in units of cylinders. TRK allocates the file in units of tracks. *Blksize* allocates the file in units whose size is *blksize*.

primary

Specifies the number of units in the primary allocation for a disk file.

secondary

Specifies the number of units in the secondary allocation for a disk file.

block-size

Specifies the blocksize to be used in creating (and accessing) a new file.

number-of-buffers

Specifies the number of buffers to be used in accessing a file.

data-class, management-class, storage-class, dataset-type

Specifies a data set's SMS data class, management class, storage class, and dataset type respectively.

dsn-variable-name

Specifies the name of a symbolic variable whose value can be used in whole or in part as a substitution parameter in the construction of a data set name.

&DD.

Specifies the DDNAME (linkname) through which the data set is referenced.

start

Specifies the position of the first character in the value of the symbolic variable that is to be used in the construction of the data set name. Start must not identify a position beyond the last non-blank character in the value.

length

Specifies the number of characters in the value of the symbolic variable that are to be used in the construction of the data set name. If not specified, *length* defaults to the number of non-blank characters from the start character to the end of the value. Trailing blank characters are truncated.

Usage

Using data set models

Data set models are currently used only to specify attributes of the work files used by the REORG utility. For more information about how it uses data set models, see REORG (see page 247).

Data set model duration

Data set models are transient entities that, once created, exist in memory for the life of the database session in which they are created. They are not stored in a dictionary.

Defining dummy files

You can define attributes for a dummy file by creating a data set model whose *dsn-expression* resolves to the character string NULLFILE. This is equivalent to specifying DUMMY on a DD JCL statement.

Example

The following example defines models for work files that are used by the REORG utility.

```
CREATE DSMODEL W* DSN 'DBDC.TEMP.ZIP9.&DD'. UNIT SYSDA SPACE CYL (1000,100)
BLKSIZE 8192;

CREATE DSMODEL WD* FROM W* SPACE CYL (10, 10) VOLSER DBA001 UNIT 3390;

CREATE DSMODEL WS* FROM W* SPACE CYL (3000, 10);

REORG ...
```

By default, all work files are allocated on SYSDA devices with a 1000 cylinder initial space allocation, have a block size of 8192, and a data set name of "DBDC.TEMP.ZIP9.*xxxxxxx*", where *xxxxxxx* is the file's DDNAME. DBKEY files (those whose DDNAME begins with "WD") are allocated on volume DBA001 with an initial allocation of 10 cylinders. Sorted files (those whose DDNAME begins with "WS") have an initial allocation of 3000 cylinders.

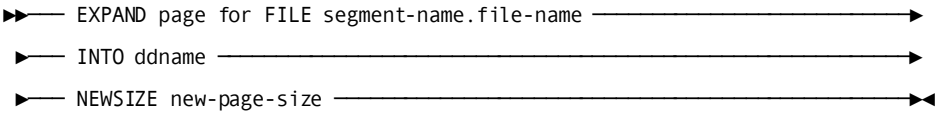
EXPAND PAGE

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.

EXPAND PAGE Syntax



EXPAND PAGE Parameter

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 z/OS or CMS name of the new file.

filename

The external z/VSE 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.

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 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. Please remember that you should only change the PAGE SIZE on the ALTER AREA segment, because altering the PRIMARY SPACE even to the current value can result in any EXTEND SPACE attribute being removed.
- Regenerate all DMCLs 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.

JCL Considerations

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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

The EXPAND PAGE utility produces the following listing after successful completion of the statement in the previous example.

```
IDMSBCF                                          IDMS Batch Command Facility                          mm/dd/yy   PAGE 1
 EXPAND FILE EMPDEMO.EMPDEMO INTO NEWFILE NEWSIZE 4820;
0UT000039 Starting Expansion of file EMPDEMO.EMPDEMO
0UT015007   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
```

Note: For more information about recompiling DMCL modules and for guidelines on when to use the EXPAND PAGE utility, see the *CA IDMS Database Administration Guide*.

EXTRACT JOURNAL

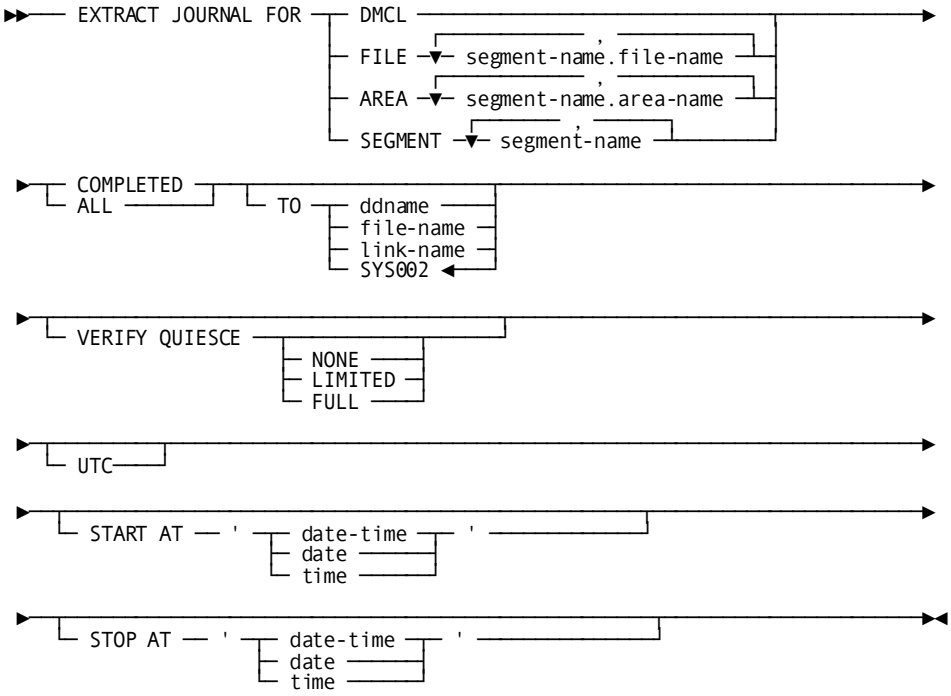
The EXTRACT JOURNAL utility conceptually extracts the most recent AFTR image for each dbkey recorded on an archived journal file and writes it to an extract file. The extract file can later be used as input to a ROLLFORWARD command for a "quick" recovery of a database area or file.

When used as part of a regular procedure to extract journal images from archived journal tapes, it allows for a quicker recovery during a critical time frame by doing much of the work during a non-critical time frame. It also speeds up recovery by extracting only the journal images it needs to recover the areas or files selected.

Authorization

DBAWRITE authority is needed for all options.

EXTRACT JOURNAL Syntax



EXTRACT JOURNAL Parameter

DMCL

Specifies that dbkeys for all areas defined in the DMCL specified in the SYSIDMS parameter file will be included in the extract file.

FILE

Specifies to include *segment-name.file-name* dbkeys in the specified file in the extract file.

AREA

Specifies to include *segment-name.area-name* dbkeys in the specified area in the extract file.

SEGMENT

Specifies to include dbkeys for all areas defined in the specified segments in the extract file.

COMPLETED

If you do not specify a STOP time, only after images for transactions that have completed by the end of the archive file will be written to the extract file. After images are written to the extract file for incomplete distributed transactions whose state is InDoubt at the end of the journal file (unless a manual recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).

If you do specify a STOP time, only after images for transactions that have completed by the first checkpoint at or following the stop time will be written to the extract file. After images are written to the extract file for incomplete distributed transactions whose state is InDoubt at the stop time (unless a manual recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).

ALL

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

If you do specify a STOP time, after images for all transactions 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 z/OS, or CMS name for the extract file.

file-name

The external z/VSE 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. 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.

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

UTC

Specifies that Start and Stop times are interpreted as UTC times instead of local times.

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.

Note: 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 or 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 from a completed transaction 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 ROLLFORWARD processing go much faster.

If a transaction does not end on the current set of input files, all AFTR images from that transaction are written to the extract file. When these extra images are processed by a subsequent EXTRACT JOURNAL or by a ROLLFORWARD command, they are included with completed AFTR images or discarded depending on the final status of that transaction. Only the most recent AFTR image taken from a completed transaction for a given dbkey is applied to the database.

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. To perform a correct recovery, you must begin the recovery operation at a quiesce point for the portion of the database being recovered. 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 that 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 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 might 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 shut down 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.

EXTRACT JOURNAL and Distributed Transactions

EXTRACT JOURNAL reports on distributed transactions and supports the use of an input manual recovery control file. The control file is used to complete InDoubt distributed transactions unless ALL is specified. For more information, see JCL Considerations (see page 95) as well as Common Facilities for Distributed Transactions (see page 625).

Note: For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Guide*

JCL Considerations

When you submit an EXTRACT 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 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.

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a manual recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

The first listing shown next, was generated after submitting the sample EXTRACT JOURNAL statement in the previous 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

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   PAGE 1


  EXTRACT JOURNAL FOR
     AREA EMPDEMO.EMP-DEMO-REGION,
          EMPDEMO.INS-DEMO-REGION,
          EMPDEMO.ORG-DEMO-REGION
     ALL
     TO SYS002
     STOP AT 'yyyy-mm-dd-hh.mm.ss.ffffff'
     ;

ROLLFORWARD STARTED yyyy-mm-dd-hh.mm.ss.ffffff
 RU_ID  00000001  PGM_ID  EMPLOAD   QUIESCE LEVELS 01  UPD 00   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 RU_ID  00000001  PGM_ID  EMPLOAD   QUIESCE LEVELS 00  UPD 00   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff

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                    mm/dd/yy   PAGE 1


  ROLLFORWARD
      SEGMENT EMPDEMO
  FROM EXTRACT SYS002
  ;

ROLLFORWARD STARTED yyyy-mm-dd-hh.mm.ss.ffffff
IMAGE FILE CREATED ON yyyy-mm-dd-hh.mm.ss.ffffff
  EXTRACT FILE IMAGES CREATED FROM yyyy-mm-dd-hh.mm.ss.ffffff TO yyyy-mm-dd-hh.mm.ss.ffffff

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                    mm/dd/yy   PAGE 2



AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

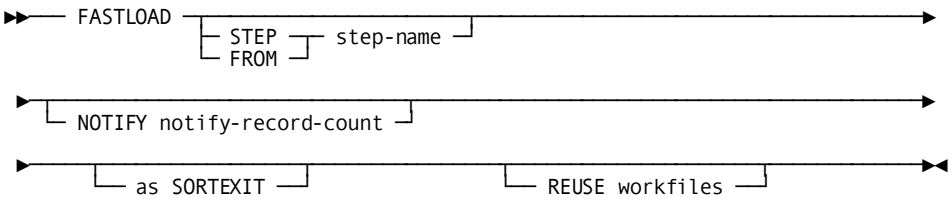
FASTLOAD

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

FASTLOAD Syntax



FASTLOAD Parameter

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 DBL*x* 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.

Usage

How to submit the FASTLOAD statement

You submit the FASTLOAD statement 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 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 (for example, 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.

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 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 restart a step only 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 data set 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 SORTEXIT option is used, the SORT*x* and IDMSDBLx 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 an 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 data set 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	■ Stores records in the database using the output from SORT1,but does not connect any sets. ■ Creates an intermediate work file for use by SORT3. ■ Prints statistics on the records written to the database.
SORT3	Sorts the file produced by IDMSDBL2.
IDMSDBL3	■ Establishes pointers for each chained setin which each record participates (that is, builds the record prefix) but does not write the prefixes to the database. ■ Creates an intermediate work file for use by SORT4. ■ Builds indexes in the database.
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	■ SYS001 contains sort control parameters from the SYSPCHfile after running the format program ■ SYS002 from running the format program	SYS004 contains the sorted contents of SYS002
IDMSDBL2	■ SYS004 from SORT1 ■ RELDCTL file from the format program;RELDCTL contains control and set information	■ SYSPCH contains sort control parameters ■ SYS005 contains set membership information ■ SYSLST contains statistics on records written to the database

Step	Input	Output
Note: Overflow statistics may be fewer than expected. 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.		
SORT3	■ SYS001 contains sort control parameters from IDMSDBL2.If running all steps at once, no sort parameters are needed. ■ SYS005 from IDMSDBL2	SYS009 contains the sorted contents of SYS005
IDMSDBL3	■ SYS009 from SORT3 ■ RELDCTL file from the format program;RELDCTL contains control and set information	■ SYSPCH contains sort control parameters ■ SYS010 contains pointer descriptors
SORT4	■ SYS001 contains sort control parameters from IDMSDBL3.If running all steps at once, no sort parameters are needed. ■ SYS010 from IDMSDBL3	SYS011 contains the sorted contents of SYS010
IDMSDBL4	■ SYS011 from SORT4 ■ RELDCTL file from the format program;RELDCTL contains control and set information	

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 programs 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 *n*) *n* = 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 'EMPDMCL'.

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, and 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.

Field	Usage	Size	Description
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: ■ If the location mode is either CALC or VIA a CALC record, IDMSDBLU determines and returns the suggested page number. ■ 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. ■ 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.
	Binary	4 bytes	Binary zeros.
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.
	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.

JCL Considerations

When you submit a FASTLOAD utility 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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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
SORT4	SYS005	SYS004
IDMSDBL4	SYS004	
FASTLOAD: SORTEXIT mode and NOT reusing workfiles		
SORT1/IDMSDBL2	SYS002	SYS005
SORT3/IDMSDBL3	SYS005	SYS010
SORT4/IDMSDBL4	SYS010	
FASTLOAD: SORTEXIT 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 IDMSDBL*x* step and used as input to a SORT*x* 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.

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;
```

Sample Output

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

```
IDMSBCF                                          IDMS Batch Command Facility                          mm/dd/yy   PAGE 1

 SET BATCH WIDTH PAGE 80;
Status = 0
   FASTLOAD;
UT010002 BEGINNING PROCESSING FOR STEP SORT1
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002    12 RECORDS WERE READ FROM SYS002
UT009003    12 RECORDS WERE WRITTEN TO SYS004
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT1    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL2
UT005001 IDMSDBL2 RELEASE nn.n TAPE volser PROCESSING STARTED
UT005002 DATABASE LOAD STATISTICS
 DATABASE LOADED ON mm/dd/77 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 nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL2 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002    12 RECORDS WERE READ FROM SYS005
UT009003    12 RECORDS WERE WRITTEN TO SYS009
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT3    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE nn.n TAPE volser 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 nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT4
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002    12 RECORDS WERE READ FROM SYS010
UT009003    12 RECORDS WERE WRITTEN TO SYS011
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT4    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL4
UT007001 IDMSDBL4 RELEASE nn.n TAPE volser PROCESSING STARTED
UT007004    NO DATABASE ERRORS WERE ENCOUNTERED
UT007005    NO LOGIC ERRORS WERE ENCOUNTERED
UT007002    12 RECORDS WERE READ FROM SYS011
UT007006 IDMSDBL4 RELEASE nn.n 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
```

Note: For more information about loading a non-SQL defined database see the *CA IDMS Database Administration Guide*.

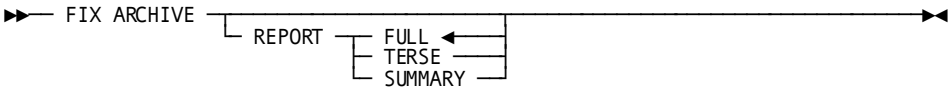
FIX ARCHIVE

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

FIX ARCHIVE Syntax



FIX ARCHIVE Parameter

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. All details of a distributed transaction record are reported. This includes local transaction ids with program names; external transaction ids, and resource manager interests. Additionally, 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. For distributed transaction records: external transaction ids, and resource manager interests are not included in the report.

SUMMARY

Indicates that only final summary information is produced.

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
- Writes additional checkpoint records to the output tape
  - For incomplete distributed transactions whose state is InDoubt at the time of the abnormal termination, additional distributed and local checkpoint records will be written to the output file to complete the transaction if a matching manual recovery control input file entry is provided.
  - For local (that is, non-distributed) transactions still active at the time of the abnormal termination, an ABRT checkpoint record will be added to the output file.

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 have to run the FIX ARCHIVE utility statement against archived journal files. Here are some exceptions:

- To identify incomplete distributed transactions whose state is InDoubt at the end of the input file. (PRINT JOURNAL can also be used for this purpose.)
- To complete transactions by writing additional checkpoint records to the output file.
  - For incomplete distributed transactions whose state is InDoubt at the end of the input file, additional distributed and local checkpoint records will be written to the output file if a matching manual recovery control input file entry is provided.
  - For local (that is, non-distributed) transactions still active at the time of the abnormal termination, an ABRT checkpoint record will be written.
- To merge multiple journal tapes onto one tape for certain rollback and rollforward operations that require multiple journal tapes to be one contiguous file

FIX ARCHIVE and Distributed Transactions

FIX ARCHIVE reports on distributed transactions and supports the use of input and output manual recovery control files. The input manual recovery control file is used to complete InDoubt distributed transactions. If an output manual recovery control file is included in the JCL, an entry will be written for each incomplete distributed transaction encountered. For more information, see JCL Considerations (see page 113) and the "Common Facilities for Distributed Transactions" appendix.

Note: For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Guide*.

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

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a manual recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

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

```
fix archive;
```

Sample Output

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

```
IDMSBCF                                    IDMS Batch Command Facility                          mm/dd/yy   PAGE 1

   FIX ARCHIVE;
 RU_ID         1  PGM_ID  EMPLOAD   QUIESECE LEVELS  1  UPD  0   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 RU_ID         1  PGM_ID  EMPLOAD   QUIESECE LEVELS  0  UPD  0   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff

  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

                                                          SINCE LAST COMMIT
   TABLESPACES OPENED                BEFORE       AFTER      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
```

Note: For more information about journaling, see the *CA IDMS Database Administration Guide*.

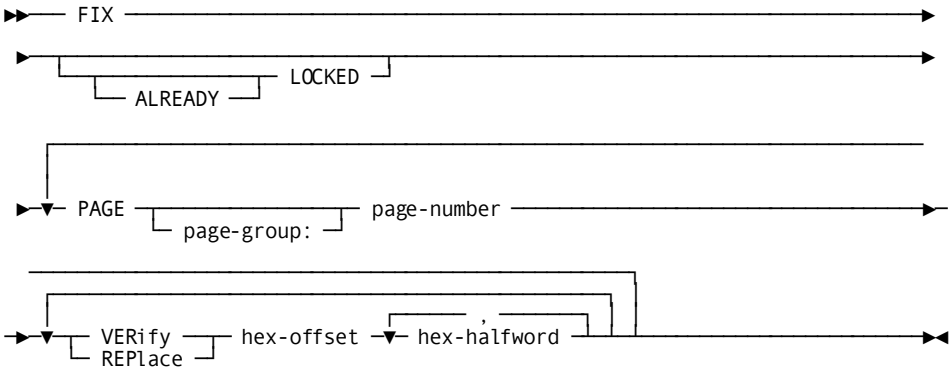
FIX PAGE

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

FIX PAGE Syntax



FIX PAGE Parameter

ALREADY LOCKED

Specifies that if the target area or areas are currently locked, processing will continue. This option is honored in local mode only. It is ignored when processing under the central version and has no effect on the processing of unlocked areas.

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.

Usage

How to submit the FIX PAGE statement

You submit the FIX PAGE statement by using either the batch command facility or the online command facility.

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 do 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.

Repairing a locked area

If a local mode application abends while an area is being updated, the lock could remain on the area. In this case, you can either explicitly unlock the area using the UNLOCK utility, or you can use the ALREADY LOCKED option.

If ALREADY LOCKED is specified and the area was locked, the area will remain locked after the FIX PAGE is completed. The ALREADY LOCKED option is not required if a page is being fixed online or through batch/CV and is ignored if specified.

Unlocking a locked area

The FIX PAGE utility cannot be used to update an area's physical area lock. Instead, use the LOCK and UNLOCK area utility statements to do this.

Committing prior work

Before executing this utility under a central version, you must commit any previous work done within the current session. For more information, see Central Version Considerations (see page 37).

JCL Considerations

When you submit a FIX PAGE statement to CA IDMS/DB through the batch command facility in local mode, the JCL to execute the facility must include statements to define the files containing the pages to be processed. To run the batch command facility under central version, include a SYSCTL statement.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

Verifying page contents

The following FIX PAGE statement 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 following FIX PAGE statement 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 following FIX PAGE statement 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;
```

Sample Output

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

```
IDMSBCF                                          IDMS Batch Command Facility                          mm/dd/yy   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 4040D3C5 C7404040 40404040  40404040 40F7F740 E2E4D5E2 C5E340E2    *Y     LEG            77 SUNSET S*
 000080   E3D9C9D7 40404040 40D5C1E3 C9C3D240  40404040 40404040 D4C1F0F2 F1F7F840    *TRIP     NATICK          MA02178 *
 0000A0   404040F6 F1F7F4F3 F2F0F9F2 F3F0F5F0  F2F8F7F7 F0F1F4F7 F7F7F1F2 F0F1F0F0    *   6174320923050 28770147771201 00*
 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
```

Note: For more information about database pages, see the *CA IDMS Database Administration Guide*.

FORMAT

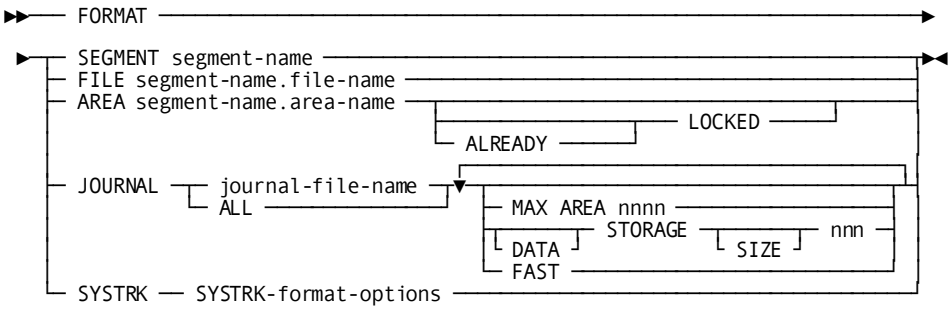
The FORMAT utility prepares a database file, area, segment, SYSTRK or disk journal file for use by CA IDMS/DB.

Authorization

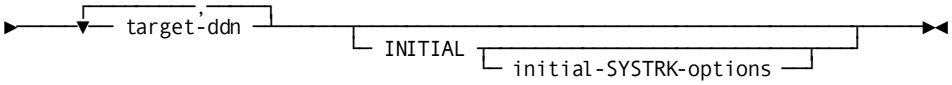
To FORMAT	You Need This Privilege	On
An area	DBAWRITE	The area
A segment	DBAWRITE	All areas of the segment

To FORMAT	You Need This Privilege	On
A file	DBAWRITE	All areas that map to the file
A disk journal file	USE	The DMCL
A SYSTRK file	USE	The DMCL

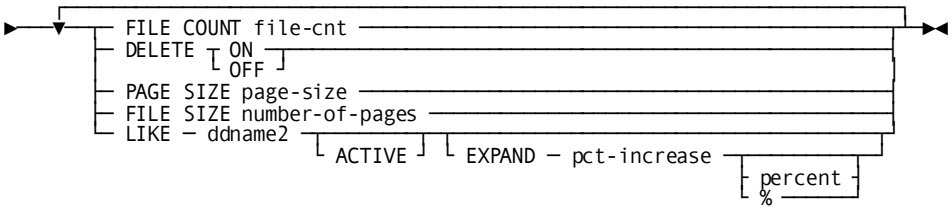
FORMAT Syntax



Expansion of *SYSTRK-format-options*



Expansion of *initial-SYSTRK-options*



FORMAT Parameter

SEGMENT

Formats all files associated with the specified segment.

segment-name

Specifies 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.

FILE

Formats the specified file.

segment-name

Identifies the segment associated with the file.

file-name

Identifies 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.

ALREADY LOCKED

Specifies that if the target area or areas are currently locked, processing will continue. This option is honored in local mode only. It is ignored when processing under the central version and has no effect on the processing of unlocked areas.

segment-name

Specifies the name of the segment associated with the area.

area-name

Specifies the name of an area included in the DMCL module.

JOURNAL

Formats the specified disk journal file.

journal-file-name

Specifies 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 100,000. The actual number of areas that CA IDMS/DB can handle can be higher because of rounding and the size of a journal block.

DATA STORAGE SIZE nnn

Specifies the amount of space to reserve in 1K (1024 bytes) increments for Data Storage in a journal file, where nnn is an integer from 1 32,767.

FAST

Formats only the journal header blocks of already existing and formatted journal files. If MAX AREA is specified, the number of JHDA entries are recalculated and the number formatted may change. If the STORAGE clause is specified, the number of JHD2 entries are recalculated and the number formatted may change.

SYSTRK

Indicates that one or more SYSTRK files are to be formatted.

target-ddn

Specifies the DDname or linkname of a SYSTRK file to be formatted.

Note: The *target-ddn* should *not* begin with the DDname prefix used for referencing SYSTRK files. Otherwise, CA IDMS attempts to use it for building the DMCL definition and fails if it cannot do so.

INITIAL

Indicates this is the first time the SYSTRK file is being formatted. If specified, no check is made to determine whether the file is in use by a CV. INITIAL must be specified the first time a file is formatted. It should not be specified when a file is being re-formatted unless you are sure that the file is not in use by a CV.

file-cnt

Specifies the number of SYSTRK files that are maintained as active mirrors. The *file-cnt* must be an integer in the range 2 through 4.

If *file-cnt* is not specified, the value is taken from the file identified by *ddname2*, if specified, or at run time from the file count currently in use by the DC/UCF system. If *file-cnt* has never been specified on a format for a related SYSTRK file, the first time that a DC/UCF system writes to a set of SYSTRK files, it sets *file-cnt* to be the lesser of the number of files currently allocated and 4. The value can be altered dynamically by a DCMT VARY CHANGE TRACKING command.

DELETE

Specifies whether the DC/UCF system is to automatically delete obsolete SYSTRK files.

ON

(z/OS and z/VM systems only) Enables automatic file deletion.

OFF

Disables automatic file deletion.

If DELETE is not specified, the value is taken from the file identified by *ddname2*, if specified, or at run time from the delete option currently in use by the DC/UCF system. If DELETE has never been specified on a format for a related SYSTRK file, the first time that a DC/UCF system writes to a set of SYSTRK files, it sets the option to OFF. The option can be altered dynamically by a DCMT VARY CHANGE TRACKING command.

page-size

Specifies the size of each page to be written to the SYSTRK file being formatted and must be an integer in the range 4088 through 32764. On z/VM, the value must be 4096. On z/OS, the maximum *page-size* is 32760. Do not specify *page-size* for VSAM files.

number-of-pages

Specifies the number of pages to be written to the SYSTRK file being formatted and must be an integer in the range 10 through 999,999.

ddname2

Specifies the DDname of a file from which the attributes and contents may be taken. If *ddname2* is specified together with either or both *page-size* and *number-of-pages*, the latter values override the respective attributes of the file identified by *ddname2*.

Note: *ddname2* should *not* begin with the DDname prefix used for referencing SYSTRK files unless the identified file contains the DMCL definition to be used during execution of the command facility.

ACTIVE

Indicates that the contents of the file identified by *ddname2* are to be copied to the file being formatted even if the file identified by *ddname2* is currently in-use by a CV.

pct-increase

Specifies the percentage increase in the number of pages written to the file being formatted over the number of pages in the file identified by *ddname2*. The *pct-increase* must be an integer in the range 0 through 1000.

Usage

How to submit the FORMAT statement

A FORMAT AREA or FORMAT SEGMENT statement can be submitted through either the batch command facility or the online command facility. The batch command facility can be run in either local mode or under the central version. When submitted in local mode, FORMAT SEGMENT will behave like a FORMAT FILE for each file in the segment. When running under central version, it will behave like a FORMAT AREA for each area in the segment.

Note: The FORMAT FILE, FORMAT SYSTRK, and FORMAT JOURNAL statements can be submitted only through 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
- SYSTRK files before they can be referenced as SYSTRK files in any CA IDMS execution JCL.

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

When not to use the FORMAT statement

Do not try to reformat a file, area, or journal that is currently active under a DC/UCF system. Do not format an existing journal if it contains journal records that might be needed for recovery.

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, 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.

A format by segment in local mode is equivalent to format by file in this regard, and can be used to format new files. Format by segment under central version is equivalent to format by area and can only be used to reformat existing areas.

Area format depends on the area lock

In local mode, 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.

Formatting a locked area

If a local mode application abends while an area is being updated, the lock could remain on the area. In this case, you can either explicitly unlock the area using the UNLOCK utility, you can format by file, or you can use the ALREADY LOCKED option.

If ALREADY LOCKED is specified and the area was locked, the area will remain locked after the format is complete. The ALREADY LOCKED option is not required if formatting an area online or through batch/CV and is ignored if specified.

Formatting by segment does not lock areas

If you format by segment in local mode, 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, except when the FAST parameter is specified. The FAST parameter, used only with previously initialized journal files, reinitializes the header files only.

Formatting disk journal files

A certain amount of space is reserved in each disk journal file for information about other systems with which a system communicates. All journal files must have the same amount of space since the data in one journal file is replicated to every other journal file.

You can specify the size or allow it to default. The actual size allocated may be higher than the value specified due to rounding. Space is allocated in blocks whose size is (journal block size minus 256). By default, one block is allocated. Additional blocks are allocated if needed until the total size meets or exceeds the size specified. If the journal block size is less than 256, no space will be reserved.

Committing prior work

Before executing this utility under a central version, you must commit any previous work done within the current session. For more information, see Central Version Considerations (see page 37).

When to use MAX AREA

Normally when a journal is formatted, a fixed number of JHDA blocks are created. 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 a Central Version can have open at one time.

The MAX AREA option lets a journal be formatted 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 might exist.

The MAX AREA option can also reduce the number of JHDA blocks that are created, which frees journal space.

To calculate the number of areas that one JHDA journal block can handle, use the following formula:

```
number_of_AREAS = (jrnl_blksize - 32) / 8
```

When MAX AREA is not specified, the FORMAT utility command will create 3 JHDA blocks. The number of areas that the default will support will vary depending on the journal block size. For example if the journal were formatted with a block size of 2032 bytes, 750 areas could be open at one time, 250 per JHDA.

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.

Formatting areas and segments under central version

The areas to be formatted must be in update mode to CV. The physical area lock for each area will remain on during and after the format. A logical area lock is acquired to prevent online transactions from updating the area during the format. If other online transactions are holding a logical area lock, Format will wait until the locks are released. If the wait would cause a deadlock, the format will be aborted. Once acquired, the logical area lock is released when a commit transaction is explicitly issued, or implicitly when the batch step ends, or at the end of the pseudo-converse when running online.

Referencing SYSTRK Files During Format

To avoid I/O errors when building the runtime environment in local mode, only previously formatted files should be referenced using a DDname that matches the SYSTRK DDname prefix. For this reason, it is recommended that non-matching DDnames always be used to identify SYSTRK files being formatted.

SYSTRK File Attributes

If a SYSTRK file is being formatted to be added as a mirror of an existing file, the page sizes of the two files must be the same and the file being formatted must have at least as many pages as the existing file. If these criteria are not met the following conditions can occur:

- Any attempt to make the newly formatted file an active mirror of the existing file fails.
- If a LIKE parameter is specified, FORMAT does not copy the contents of the file specified by *ddname2* to the newly formatted file.

If INITIAL is not specified, the page size and number of pages of a file being formatted remain unchanged.

If INITIAL is specified, the number of pages written to the file is determined according to the following precedence rules:

- If a FILE SIZE parameter is specified, then the number of pages is *number-of-pages*.
- If a LIKE parameter is specified, then the number of pages is a value based on the number of pages in the file identified by *ddname2*. The value is calculated as:

  ```
  page-cnt * (100 + pct-increase) / 100
  ```

  page-cnt

  Specifies the number of pages in the file identified by *ddname2*.

  pct-increase

  Specifies the value in the EXPAND parameter, if specified or 0.

- The number of pages is a value based on the size of the current DMCL calculated as:

  ```
  ((DMCL-size + page-size -1) / page-size)*4
  ```

  DMCL-size

  Specifies the size of the DMCL load module.

  page-size

  Specifies the page size of the file being formatted.

In the latter two cases, the number calculated is rounded up to the next larger integer value. If the calculated value is less than the minimum, it is set to the minimum of 10. If the calculated value is larger than the maximum, it is set to the maximum of 999,999.

If INITIAL is specified, the size of the pages written to a non-VSAM file is determined according to the following precedence rules:

- If PAGE SIZE is specified, then the page size is *page-size*.
- If a block size has been assigned (for example, specified in JCL or at the time the file was created), then page size is the block size.
- If a LIKE parameter is specified, then page size is the *page-size* of the file identified by *ddname2*.
- Otherwise, the page size is 7548.

For VSAM files, the page size is the file record size. Any attempt to override this through a PAGE SIZE parameter fails.

Choosing a SYSTRK Page Size

In most cases, the FORMAT utility's default page size for SYSTRK files provides an acceptable trade-off between memory, I/O, and disk space. Consider overriding the default only if the size of the DMCL is extremely large (500K or more). A larger page size will reduce I/Os and disk space requirements at the expense of slightly increased memory usage for buffers.

Estimating the Minimum Number of Pages for a SYSTRK File

To estimate the minimum number of pages needed for a SYSTRK file, perform the following steps:

1. Take the size of the DMCL load module used by the CV, divide it by the SYSTRK page size and multiply it by 2.5.
2. Multiply the resulting value with a factor to allow for overrides and growth. Overrides require approximately 100 bytes of space each and are generated for:
   - Each database or journal file defined in the execution JCL
   - Each dynamic change in the data set name of a database or journal file
   - Each dynamic change in the permanent status of an area
   - Each dynamic change in the status of a journal file

Copying SYSTRK File Contents

If a LIKE parameter is specified, the contents of the file identified by *ddname2* are copied to the files being formatted unless the file identified by *ddname2* is in use by a CV or the attributes of the two files are incompatible. If the contents of *ddname2* are not copied, a message indicates the reason.

If the file attributes are compatible, specify the keyword ACTIVE to force the copy to occur even if the file identified by *ddname2* is in use by CV. Only do this if you are sure that CV will not update the file while the copy is in progress, otherwise, the contents of the two files may not be the same which can lead to unpredictable results during CV restart. Ensure that a CV does not update its SYSTRK files by varying change tracking inactive before doing the format.

There is normally no need to force the contents of SYSTRK files to be copied. CV automatically updates newly formatted SYSTRK files as part of making them active mirrors.

JCL Considerations

When you submit a FORMAT statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the following:

- Database files to be processed (or that map to the areas to be processed)
- Journal files to be processed

- Dictionary containing table definitions, if formatting all or part of an SQL-defined database by area or by segment.
- SYSTRK files to be processed

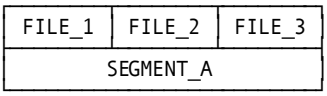
To run under the batch command facility under central version, include a SYSCTL statement.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

Formatting a segment

The following illustration shows a segment that includes three files:



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 following left-hand illustration 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.

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 following left-hand illustration 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, 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 sysjrnl1;
```

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.

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;
```

Formatting SYSTRK Files

The following sample IDMSBCF statement instructs the FORMAT utility to format three new SYSTRK files (track01, track02, track03). It directs the utility to format the files to have the default page size of 7548 and contain 60 pages each. CA IDMS will maintain 3 active mirrors.

```
format systrk track01, track02, track03
   initial file count 3
   file size 60;
```

Sample Output

Formatting by SEGMENT

The following listing was generated after the successful completion of the previous FORMAT SEGMENT USERDB example.

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   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
```

Note: For more information about allocating CA IDMS/DB files, see the *CA IDMS Database Administration Guide*.

Formatting SYSTRK Files

The following listing was generated after the successful completion of the previous FORMAT SYSTRK example.

```
IDMSBCF                          CA IDMS Batch Command Facility


FORMAT SYSTRK TRACK01, TRACK02, TRACK03
       INITIAL FILE COUNT 3 FILE SIZE 60;

Systrk file TRACK01 page size 7,548 file size 60
        delete NULL file count 3.
Systrk file TRACK02 page size 7,548 file size 60
        delete NULL file count 3.
Systrk file TRACK03 page size 7,548 file size 60
        delete NULL file count 3.
Status = 0       SQLSTATE = 00000

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

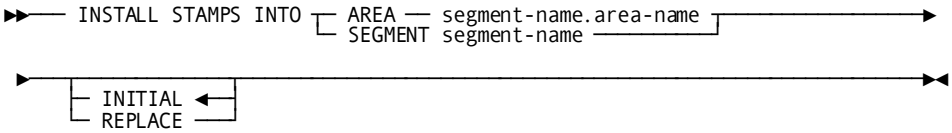
INSTALL STAMPS

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

INSTALL STAMPS Syntax



INSTALL STAMPS Parameter

AREA

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

segment-name

Specifies the name of the segment containing the area.

area-name

Specifies the name of the area.

SEGMENT

Specifies the segment whose areas will have their stamps installed.

segment-name

Specifies the name of a segment included in the DMCL module.

INITIAL

Specifies that the area(s) contain no synchronization stamps because they were formatted using the file or segment option of the FORMAT utility statement executing in local mode. INITIAL is the default.

REPLACE

Specifies that the area(s) contain synchronization stamps that should be replaced with those from the catalog.

Usage

How to submit the INSTALL STAMPS statement

You submit the INSTALL STAMPS statement 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.

Use caution when replacing stamps

By replacing stamps in an area, you are asserting that the catalog's definition accurately describes data in the area. You should be sure that this is true or that the area contains no data before replacing stamp values. No data validation is performed by the utility.

JCL Considerations

When you submit an INSTALL STAMPS statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the following:

- File(s) that map to the area to be processed
- Dictionary that defines the table(s) in the area
- Journal file(s) if backups are not taken

To run under central version, a SYSCTL statement is needed.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

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

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

Sample Output

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

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   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
```

Note: For more information about synchronization stamps, see the *CA IDMS Database Administration Guide*. and the *CA IDMS SQL Reference Guide*.

LOAD

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.

Note: For more information about the phases and steps involved in loading a database, and help in deciding what kind of load you should run, see the *CA IDMS Database Administration Guide*.

Authorization

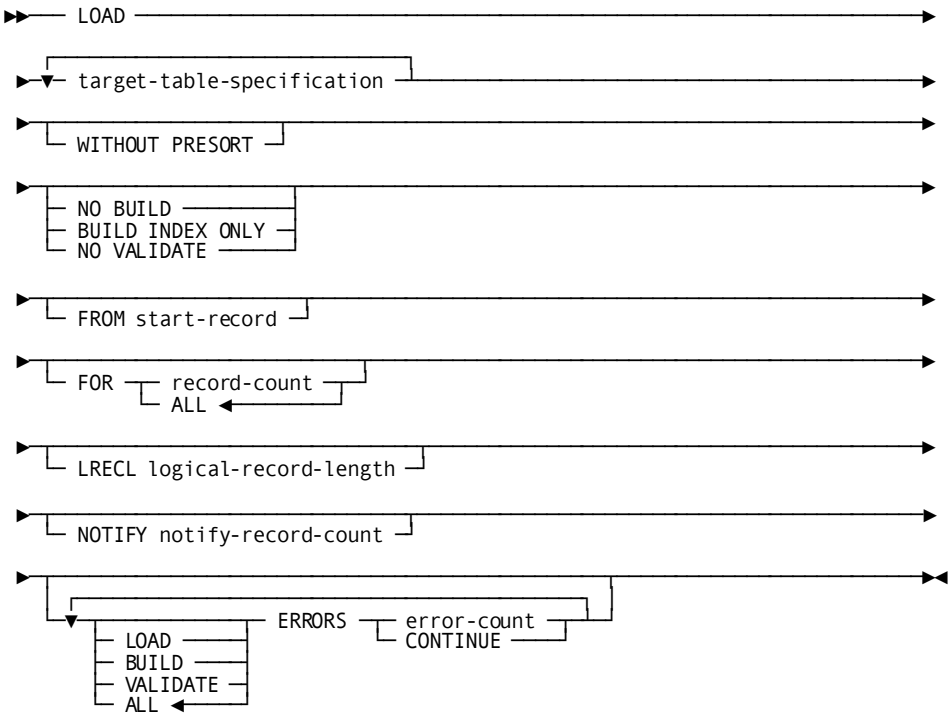
To	You need this privilege	On
Load into a table	INSERT	The table

Note: Only one LOAD, BUILD, or VALIDATE statement can be performed during one execution of the Batch Command Facility (IDMSBCF).

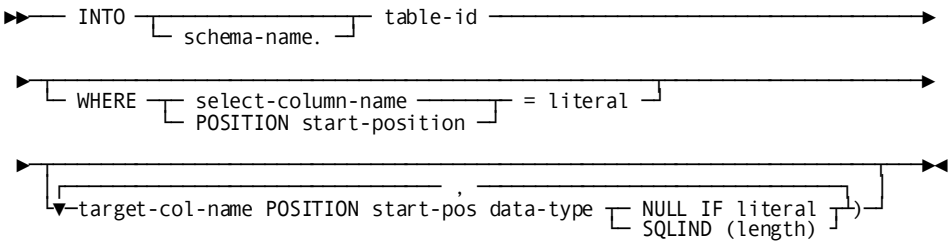
LOAD Syntax

Complete LOAD:

Syntax for complete or phased LOAD

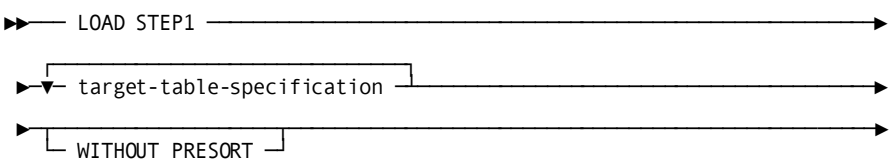


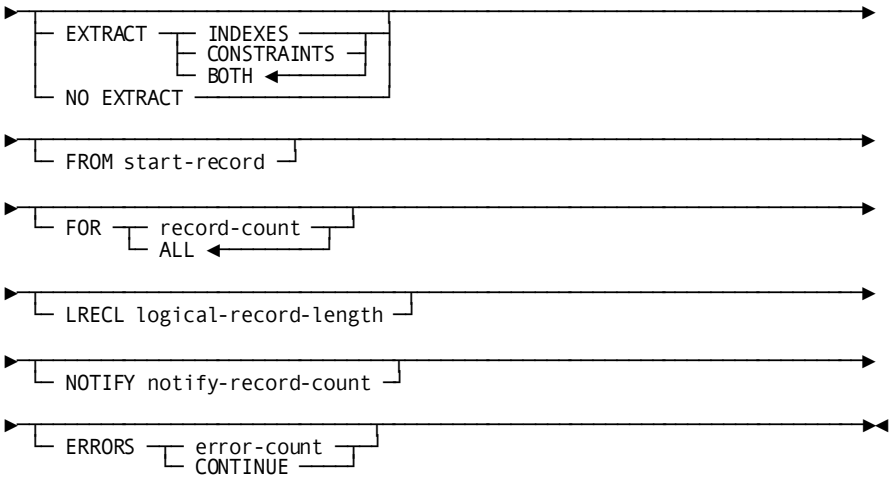
Expanded syntax for target table specification



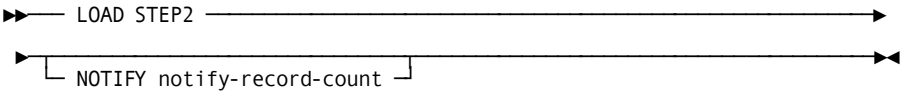
Stepped LOAD

Syntax for STEP 1





Syntax for STEP 2



LOAD Parameter

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".

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.

Note: For more information and help in deciding whether to suppress the presort, see the *CA IDMS Database Administration Guide*.

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 might 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

Specifies 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

Specifies 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

Specifies 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

Specifies 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

Specifies 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

Specifies the name of the schema that contains the table.

table-id

Specifies 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

Specifies 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

Specifies the value the column must contain.

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

Note: For more information about comparable data types, see the *CA IDMS SQL Reference Guide*.

POSITION *start-pos*

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

literal

Specifies 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 and the order, data type, and null indications must exactly match those of the table as defined in the dictionary. VARCHAR and VARGRAPHIC input values must be preceded by a 2-byte binary length of the value.

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
  - The size of VARCHAR is the maximum length, plus 2.
  - The size of VARGRAPHIC is the maximum length times the number of bytes for one character, plus 2.

POSITION

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

start-pos

Specifies 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.

Note: For a list of SQL data types and for the SQL standards of data-type specification, see the *CA IDMS SQL Reference Guide*.

NULL IF

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

literal

Specifies 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:

Value	Meaning
Binary zeros	The column value is not null
Binary X'FF's	The column value is null

(*length*)

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

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.

Note: For more information and help in deciding whether to suppress the presort, see the *CA IDMS Database Administration Guide*.

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 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. Therefore, 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. Therefore, 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.

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.

Usage

How to submit the LOAD statement

You submit the LOAD statement 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 or to expand it afterwards.

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 the *CA IDMS Database Administration Guide*.

SYS001 input file contents

The LOAD utility does not perform any type of data translation against the input records within the SYS001 file. As a result, each column value in the SYS001 file must be in the proper internal format as specified by the corresponding column's data format. For example, if a column has a format of DECIMAL(5), the corresponding field in the SYS001 record must be a valid packed decimal number occupying 3 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.

Checking error messages

Use the *CA IDMS Messages and Codes Guide* 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.

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, if doing a complete LOAD

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 data set, it is required that all extents be 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 SCHEMA RECORD 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

Note: Note: For more information about the generic JCL used to execute the batch command facility, see the chapter specific to your operating system.

Examples

Phased LOAD

The following sample statement 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 next example 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;
```

Sample Output

Complete LOAD

The following report was generated after executing the LOAD statement in example 1 shown previously.

```
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  -   volser     PRESORT TABLES FOR LOAD     yy-mm-dd-hh.mm.ss
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  -   volser     LOAD TABLES AFTER SORT     yy-mm-dd-hh.mm.ss
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  -   volser     FILL IN OWNERS DBKEY     yy-mm-dd-hh.mm.ss

IDMSLOAD  -  12 intermediate records were written to SYS003
IDMSLOAD  -       largest SYS003 record size is 56 characters
IDMSLOAD  -         FILL IN OWNERS DBKEY     processing completed
IDMSLOAD  -   volser     CONNECT UP INDEXES     yy-mm-dd-hh.mm.ss
IDMSLOAD  -  9 intermediate records were written to SYS003
IDMSLOAD  -       largest SYS003 record size is 56 characters
IDMSLOAD  -         CONNECT UP INDEXES     processing completed
IDMSLOAD  -   volser     BUILD RECORD PREFIX     yy-mm-dd-hh.mm.ss
IDMSLOAD  -         BUILD RECORD PREFIX     processing completed


AutoCommit will COMMIT transaction

Command Facility ended with warnings
```

Complete LOAD specifying column name

The next report was generated after executing the LOAD statement in example 2.

Note: 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 the *CA IDMS Messages and Codes Guide*.

```
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  -   volser     PRESORT TABLES FOR LOAD     yy-mm-dd-hh.mm.ss
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  -   volser     LOAD TABLES AFTER SORT     yy-mm-dd-hh.mm.ss
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  -   volser     FILL IN OWNERS DBKEY     yy-mm-dd-hh.mm.ss
IDMSLOAD  -  34 intermediate records were written to SYS003
IDMSLOAD  -       largest SYS003 record size is 56 characters
IDMSLOAD  -         FILL IN OWNERS DBKEY     processing completed
IDMSLOAD  -   volser     CONNECT UP INDEXES     yy-mm-dd-hh.mm.ss
IDMSLOAD  -  19 intermediate records were written to SYS003
IDMSLOAD  -       largest SYS003 record size is 56 characters
IDMSLOAD  -         CONNECT UP INDEXES     processing completed
IDMSLOAD  -   volser     BUILD RECORD PREFIX     yy-mm-dd-hh.mm.ss
IDMSLOAD  -         BUILD RECORD PREFIX     processing completed
IDMSLOAD  -   volser     VALIDATE INDEXES STEP 1     yy-mm-dd-hh.mm.ss
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  -   volser     VALIDATE INDEXES STEP 2     yy-mm-dd-hh.mm.ss
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
```

More Information

- For more information about procedures for loading a database, see the *CA IDMS Database Administration Guide*.
- For more information about SQL data-type specifications, see the *CA IDMS SQL Reference Guide*.

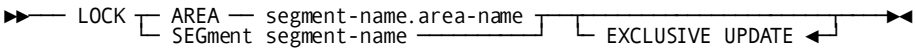
LOCK

The LOCK utility allows a DBA to explicitly lock an area. This allows the DBA to place a lock on an area that will remain in effect across several commands. Therefore, access to an area can be prevented while a series of operations is performed on it.

Authorization

To	You Need This Privilege	On
Lock an area	DBAWRITE	The area

LOCK Syntax



LOCK Parameter

AREA

Directs the LOCK utility statement to lock a specified area.

segment-name

Identifies the name of the segment associated with the area to be locked.

area-name

Identifies the name of the area to be locked.

SEGMENT

Specifies the segment whose areas will be locked.

segment-name

Identifies the name of the segment to be locked.

EXCLUSIVE UPDATE

Specifies the update mode. EXCLUSIVE UPDATE is the default mode and the only mode currently supported.

Usage

Local mode execution

If the LOCK AREA statement is issued through the batch command facility executing in local mode, a physical lock will be placed on the area. The lock remains in effect until an explicit UNLOCK is issued. If the area is already locked, the LOCK statement will fail with a DB002352 error message as illustrated in the sample output.

If the LOCK SEGMENT statement is issued in local mode, a physical lock will be placed on each area in the segment that is not already locked. A status of each area will be reported. Once all areas have been locked or checked, if any area in the segment was already locked, a DB002406 error message will be issued, as illustrated in the sample output.

Online execution

If the LOCK statement is issued through the online command facility (OCF) or through the batch command facility executing in a batch/CV mode, a logical lock will be placed on the area. This lock prevents all access to the area by other users until the session in which the LOCK statement is issued is committed. If executing online, a commit is automatically issued at end of task prior to the pseudo-converse unless autocommit is disabled through a SET OPTIONS statement.

The areas being locked must be varied in update mode to the central version. If another task is updating an area being locked, the LOCK command will wait until the logical area lock is available. If the wait would cause a deadlock, the LOCK command fails.

Committing prior work

Before executing this utility under a central version, you must commit any previous work done within the current session. For more information, see Central Version Considerations (see page 37).

JCL Considerations

When you submit a LOCK statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the file containing the first page of the area to be processed. To run under central version, a SYSCTL statement is needed.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

Lock by area

The following example directs the LOCK utility to lock the EMPDEMO.EMP-DEMO-REGION area.

```
lock area empdemo.emp-demo-region;
```

Lock by segment

The following example directs the LOCK utility to lock the EMPDEMO segment.

```
lock segment empdemo;
```

Sample Output

Lock by area

When LOCK processing is completed, the following listing is generated when processing is successful.

```
IDMSBCF nn.n                          CA IDMS Batch Command Facility          mm/dd/yy            PAGE 1

   LOCK AREA EMPDEMO.EMP-DEMO-REGION;
  Status = 0 SQLSTATE = 00000
```

When LOCK processing is completed, the following listing is generated when processing fails.

```
IDMSBCF nn.n                          CA IDMS Batch Command Facility          mm/dd/yy            PAGE 1

   LOCK AREA USERDB.EMP_AREA;
  Status = -4      SQLSTATE = 5000B       Messages follow:
DB002352 C-4M353: Area USERDB.EMP_AREA required area lock mode not available
```

Lock by segment

When LOCK processing is completed, the following listing is generated.

```
IDMSBCF nn.n                          CA IDMS Batch Command Facility          mm/dd/yy          PAGE 1

   LOCK SEGMENT EMPDEMO;
   Area EMPDEMO.EMP-DEMO-REGION was not available
   Area EMPDEMO.INS-DEMO-REGION Lock Mode Acquired
   Area EMPDEMO.ORG-DEMO-REGION Lock Mode Acquired
 Status = -4  SQLSTATE = 28000   Messages follow:
 DB002406 C-4M363: Area lock mode unavailable for one or more areas
```

Note: For more information about area locks, see the *CA IDMS Database Administration Guide*.

MAINTAIN INDEX

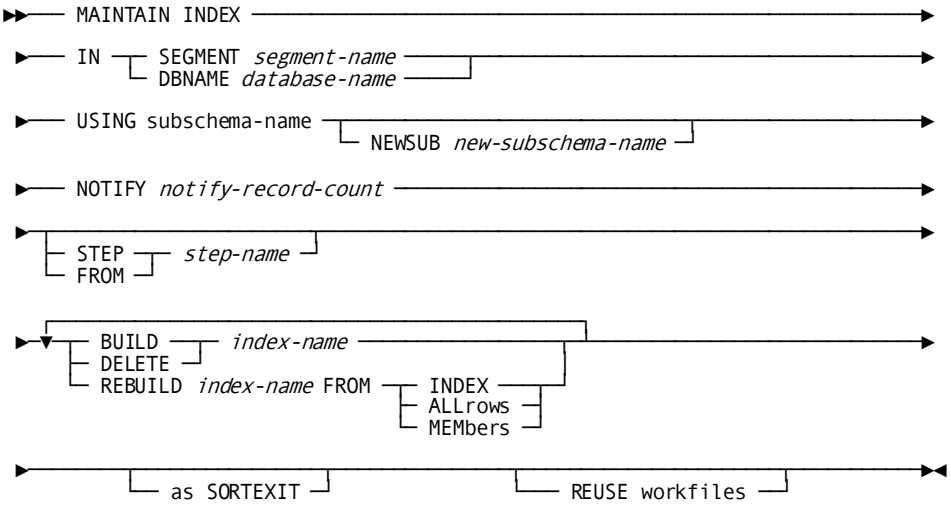
The MAINTAIN INDEX utility builds, rebuilds, or deletes one or more indexes in a non-SQL-defined database.

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

MAINTAIN INDEX Syntax



MAINTAIN INDEX Parameter

IN SEGMENT

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

segment-name

Specifies the name of the segment.

IN DBNAME

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

database-name

Specifies the name of the database.

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 zeros.

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

subschema-name

Specifies 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.

Note: 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

Specifies 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

Specifies 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

Specifies 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

Specifies 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 DBL*x* 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.

Usage

How to submit the MAINTAIN INDEX utility

You submit the MAINTAIN INDEX utility 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 reentrancy

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.

BUILD-option with non-MA indexes

Specifying the BUILD-option will connect every potential member record to the system-owned index. Therefore, if the index is not MANDATORY AUTOMATIC and there are occurrences of the member record that should not participate in the index, do not use the BUILD-option. Instead, use the REBUILD FROM MEMBERS-option.

Rebuilding indexes

When rebuilding an index, MAINTAIN INDEX might 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:

Function	REBUILD Option	Subschema Specification
Reorganize an existing index (for example, after deleting many member occurrences)	FROM INDEX	USING *old-subschema-name*
Rebuild a damaged index	FROM MEMBERS	USING *old-subschema-name*
Modify the following index tuning options: ■ Key compression ■ Number of entries in an SR8 record ■ Index displacement ■ Location of an index (its area or page range) ■ Linked or unlinked attribute	FROM INDEX	USING *old-subschema-name* NEW SUB *new-subschema-name*
Change the following index characteristics: ■ Sort key ■ Collating sequence ■ Sorted or unsorted attribute ■ Duplicates option	FROM MEMBERS	USING *new-subschema-name*
Rebuild an index from all member occurrences without making index changes	FROM ALLROWS	USING *old-subschema-name*
Rebuild an index from all member occurrences making index changes	FROM ALLROWS	USING *new-subschema-name*

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) reorganize 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 ensure 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 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 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 data set 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, see 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 IDMSDBLx 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 data set 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:

Step	Description
IDMSTABX	■ Interprets the control statements ■ Reads the database ■ Deletes system-owned indexes ■ Writes index descriptors to the SYS003 file ■ Writes set descriptors to the RELDCTL file ■ Writes sort control parameters to the SYSPCH file
SORT3	Sorts the contents of the SYS003 file and puts the results in SYS004.

Step	Description
IDMSDBL3	■ Erases old index structures, if there are any ■ Builds the new index structures ■ Establishes pointers for user-owned index sets, and puts them in the SYS005 file ■ Establishes index pointers for system-owned linked indexes and puts them in the SYS005 file
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	■ Control parameters from your submission of the utility ■ The database	■ SYS003 contains index membership information ■ SYSPCH contains sort parameters ■ RELDCTL file **Note:** The SYS003 record length can be calculated as follows: (largest sortkey length in the subschema * 2) + 28
SORT3	■ SYS003 from IDMSTABX ■ SYS001 contains the sort parameters from IDMSTABXSYSPCH file. If running in step mode or if it is the first step in a sort, no sort parameters are needed.	SYS004 contains the sorted contents of SYS003
IDMSDBL3	■ SYS004 from SORT3 ■ RELDCTL file	■ SYS005 contains prefix pointer information for user-owned indexes ■ SYSPCH contains sort parameters **Note:** The SYS005 record length is 32.

Step	Input	Output
SORT4	■ SYS005 from IDMSDBL3 ■ SYS001 contains the sort parameters from IDMSDBL3SYSPCH file. If running in step mode or if this is the first step in a sort, sort control parameters are not needed.	SYS006 contains the sorted contents of SYS005
IDMSDBL4	■ SYS006 from SORT4 ■ RELDCTL file	SYSLST contains a summary of the results of the MAINTAIN INDEX operation

Note: This table describes the input and output files when you execute 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 *n*) *n* is the length of the longest sort or CALC key in the subschema	Byte 5
SORT4	SYS005	Ascending	16 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 z/VSE. 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 that is identical to the original schema.
2. Create a global subschema for the new schema with a name that 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. Back up 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 next
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 two 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 you are 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  dmclnm 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 (for example, 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.
```

JCL Considerations

When you submit a MAINTAIN INDEX utility 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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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 SORT*x* 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.

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: SORTEXIT mode and NOT reusing workfiles		
IDMSTABX		SYS003
SORT3/IDMSDBL3	SYS003	SYS005
SORT4/IDMSDBL4	SYS005	
MAINTAIN INDEX: SORTEXIT mode and REUSING workfiles		
IDMSTABX		SYS003
SORT3/IDMSDBL3	SYS003	SYS003
SORT4/IDMSDBL4	SYS003	

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;
```

Sample Output

The following listing is generated after successful completion of the MAINTAIN INDEX utility statement in the previous example.

```
 IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   PAGE 1

 MAINTAIN INDEX IN SEGMENT EMPDEMO USING EMPSS01
 REBUILD "SKILL-NAME-NDX" FROM MEMBERS ;
UT010002 BEGINNING PROCESSING FOR STEP IDMSTABX
UT012001 IDMSTABX RELEASE nn.n TAPE volser STARTED
UT012004 69 INTERMEDIATE RECORDS WERE WRITTEN TO SYS003
UT012006 SYS003   MAXIMUM RECORD SIZE IS 78
UT012007 IDMSTABX RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSTABX HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002    69 RECORDS WERE READ FROM SYS003
UT009003    69 RECORDS WERE WRITTEN TO SYS004
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT3    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE nn.n TAPE volser 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 nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT4
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002    68 RECORDS WERE READ FROM SYS005
UT009003    68 RECORDS WERE WRITTEN TO SYS006
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT4    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL4
UT007001 IDMSDBL4 RELEASE nn.n TAPE volser PROCESSING STARTED
UT007004    NO DATABASE ERRORS WERE ENCOUNTERED
UT007005    NO LOGIC ERRORS WERE ENCOUNTERED
UT007002    68 RECORDS WERE READ FROM SYS006
UT007006 IDMSDBL4 RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL4 HAS COMPLETED SUCCESSFULLY
UT013001 DATABASE INDEX/ASF TABLE HAS COMPLETED SUCCESSFULLY
Status = 0
```

More Information

- For more information about designing indexes, see the *CA IDMS Database Design Guide*.
- For more information about defining and maintaining indexes, see the *CA IDMS Database Administration Guide*.

MERGE ARCHIVE

The MERGE ARCHIVE 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.

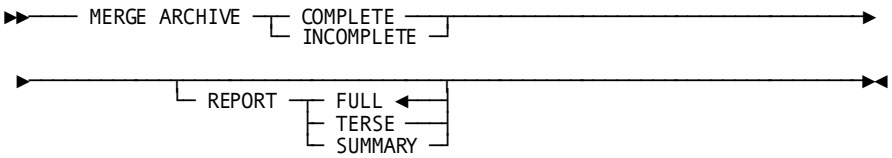
Under certain circumstances, this utility must be used if members of a data sharing group are sharing update access to data.

Note: For more information about when the use of this utility is mandated, see Backup and Recovery in the *CA IDMS Database Administration Guide*.

Authorization

To	You Need This Privilege	On
merge archived journal files	USE	The DMCL

MERGE ARCHIVE Syntax



MERGE ARCHIVE Parameter

COMPLETE

Indicates that all journal files that contain images needed for recovery have been included as input. To complete transactions that are incomplete at the end of the process, additional checkpoint records will be written to the merged output file:

- For incomplete distributed transactions whose state is InDoubt, additional distributed and local checkpoint records will be written if a matching manual recovery control input file entry is provided.
- For local (that is, non-distributed) transactions still active at the end of the process, an ABRT checkpoint record will be written.

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 additional checkpoints are generated. The output file can be used only 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. All details of a distributed transaction record are reported. This includes local transaction ids with program names, external transaction ids, and resource manager interests. Additionally, 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. For distributed transaction records: external transaction ids and resource manager interests are not included in the report.

SUMMARY

Indicates that only final summary information is produced.

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 might need to be provided before the merged file can be used for recovery purposes. For more information, see the ROLLFORWARD or ROLLBACK utility commands.

MERGE ARCHIVE and Distributed Transactions

MERGE ARCHIVE reports on distributed transactions and supports the use of input and output manual recovery control files. If COMPLETE is specified, the input manual recovery control file is used to complete InDoubt distributed transactions. If an output manual recovery control file is included in the JCL, an entry will be written for each incomplete distributed transaction encountered. For more information, see JCL Considerations (see page 179) and the "Common Facilities for Distributed Transactions" appendix.

Note: For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Guide*.

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 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

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a manual recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of these files is fixed block with a record length of 80.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

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  yyyy-mm-dd-hh.mm.ss.ffffff  GLBQU
 NODE SYSTEM74 RU_ID         42 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  0   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff  GLBQU
 NODE SYSTEM74 RU_ID         43 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  1   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM74 RU_ID         45 PGM_ID  DBCRUPD   QUIESCE LEVELS  3  UPD  2   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM72 RU_ID         46 PGM_ID  DBCRUPD   QUIESCE LEVELS  0  UPD  0   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM72 RU_ID         50 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  0   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM72 RU_ID         51 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  1   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM72 RU_ID         52 PGM_ID  DBCRUPD   QUIESCE LEVELS  3  UPD  2   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM72 RU_ID         53 PGM_ID  DBCRUPD   QUIESCE LEVELS  4  UPD  3   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM74 RU_ID         42 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  2   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM73 RU_ID         49 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  0   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
...
 NODE SYSTEM73 RU_ID       1499 PGM_ID  DBCRUPD   QUIESCE LEVELS  6  UPD  6   ABRT  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM73 RU_ID       1458 PGM_ID  DBCRUPD   QUIESCE LEVELS  5  UPD  5   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM73 RU_ID       1359 PGM_ID  DBCRUPD   QUIESCE LEVELS  4  UPD  4   ABRT  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM74 RU_ID       1226 PGM_ID  DBCRUPD   QUIESCE LEVELS  0  UPD  0   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM73 RU_ID       1472 PGM_ID  DBCRUPD   QUIESCE LEVELS  3  UPD  3   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM73 RU_ID       1498 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  2   ABRT  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM73 RU_ID       1448 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  1   ABRT  yyyy-mm-dd-hh.mm.ss.ffffff
 NODE SYSTEM73 RU_ID       1392 PGM_ID  DBCRUPD   QUIESCE LEVELS  0  UPD  0   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff  GLBQ


     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
```

More Information

- For more information about manual recovery, see the *CA IDMS Database Administration Guide*.
- For more information about data sharing, see the *CA IDMS System Operations Guide*.

PRINT INDEX

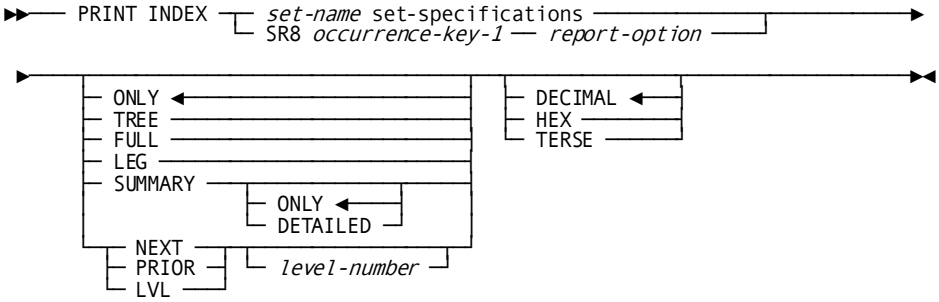
The PRINT INDEX utility reports on the structure of system-owned indexes and indexed sets. Using the PRINT INDEX utility, you can review:

- 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

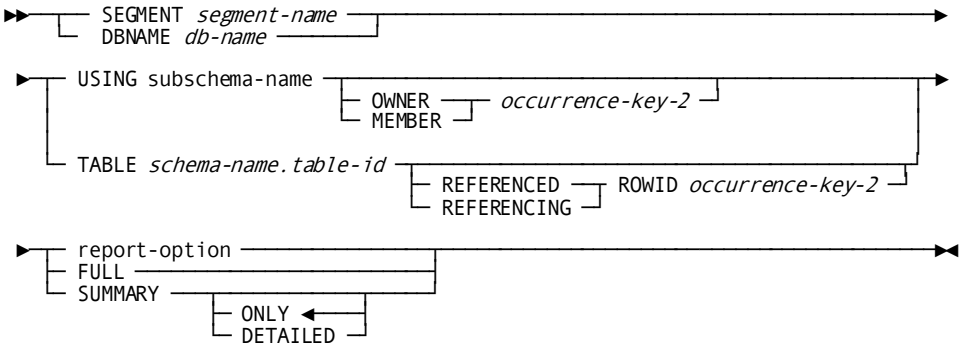
Authorization

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

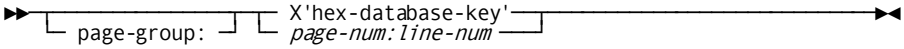
Syntax: PRINT INDEX



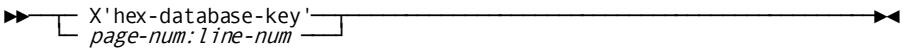
Expansion of set-specifications



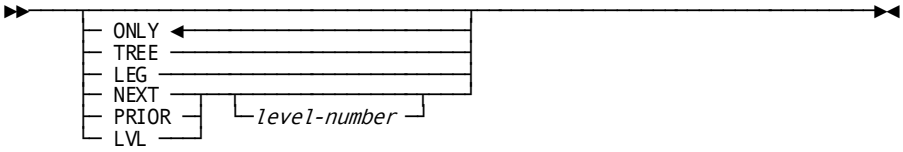
Expansion of occurrence-key-1



Expansion of occurrence-key-2



Expansion of report-option



PRINT INDEX Parameter

set-name

Specifies 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. When using the FULL option, or when specifying a starting member dbkey, the member area must also exist in this segment.

segment-name

Specifies the name of the segment.

DBNAME

Specifies the database containing the index structures to be reported.

db-name

Specifies the name of the database.

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.

REFERENCED ROWID

For the named table, directs the PRINT INDEX utility to report on the index occurrence whose owner is the referenced row identified by *occurrence-key*.

REFERENCING ROWID

For the named table, directs the PRINT INDEX utility to report on the index occurrence containing the row ID of the referencing row identified by *occurrence-key*.

X'*hex-database-key*'

Specifies the hexadecimal database key of an owner or member record in the specified indexed set.

page-num

Specifies the page number of an owner or member record in the specified indexed set.

line-num

Specifies 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*'

Specifies the hexadecimal database key of the SR8 record.

page-num

Specifies the page number of the SR8 record.

line-num

Specifies 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 effect 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.

LVL

Directs the PRINT INDEX utility to report on all the SR8 records in a single level of the index.

level-number

Specifies 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.

SUMMARY

Requests a summary report for the target index. A summary report consists of three parts:

- Part 1 (header) provides general information on the index definition.
- Part 2 (main body) provides information on index owner occurrence(s). A system-owned index contains a single index owner; a user-owned index can contain more than one index owner.
- Part 3 (index overview) provides global statistical information for a user-owned index only.

A summary report on a system-owned index contains parts 1 and 2.

A summary report on a user-owned index always contains parts 1 and 3. Part 2 is included only in a detailed summary report.

ONLY

Requests a summary report with parts 1 and 3 for the target user-owned index. This parameter is ignored for a system-owned index. ONLY is the default.

DETAILED

Requests a summary report with parts 1, 2, and 3 for the target user-owned index. This parameter is ignored for a system-owned 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.

Usage

How to submit the PRINT INDEX statement

You submit the PRINT INDEX statement by using the batch command facility or the online command facility.

When to use PRINT INDEX

The PRINT INDEX utility can help you determine whether an index needs to be rebuilt. For example, you should consider rebuilding an index when the PRINT INDEX utility report on the index indicates one of the following:

- The number of index levels is greater than anticipated for the original index structure.
- Twenty-five percent or more of the member records are orphans.

An index can be rebuilt using MAINTAIN INDEX or TUNE INDEX. For more information about index rebuilding and indexing in general, see the *CA IDMS Database Administration Guide*.

Note: The output of PRINT INDEX without the SUMMARY parameter is proportional to the number of index members that are being reported. If PRINT INDEX is run online or in batch through CV, the output is buffered in scratch. If the scratch area cannot contain all the output, PRINT INDEX fails with a task abend.

Hexadecimal display of symbolic keys

The HEX parameter of the SET/SR8 statement is useful when the symbolic key for the index is a non-displayable data type, such as binary or packed.

When to use DBNAME

You can use DBNAME instead of SEGMENT at any time. You must use it when an index member resides in a different segment from the index structure, and the FULL option is used, or you specify a starting MEMBER dbkey.

JCL Considerations

When you submit a PRINT INDEX utility 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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

Printing an entire index

The following example directs the PRINT utility to report on the EMP-IDX-SET using the FULL option.

```
PRINT INDEX "EMP-IDX-SET" DBNAME VLDBDBN USING VLDBSUBC FULL;
```

Printing the bottom level of an index

The following example directs the PRINT utility to report on the COV-IDX-SET using the LEVEL and TERSE options.

```
PRINT INDEX "COV-IDX-SET" SEGMENT VLDBSPG1 USING VLDBSUBC  MEMBER X'01390448' LVL 0
TERSE;
```

Printing individual SR8 record

The following example directs the PRINT utility to report on a specific SR8 record.

```
PRINT INDEX SR8 5:80130:03 NEXT 2 HEX;
```

Printing an index from an SQL-defined database

The following example directs the PRINT utility to report on the EMP-COVERAGE index that is part of the SQLSPG.EMPLOYEE table.

```
PRINT INDEX "EMP-COVERAGE" SEGMENT VLDBSPG1 TABLE SQLSPG.EMPLOYEE SUMMARY;
```

Printing a summary report of an index

The following example directs the PRINT utility to report on the DEPT_EMPL index using the SUMMARY option.

```
PRINT INDEX DEPT_EMPL SEGMENT USERDB TABLE DEMO.DEPT SUMMARY;
```

Printing a REFERENCING ROWID summary report of an index

The following example directs the PRINT utility to report on the index occurrence containing the row ID of the referencing row identified by X'01390201'.

```
PRINT INDEX "COV-IDX-SET" SEGMENT VLDBSPG1 TABLE SQLSPG.COVERAGE REFERENCING ROWID
X'01390201' SUMMARY;
```

Sample Output

Printing an entire index

The PRINT INDEX utility generates the following report after successful completion of the statement in the previous "Printing an Entire Index" example.

```
IDMSBCF  18.0                                  CA IDMS Batch Command Facility                              mm/dd/yy


PRINT INDEX "EMP-IDX-SET"
      DBNAME VLDBDBN
      USING VLDBSUBC FULL;
SET=EMP-IDX-SET      OWNER=SR7             PAGE GROUP=2     RECORDS PER PAGE=255
                                           ODBK=01394301 SR8 N01394303 SR8 P01394306 ASC CUSH=12  SYM TKL=3    COMP
                  MEMBER=EMPLOYEE          PAGE GROUP=1     RECORDS PER PAGE=255
L1   01394303  NUME=5                 U=FFFFFFFF N=01394302 P=01394301 RECL=224   SPA=3164
             01394302 0028  01394304 0053  01394307 0106  01394305 0329  01394306 0479
L0   01394302  NUME=15                U=01394303 N=01394304 P=01394303 RECL=184   SPA=3164
             0138DF01 0001  0138DE01 0003  0138CE01 0004  0138D801 0007  0138C501 0011  0138D401 0013  0138D101 0015

             0138CE02 0016  0138DE02 0019  0138DE03 0020  0138DE04 0021  0138DE05 0023  0138DE06 0024  0138DE07 0027
             0138DE08 0028
     01394304  NUME=10    ORPH=8      U=01394303 N=01394307 P=01394302 RECL=140   SPA=3164
.
.
.


MEM  0138DF01                  U=01394302
     0138DE01                  U=01394302
     0138CE01                  U=01394302
     0138D801                  U=01394302
     0138C501                  U=01394302
     0138D401                  U=01394302
     0138D101                  U=01394302
     0138CE02                  U=01394302
     0138DE02                  U=01394302
     0138DE03                  U=01394302
.
.
.
0138CF02     *ORPHAN*OF*  U=01394305
     0138CF03     *ORPHAN*OF*  U=01394305
     0138D107                  U=01394306
     0138D105     *ORPHAN*OF*  U=01394305
     0138D108                  U=01394306
     0138D106                  U=01394306
     0138DB05                  U=01394306
     0138DF03                  U=01394306
     0138DF04                  U=01394306
     0138D807                  U=01394306
     0138DB04                  U=01394306
TOTAL SR8=6
Status = 0       SQLSTATE = 00000
```

Printing the bottom level of an index

The next report illustrates the use of the LVL and TERSE options to request the printing of the bottom level of an index.

```
IDMSBCF  18.0                                  CA IDMS Batch Command Facility                          mm/dd/yy


 PRINT INDEX "COV-IDX-SET"
       SEGMENT VLDBSPG1 USING VLDBSUBC

       MEMBER X'01390448' LVL 0 TERSE;
SET=COV-IDX-SET      OWNER=SR7             PAGE GROUP=5     RECORDS PER PAGE=255
                                           ODBK=01390201 SR8 N01390203 SR8 P01390202 UNS CUSH=4
                    MEMBER=COVERAGE        PAGE GROUP=5     RECORDS PER PAGE=255
L0   01390203  NUME=4                U=FFFFFFFF N=01390202 P=01390201 RECL=52    SPA=3820
     01390202  NUME=70               U=FFFFFFFF N=01390201 P=01390203 RECL=316   SPA=3820
TOTAL SR8=2
Status = 0       SQLSTATE = 00000
```

Printing individual SR8 records

The following report illustrates the use of the SR8 option to request the printing of a specific SR8 record.

```
IDMSBCF  18.0                                  CA IDMS Batch Command Facility                          mm/dd/yy



PRINT INDEX SR8 5:80130:03 NEXT 2 HEX;
SET=COV-IDX-SET      OWNER=SR7             PAGE GROUP=5     RECORDS PE
      R PAGE=255
                                           ODBK=01390201 SR8 N01390203
       SR8 P01390202 UNS CUSH=4
L0   01390203  NUME=4                U=FFFFFFFF N=01390202 P=01390201
      RECL=52    SPA=3820
              0139044A  01390449  01390448  01390447
     01390202  NUME=70               U=FFFFFFFF N=01390201 P=01390203
      RECL=316   SPA=3820
              01390446  01390445  01390444  01390443  01390442  01390441
              01390440  0139043F  0139043E  0139043D  0139043C  0139043B
              0139043A  01390439  01390438  01390437  01390436  01390435
              01390434  01390433  01390432  01390431  01390430  0139042F
              0139042E  0139042D  0139042C  0139042B  0139042A  01390429
              01390428  01390427  01390426  01390425  01390424  01390423
              01390422  01390421  01390420  0139041F  0139041E  0139041D
              0139041C  0139041B  0139041A  01390419  01390418  01390417
              01390416  01390415  01390414  01390413  01390412  01390411
              01390410  0139040F  0139040E  0139040D  0139040C  0139040B
              0139040A  01390409  01390408  01390407  01390406  01390405
              01390404  01390403  01390402  01390401
TOTAL SR8=2
```

Printing an index from an SQL-defined database

The following example provides a report, using the FULL option, on an SQL-defined index.

```
PRINT INDEX "COV-IDX-SET" SEGMENT VLDBSPG1
       TABLE SQLSPG.COVERAGE FULL;
SET=COV-IDX-SET      OWNER=SR7             PAGE GROUP=5     RECORDS PER PAGE=255

                                           ODBK=01390201 SR8 N01390203 SR8 P01390202 UNS CUSH=4
                     MEMBER=COVERAGE         PAGE GROUP=5     RECORDS PER PAGE=255
L0   01390203  NUME=4                U=FFFFFFFF N=01390202 P=01390201 RECL=52    SPA=3820
     0139044A  01390449  01390448  01390447
     01390202  NUME=70                U=FFFFFFFF N=01390201 P=01390203 RECL=316   SPA=3820
     01390446  01390445  01390444  01390443  01390442  01390441
     01390440  0139043F  0139043E  0139043D  0139043C  0139043B
.
.
.



MEM  0139044A                  U=01390203
     01390449                  U=01390203
     01390448                  U=01390203
     01390447                  U=01390203
     01390446                  U=01390202
…
     01390405                  U=01390202
     01390404                  U=01390202
     01390403                  U=01390202
     01390402                  U=01390202
     01390401                  U=01390202
TOTAL SR8=2
```

Printing a summary report of an index

The following report illustrates the use of the SUMMARY option to request the printing of a user-owned index.

```
  PRINT INDEX "EMP-COVERAGE"       SEGMENT VLDBSPG1
       TABLE SQLSPG.EMPLOYEE SUMMARY;
SET Name: EMP-COVERAGE
   IBC 70                              Displacement       0
   Sort option NOT SORTED              Key length         N/A
   Duplicates FIRST                    Compression        No
OWNER: EMPLOYEE
   AREA VLDBSPG1.EMPL-AREA             Low page  (SUB-    80056
   Page size 4276                      High page  AREA)   80100
   Page group 5                        Records per page   255
MEMBER: COVERAGE                       Set membership     Optional Automatic
   Located VIA index Yes Displ't 0     Index is           Linked
   AREA VLDBSPG1.COVE-AREA             Low page  (SUB-    80106
   Page size 4276                      High page  AREA)   80150
   Page group 5                        Records per page   255

Index overview
   Nr of owner occurrences                    56
   Nr of owner occurrences                    56
   Nr of empty owners                         55      98.2%
   Nr of displaced top level SR8s              0       0.0%
   Nr of SR8s:              Total              2
                          Average             0.0
                          Highest             2      Owner X'0138D404'
   Min. nr of SR8s:         Total              2
                          Average             0.0
                          Highest             2      Owner X'0138D404'
   Nr of levels:          Average             0.0
                          Highest             1      Owner X'0138D404'
   Min. nr of levels:     Average             0.0
                          Highest             1      Owner X'0138D404'
   Nr of pages:           Average             0.0
                          Highest             1      Owner X'0138D404'
   Min. nr of pages:      Average             0.0
                          Highest             1      Owner X'0138D404'
   Nr of occurrences with orphans             0
   Nr of Orphans:           Total             0       0.0%
                          Highest             0      Owner *** N/A ***
   Total size of all SR8s                   368
   Size of largest SR8                      316


Distribution of Index Levels
       ....+....20...+....40...+....60...+....80...+....
    2+|                                                      0    0.0%
    1 |+                                                     1    1.7%
    0 |************************************************-    55   98.2%

Distribution of Minimum Index Levels
       ....+....20...+....40...+....60...+....80...+....
    2+|                                                      0    0.0%
    1 |+                                                     1    1.7%
    0 |************************************************-    55   98.2%

Distribution of Number of SR8s
       ....+....20...+....40...+....60...+....80...+....
    3+|                                                      0    0.0%
    2 |+                                                     1    1.7%
    1+|                                                      0    0.0%
    0 |************************************************-    55   98.2%

Distribution of Number of Index Members
       ....+....20...+....40...+....60...+....80...+....
   76+|                                                      0    0.0%
   72 |+                                                     1    1.7%
    1+|                                                      0    0.0%
    0 |************************************************-    55   98.2%

Distribution of Estimated IOs for Sequential Bottom Level access using 1 Buffer
       ....+....20...+....40...+....60...+....80...+....
    2+|                                                      0    0.0%
    1 |+                                                     1    1.7%
```

```
    0 |***********************************************-        55   98.2%

Distribution of Nr of pages with Intermediate Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    1+|                                                      0    0.0%
    0 |************************************************        56  100.0%

Distribution of Minimum Nr of pages with Intermediate Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    1+|                                                      0    0.0%
    0 |************************************************        56  100.0%

Distribution of % Displaced Intermediate Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    1+|                                                      0    0.0%
    0 |************************************************        56  100.0%

Distribution of Nr of pages with Bottom Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    2+|                                                      0    0.0%
    1 |+                                                     1    1.7%
    0 |***********************************************-        55   98.2%

Distribution of Minimum Nr of pages with Bottom Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    2+|                                                      0    0.0%
    1 |+                                                     1    1.7%
    0 |***********************************************-        55   98.2%

Distribution of % Displaced Bottom Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    1+|                                                      0    0.0%
    0 |************************************************        56  100.0%
Status = 0        SQLSTATE = 00000
```

Printing a REFERENCING ROWID summary report of an index

The following report illustrates the use of the REFERENCING ROWID option to request the printing of the index occurrence containing the row ID of the referencing row identified by X'01390201'.

```
PRINT INDEX "COV-IDX-SET" SEGMENT VLDBSPG1
      TABLE SQLSPG.COVERAGE
      REFERENCING ROWID X'01390201' SUMMARY;
SET Name: COV-IDX-SET
   IBC 70                              Displacement       0
   Sort option NOT SORTED              Key length         N/A
   Duplicates FIRST                    Compression        No
OWNER: SR7
   AREA VLDBSPG1.COVE-AREA             Low page  (SUB-    80106
   Page size 4276                      High page  AREA)   80150
   Page group 5                        Records per page   255
MEMBER: COVERAGE                       Set membership     Optional Automatic
   Located VIA index No                Index is           Linked
   AREA VLDBSPG1.COVE-AREA             Low page  (SUB-    80106
   Page size 4276                      High page  AREA)   80150
   Page group 5                        Records per page   255

OWNER X'01390201' on page 80130
   Top level SR8  on page 80130        utilization     5.7%
Index occurrence totals
   Nr of members                              74
   Nr of levels                                1                1 Minimum
   Size of largest SR8                       316
   Nr of SR8s                                  2                2 Minimum
   Nr of pages with SR8s                       1                1 Minimum
   Nr of displaced SR8s                        0      0.0%
   Nr of entries in use                       74     52.8%
   Nr of Orphans                               0      0.0%
   Total size of all SR8s                    368

Nr of Buffers versus Estimated IOs for Sequential Bottom Level access
-------------        -------------
     1  - 20                     1
Status = 0       SQLSTATE = 00000
```

Report Output Description

- **Part 1—Header**

  The report header provides general information on the index definition, the index owner record or SQL table, and the index member record or SQL table.

- **Part 2—Details for each index occurrence**

  A detailed report on index run-time data per index occurrence is always output for a system-owned index. For a user-owned index, it is output only when explicitly requested using SUMMARY DETAILED. The report provides the following:

  - The DBKEY of the index owner record occurrence and its page number.
  - The page number of the first (top level) SR8. Ideally, the top level SR8 should reside on the same page as the index owner, except for an index with only one level and a non-zero index displacement.
  - The number of entries that are used in the top level SR8 is expressed as a percentage utilization of the maximum IBC count assigned to the index.
  - At the intermediate and bottom level (output only if the index occurrence has more than 1 level):
    - Number of SR8's and its computed minimum value
    - Number of pages with SR8's and its computed minimum value
    - Number of displaced SR8's and as a percentage of SR8's
    - Number of entries in use and as a percentage of available entries
    - Number of orphans and as a percentage of used entries
    - Total size of all SR8's
  - Index occurrence totals:
    - Number of levels in the index and its computed minimum value
    - Number of members in the index
    - Size of the largest SR8
    - Number of SR8's and its computed minimum value
    - Number of pages with SR8's and its computed minimum value
    - Number of displaced SR8's and as a percentage of SR8's
    - Number of entries in use and as a percentage of available entries
    - Number of orphans and as a percentage of used entries
    - Total size of all SR8's
  - Estimated IO's versus number of database buffers for sequential bottom level access indicates the physical "sequentiality" of the index. Ideally, the number of I/O's should not vary with the number of buffers and should be equal to the number of pages with bottom level SR8's.

A displaced SR8 is a bottom level SR8 located within the index displacement or a non-bottom level SR8 located outside the index displacement.

A computed minimum value is obtained by using the current number of entries in the index, filling SR8's to 100% using the current value of INDEX BLOCK CONTAINS for the index, and assuming that all space on a database page is available to hold the index owner and the associated SR8's.

- **Part 3—Index overview and distribution diagrams for a user-owned index**
  - Index overview

    An index overview provides the following information:

    - Number of owner occurrences
    - Number of empty owners and as a percentage of owner occurrences
    - Number of displaced (not on same page as owner) top level SR8's
    - Total, average, and highest value of the number of SR8's
    - Total, average, and highest value of the computed minimum number of SR8's
    - Average and highest value of the index level
    - Average and highest value of the computed minimum level
    - Average and highest values of the computed minimum number of pages
    - Number of index occurrences with orphans
    - Number of orphans: total and as a percentage of the number of entries and highest plus its owner DBKey
    - Total size of all SR8's
    - Size of largest SR8
  - Distribution diagrams

    A distribution diagram provides the number and percentage of index occurrences for a certain property in both a numeric and a pseudo-graphical way. Properties for which a distribution diagram is output are:

    - Index level
    - Minimum index level
    - Number of SR8's
    - Number of members in the index occurrence
    - Estimated IOs using 1 buffer for sequential bottom level access
    - Number of pages with intermediate level SR8's
    - Minimum number of pages with intermediate level SR8's

- Percentage displaced intermediate level SR8's
- Number of pages with bottom level SR8's
- Minimum number of pages with bottom level SR8's
- Percentage displaced bottom level SR8's

More Information

- For more information about designing indexes, see the *CA IDMS Database Design Guide*.
- For more information about defining and maintaining indexes, see the *CA IDMS Database Administration Guide*.

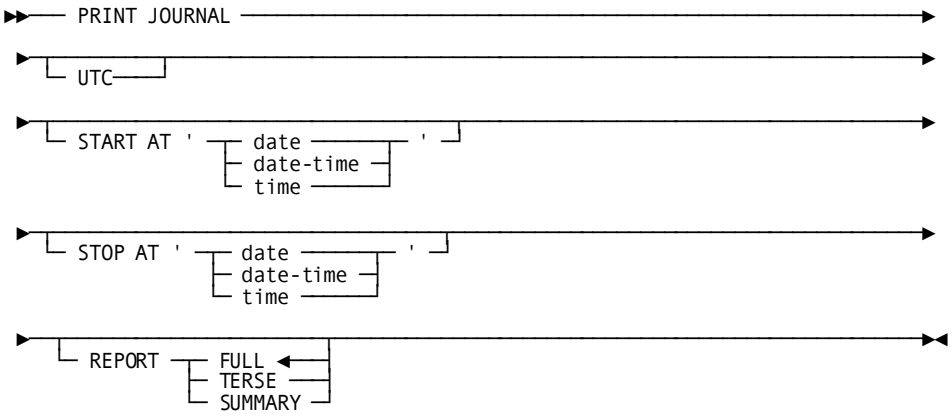
PRINT JOURNAL

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

PRINT JOURNAL Syntax



PRINT JOURNAL Parameter

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 previously. 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.
- **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 the following format:

- *hh:mm:ss*

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

When specifying *time*, the date defaults to the current date.

REPORT

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

FULL

Specifies that all details are to be reported. For every transaction, this includes checkpoints, database statistics, and area usage. All details of a distributed transaction record are reported. This includes local transaction ids with program names, external transaction ids, and resource manager interests. Additionally, 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. For distributed transaction records: external transaction ids and resource manager interests are not included in the report.

SUMMARY

Indicates that only final summary information is produced.

UTC

Specifies that Start and Stop times are interpreted as UTC times instead of local times.

Usage

How to submit the PRINT JOURNAL statement

You submit the PRINT JOURNAL statement only through the batch command facility. When submitting PRINT JOURNAL statements, you must run the batch command facility in local mode.

PRINT JOURNAL and Distributed Transactions

PRINT JOURNAL reports on distributed transactions and supports the use of input and output manual recovery control files. The input manual recovery control file is used to complete InDoubt distributed transactions. If an output manual recovery control file is included in the JCL, an entry will be written for each incomplete distributed transaction encountered. For more information, see JCL Considerations (see page 200) and the "Common Facilities for Distributed Transactions" appendix.

Note: For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Guide*.

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.

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a manual recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

The PRINT JOURNAL utility produces the following report.

```
IDMSBCF                                    IDMS Batch Command Facility                          mm/dd/yy   PAGE 1

   PRINT JOURNAL;
 RU_ID         1  PGM_ID  EMPLOAD   QUIESECE LEVELS  1  UPD  0   BGIN  yyyy-mm-dd-hh.mm.ss.ffffff
 RU_ID         1  PGM_ID  EMPLOAD   QUIESECE LEVELS  0  UPD  0   ENDJ  yyyy-mm-dd-hh.mm.ss.ffffff

  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: yyyy-mm-dd-hh.mm.ss.ffffff  TIME OF LAST COMMIT: NONE                    NUMBER OF COMMITS:       0

                                                          SINCE LAST COMMIT
   TABLESPACES OPENED                BEFORE       AFTER      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
```

Note: For more information about journaling and transaction checkpoints, see the *CA IDMS Database Administration Guide*.

PRINT LOG

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.

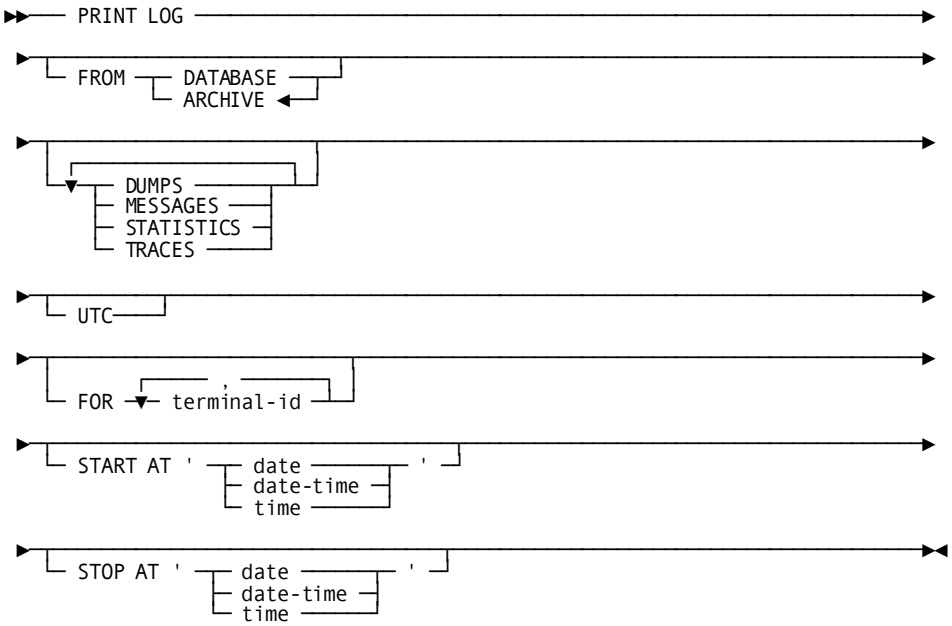
You can select portions for printing based on:

- 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

PRINT LOG Syntax



PRINT LOG Parameter

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 previously. 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 the following format:

- *hh:mm:ss*

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

When specifying only the TIME option, the date defaults to the current date.

UTC

Specifies that Start and Stop times are interpreted as UTC times instead of local times. If coded, all times the utility writes are also displayed as UTC times instead of local times.

Usage

How to submit the PRINT LOG statement

You submit a PRINT LOG statement only through the batch command facility. You can run the batch command facility only 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 z/OS systems or DITTO for z/VSE systems) to print the contents of the log file.

JCL Considerations

When you submit a PRINT LOG statement 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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

Printing everything except dumps

The following PRINT LOG statement 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 following PRINT LOG statement 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 following PRINT LOG statement 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';
```

Sample Output

The following PRINT LOG report 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
CA IDMS-DB/DC  Print Log Utility             CA IDMS-DB/DC is a Proprietary Software Product         DATE       TIME    PAGE
volser         Release nn.n                  Licensed from CA                                     09/19/99   09:26:36     2

091999 09.23.57 IDMS DC013002 V72 T0 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.DDLDCLSCR
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 T0 OPENING SYSTEM RUN UNITS
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCLGSD
091999 09.23.57 IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCLGSD
091999 09.23.57 IDMS DC050001 V72 T0 DCLOG IS 00% FULL
 .
 .
 .
CA IDMS-DB/DC  Print Log Utility             CA IDMS-DB/DC is a Proprietary Software Product         DATE       TIME    PAGE
volser         Release nn.n                  Licensed from CA                                     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 T0 SECURITY SYSTEM INITIALIZATION BEGINNING
091999 09.24.04 IDMS DC013017 V72 T0 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      0005CBC0  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
 .
 .
```

.

More Information

- For more information about viewing the contents of the DC/UCF system log online, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about defining the DC/UCF system log, see the *CA IDMS System Generation Guide*.
- For more information about maintaining the DC/UCF system2 log, see the *CA IDMS System Operations Guide*.
- For more about information statistics written to the DC/UCF system log, see the *CA IDMS Reports Guide*.

PRINT PAGE

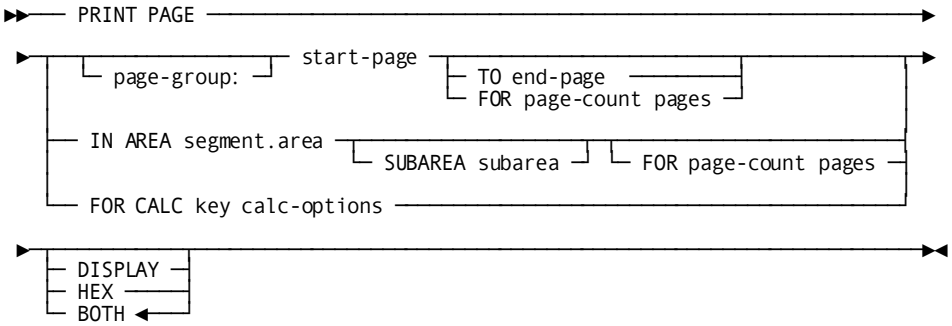
The PRINT PAGE utility prints the contents of one or more database pages in display (decimal) and/or hexadecimal format. You can request printing of:

- The target page for a specified CALC key
- A specified range of pages
- All or some of the pages in an area or subarea

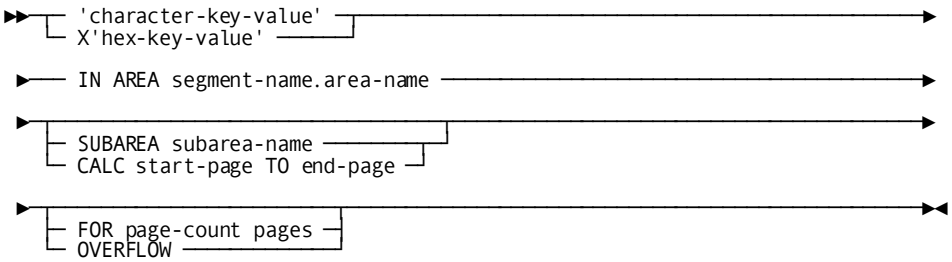
Authorization

To	You Need This Privilege	On
Print one or more pages in an area	DBAREAD	The area

PRINT PAGE Syntax



Expansion of calc-options



PRINT PAGE Parameter

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 continues with the first page of the area. Processing stops 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

Specifies the name of the segment associated with the area whose pages are to be printed.

area

Specifies 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

Specifies 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.

Note: You cannot use the FOR CALC option for locating the target page of a large key value, because you can only specify key values that are less than or equal to 256 bytes.

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

Specifies the name of the segment associated with the area containing the target page to be printed.

area-name

Specifies 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

Specifies 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

Specifies the number of the first page in the page range.

TO

Identifies the end of the page range.

end-page

Specifies 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.

Usage

How to submit the PRINT PAGE statement

You submit the PRINT PAGE statement by using either the batch command facility or the online command facility. When submitting the PRINT PAGE utility through the batch command facility, you must run the batch command facility in local mode.

JCL Considerations

When you submit a PRINT PAGE statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the files containing the pages to be processed. To run under central version include a SYSCTL statement.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Example

Print a specific page in an area

The following example directs PRINT PAGE to print page 75020.

```
print page 75020;
```

Sample Output

The following report lists the contents of page 75020 in response to the PRINT PAGE statement in the previous example.

```
IDMSBCF                                          IDMS Batch Command Facility                          mm/dd/yy   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    617432092305028770147712*
                           *010000003404O5..*
-   420     28     2  00CC     75,020-001     75,020-001     75,020-001
                              75,027-015     75,197-005     75,197-005
                           *7712017806015З......@.......*
-   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
```

More Information

- For more information about defining CALC keys, see *CA IDMS Database Administration Guide*.
- For more information about database pages, see *CA IDMS Database Administration Guide*.

PRINT SPACE

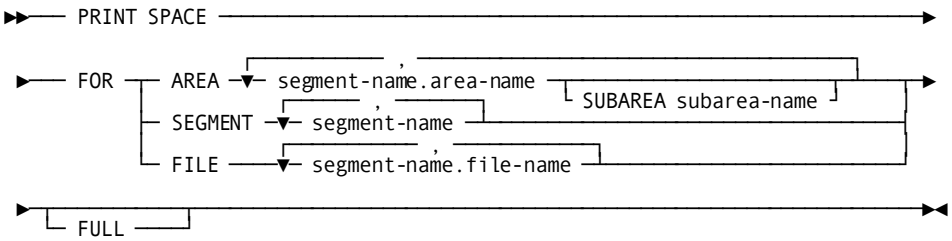
The PRINT SPACE utility reports on space utilization in one or more areas, subareas, or segments.

You can specify whether the report is to be based on information in the space management pages (SMPs) or on information in the database page headers.

Authorization

To	You Need This Privilege	On
Report on space utilization in an area	DBAREAD	The area
Report on space utilization in a segment	DBAREAD	All areas of the segment
Report on space utilization in a file	DBAREAD	Each area of the file

PRINT SPACE Syntax



PRINT SPACE Parameter

FOR

Identifies the areas on which the PRINT SPACE utility is to report.

AREA

Directs the PRINT SPACE utility to report on space utilization in one or more areas or subareas. If no SUBAREA clause is specified, this option produces a report for the entire area plus a report for each file in the area. If a SUBAREA clause is specified, reporting is restricted to the specified subarea.

Note: Native VSAM files are ignored if used in this utility.

segment-name

Specifies the name of the segment associated with the area.

area-name

Specifies the name of the area.

subarea-name

Specifies the name of the subarea associated with 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 are ignored if used in this utility.

segment-name

Specifies 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 you specify the FULL parameter.

segment-name

Specifies the name of the segment associated with the file.

file-name

Specifies 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.

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.

Running Under Central Version

Only PRINT SPACE FOR AREA and PRINT SPACE FOR SEGMENT are supported under central version.

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.

JCL Considerations

When you submit a PRINT SPACE statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the file(s) containing the area(s) to be processed. When running under central version a SYSCTL statement is needed.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

The following report is produced by the PRINT SPACE utility after processing the PRINT SPACE statement by area with the FULL option.

```
IDMSBCF                                          IDMS Batch Command Facility                          mm/dd/yy   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  nn.n          CA IDMS Batch Command Facility     mm/dd/yy    PAGE    1

  PRINT SPACE FOR FILE EMPDEMO.EMPDEMO;

                                    AVAILABLE Space Distribution Report

                                              AVAIL         NUMBER
                                              SPACE        OF PAGES

AREA       EMPDEMO.EMP-DEMO-REGION
 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

             Maximum                                          Percent of
Record Type  Length       Occurrences   Total Space Used      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
```

Note: For more information about space utilization and database pages, see the *Database Administration Guide*.

```
PRINT SPACE FOR AREA SYSSQL.DDLCATX FULL;
                                    AVAILABLE Space Distribution Report
                                              AVAIL         NUMBER
                                              SPACE        OF PAGES
AREA       SYSSQL.DDLCATX
 PAGE SIZE          4,276
 PAGE RESERVE         212
    PAGES          10,001 THRU         13,000
                                             91-100%             56
                                             81-90 %            117
                                             71-80 %            125
                                             61-70 %            197
                                             51-60 %            197
                                             41-50 %            229
                                             31-40 %            230
                                             21-30 %            225
                                             11-20 %            203
                                             00-10 %          1,419
                                             SMPS                 2
                                             TOTAL            3,000
    FILE   SYSSQL.DDLCATX
    BLOCKS              1 THRU          3,000
Total Space Allocated                                12,828,000
Total Space Available            (Percent)            3,240,368     (25.26%)
Total Reserved Space Available   (Percent)              511,128     ( 3.98%)
Total Unreserved Space Available (Percent)            2,729,240     (21.28%)
Total Space Used                                      9,587,632
Logically Full Pages                                          0
Total Space Unusable     (Percent)                            0    (  0.00%)
AREA       SYSSQL.DDLCATX              Distribution of USED Space Report
            Maximum                                     Percent of
Record Type  Length       Occurrences   Total Space Used      Total Used
Null Line         8              54                432            0.00
  SR5            48               1                 48            0.00
  SR7            40              12                480            0.00
  SR8         1,404          28,567          9,482,184           99.93
Space Inv.    4,244               2              8,488            0.08
Overhead         32           3,000             96,000            0.98
        *** NO logically deleted records found ***
```

PRINT TRACE

The PRINT TRACE utility prints all or selected trace entries that reside in a DDLDCTRC area, a DDLDCLOG area, or one or more archive files created by the ARCHIVE TRACE or ARCHIVE LOG utilities.

You can select entries for printing based on the following:

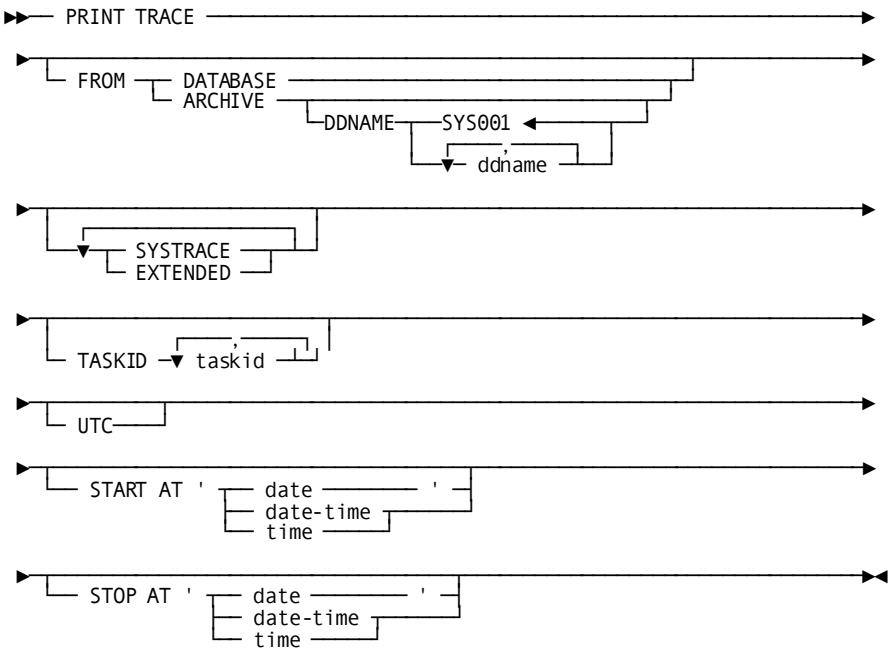
- Type of information
- Date and time
- Taskid

Authorization

To	You Need This Privilege	On
To print trace information	DBAREAD	SYSTEM.DDLDCTRC or SYSTEM.DDLDCLOG areas

PRINT TRACE Syntax

The following diagram shows the syntax for the PRINT TRACE utility statement:



PRINT TRACE Parameters

This section describes the PRINT TRACE command parameters:

FROM

Indicates the location from which to print trace information: a database area or an archive file.

DATABASE

Prints trace information from either the DDLDCTRC or the DDLDCLOG area. If the DMCL contains a DDLDCTRC area, only trace information in that area is printed; otherwise, only trace information in the DDLDCLOG area is printed.

ARCHIVE

Prints trace information from one or more archive files.

DDNAME

Identifies the ddname of one or more archive files whose contents are to be printed. If more than one ddname is specified, trace information is merged and displayed in chronological sequence.

ddname

Specifies the ddname of an archive file.

Default: SYS001

Limit: 32 ddnames

Note: Printing trace information from multiple archive files is only available to z/OS users. z/VSE users need to consolidate multiple archive files into a single file (in the order in which they were created) and use this file with the PRINT TRACE utility.

SYSTRACE

Includes basic system trace information in the output.

EXTENDED

Includes extended trace information in the output.

Note: If you specify SYSTRACE or EXTENDED, only the specified type of information is printed. If you do not specify either option, both types of information are printed.

TASKID

Prints trace information for selected tasks.

taskid

Specifies the taskid of a task whose trace information is to be printed.

Default: Trace information associated with all tasks is printed.

Limit: 32 task identifiers.

START AT *date-time*

Prints only trace information recorded at or after the specified time.

Default: Prints information from the beginning of the database area or archive file.

STOP AT *date-time*

Prints only trace information recorded at or before the specified date and time.

Default: Prints information recorded in the database area or archive file (starting at the time specified in the START AT parameter, if any).

Expansion of date-time

date

Specifies a date, in one of the following formats:

- yyyy-mm-dd
- mm/dd/yyyy

The date components are:

- yyyy specifies the year.

  Limit: 0001–9999 (leading zeros are optional)

- mm specifies the month within the year.

  Limit: 01–12 (leading zeros are optional)

- dd specifies the day within the month.

  Limit: 01–31 (leading zeros are optional)

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

date-time

Specifies a date and time in one of the following formats:

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

The date components are the same as those that can be specified for *date*.

The time components are:

- hh specifies the hour on a 24-hour clock.

  Limit: 00-23 (leading zeros are optional)

- mm specifies the number of minutes past the hour.

  Limit: 00-59 (leading zeros are optional)

- ss specifies the number of seconds past the minute.

  Limit: 00-59 (leading zeros are optional)

- ffffff specifies the number of millionths of a second past the specified second.

  Limit: 000000-999999 (trailing zeros are optional)

  Default: 000000

time

Specifies a time in the following format:

- hh:mm:ss

The time components are the same as those that can be specified for *date-time*.

Note: The date is assumed to be the current date.

UTC

Specifies that Start and Stop times are interpreted as UTC times instead of local times.

PRINT TRACE Usage

How to submit the PRINT TRACE statement

You submit a PRINT TRACE statement only through the batch command facility. You can run the batch command facility only in local mode.

When to use PRINT TRACE

Use the PRINT TRACE utility when the system trace is being written to the DDLDCTRC area or DDLDCLOG area, if no DDLDCTRC area is defined in the dictionary.

JCL Considerations

When you submit a PRINT TRACE statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The system trace area (DDLDCTRC) or system log area (DDLDCLOG) if you specify FROM DATABASE
- The archive trace file whose contents you want to print if you specify FROM ARCHIVE
- The system message area (DDLDCMSG)
- The dummied journal file

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

Printing All Trace Entries

The following PRINT TRACE statement requests printing of all SYSTRACE and EXTENDED trace entries from the trace area.

```
PRINT TRACE FROM DATABASE;
```

Printing All Trace Information for a Specified Time Period

The following PRINT TRACE statement requests printing of all SYSTRACE and EXTENDED trace entries from the archive file, starting from March 02, 2011 at 4:12 p.m. until just before 4:30 p.m. on March 02, 2011 using the archive file with DDNAME SYS001.

```
PRINT TRACE FROM ARCHIVE
  DDNAME SYS001
  START AT '2011-03-02-21.12.00'
  STOP AT  '2011-03-02-21.30.00';
```

Printing SYSTRACE Entries from an Archive File

The following PRINT TRACE statement requests printing of all SYSTRACE trace entries from the archive file with DDNAME SYS001.

```
PRINT TRACE FROM ARCHIVE
  DDNAME SYS001
  SYSTRACE;
```

Printing SYSTRACE and EXTENDED TRACE Entries from an Archive File

The following PRINT TRACE statement requests printing of all trace entries from the ARCHIVE file with DDNAME LOGARC.

```
PRINT TRACE FROM ARCHIVE
  DDNAME LOGARC;
```

PRINT TRACE Output for System Trace Entries

The following items are included in the PRINT TRACE output for system trace entries.

TOD TIME/DATE

The UTC time and date the trace entry was generated.

SCA

The relative SCA number of the subtask that generated the trace entry.

TASKID

Taskid of the task that generated the trace entry.

A(LTE)

The address of the generating task's LTE.

MOD

The internal module number and four-character identifier of the module generating the trace entry.

MAC

The issuing macro number or entry point.

CALL

The relative macro expansion within the issuing program.

R11/R2-R1/R8

The contents of registers 11 through 1 and 2 through 8 at the time the entry was generated.

TOD CLOCK

The contents of the TOD clock at the time the entry was generated.

<SYSTEM73>

Node or data sharing member name of issuing system.

PRINT TRACE Output for Extended Entries

The following PRINT TRACE statement requests printing of all SYSTRACE and EXTENDED trace entries from the archive file starting from March 02,2011 at 4:12 p.m. until just before 4:30 p.m. on March 02,2011 using the archive file with DDNAME SYS001.

```
PRINT TRACE FROM ARCHIVE
   DDNAME SYS001
   START AT '2011-03-02-21.12.00'
   STOP AT  '2011-03-02-21.30.00';
```

Sample Output

The following example shows the output from the Print Trace Utility:

```
PRINT TRACE;
CA IDMS DB/DC  Print Trace Utility           CA IDMS DB/DC is a Proprietary Software Product          DATE       TIME      AGE
CAGJI0         Release 1800                                Licensed from CA, Inc.                  01/19/10   14:31:32      1


*** DDLDCTRC AREA FROM PAGES 000060001 TO 000062000
*** FIRST AND LAST PAGES SELECTED ARE 000060019 AND 000060041

*** Trace entries generated on: 2010-01-19

TRACE ENTRIES, ORDERED OLDEST TO NEWEST.
                    TASKID/
  TOD TIME/DATE SCA   A(LTE)   MOD   MAC  CALL    R11/R2   R12/R3   R13/R4   R14/R5   R15/R6    R0/R7    R1/R8        TOD CLOCK
19.31.33.776391 00       122   292    22    13  801D8208 38BD5456 36EC82AC B8BD5500 382B4CB8 00027350 001D820C C5699CD3C0807706
       SYSTEM73     36E2B7C8  MTSY TSKCEP1      36E99D88 00000000 0025E0C0 36E99D88 3701CFBC 39C07000 B8265E42
19.31.33.776391 00       122     5   240     1  801D8208 382B4CB8 36EC82D8 B82B4CC8 382B4CB8 00027350 001D820C C5699CD3C0807786
       SYSTEM73     36E2B7C8  TSKC #GETSTK      36E99D88 00000000 0025E0C0 36E99D88 3701CFBC 39C07000 B8265E42
19.31.33.776391 00       122     5    12     4  36EC82D4 382B4CB8 36EC8320 B82B5326 3825C888 0025E030 36EC82D8 C5699CD3C0807906
       SYSTEM73     36E2B7C8  TSKC RMGREP2      0025E680 0025E000 0025E700 000263F0 00000000 00000000 00000000
19.31.33.776393 00       122     5     9     5  36EC82D4 382B4CB8 36EC8320 B82B5558 38289CB8 00000000 00000000 C5699CD3C0809686
       SYSTEM73     36E2B7C8  TSKC TIMPGET      0026EEC8 00000063 0025E700 0026E9C0 00000000 00000000 00000000
19.31.33.776393 00       122    14   240     1  0000003C 38289CB8 36EC834C B8289CCC 38289CB8 00000000 00000000 C5699CD3C0809706
.
.
.
```

```
PRINT TRACE;
CA IDMS DB/DC  Print Trace Utility           CA IDMS DB/DC is a Proprietary Software Product            DATE       TIME
AGE
CAGJI0         Release 1800                              Licensed from CA, Inc.                       01/19/10   14:31:32


*** Trace entries generated on: 2010-01-19

19.31.33.872678 00       122     8   109    42  36EC8278 382B73B8 36EC82C4 B82B76C4 383D7BD8 00000001 36EC827C C5699CD3D8026486
       SYSTEM73     36E2B7C8  TSKI TMGREP1      00000000 36E2B7C8 36E63708 379CAC88 00000000 38262798 382B83B8
19.31.33.872678 00       122   329   240     1  00000238 383D7BD8 36EC82EC B83D7C0C 383D7BD8 00000001 36EC827C C5699CD3D8026586
       SYSTEM73     36E2B7C8  TMGR #GETSTK      36EC827C 0000008E 00000052 383D8D50 382B76CA 00000001 382B83B8
19.31.33.872678 00       122   442  TMGD  TM1*    SEQ=1       LTE=36E2B7C8                                    C5699CD3D8026E06
       SYSTEM73     E3D4F15C 36E2B7C8 E3C5D5C4 00000000  00008000 00000000 00000000 0000007A *TM1*.S.HTEND.....................:*
                    00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 *.............................*
19.31.33.872679 00       122   442  TMGD  TMR*    SEQ=1       LTE=36E2B7C8                                    C5699CD3D8027906
       SYSTEM73     E3D4D95C 36E2B7C8 E2A48299 E3C5D5C4  36E2B7C8 5BE3C3D6 D4404040 0000007A *TMR*.S.HSubrTEND.S.H$TCOM    ...:*
19.31.33.872680 00       122   442  TMGD  TMR*    SEQ=1       LTE=36E2B7C8                                    C5699CD3D8028006

.
.
```

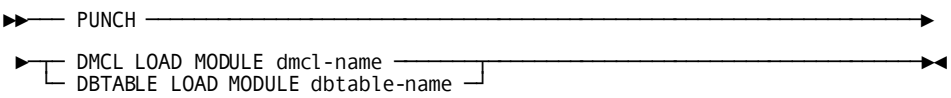
PUNCH

The PUNCH utility retrieves the DMCL or database name table load module from the dictionary and writes it, 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

PUNCH Syntax



PUNCH Parameter

DMCL LOAD MODULE

Directs the PUNCH utility to punch a DMCL load module from the dictionary.

dmcl-name

Specifies 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

Specifies the load module name of the DBTABLE to be punched.

Usage

How to submit the PUNCH statement

You submit the PUNCH statement only through the batch command facility.

JCL Considerations

When you submit a PUNCH statement 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.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

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
```

More Information

- For more information about defining DMCL and DBTABLE modules, see the *CA IDMS Database Administration Guide*.
- For more information about deleting DMCL and DBTABLE load modules, see the *CA IDMS Database Administration Guide*.

RELOAD

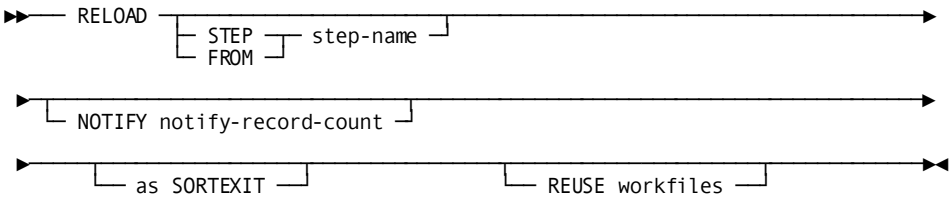
The RELOAD utility reloads a database using input created by the UNLOAD utility statement.

Note: You cannot use RELOAD to reload an SQL database that contains keys longer than 256 bytes. Instead, use the REORG utility to reorganize a large key database.

Authorization

To	You Need This Privilege	On
Reload data into a segment	DBAWRITE	All areas associated with the segment

RELOAD Syntax



RELOAD Parameter

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

Specifies 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

Specifies the name of the step to execute.

The name must be one of the following:

- SORT1
- IDMSDBL2
- SORT2
- IDMSDBLX
- SORT3
- IDMSDBL3
- SORT4
- IDMSDBL4

Note: For more information and an explanation about when you can restart steps IDMSDBL2, IDMSDBL3, or IDMSDBL4, see "Usage" in this section.

NOTIFY

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

notify-record-count

Specifies 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 DBL*x* 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.

Usage

How to submit the RELOAD utility

You submit the RELOAD utility 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 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 could 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 the one 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, see 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 overlayed by the updates in the restarted IDMSDBL4.

Note: For more information and help in deciding whether to run the RELOAD utility all at once or in steps, see the *CA IDMS Database Administration Guide*.

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 following:
  - Unloaded data from UNLOAD processing
  - 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 restart a step only 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 data set 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 see the "Intermediate Work File" tables under "JCL Considerations." Partially created output files should be deleted before you restart 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 IDMSDBLx 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 data set 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

Step	Input	Output
IDMSDBLX	SYS004	SYS006
SORT3	SYS005 SYS006	SYS004
IDMSDBL3	SYS004	SYS005
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	■ Populates the database by means of an area sweep but does not connect any sets. ■ Prints statistics on the records written to the database. **Note:** 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.
SORT2	Sorts the contents of SYS003 and SYS006.
IDMSDBLX	■ Establishes pointers for each index in which each record participates but does not build the indexes in the database.

Step	Description
SORT3	Sorts the contents of SYS005 and SYS008.
IDMSDBL3	■ 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. ■ Builds indexes in the database.
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
SORT1	■ SYS001 contains sort parameters from UNLOAD SYSPCH file ■ SYS002 contains the record information from UNLOAD	SYS004 contains the sorted record information	SYS004 record length = SYS002 record length from UNLOAD
IDMSDBL2	■ SYS004 from SORT1 ■ RELDCTL file from UNLOAD utility	■ SYS005 contains member descriptors for chained sets ■ SYS006 contains member descriptors for user owned index sets ■ SYSPCH contains sort parameters ■ SYSLST contains statistics report	SYS005 record length = 32 bytes SYS006 record length = 32 bytes

Step	Input	Output	Size
SORT2	■ SYS001 contains sort parameters from IDMSDBL2 SYSPCH* ■ SYS003 contains index information from UNLOAD ■ SYS006 contains member descriptors for user owned index sets from IDMSDBL2	SYS007 contains the sorted contents of SYS003 and SYS006	SYS007 record length = larger of RELOAD's SYS006 or UNLOAD's SYS003
IDMSDBLX	■ SYS007 contains sorted index information from SORT2 ■ RELDCTL file from UNLOAD utility	SYS008 contains reformatted index information	SYS008 record length = same as SYS007
SORT3	■ SYS001 contains sort parameters from IDMSDBL2 SYSPCH* ■ SYS005 from IDMSDBL2 ■ SYS008 from IDMSDBLX	SYS009 contains sorted index set descriptors	SYS009 record length = larger of RELOAD's SYS008 or IDMSDBL2's SYS005
IDMSDBL3	■ SYS009 from SORT3 ■ RELDCTL file from UNLOAD utility	■ SYS010 contains prefix pointer information ■ SYSPCH contains sort parameters ■ SYSLST contains a statistics report	SYS010 record length = 28 bytes

Step	Input	Output	Size
SORT4	■ SYS001 contains sort parameters from IDMSDBL3 SYSPCH* ■ SYS010 from IDMSDBL3	SYS011 contains sorted prefix pointer information	SYS011 record length = same as SYS010 from prior step
IDMSDBL4	■ SYS011 from SORT4 ■ RELDCTL file from UNLOAD utility	SYSLST contains messages on the results of the reload operation	

* The sort parameters noted in the previous 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 z/VSE. 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.

DBLUEREX User Exit

RELOAD modules IDMSDBL2, IDMSDBL3, and IDMSDBL4 have the ability to call a user exit named DBLUEREX when processing errors are encountered. This exit will be called if an unexpected error status is returned from a call to CA IDMS for all three modules. The exit is also called by IDMSDBL3 if an error is encountered while processing the intermediate file containing records to rebuild the database's chain sets.

The DBLUEREX module must be linked with each IDMSDBLx module from which the user wishes it to be invoked. The modules must be coded so that they are able to run with AMODE=31 and RMODE=ANY so that they will match the modes of the IDMSDBL? modules. The following statements should be used to link DBLUEREX with the desired module(s).

```
INCLUDE LIB1(DBLUEREX)
INCLUDE LIB2(IDMSDBL?)
ENTRY #DDNHD
```

where the ? represents 2, 3, or 4 depending on the module to which the exit is being linked.

Upon entry to DBLUEREX, Register 1 will point to a list of three fullwords with the following contents. The last fullword in the list will have its high-order bit turned on to signal the end of the parameter list.

Offset	Description
x'00'	Address of an eight byte field containing the name of the module making the call.
x'04'	Address of the error status field in the run-unit's subschema control for calls from IDMSDBL2, IDMSDBL4, and IDMSDBL3 if a database error is encountered. This parameter will contain zeros if called by IDMSDBL3 when an error is encountered while processing the intermediate file containing records defining the database's chain sets.
x'08'	Address of the current input record being processed by the calling module.

JCL Considerations

When you submit a RELOAD statement through the batch command facility, the JCL to execute the facility must include statements to define:

- Files containing the areas to be reloaded
- 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 SORT*x* 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.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

The RELOAD utility generates the following listing after the successful execution of the RELOAD statement in the previous example. Note that the file the unloaded database is reloaded into is formatted first.

```
IDMSBCF                                          IDMS Batch Command Facility                          mm/dd/yy   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 nn.n TAPE volser SORT STARTED
UT009002    105 RECORDS WERE READ FROM SYS002
UT009003    105 RECORDS WERE WRITTEN TO SYS004
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT1    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL2
UT005001 IDMSDBL2 RELEASE nn.n TAPE volser PROCESSING STARTED
UT005002 DATABASE LOAD STATISTICS
 DATABASE LOADED ON mm/dd/yy 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 nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL2 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT2
UT009001 IDMSDBLY RELEASE nn.n TAPE volser 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 nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT2    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBLX
UT008001 IDMSDBLX RELEASE nn.n TAPE volser PROCESSING STARTED
UT008002    67 RECORDS WERE WRITTEN TO SYS008
UT008006 IDMSDBLX RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBLX HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
UT009001 IDMSDBLY RELEASE nn.n TAPE volser 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 nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT3    HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE nn.n TAPE volser 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 nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT4
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002    42 RECORDS WERE READ FROM SYS010
UT009003    42 RECORDS WERE WRITTEN TO SYS011
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT4    HAS COMPLETED SUCCESSFULLY
```

```
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL4
UT007001 IDMSDBL4 RELEASE nn.n TAPE volser PROCESSING STARTED
UT007004    NO DATABASE ERRORS WERE ENCOUNTERED
UT007005    NO LOGIC ERRORS WERE ENCOUNTERED
UT007002    42 RECORDS WERE READ FROM SYS011
UT007006 IDMSDBL4 RELEASE nn.n 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
```

Note: For more information about unloading and reloading a CA IDMS/DB database, see the *CA IDMS Database Administration Guide*.

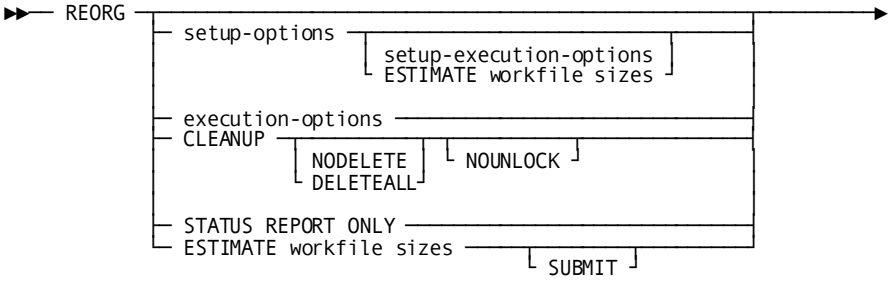
REORG

The REORG utility reorganizes a database by unloading data from one or more areas in one database and reloading it into one or more areas of another database.

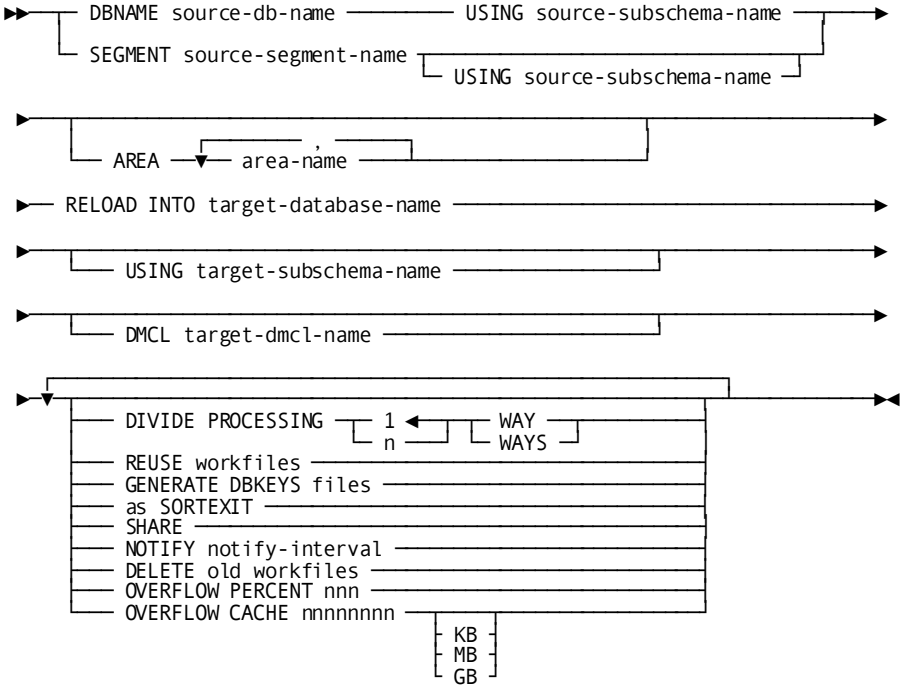
Authorization

To	You Need This Privilege	On
Unload an area	DBAREAD	The area itself Any area(s) with set connections to the area Any area(s) containing an index defined on records in the area
Unload a segment	DBAREAD	All areas of the segment
Reload data into an area	DBAWRITE	The area itself Any area(s) with set connections to the area Any area(s) containing an index defined on records in the area
Reload data into a database	DBAWRITE	All areas into which the data is being reloaded Any area(s) with set connections to target areas Any area(s) containing an index defined on records in the target areas

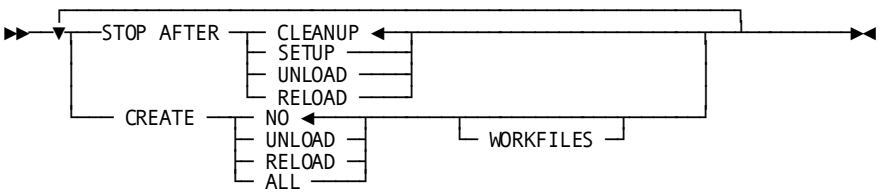
REORG Syntax



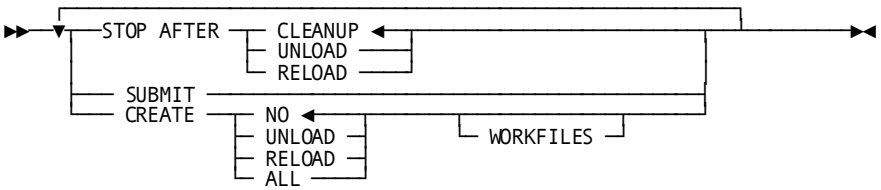
Expansion of *setup-options*



Expansion of *setup-execution-options*



Expansion of *execution-options*



REORG Parameter

CLEANUP

Specifies to perform a cleanup to delete work files and unlock target areas. You can use this option to complete a successful unload/reload operation if cleanup was not performed automatically. It can also be used to delete work files created by an incomplete unload/reload operation that is not being continued.

Note: For more information, see Considerations for running REORG on VSE.

NODELETE

Specifies to not delete work files during the cleanup phase.

DELETEALL

Directs an explicit cleanup job to delete all work files associated with the current control file, including those that were not dynamically allocated by the current REORG operation. This option does not apply to DBKEYS files.

By default, only files dynamically allocated by the current REORG operation are deleted.

NOUNLOCK

Specifies to not unlock the reloaded database areas during the cleanup phase.

STATUS REPORT ONLY

Specifies to produce a status report by reading the reorganization control file. This can be run while other REORG jobs are executing to obtain a current status of the entire REORG process.

ESTIMATE workfile sizes

Directs REORG to estimate the size of work files by gathering statistics in a separate pass of the database. This option generates estimates for both UNLOAD and RELOAD work files.

By default, statistics are collected during the unload phase that can be used for sizing RELOAD work files only. UNLOAD work files must be sized manually.

If specified together with *setup-options*, statistics gathering starts as soon as the setup phase is complete and processing stops after the statistics have been gathered. Additional jobs are automatically submitted to help gather statistics and process the database by UNLOAD slice and index.

If specified independently of *setup-options*, it must be specified in a separate execution of the REORG utility that occurs between the setup and UNLOAD phases. Multiple jobs are submitted to gather statistics only if the SUBMIT option is specified. Processing stops when file estimation is complete.

When file estimation is complete, processing must be restarted by specifying a new STOP AFTER point.

File size estimates are automatically used when dynamically allocating work files if no primary space value is specified for the file's DSMODEL.

source-db-name

Specifies the database name the source subschema with be bound to. All bound areas will be unloaded, unless restricted with the area-name option. The source subschema name must be specified with this option.

source-segment-name

Specifies the segment of the database whose areas are to be unloaded. The segment is referred to as the source segment, and the database is referred to as the source database.

source-subschema-name

Specifies the subschema that describes the source database.

By default, if the source-subschema-name is not specified, it indicates that an SQL-defined segment is to be unloaded. In this case, the REORG utility extracts the description of the database from the dictionary to which the session is connected.

area-name

Specifies an area in the source segment to unload. Records in this area may target different areas in the target database.

By default, if no areas are specified for a non-SQL defined database, all areas in the source segment that are also defined in the source subschema are unloaded. If no areas are specified for an SQL-defined database, all areas in the segment are unloaded.

Target-database-name

Specifies the name of the database into which data is reloaded. This is referred to as the target database.

If an SQL-defined database is being reorganized, the *target-database-name* must be the same as the *source-segment-name* specified in the SEGMENT clause.

If a non-SQL-defined database is being reorganized, *target-database-name* must be the name of either a segment or a DBNAME identifying the database into which data is to be reloaded.

Target-subschema-name

Specifies the subschema that describes the database into which the data is reloaded.

Do not specify a target subschema if reorganizing an SQL-defined database.

By default, if reorganizing a non-SQL-defined database and *target-subschema-name* is not specified, the source subschema is also the target subschema.

Target-DMCL-name

Specifies the DMCL that defines the database into which the data is reloaded.

By default, if a *target-dmcl-name* is not specified, the DMCL that describes the source database is used during the reload phase and therefore must also describe the target database.

See the source/target DMCLs in Initiating a REORG Operation (see page 259) for additional information.

DIVIDE PROCESSING n WAYS

Specifies the degree to which parallel processing is to be used to reduce elapsed time. The default is 1, which means that one job processes data and another processes system indexes.

Use of this option exponentially increases the number of work files needed. For more information, see Work files and the n WAYS parameter in Work Files (see page 267).

REUSE workfiles

Specifies to reuse work files as much as possible. Use of this option reduces the number of work files that are needed.

GENERATE DBKEYS files

Specifies that each task that stores a record into the new database generates a DBKEYS file that maps the old db-key of the record with the new db-key of the record.

For more information, see Generating db-key cross-reference information in Using the REORG Utility (see page 254).

as SORTEXIT

Specifies to process sorted output data directly from the SORT, instead of writing it to an intermediate work file. Use of this option reduces the number of files and the amount of disk space needed.

SHARE

Specifies to not lock source areas to prevent concurrent update. By default, source areas are locked at the start of unload processing.

When source areas are locked, they remain locked even after the REORG has completed. This prevents updates to the source area under the assumption that the target database is now the current database.

notify-interval

Specifies a time-interval in minutes. Each time this interval expires, a message is written indicating the current task being processed and the percentage of records processed to that point. This message is not produced while data is being sorted.

DELETE old workfiles

Directs REORG setup to delete work files that may be left from a previous run. All existing work files that match the name of a new work file are deleted, including DBKEYS files. For more information see, Considerations for running REORG on VSE.

By default, old work files are reused.

OVERFLOW PERCENT *nnn*

Specifies the percentage used to estimate the size of SYSOF2 and SYSOF8 work files and all output work files for the two overflow tasks: RELOAD2 and RELOAD5. The size of these files cannot be predicted, so they are estimated to be a percentage of related work files.

By default, 10% is used.

OVERFLOW CACHE *nnnn*

Specifies a limit on the size of the overflow cache used to temporarily hold records that do not fit on their target page.

nnn is the maximum size of the overflow cache specified in bytes, Kilobytes (2**10), Megabytes (2**20), or Gigabytes (2**30), depending on whether it is followed by no qualifier or by KB, MB, or GB respectively.

A value larger than 2**31-1 is limited to 2**31-1 bytes.

By default, 32 KB is used.

STOP AFTER

Specifies the processing phase after which execution halts. The options are:

CLEANUP

Continues processing until the entire reorganization operation is complete. CLEANUP is the default if *setup-options* are specified.

SETUP

Terminates processing after setup is complete and before unload is initiated. This option is valid only if *setup-options* are specified.

UNLOAD

Terminates processing after the unload phase is complete and before reload is initiated.

RELOAD

Terminates processing after the reload phase is complete and before cleanup is initiated.

If neither *setup-options* nor STOP AFTER is specified on a REORG statement, the processing stop point remains unchanged from that previously established.

If STOP AFTER is specified, and the reorganization control file indicates that processing has already been initiated for a phase beyond that specified as a stop point, a warning is issued and processing is terminated.

SUBMIT

Specifies to submit JCL for additional REORG jobs to the internal reader when a REORG operation is restarted.

This option is not valid during the initial setup phase, because by default, if processing continues beyond the initial setup phase, JCL is automatically submitted before starting the unload phase.

The JCL to be submitted is retrieved from the RORGJCL file. If no RORGJCL is present in the execution JCL, the SUBMIT option is ignored.

CREATE NO/UNLOAD/RELOAD/ALL WORKFILES

Specifies whether to create work files and for which phases. The options are:

NO

Does not dynamically create work files. This is the default.

UNLOAD

Creates only the files used during unload processing.

RELOAD

Creates only the files used during reload processing.

ALL

Creates both unload and reload work files.

Note: For information about REORG's work files and using data set models to create them, see Work Files (see page 267).

Usage

This section describes the reorganization process and provides guidelines and discusses considerations in the use of the REORG utility.

Using the REORG Utility

How to submit the REORG statement

You submit the REORG utility only through the batch command facility. You must run the batch command facility in local mode.

The REORG utility is currently supported only in z/OS environments.

REORG processing overview

The REORG utility unloads and reloads a CA IDMS database using parallel processing to reduce the amount of time it takes to reorganize data. To enable parallel operations, the utility divides the source and target databases into slices and groups of system-owned indexes. Each slice or index group can be processed (unloaded or reloaded) in parallel by concurrently executing jobs. The amount of concurrency is generally controlled by the DIVIDE PROCESSING n WAYS option; the higher the value specified, the more concurrency is possible.

A REORG operation is controlled through a control file that is built during operation initiation and is accessed by all jobs executing as part of the same REORG operation. The control file describes the specifics of the operation and is updated with status information reflecting the progress that has been made in operation execution. Each job that performs work on behalf of a REORG operation updates the control file. Operating system facilities are used to serialize these updates and to coordinate concurrently executing jobs.

A REORG operation is comprised of the following major execution phases:

- Setup—in which the specifics of the operation are determined and its control file built.
- Unload—in which the data is extracted from the source database and written to sequential files.
- Reload—in which the extracted data is loaded into the target database and inter-record relationships and indexes are built.
- Cleanup—in which work files may be deleted and target areas unlocked.

General reorganization procedures

Use the following procedures to reorganize a database. These procedures may vary slightly depending on the type of change being made and other operational considerations.

To REORG a database, take the following steps:

- Create the new subschema, segments and/or DMCL reflecting the changes to be made.
- Back up all areas that may be updated during the process.
  - Areas being unloaded that contain logically deleted records.
  - Areas being unloaded if data will be reloaded back into the same areas.
  - Areas physically linked to the unloaded areas through sets, linked constraints, or system indexes.
- If necessary, execute the CLEANUP utility to eliminate any logically deleted records from the areas being unloaded.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

If the target database files are the same as the source database files, take the following steps:

- Execute the REORG utility specifying STOP AFTER UNLOAD.
- If the SHARE option was not specified, use the UNLOCK command to unlock all areas into which the data will be reloaded.
- Format the areas into which the data will be reloaded.
- Execute the REORG utility specifying STOP AFTER CLEANUP and SUBMIT.
- Back up the updated areas. The new areas just loaded and any linked areas that were updated.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

If the target database files are different from the source database files, and the two are described by different segment definitions whose files have different DDNAMEs and data set names, take the following steps:

- Format the areas into which the data will be reloaded.
- Execute the REORG utility specifying STOP AFTER CLEANUP.
- Back up the updated areas. The new areas just loaded and any linked areas that were updated.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

If the target database files are different from the source database files, but the two have the same DDNAMEs, take the following steps:

- Format the areas into which the data will be reloaded.
- Execute the REORG utility specifying STOP AFTER UNLOAD. Ensure that the REORG JCL and the JCL in the RORGJCL file reference the source database files, the files containing the data to be unloaded.

- Execute the REORG utility specifying STOP AFTER CLEANUP and SUBMIT. Ensure that the REORG JCL and the JCL in the RORGJCL file reference the target database files, the files into which the data is to be loaded.
- Back up the updated areas. The new areas just loaded and any linked areas that were updated.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

More Information

- For more information about the use of the RORGJCL file, see Automatic job submission in Controlling REORG Execution (see page 261).
- For more information about the JCL needed to execute the REORG utility, see JCL Considerations (see page 282).
- For more information about general guidelines and considerations on database reorganization, see the *CA IDMS Database Administration Guide*.

Reducing REORG elapsed time

You can substantially reduce the time it takes the REORG utility to unload and reload a database by using chained reads. This enables multiple pages to be read with a single I/O. Its use benefits both the unloading of data, which is primarily accomplished through an area sweep and the reloading of data, which REORG does by loading data in a forward direction. You will get the most benefit if you use chained reads for all areas being updated by the reload phase, which is most easily done using the PREFETCH facility. IDMSQSAM can be used instead, but because it applies to at most one area at a time, its benefit is limited.

Note:

- For more information about using chained reads, see the *CA IDMS Database Administration Guide*.
- For more information about PREFETCH and IDMSQSAM related SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.

Relative REORG performance

While in many cases REORG will require less time and resources to unload and reload a database than the UNLOAD/RELOAD utilities, this may not always be the case. For certain database structures, REORG will take longer to run and cost more in terms of CPU and I/Os. The best means of determining how REORG performs against a specific database is to do a trial run during a non-critical processing window. This should be done as part of planning a database reorganization.

Generating db-key cross-reference information

The REORG utility optionally creates a set of DBKEYS work files whose records relate the db-key of a record in the source database with its corresponding db-key in the target database. These files are not used by REORG, but can be used by a user-written application or a third-party product for the purpose of cross-referencing the old and new locations of a record. You may need to merge these files together prior to using the information.

Note: For more information about and a description of the records in the DBKEYS files, see DBKEYS File Layout (see page 293).

Efficiency of the reorganized database

The database resulting from a REORG operation should be as efficient as one reorganized through UNLOAD and RELOAD even though the two may not be identical.

A database processed by RELOAD is loaded back to front. CALC and VIA records that overflow are usually written to pages that have already been loaded and have room. A database processed by REORG is loaded from front to back. CALC and VIA records that overflow are saved in a memory cache so that they do not displace records targeting later pages. If the cache is not large enough, the records are written to an overflow file and loaded in a later step.

The resulting databases should be similar in terms of the number of records that are stored on their intended target page and the number of records that overflow, but the two databases will not be exactly the same.

You can use the IDMSDBAN utility to obtain a report of the number of page changes needed to traverse all occurrences of each CALC and VIA set. By executing this utility before and after reorganization, you can determine the effect that REORG has had on these statistics and therefore the relative efficiency of the resulting database.

Implementing Changes Through REORG

Changing an SQL-defined database

You use REORG on an SQL-defined database to modify the database based on changes made to its segment definition. These changes must be reflected in the target DMCL. The modified DMCL must be available during the entire REORG operation, including the setup and unload phases.

You can make the following changes using REORG:

- Change area page ranges
- Change page size

  See EXPAND PAGE (see page 84) for an alternative means of changing a page size.
- Change the maximum number of records per page

  See CONVERT PAGE (see page 71) for an alternative means of changing the maximum number of records per page.

The names of the source and target segments for a REORG operation on an SQL-defined database must be the same. Consequently, if you are making any of the previous changes, you must specify a target DMCL whose name differs from that of the source DMCL. For more information about the DMCLs used by the REORG utility, see the source/target DMCLs in Initiating a REORG Operation (see page 259).

You cannot make changes to table definitions in an SQL-defined database between the setup and reload phases of a REORG operation.

Changing a non-SQL-defined database

You use REORG on a non-SQL-defined database to modify the database based on changes made to its schema or segment definitions. These changes must be reflected in the target subschema and DMCL. The modified DMCL and subschema must be available during the entire REORG operation, including the setup and unload phases.

You can make the following changes using REORG:

- Change area page ranges
- Change page size

  See EXPAND PAGE (see page 84) for an alternative means of changing a page size.
- Reassign records to different areas or page ranges

  Note: The record names must match in the source and target subschemas.
- 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

  See CONVERT PAGE (see page 71) for an alternative means of changing the maximum number of records per page.

- Add fields to an existing index key provided the fields already exist
- Remove fields from an index
- Change an index from unsorted to sorted provided the key fields already exist

  Note: Duplicate key order is random.
- Change an index from sorted to unsorted

Changes that cannot be implemented using REORG

The REORG utility cannot be used against native VSAM files.

During REORG processing, record formats and set relationships are preserved. This limits the modifications that you can make during a REORG operation. For example, you cannot use a REORG operation to:

- Remove a set
- Remove a record type
- Insert new data fields into a record
- Remove fields from a record
- Change the order of a sorted chained set
- Connect records to new chained or user-owned index sets
- Change the size, location, or data type of sort or index keys
- Change a chained set from unordered to ordered

These types of changes require other utilities, such as RESTRUCTURE, and possibly user-written programs.

Initiating a REORG Operation

Specifying setup options

A reorganization is initiated when a REORG statement is executed that specifies *setup-options*. The *setup-options* describe the segment to reorganize and the processing options to use. The first phase of the reorganization is called setup. During this phase, the parameters and the structure of both the source and target databases are analyzed to determine the tasks to perform and the work files that are needed. The result of this analysis is then written to the reorganization control file (RORGCTL). All subsequent reorganization jobs read the control file to determine what to do and how to do it.

Important! Be sure that setup-options are specified only when you intend to initiate a new REORG operation. Their presence causes the RORGCTL file to be overwritten, which results in the failure of any in-progress REORG operation using that control file.

The source/target subschemas

The REORG utility uses information in the dictionary when processing an SQL-defined database. When processing a non-SQL-defined database, the description of the data is contained in the source and target subschemas.

The source subschema describes the data to be unloaded and 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 that can participate in the defined sets

The target subschema describes the data to be reloaded and must:

- Define the same record names and sets as the source subschema
- Include the areas into which data is 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

The source subschema must be available during the setup and unload phases. The target subschema must be available during the setup, unload, and reload phases.

The source segment and the target database

The source segment, which is the segment named in the SEGMENT clause of the REORG statement, must include all areas being reorganized. If these areas are physically connected to other areas, either:

- These other areas must be part of the source segment, or
- These areas must be part of segments that are included in a DBNAME whose name is the same as that of the source segment

The target database, which is the database named in the INTO clause of the REORG statement, must include all areas into which data is to be reloaded as part of the REORG operation. If these areas are in different segments or are physically connected to areas in other segments, the target database must be a DBNAME that includes all of these segments.

Note: For more information about inter-area considerations, see Cross-area dependencies and Cross-segment dependencies in Special Database Considerations (see page 264).

The source/target DMCLs

The source DMCL must include the source segment and all segments with which the source areas are physically linked. The target DMCL must include the target segments and all segments with which the target areas are physically linked. Both the source and target DMCLs must be available during the setup and unload phases of the REORG operation. The target DMCL must also be available during the reload and cleanup phases.

The source DMCL is the DMCL specified in the SYSIDMS file. If no DMCL is specified in the SYSIDMS file, the DMCL named IDMSDMCL is used.

The target DMCL is the DMCL named in the INTO clause of the REORG statement. If no DMCL is specified in the INTO clause, the source DMCL is used as the target. If the same DMCL is used as both source and target, then either this DMCL must include both the source and target segments or no segment-related changes can be made to the data being reorganized.

Controlling REORG Execution

Specifying a stop point

By default, when a reorganization is initiated, it executes to completion or until a system failure or an unrecoverable error is encountered. However, you can break up execution into major processing phases by specifying a STOP AFTER parameter. You might want to halt processing after setup to examine the generated report to determine how much disk space is needed. Or, you might halt execution after the unload phase to delete the source database files before allocating the target files. Once a stop point has been established, which always occurs during setup, it remains in effect until another REORG statement is executed that specifies a different STOP AFTER value. The specified stop point must be the same as or later than the reorganization phase most recently processed.

Automatic job submission

The REORG utility relies on concurrently executing jobs for its parallel processing. To facilitate this, REORG has the ability to automatically submit jobs through the internal reader. It does this at the end of the setup phase unless STOP AFTER SETUP has been specified. It also does this anytime a REORG statement is executed that specifies SUBMIT. Typically, the SUBMIT keyword is used only when restarting a reorganization and no other jobs are active, such as when execution has been interrupted because a stop point has been reached or an error such as a system failure has caused all jobs to fail. If specified at other times, it may result in more jobs than there is work to perform, therefore needlessly tying up initiators.

JCL for jobs submitted automatically is retrieved from the RORGJCL file. The JCL should have the following characteristics:

- The prefix of the included job name should be short enough to allow for appending one or more characters to make the job name unique.
- The JCL should include a step that executes a REORG statement through the batch command facility. No parameters should be specified on the REORG statement.
- The RORGJCL file should not be included.
- The JCL should have the same considerations as listed in JCL Considerations (see page 282).

Important! The JCL or the DMCL(s) being used should either allow simultaneous access to both the source and target files or you must stop execution between the unload and reload phases and may need to change the RORGJCL contents so that instead of referencing the source files, it references the target files. Failure to do this can result in corruption of the source database as REORG may attempt to reload into the source files.

Note: For more information about the JCL needed to execute a REORG statement, see the JCL Considerations (see page 282).

REORG submits one job for each slice and each index group up to two times the number of slices. If there are two slices and one index group, three jobs are submitted. If there are two slices and three index groups, four jobs are submitted.

Manual job submission

Once a REORG operation has been initiated, you can manually submit as many REORG jobs as you want. Each job examines the control file to determine if there is any work to do on behalf of the reorganization. If no such work exists, perhaps because pre-requisite tasks have not yet completed, the job waits until some other job completes a task before re-examining the control file. If it finds work to do, it begins to perform that work provided no other job claims it first.

While manual job submission is possible, automatic submission should normally be used to avoid having either too few or too many jobs active at one time. Manual job submission can be used to resubmit one or more failed jobs.

Restarting REORG

If a REORG job fails, other REORG jobs try to automatically restart the task that failed after they have completed the task they were processing. If the problem was a temporary environment-related problem, and the task successfully completes, nothing additional needs to be done. Although, you may want to submit another REORG job to make up for the one that failed.

If the problem was not temporary, for example, a work file was too small; the remaining REORG jobs also fail when they try to restart the task, and at some point all jobs end. In this case, if the problem can be corrected, REORG can be restarted simply by resubmitting one or more REORG jobs. If the first one specifies the SUBMIT option, it submits the additional jobs.

REORG automatically restarts at the failing task or in some cases on a prior task if that is what is needed to recover from the failure. Depending on the requirements of the failing task, a portion of the database may be automatically formatted, or an index may be deleted as part of the restart process.

Restarting REORG from the beginning

Some problems may be severe enough to require restarting REORG from the beginning, such as if the wrong subschema was specified causing REORG to fail because the data did not match its description. Should it be necessary to restart REORG processing from the beginning, take the following actions:

- Format the target database if it had been partially loaded
- Delete any automatically created work files by executing a REORG statement that specifies CLEANUP
- Restart the REORG operation by executing a REORG statement that specifies *setup-options*, corrected if necessary.

Recovering from an out-of-space condition on a work file

You can recover from an out-of-space condition on a work file by taking the following actions:

- Manually delete the work file and reallocate it with more space
- Resubmit one or more REORG jobs. If all previously active jobs failed, submit only one job and specify the SUBMIT option to enable the automatic submission of the remainder of the jobs. If some jobs are still active, you can manually submit a job for each one that failed.

REORG serialization in a multi-image environment

Serialization is used to coordinate execution between jobs concurrently executing a REORG operation. These jobs serialize access to various internal resources by using a SYSTEMS ENQ with a QNAME of IDMSUT. In a multi-image environment, the SYSTEMS ENQ allows REORG jobs running on different images to serialize properly. However, if a multi-image manager is used, a SYSTEMS ENQ may be converted to a SYSTEM ENQ, which would only serialize REORG jobs running on the same image. In this environment, either restrict REORG jobs to run on one image or configure the multi-image manager to exempt REORG ENQ requests from being converted.

Special Database Considerations

Reorganizing a dictionary

When reorganizing a dictionary, the source and target subschemas are always the same and are determined by the areas of the dictionary being reorganized.

- Use the IDMSNWKU subschema for all areas, except DDLCAT, DDLCATX, and DDLCATLOD.
- Use the IDMSCATZ subschema for the DDLCAT and DDLCATX areas.
- Use the IDMSCATL subschema for the DDLCATLOD area.

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.

REORG an ASF database

If you want to run the REORG utility against an ASF data or definition area, see the *CA IDMS ASF User Guide* for more information. The *CA IDMS ASF User Guide* refers specifically to the UNLOAD and RELOAD utilities for unloading and reloading an ASF database, but REORG may be substituted for UNLOAD/RELOAD in this case.

Logically deleted records

You must remove any logically deleted records from the areas being unloaded before executing the REORG utility. To determine if an area has logically deleted records, use PRINT SPACE with the FULL option. If logically deleted records exist, use the CLEANUP utility to remove them from the area.

SQL-defined databases and synchronization stamps

Area and table synchronization stamps in SQL-defined segments are unloaded and reloaded by the REORG utility. For this reason, you must format the target areas using the FILE option so that the format operation does not also store synchronization stamps.

The one exception to this is when unloading and reloading the DDLCATLOD area in a segment defined for SQL. In this case, use the AREA option of the FORMAT statement or use the INSTALL STAMPS utility so that the format operation stores synchronization stamps.

Cross area dependencies

Areas that have physical connections to the areas being reorganized are referred to as **dependent areas**. You can reorganize an area that has physical connections to other areas without also unloading and reloading its dependent areas. Physical connections include:

- Sets that cross area boundaries
- Indexes that cross area boundaries
- Linked constraints in which the referenced and referencing tables reside in different areas

The REORG utility keeps track of the inter-area linkages and rebuilds them during the RELOAD phase. For this reason, not only must the target areas be updatable by REORG, but their dependent areas must also be updatable.

Cross segment dependencies

A source area can have physical connections to an area in another segment. To reorganize all related areas, the DBNAME setup option must be used, and it must include all segments with physical connections. When the SEGMENT setup option is used, only areas in the name segment can be reorganized. Areas in connected segments will be processed as dependent areas. Their pointers will be adjusted, but they will not be reorganized. To use the SEGMENT option, the segment named on the setup clause of the REORG statement must also be the name of a DBNAME that includes both the source segment and all segments whose areas are physically connected to the areas being reorganized.

A target area can have physical connections to an area in another segment. To reorganize into such an area, the INTO clause of the REORG statement must identify a DBNAME whose segments include all of the target areas and all of their dependent areas.

Mixed page groups

Mixed page groups are supported. A source DBNAME may include segments with different *page-group* and different maximum *records-per-page* values. A segment may have physical connections to segments with a different *page-group* and/or maximum *records-per-page* value. Cross segment chained sets must have the same *page-group* and maximum *records-per-page*.

Ignoring the order of CALC duplicates

By default, when REORG unloads a CALC record for which duplicates are allowed, it preserves the order of duplicate entries by walking the CALC set in a prior direction and determining the relative position of the record with respect to others with the same CALC key value.

This processing can result in significant overhead. If preserving the order of duplicate entries is not important, you can eliminate this processing by turning on the OPT00093 bit in the RHDCOPTF module.

Processing of DIRECT records

Because DIRECT records have been placed on a specific page by a user, the REORG utility may have difficulty in determining where they should be placed in the target area. If a DIRECT record's old page exists in its target page range, REORG attempts to place the occurrence on the same page. However, if the old page does not exist in the target page range, the record is stored in the target area proportionally to its position in the source area.

If this is not acceptable, you may need to write a user-written program to unload the database and the FASTLOAD utility to reload it.

DCMT VARY PERMANENT considerations

If you run the REORG utility to change an area's low page number and Change Tracking is not used, it is recommended that you remove any permanent status on the affected area before making the new DMCL active within the CV.

When Change Tracking is not used, the PERMANENT feature is implemented by carrying the area's low page number in the journals across cycles of the CV. Changing an area's low-page prohibits future cycles of the CV from properly identifying the area once the new page range is implemented.

If a DCMT VARY SEGMENT/AREA PERMANENT command is still in effect when the new page range is implemented, the area's usage-mode at startup is determined by the value specified in the DMCL. The entry in the journals for the old area's page range remains until the next format of the journals.

The journal entry for the old starting page can be removed without formatting the journal by doing a non-permanent DCMT VARY AREA command against the area prior to changing the DMCL definition in the CV.

Work Files

The REORG process requires many work files. These files can be dynamically created and allocated for you by the REORG utility, or you can manually assume these responsibilities. This section discusses considerations associated with the creation and use of work files.

Work files and the n WAYS parameter

The number of work files required by REORG can grow exponentially as the value in the DIVIDE PROCESSING n WAYS parameter is increased. Therefore, you should exercise care in choosing this parameter.

For example, a REORG that specifies DIVIDE PROCESSING 2 WAYS usually results in 2 unload slices, 2 reload slices, and 4 SYS002 work files. If 3 WAYS were specified, this usually results in 3 unload slices, 3 reload slices, and 9 SYS002 work files. A similar expansion exists for each RELOAD phase.

REBUILD phases use fewer work files, but the number of index work files generated by RELOAD1 increases with the number of reload slices.

Reducing the number of work files

There are two ways to reduce the number of work files needed by the REORG utility other than reducing the n-WAYS parameter. You can direct REORG to:

- Reuse work files, which allows work files that are no longer needed to be reused for other purposes during later phases of processing
- Execute as a sort exit, which eliminates the use of sorted output files

It is recommended that both of these options be used to reduce the number of work files, and the amount of disk space needed.

Work file creation

REORG can create work files dynamically or you can manually create them prior to beginning the unload and/or reload phases of REORG execution. Regardless of how the files are created, it is a good idea to halt execution after setup to determine what work files are needed by examining the report produced by REORG.

If using dynamic work file creation, you must specify the attributes of the work files using one or more CREATE DSMODEL statements. REORG creates the files either as directed by the CREATE WORKFILE clause or at the time they are first accessed. Dynamically created work files, other than DBKEYS files, are deleted automatically during the cleanup phase.

If you want to use REORG's size estimates to create a file, code a DSMODEL without a primary SPACE allocation. You can code a SPACE parameter with just a unit type (TRK, CYL, or blksize) and no value for primary allocation.

You must code a primary space allocation value or delay creating work files until estimates are available. This means, for example, that you cannot direct REORG to create RELOAD work files during the setup phase unless the DSMODEL contains a primary allocation value.

If you code a zero primary space allocation value and a non-zero secondary value, the secondary value is replaced by a value derived from the estimated primary value.

Note: For more information, see Considerations for running REORG on VSE.

Work file allocation

REORG can allocate work files dynamically or you can reference them manually by including an appropriate DD statement in the REORG job's JCL and in the JCL included in the RORGJCL file if automatic job submission is used.

If using dynamic work file allocation, you must specify the data set names of the work files using one or more CREATE DSMODEL statements. REORG allocates the files as they are needed. If a DD card for a given work file is included in the execution JCL, it is used, and that file is not dynamically allocated.

If not using dynamic work file allocation, you must include a DD statement for every work file in every REORG job and in the RORGJCL file.

Work file deletion

By default during the cleanup phase, REORG deletes all work files created during the current operation other than DBKEYS files. It deletes only those files for which a matching data set model was specified either at setup or when the file was created. If a matching model is detected, REORG attempts to dynamically allocate the file thereby determining its data set name which may be derived from the model or overridden by a DD statement in the JCL. Regardless of how the data set name is determined, if the file is created during the execution of a REORG statement, it will be deleted during the cleanup phase of the operation unless the file is subsequently overridden with a different data set name in some later job. You can determine which files cleanup will delete by examining the work file summary sections of the REORG Status Report.

Should it be necessary to restart a REORG operation from the beginning, you should first execute a REORG statement that specifies CLEANUP to delete any work files created by the interrupted operation. If you do not do this, none of the work files created during the first operation execution will be deleted by REORG even if they are reused during the second execution.

If REORG is restarted without first cleaning up the old work files, you can still direct REORG to delete them using one of the following methods:

- You can allow REORG to reuse the old files and then after REORG has ended normally, run a REORG CLEANUP job with the DELETEALL option. This option directs REORG to delete all work files, other than DBKEYS files, whether they were created by the most recent REORG operation or not
- You can delete the old work files during the setup phase of the restarted run. Specify DELETE OLD WORKFILES during setup and REORG checks for and deletes any files it finds that match the file names it plans to use. This includes DBKEYS files. This approach is preferable when the old files may be too small to be reused.

  Note: For more information, see Considerations for running REORG on VSE.

Using DSMODELs with REORG

Data set models (DSMODELs) are used to specify a set of attributes for data sets. REORG can use DSMODELs to specify the attributes for its work files, thereby eliminating the need for coding DD statements for every work file used by the utility.

The REORG utility checks for the presence of DSMODELs during the setup phase. If they are present, REORG saves the model information in the control file for later use. This saved information is used by subsequent tasks and by other jobs. No additional DSMODEL statements need to be specified in any submitted or subsequent job.

When REORG creates a work file, either because a CREATE WORKFILE clause has been specified or because a file does not exist at the time it is accessed, REORG uses the saved model information unless another CREATE DSMODEL statement has been specified in the same session as the one in which the file is being created. If another CREATE DSMODEL statement has been specified, it overrides the previously saved information. REORG updates the control file with the new information and uses it to create the work file.

To use a data set model's attributes for a given work file, the model's name must match the file's DDNAME according to the rules discussed in the CREATE DSMODEL statement. The REORG utility uses the following DDNAME prefixes:

- WU—Unload/Reload work files
- WI—Index REBUILD work files
- WS—Sort output work files
- WD—DBKEYS work files

Each prefix is followed by a generated number. To use a model for a work file, the model name must match the file's DDNAME. If more than one DSMODEL matches that of a work file, the one that most closely matches is used. For information about how to specify model names that match character strings such as a DDNAME, see CREATE DSMODEL (see page 81).

For the REORG utility to dynamically allocate a work file, its matching model must minimally specify a DSN attribute so that the name of the data set can be determined. Symbolic parameters in the DSN value can be used to generate different names for each work file.

For the REORG utility to create a work file, the DSMODEL must also include information to identify where to create the file, the space to allocate, and a block size to use.

When specifying a DSMODEL for a DBKEYS file, the BLKSIZE must be a multiple of 16. All other work files are variable length. The BLKSIZE for these files must be four bytes larger than the largest work record.

If a block size is not coded or BLKSIZE 0 is coded in the DSMODEL for a REORG file, REORG chooses one based on the device type. If the device type can not be determined, a 3390 will be assumed. Normally, 1/2 track blocking is used for a 3390 device.

Sizing work files

The simplest way to size work files is to let REORG do it for you. REORG automatically estimates the size of all files after making a pass of the database and gathering statistics. This occurs by default when the data is unloaded. While this does not require an extra pass of the data, the estimates that are generated can only be used for allocating RELOAD work files because the UNLOAD files have already been created and used.

To use REORG-generated estimates for allocating UNLOAD work files, use the ESTIMATE WORKFILE SIZES option. This directs REORG to make a preliminary pass of the data without opening or writing to any work files. The generated file estimates are stored in the control file and can be used to allocate both UNLOAD and RELOAD work files when the REORG operation is resumed.

For REORG to more accurately estimate the size of overflow files, it may be necessary to specify an OVERFLOW PERCENT parameter. REORG assumes that 10% of the records to be stored in the database will overflow. If this assumption is not valid for a particular database, you may need to specify a different overflow percent value so that REORG can generate better estimates for the size of overflow files.

While REORG attempts to accurately estimate the size needed for work files, it may not always be able to do so. In certain cases, it may be necessary to manually estimate the size needed for one or more work files.

Estimating the size needed for work files is difficult for two reasons: many classes of files contain different types of records, and the number of records written to each file within a class varies depending on how REORG chooses to divide the page ranges into slices. Consequently, there are no simple formulas that can be used to estimate work file sizes. There are however, some techniques that can be used to facilitate file sizing and allocation.

In planning for a reorganization, do one or more trial runs to determine the actual file sizes needed for a given n-WAY value. This can be done during a non-critical time against a copy of the source database. This is the easiest way to obtain accurate size estimates. If it is impractical to do a trial reorganization on a full copy of the database, do it on a reasonably-sized sample that is representative of the original and then scale up the work file sizes proportionately. Be sure that the sample database is large enough relative to the n-WAY parameter so that the slicing algorithm does not reduce the number of slices due to their small size. Size estimates determined using a sample database will not be as accurate as those determined using a full copy of the database and so should be increased to account for this.

The following is a list of additional techniques that may prove helpful in allocating work files under z/OS:

- Specify a relatively modest primary allocation and a large secondary allocation. This allows the file to extend to handle large amounts of information without wasting unneeded space.
- Use IBM's Storage Management Subsystem (SMS) to determine where files are to be allocated rather than trying to manually place files on specific volumes. SMS can automatically spread the files across volumes wherever there is available space. Use the DATACLAS and STORCLAS options to allow files to extend across multiple volumes.
- If SMS is not available, allow the files to be allocated across multiple volumes by specifying multiple volume serial numbers. There must be enough space across all listed volumes to satisfy the space needs of all of the work files. The order of the entries in the list is not important.
- Consider using extended format data sets for large work files. Although this adds 32 bytes of overhead to each block, extended format data sets can have more extents across more volumes than a basic format data set.

Under z/VSE it is recommended to not specify a primary space value, and let REORG calculate one. If this value is not large enough, or will not fit on the specified volume, a DLBL and EXTENT for the individual file should be manually coded to override the generated label.

Work file restrictions

You cannot use tape or virtual tape for work files.

Note: For more information about the types of work files used by the REORG utility, see the Classes of work files in REORG Processing Details (see page 272).

REORG Processing Details

How the unload phase works

During the unload phase, most record types are unloaded as part of an area sweep. Certain database structures necessitate exceptions to this general processing flow.

- A record that is stored VIA a System Index in which sequence order cannot be determined from the member record (such as an unsorted index or one in which duplicates are FIRST or LAST), is offloaded by walking its index.
- Members of a VIA set are unloaded at the time their owner is unloaded by walking the VIA set. If the VIA member in turn owns VIA members, they too are unloaded in a similar fashion.
- 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 its duplicates.

Records that are unloaded are assigned a target page number during the unload phase. Target page numbers are assigned as follows:

- CALC records are assigned a target page based on their CALC key and target page range
- VIA records that are members of their VIA set are assigned a target page based on their owner's assigned or target page. If a VIA record is in the same target page range as its owner, it is assigned the same target page plus any VIA displacement. If in a different page range, it is assigned a target page in its range proportional to its owner's page in its assigned or target page range plus any VIA displacement.
- VIA records that are not members of their VIA set are assigned a target page that is proportional to their position in their source page range.
- DIRECT records are assigned a target page that is the same as their source page if the page is in both the source and target page ranges; otherwise, DIRECT records are assigned a target page that is proportional to their position in their source page range.

REORG slicing

A database is divided into slices based upon the value coded in the DIVIDE PROCESSING n WAYS parameter. Generally, each area is divided symmetrically, but if an area contains a subarea and no outside records span that subarea, REORG divides the subarea independently of the rest of the area.

If dividing an area or subarea produces a piece that is too small, smaller than 100 pages, that area or subarea is divided again into fewer, but larger pieces. If an area or subarea is less than 200 pages, it is not divided at all.

After all areas and subareas have been divided into pieces, each piece is assigned to a slice. Each slice typically ends up with approximately the same number of pages.

Usually, the number of slices equals the n-WAYS value, but if the total number of pages being processed is small, there may be fewer slices. If the total size of all areas is less than 200 pages, all pages are assigned to one slice, regardless of the n-WAYS value.

Dependent areas are areas with physical linkages to the areas being processed. They are also divided and assigned to slices.

Areas and subareas that contain only system indexes are not divided and are not assigned to a slice.

The source and target database are sliced independently of one another and consequently may not be divided into the same number of slices.

Index groups

A user-owned index occurrence is processed with the slice to which its owner record is assigned. System-owned indexes are assigned to groups and processed independently of slices. System indexes that have the same or overlapping target page ranges are assigned to the same group; otherwise, they are assigned to different groups. If an index has a distinct (non-overlapping) target page range, it is assigned its own index group. Index groups are processed independently of each other and therefore can be processed in parallel. Once assigned to a group, an index is in that group for the duration of REORG processing.

REORG tasks and phases

REORG processing is divided into tasks and grouped into phases. Each phase processes a type of work needed to unload or reload a database or rebuild an index. Each task processes a slice or index group within a phase.

For example, if a database were divided into two unload slices and had three index groups, the unload phase could have up to five tasks: one for each slice and possibly one for each index group. All unload tasks must successfully complete before REORG can begin to process tasks in the RELOAD or REBUILD phases.

RELOAD contains six phases numbered 1 through 6. These phases reload slices, rebuild user indexes, and reconnect pointers. Each reload task in a given phase must successfully complete before processing can move to the next RELOAD phase.

REBUILD can contain up to three phases numbered 1 through 3. Each index group has one task per phase, but each index group can run independently of the others and independently of RELOAD phases 3 through 5.

If a system index related page range conflict exists between tasks in RELOAD and REBUILD, or between REBUILD tasks, updates of the page range are serialized.

The phases are:

Phase	Description
SETUP	Analyzes syntax, subschemas, and DMCLs to initialize the REORG process. It decides how to divide the source and target databases for parallel processing. It creates tasks and decides what work files are needed to support those tasks. It builds the RORGCTL file.
UNLOAD	Consists of Unload Slice and Unload Index tasks. Slices are offloaded through area sweeps and by walking VIA sets. If an index needs its sequence preserved, the index is walked to determine the relative sequence number of each member. The sequence data is merged with information extracted from the loaded record in REBUILD2. If VIA System Indexes exist and their sequence needs to be preserved, the member records are offloaded by walking the index. VIA members are offloaded when their owners are offloaded regardless of how the owners are offloaded.
REBUILD1	This phase exists only when there is a VIA SYSTEM INDEX that spans multiple reload slices. If needed, it runs prior to RELOAD1. It sorts VIA INDEX member records into index order and assigns the records to target slices.
RELOAD1	Loads data into the target database. Set connection and index key data is generated and processed by later phases. If a task in this phase fails, when restarted, it formats only the page ranges assigned to its slice and restarts processing from the beginning.
RELOAD2	This is an overflow phase. If a reload slice fills up, the data is saved and processed by this phase. There is only one task in this phase, and it updates across slices. It generates the same set connection and index key data as RELOAD1 for records it stores. If an area fills up during this phase, processing stops. If the task in this phase fails, when restarted, REORG starts over by reprocessing all RELOAD1 tasks. Each formats its slice and restarts.
RELOAD3	Merges user-owned index data generated by UNLOAD, RELOAD1, and RELOAD2. Member keys and/or sequence numbers are merged in owner sequence and generated records are passed to RELOAD4.

Phase	Description
RELOAD4	Merges data produced by RELOAD1 and RELOAD2 about old and new db-keys for chained sets and passes it to RELOAD6. If data is passed from RELOAD3, it rebuilds user indexes using that data. INDEX and OWNER pointer data is generated and passed to RELOAD6. If a task in this phase fails, when restarted, it reprocesses data from the beginning. It erases and rebuilds all user index occurrences in its slice. If a target slice fills before all data is processed, unconnected data is passed to RELOAD5. Tasks in this phase serialize on system index related page ranges.
RELOAD5	This is an overflow phase for RELOAD4. Data that would not fit into a slice is processed here. There is only one task in this phase, and it updates pages across all slices. If an area fills up during this phase, processing stops. Index key connection begins with the key that failed in RELOAD4. If the task in this phase fails, when restarted, REORG reprocesses all tasks in RELOAD4. User indexes that were already built are rebuilt. Tasks in this phase serialize on system index related page ranges.
RELOAD6	Updates prefix pointers using data generated by RELOAD phases 4 and 5; and REBUILD phase 3.
REBUILD2	This phase merges sequence data from an UNLOAD INDEX task with member data from RELOAD1 and RELOAD2. It is only generated for groups that have indexes for which sequence must be preserved. It can run in parallel with all REBUILD phases 2 through 4 for other index groups and in parallel with RELOAD phases 3 through 6.
REBUILD3	This phase uses member data from RELOAD1, RELOAD2, and REBUILD2 to rebuild system indexes. Tasks in one group can run in parallel with reload tasks and index tasks from other groups as long as there are no page range conflicts. Tasks in this phase serialize on system index related page ranges.
CLEANUP	There are no tasks in this phase. When all other processing has successfully completed, this phase optionally unlocks the reload areas and deletes work files created by the REORG utility.

Classes of work files

Each phase of REORG processing can produce different classes of work files. A task can produce one or more occurrences of a given class of work file. Each work file is read by a task in a later phase along with similar work files from other tasks.

The following table describes each class of work file. When discussing how many work files are generated, sometimes a task's phase name is referenced and sometimes the number of reload or unload slices is referenced. For each unload slice, there is one UNLOAD SLICE task and one RELOAD3 task. For each reload slice, there is one RELOAD1, RELOAD4, and RELOAD6 task. The number of reload slices is the same as the number of RELOAD1 tasks. Sometimes these terms are interchanged when it is thought to clarify a description.

The classes of work files are:

Work File Classes	Description
SYS002	Contains offloaded record and set information. Output: UNLOAD SLICE and UNLOAD INDEX Input: RELOAD1 One is created for each unload slice times the number of reload slices. One is created for each index group that contains a VIA system index where all members target one reload slice.
SYSX02	Contains offloaded record and set information for members of a VIA index. Output: UNLOAD SLICE and UNLOAD INDEX Input: REBUILD1 One is created for each unload slice times the number of index groups that contain VIA system indexes that target multiple reload slices. One is created for each index group that contains VIA system indexes that target multiple reload slices
SYSIX2	Contains sorted record and set information for members of a VIA index. Output: REBUILD1 Input: RELOAD1 One is created for each index group that contains VIA system indexes that target multiple reload slices times the number of reload slices.

Work File Classes	Description
SYSOF2	Contains record and set information for records that overflowed their target slices. Output: RELOAD1 Input: RELOAD2 One is created for each reload slice or RELOAD1 task.
SYS003	Contains offloaded index sequence information for user indexes. Output: UNLOAD SLICE Input: RELOAD3 One is created for each unload slice.
SYSX03	Contains offloaded index sequence information for system indexes. Output: UNLOAD INDEX Input: REBUILD2 One is created for each index group that contains system indexes that are not VIA and where member sequence is not being preserved.
SYS004	Work file containing sorted record and set information. Not created when SORTEXIT is specified. One is created for each RELOAD1 and RELOAD2 task.
SYS005	Contains old and new db-key values for chained sets. Output: RELOAD1 and RELOAD2 Input: RELOAD4 One is created for each (RELOAD1 and RELOAD2 task) times the number of RELOAD4 tasks.
SYS006	Contains user index owner and member db-key and symbolic key information. Output: RELOAD1 and RELOAD2 Input: RELOAD3 One is created for each reload slice times the number of unload slices.
SYSX06	Contains system index member db-key and symbolic key information. Output: RELOAD1 and RELOAD6 Input: REBUILD2 One is created for each reload slice times the number of index groups that contained a non-VIA system index where sequence was being preserved.

Work File Classes	Description
SYS007	Work file containing sorted user index owner and member work record information. Not created when SORTEXIT is specified. One is created for each RELOAD3 task.
SYSX07	Work file containing sorted user index owner and member work record information. Not created when SORTEXIT is specified. One is created for each REBUILD2 task.
SYS008	Contains user index information that is used to rebuild indexes. Output: RELOAD3 Input: RELOAD4 One is created for each unload slice times the number of reload slices.
SYSX08	Contains system index information that is used to rebuild indexes. Output: RELOAD1, RELOAD2, and REBUILD2 Input: REBUILD3 One is created for each REBUILD2 task that is generated. Also, one is created for each index group (that contains a VIA system index or a non-VIA system index where sequence was not being preserved) times the number of RELOAD1 and RELOAD2 tasks.
SYS009	Work file containing sorted user index work records and sorted old/new db-keys for chained sets. Not created when SORTEXIT is specified. One is created for each RELOAD4 and RELOAD5 task.
SYSX09	Work file containing sorted system index work records. Not created when SORTEXIT is specified. One is created for each REBUILD3 task.
SYS010	Contains new db-key information for updating record prefixes. Output: RELOAD4, RELOAD5, and REBUILD3. Input: RELOAD6 One is created for each (RELOAD4 and RELOAD5) task times the number of RELOAD6 tasks, and one is created for each index group (that contains indexes where the member contains UP or OWNER pointers) times the number of RELOAD6 tasks.

Work File Classes	Description
SYS011	Work file containing sorted prefix pointer information for all records. Not created when SORTEXIT is specified. One is created for each RELOAD6 task.
DBKEYS	A file created when the DBKEYS option is specified. Output: RELOAD1 and RELOAD2 One is created for each RELOAD1 and RELOAD2 task. All should be concatenated together when processed.

Considerations for Running REORG on VSE

Work File Creation and Deletion

REORG on VSE does not support creating and deleting work files. It will create and delete user labels, but the work files will not be created, a VTOC entry will not be created, until the file is opened for output. When REORG processing is complete, work files must be manually deleted, or overwritten to reclaim the space. The options: DELETE OLD WORKFILES, CREATE WORKFILES, and DELETEALL,only apply to user labels for the current job. Automatic deletion of work files during the CLEANUP phase will only delete user labels.

CA DYNAM/D is required to create labels

The extent of a generated label will have a relative starting track of 1 for the number of tracks based onthe primary space value. These labels require CA DYNAM/D to convert them to an actual track address at open time.

If CA DYNAM/D is not installed, labels for work files must be coded manually. The recommended procedure in this case, is to use the ESTIMATE FILE SIZES option, which does not use any work files. Code the JCL statements based on the reported file sizes; and include them in all REORG jobs.

SYSIDMS

REORG requires the RORGCTL and RORGJCL files to be defined in SYSIDMS using FILENAME= parameters:

```
FILENAME=RORGCTL RECFM=F BLKSIZE=8192
FILENAME=RORGJCL RECFM=F BLKSIZE=NNNN LRECL=80
DLBLMOD=ON
```

Where NNNN is the block size of the JCL file and must be a multiple of 80.

The SYSIDMS DLBLMOD=ON option must be specified to allow for sequential and random processing of the RORGCTL file. Either a DA or SD label m may be used for RORGCTL.

DSMODEL

The only options which apply to z/VSE are: DSN; BLKSIZE; the allocation unit value and primary value for the SPACE option; and the first volume of the VOLSER Option. The rest can be coded, but is ignored.

REORG generates labels using track sizes. If CYL is coded for a space allocation unit, the primary value is converted to tracks, but there is no cylinder alignment. If a block size is coded for an allocation unit, the coded value is used to calculate the number of tracks required.The coded value must match the BLKSIZE value, otherwise the calculated number of tracks may not be accurate.

An example of a DSMODEL follows; note that the primary space value was not coded. This allows REORG to generate the value for each file, using file size estimates. If a primary space value had been coded, this value would be used for all files regardless of the estimated size.

```
CREATE DSMODEL W*
 DSN 'USERID.EMPDB.WORKFILE.&DD'.
BLKSIZE 4096
 SPACE TRK
 VOLSER IDMS05
;
```

RORGJCL

JCL submitted by REORG is read from the RORGJCL file. This must be a sequential file built on disk and contain all the JECL and JCL statements for the submitted job. However the JECL and some JCL statements cannot be directly copied to this file using normal JCL because POWER will try to interpret these statements when the copy job is run. These statements must be "hidden" from power by changing the first characters as follows:

- * $$ statements must be coded as $ $$ statements
- // JOB statements must be coded as #/ JOB statements
- /* coded as #*
- /& coded as #&
- Any statement starting with a "/" may be coded as starting with a "#"

This is the same method used by the IESINSRT program to hide JCL, which is documented in the *z/VSE Administration Guide*.

The JCL will get stored on disk in the format it is coded. REORG will convert the hiding characters back to their correct values prior to submitting the job to POWER. For example the JCL to copy a job to the RORGJCL file can look like this:

```
// DLBL RORGJCL,'USERID.EMPDB.RORGJCL',1,SD
// EXTENT SYS020,CULLD9,,,1,5
// ASSGN SYS020,DISK,VOL=CULLD9,SHR
// EXEC IDCAMS,SIZE=386K
   REPRO INFILE(SYSIPT) -
         OUTFILE(RORGJCL ENV( BLKSZ(4080) RECFM(FB) RECSZ(80) ) )
$ $$ JOB JNM=RORGJOB,CLASS=B,DISP=D
$ $$ LST CLASS=R,DEST=(,USERID),JSEP=0
$ $$ PUN CLASS=R,DEST=(,USERID)
#/ JOB RORGJOB
* JCL THAT REORG SUBMITS
#/ DLBL SYSIDMS,'#SYSIPT'
#/ EXEC IDMSBCF,SIZE=256K
ECHO=ON  JOURNAL=OFF  DLBLMOD=ON
DMCL=IDMSDMCL DBNAME=EMPDEMO
FILENAME=RORGCTL RECFM=F  BLKSIZE=8192
#*
REORG;
#*
#&
$ $$ EOJ
/*       EOF for REPRO
```

The statements starting with $ $$ JOB and ending with $ $$ EOJ are copied to RORGJCL and are submitted by REORG.

JCL Considerations

The REORG command must be submitted through the batch command facility. The JCL to execute the facility must include statements to define:

- The standard BCF JCL statements.
- A RORGCTL file. It is a fixed-length, LRECL 8192, unblocked BDAM file that is initialized during setup and read and updated by subsequent and parallel jobs.
- An optional RORGJCL file that references a card-image file containing JCL that is automatically submitted by REORG. This file must be fixed or fixed blocked with an LRECL of 80.
- The files containing the source areas and their dependent areas for the unload phase if dynamic database allocation is not being used to reference them through the source DMCL.
- The files containing the target areas for the RELOAD and REBUILD phases if dynamic database allocation is not being used to reference them through the target DMCL.
- The journal file(s), which can be dummied.
- The work files if dynamic work file allocation is not being used to reference them.

If performing parallel processing, do not specify any syntax that would cause the jobs to serialize.

- Do not specify DISP=OLD or DISP=NEW for database or work files. Specify DISP=SHR instead. If a file is new and must be created, create it prior to submitting any REORG jobs.
- DISP=OLD or DISP=NEW is acceptable for a file that is unique to a job. For example SYSLST, SYSOUT, or SYSJRNL are unique to a job. Sort files such as SORTWK01 and SORTMSG are also unique to a given job.

If you define the same DDNAMEs for the source and target database files, the database cannot be unloaded and reloaded in the same job. You must halt processing after the unload phase and then restart processing ensuring that the target files and not the source files are being referenced. Follow these general steps:

- Submit JCL for the unload phase. The source database files must be referenced in the REORG JCL and in JCL submitted from RORGJCL, unless using dynamic database file allocation.
- Use the STOP AFTER UNLOAD option to halt the REORG process after the unload phase.
- After all unload jobs have ended and the unload phase has completed, format the target database files if they have not already been formatted.

- Submit JCL for the reload phases. The submitted job and JCL submitted from RORGCTL must reference the target database files, unless using dynamic database file allocation.
- The submitted job must say STOP AFTER CLEANUP or STOP AFTER RELOAD to allow tasks in the reload phases to run.

Work files can be manually or dynamically allocated. Considering the number of work files needed, the use of dynamic work file allocation is recommended. A CREATE DSMODEL must be coded in the SETUP job, or in the job that allocates the files to use dynamic allocation.

Note: When using CREATE DSMODEL statements, it is recommended that you force the batch command facility to terminate if an error is encountered. You do this by specifying a SET OPTIONS statement with the ON ERROR END clause. By forcing a termination, it ensures that REORG will not execute with incomplete data set models due to an error in a CREATE DSMODEL statement.

When manually allocating work files, you should run a preliminary REORG with the STOP AFTER SETUP option. This produces a report of all required work files. You must allocate these files prior to restarting the REORG process and include the necessary DD statements with DISP=SHR in the JCL that is submitted to restart processing and in the JCL in the RORGJCL file.

The RORGCTL file should be created prior to submitting the first REORG job, unless STOP AFTER SETUP is specified. REORG jobs running in parallel must specify DISP=SHR on this file, or they will serialize with each other on the DSNAME. If processing is stopped after SETUP, the SETUP job can create the file, but subsequent jobs must specify DISP=SHR.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

Example 1

The following example reorganizes the EMPDEMO database. To speed execution, the data is divided into three slices that can be processed in parallel. All work files are dynamically allocated because data set model information is present. Because no STOP AFTER parameter is specified, the reorganization runs to completion.

```
Create dsmodel w* …
Reorg segment empdemo using empss01
reload into empdemo dmcl newdmcl
divide processing 3 ways
reuse workfile;
```

Example 2

The following example separates the setup phase from the later phases. After executing setup multiple times, database reorganization is begun using the second REORG statement. Again, work files are allocated dynamically. The SUBMIT directive submits multiple jobs to to process the work in parallel.

```
Create dsmodel w* …
Reorg segment empdemo using empss01
reload into empdemo dmcl newdmcl
divide processing 3 ways
reuse workfiles
stop after setup;

Reorg submit stop after cleanup;
```

Example 3

The following example restarts a reorganization that was interrupted by a system failure. It too submits multiple jobs to process the work.

```
Reorg submit;
```

Example 4

The following example shows an extra DSMODEL being specified for a DBKEYS file. The DBKEYS file DDNAME starts with WD so it matches on the WD* DSMODEL. The SPACE and BLKSIZE attributes are taken from this model and remaining attributes are taken from the W* DSMODEL because of the FROM syntax. The remaining REORG work files match on the W* DSMODEL and take attributes directly from it.

```
CREATE DSMODEL W*
DSN 'IDMS.REORG.WORKFILE.&DD'.
UNIT SYSDA
SPACE TRK(5,5)
BLKSIZE 12004
;
CREATE DSMODEL WD*
FROM W*
SPACE TRK(2,2)
BLKSIZE
16000
;
REORG SEGMENT REOGSMAL USING REOGSUB
RELOAD INTO REOGSMAL
DIVIDE PROCESSING 4 WAYS
 STOP AFTER UNLOAD
 CREATE ALL WORKFILES
 GENERATE DBKEYS
NOTIFY 10
;
```

Example 5

The following example shows the syntax required for a job submitted through RORGJCL. DSMODEL and other options are not coded because they are taken from the control file and are not needed by this job. A job with this syntax can also be submitted manually, and it also uses existing options from the control file and looks for tasks to process.

```
REORG;
```

Sample Output

When a REORG job completes, it prints a status report of the current REORG process. This report can also be produced using the REORG STATUS REPORT ONLY option. The report is made up of several sections, which contains an analysis of the current REORG operation and a status of the REORG tasks.

REORG Status Report - Section 1

The first section of the REORG Status Report identifies an overall status of the REORG operation.

```
 ************************************************************************
 *                                                                      *
 *                 REORG Status Report                                  *
 *     Identifying time stamp: 2005-11-17-13.44.06.388141      *
 *  Unload subschema=EMPTSS01  Unload segment=EMPDEMO   Unload DMCL=EMPTDMCL  *
 *  Reload subschema=EMPTSS01  Reload segment=EMPBDEMO  Reload DMCL=EMPBDMCL  *
 *                                                                      *
 ************************************************************************
                   Options in effect
                   -----------------
Divide processing 3 ways    Notify interval=1    Job submission=YES  SHARE=YES
STOP AFTER CLEANUP  CREATE ALL WORKFILES     reuse workfiles=NO   SORTEXIT=NO
Overflow Percentage=10       Concurrent jobs=5       GENERATE DBKEYS FILES=YES
                    Current status
                    --------------
SETUP=completed    UNLOAD=completed    RELOAD=completed    CLEANUP=completed
     total tasks=23     tasks completed=23     reload areas are not locked
```

REORG Status Report - Section 2

The next section identifies all the records and sets being processed by the REORG operation. The sequence numbers are assigned by REORG and are used to identify RECORDS and SETS in REORG work files. They have no relation to record or index IDs that are assigned to database records.

System indexes are assigned to groups. All indexes in the same group are offloaded and rebuilt together. Indexes are assigned to the same group when they share a common page range.

```
                Records and Sets
                ----------------

      Set=EMP-NAME-NDX                Sequence=1      Index group=1
      Set=JOB-TITLE-NDX               Sequence=2      Index group=2
      Set=SKILL-NAME-NDX              Sequence=3      Index group=2
      Set=COVERAGE-CLAIMS             Sequence=4
      Set=DEPT-EMPLOYEE               Sequence=5
      Set=EMP-COVERAGE                Sequence=6
      Set=EMP-EXPERTISE               Sequence=7
      Set=EMP-EMPOSITION              Sequence=8
      Set=MANAGES                     Sequence=9
      Set=REPORTS-TO                  Sequence=10
      Set=JOB-EMPOSITION              Sequence=11
      Set=OFFICE-EMPLOYEE             Sequence=12
      Set=SKILL-EXPERTISE             Sequence=13
   Record=COVERAGE                    Sequence=1
   Record=DENTAL-CLAIM                Sequence=2
   Record=DEPARTMENT                  Sequence=3
   Record=EMPLOYEE                    Sequence=4
   Record=EMPOSITION                  Sequence=5
   Record=EXPERTISE                   Sequence=6
   Record=HOSPITAL-CLAIM              Sequence=7
   Record=INSURANCE-PLAN              Sequence=8
   Record=JOB                         Sequence=9
   Record=NON-HOSP-CLAIM              Sequence=10
   Record=OFFICE                      Sequence=11
   Record=SKILL                       Sequence=12
   Record=STRUCTURE                   Sequence=13
```

REORG Status Report - Section 3

The next section of the report identifies unique page ranges within the source and target databases. All records in the same page range share one or more pages and do not overlap with records outside the page range. These ranges are used by REORG setup to determine how to slice up a database.

Page ranges that contain only indexes are also identified.

```
                  Areas and Slices
                  ----------------

Unload area=EMPDEMO.EMP-DEMO-REGION      low page=175001    high page=175400
               Records only area subrange  low page=175021    high page=175400
                 Index only area subrange  low page=175002    high page=175017
Unload area=EMPDEMO.INS-DEMO-REGION      low page=175401    high page=175600
               Records only area subrange  low page=175421    high page=175600
               Records only area subrange  low page=175402    high page=175419
Unload area=EMPDEMO.ORG-DEMO-REGION      low page=175601    high page=175800
               Records only area subrange  low page=175621    high page=175800
                 Index only area subrange  low page=175602    high page=175619

Reload area=EMPBDEMO.EMP-DEMO-REGION     low page=275001    high page=275800
               Records only area subrange  low page=275041    high page=275800
                 Index only area subrange  low page=275002    high page=275033
Reload area=EMPBDEMO.INS-DEMO-REGION     low page=276001    high page=276400
               Records only area subrange  low page=276041    high page=276400
               Records only area subrange  low page=276002    high page=276037
Reload area=EMPBDEMO.ORG-DEMO-REGION     low page=277001    high page=277400
               Records only area subrange  low page=277041    high page=277400
                 Index only area subrange  low page=277002    high page=277037
                 Index only area subrange  low page=277002    high page=277037
```

REORG Status Report - Section 4

The next section reports the portions of each page range assigned to each slice.

```
      Unload slice=1
          Unload slice range    low page=175021    high page=175147
          Unload slice range    low page=175421    high page=175520
          Unload slice range    low page=175402    high page=175419
          Unload slice range    low page=175621    high page=175720
      Unload slice=2
          Unload slice range    low page=175148    high page=175274
          Unload slice range    low page=175521    high page=175600
          Unload slice range    low page=175721    high page=175800
      Unload slice=3
          Unload slice range    low page=175275    high page=175400

      Reload slice=1
          Reload slice range    low page=275041    high page=275294
          Reload slice range    low page=276281    high page=276400
          Reload slice range    low page=277161    high page=277280
      Reload slice=2
          Reload slice range    low page=275295    high page=275547
          Reload slice range    low page=276041    high page=276160
          Reload slice range    low page=276002    high page=276037
          Reload slice range    low page=277281    high page=277400
      Reload slice=3
          Reload slice range    low page=275548    high page=275800
          Reload slice range    low page=276161    high page=276280
          Reload slice range    low page=277041    high page=277160
```

REORG Status Report - Section 5

The next section describes the status of each task and the work files assigned to the task. The data set name is displayed only if dynamic work file allocation is used.

```
                Tasks and Work File Usage
                -------------------------

Task#=1       Task=UNLOAD               type=S  slice=1     status=completed

 SYS002  O/P   DDNAME=WU00001   DSN=USERA01.EMPDEMO.WORKFILE.WU00001
 SYS002  O/P   DDNAME=WU00002   DSN=USERA01.EMPDEMO.WORKFILE.WU00002
 SYS002  O/P   DDNAME=WU00003   DSN=USERA01.EMPDEMO.WORKFILE.WU00003
 SYS003  O/P   DDNAME=WU00010   DSN=USERA01.EMPDEMO.WORKFILE.WU00010

Task#=2       Task=UNLOAD               type=S  slice=2     status=completed

 SYS002  O/P   DDNAME=WU00004   DSN=USERA01.EMPDEMO.WORKFILE.WU00004
 SYS002  O/P   DDNAME=WU00005   DSN=USERA01.EMPDEMO.WORKFILE.WU00005
 SYS002  O/P   DDNAME=WU00006   DSN=USERA01.EMPDEMO.WORKFILE.WU00006
 SYS003  O/P   DDNAME=WU00011   DSN=USERA01.EMPDEMO.WORKFILE.WU00011

Task#=3       Task=UNLOAD               type=S  slice=3     status=completed

 SYS002  O/P   DDNAME=WU00007   DSN=USERA01.EMPDEMO.WORKFILE.WU00007
 SYS002  O/P   DDNAME=WU00008   DSN=USERA01.EMPDEMO.WORKFILE.WU00008
 SYS002  O/P   DDNAME=WU00009   DSN=USERA01.EMPDEMO.WORKFILE.WU00009
 SYS003  O/P   DDNAME=WU00012   DSN=USERA01.EMPDEMO.WORKFILE.WU00012
```

REORG Status Report - Section 6

The next section shows the work file summary. This example shows the general-purpose files used during the unload and reload phases. Other sections show index files, sorted data files, and DBKEYS files.

All work file summary reports show the estimated size of each file and the attributes used in determining those sizes. BPT is Blocks Per Track; TPC is Tracks Per Cylinder; and *3390* indicates that attributes were based on a generic 3390 device, as opposed to a specific volume.

An asterisk (*) following a DDNAME indicates that the file was created by a REORG job executing as part of the current operation and therefore will be deleted automatically during cleanup.

```
                   Unload/Reload work file summary
                   -------------------------------
DDname    DSN                                            Estimated Size
------    ---                                            --------------
WU00001 * USERA01.EMPDEMO.WORKFILE.WU00001                        1 Trk
          Estimate based on: VOLSER=*3390*  BLKSIZE=27998  BPT=02  TPC=15
          UNIT=SYSDA   SPACE=Trk  PRI=1  SEC=1
WU00002 * USERA01.EMPDEMO.WORKFILE.WU00002                        1 Trk
          Estimate based on: VOLSER=*3390*  BLKSIZE=27998  BPT=02  TPC=15
          UNIT=SYSDA   SPACE=Trk  PRI=1  SEC=1
&vellip.
WU00039 * USERA01.EMPDEMO.WORKFILE.WU00039                        1 Trk
          Estimate based on: VOLSER=*3390*  BLKSIZE=27998  BPT=02  TPC=15
          UNIT=SYSDA   SPACE=Trk  PRI=1  SEC=1
Total primary space:          0 Cyl         39 Trk          0 Blk
```

REORG Status Report - Section 7

The next section is a recap of tasks that have completed in task order. It shows when the task was started, when it was completed, and the job that processed the task.

```
 ************************************************************************
 *                                                                      *
 *                 Recap of TASK activity                               *
 *                                                                      *
 ************************************************************************

Task#=1       Task=UNLOAD               type=S  slice=1     status=completed
      jobname=USERA01Y  jobid=JOB01048
 UT019000 Task 1 UNLOAD   slice 1 starting at 2005-11-17-13.44.35.255392
 UT019003 Task 1 UNLOAD   slice 1 completed at 2005-11-17-13.44.35.884474

Task#=2       Task=UNLOAD               type=S  slice=2     status=completed
      jobname=USERAXXA  jobid=JOB05591
 UT019000 Task 2 UNLOAD   slice 2 starting at 2005-11-17-13.44.35.752587
 UT019003 Task 2 UNLOAD   slice 2 completed at 2005-11-17-13.44.36.077111
```

REORG Status Report - Section 8

The next section shows the tasks processed by each job in job name sequence.

```
 ************************************************************************
 *                                                                      *
 *                 Recap of JOB activity                                *
 *                                                                      *
 ************************************************************************


      jobname=USERA01Y  jobid=JOB01048

 UT019000 Task 1 UNLOAD   slice 1 starting at 2005-11-17-13.44.35.255392
 UT019003 Task 1 UNLOAD   slice 1 completed at 2005-11-17-13.44.35.884474

      jobname=USERAXXA  jobid=JOB05591

 UT019000 Task 2 UNLOAD   slice 2 starting at 2005-11-17-13.44.35.752587
 UT019003 Task 2 UNLOAD   slice 2 completed at 2005-11-17-13.44.36.077111

      jobname=USERAXXB  jobid=JOB05592

 UT019000 Task 3 UNLOAD   slice 3 starting at 2005-11-17-13.44.35.826364
 UT019003 Task 3 UNLOAD   slice 3 completed at 2005-11-17-13.44.36.103013
```

REORG Status Report - Section 9

The last section shows task activity in chronological order. When completed, it shows the elapsed time from when REORG was started to when it completed.

```
 ************************************************************************
 *                                                                      *
 *                 Recap of activity by time of day                     *
 *                                                                      *
 ************************************************************************

 USERA01Y JOB01048 UT019000 Task 1 UNLOAD   slice 1 starting at 2005-11-17-13.44.35.255392
 USERAXXA JOB05591 UT019000 Task 2 UNLOAD   slice 2 starting at 2005-11-17-13.44.35.752587
 USERAXXB JOB05592 UT019000 Task 3 UNLOAD   slice 3 starting at 2005-11-17-13.44.35.826364
 USERAXXC JOB05593 UT019002 Task 4 UNLOAD   index group 1 starting at 2005-11-17-13.44.35.866943
 USERA01Y JOB01048 UT019003 Task 1 UNLOAD   slice 1 completed at 2005-11-17-13.44.35.884474
 USERAXXC JOB05593 UT019005 Task 4 UNLOAD   index group 1 completed at 2005-11-17-13.44.35.985660
 USERAXXA JOB05591 UT019003 Task 2 UNLOAD   slice 2 completed at 2005-11-17-13.44.36.077111
 USERAXXB JOB05592 UT019003 Task 3 UNLOAD   slice 3 completed at 2005-11-17-13.44.36.103013
 USERA01Z JOB01069 UT019000 Task 5 RELOAD1  slice 1 starting at 2005-11-17-13.46.56.274321
 USERA01Z JOB01069 UT019003 Task 5 RELOAD1  slice 1 completed at 2005-11-17-13.46.57.381739
.
.
.
USERAXXA JOB05622 UT019005 Task 22 REBUILD4 index group 1 completed at 2005-11-17-13.47.21.060599
 USERA01Z JOB01069 UT019005 Task 23 REBUILD4 index group 2 completed at 2005-11-17-13.47.21.228884
 USERA01Z JOB01069 UT019014 All REORG phases have been completed at 2005-11-17-13.47.48.448372
 USERA01Z JOB01069 UT019015 REORG elapsed time: 0 days 0 hours 3 minutes 42 seconds
```

RELOAD Statistics Report - Section 1

The following example shows standard database statistics as returned from an ACCEPT DATABASE STATISTICS command. The values reflect the processing done by the current RELOAD1 or RELOAD2 task.

```
UT005002 DATABASE LOAD STATISTICS
 DATABASE LOADED ON 01/15/08 AT 21:24:53
PAGES READ ..................            62
PAGES WRITTEN ...............            59
PAGES REQUESTED .............            86
CALC RCDS IN TARGET PAGE ....           268
CALC RCDS OVERFLOWED ........             0
VIA RCDS IN TARGET PAGE .....         3,869
VIA RCDS OVERFLOWED .........             0
LINES REQUESTED BY IDMS .....         1,742
RCDS MADE CURRENT OF R/U ....         4,137
CALLS TO IDMS ...............         4,144
FRAGMENTS STORED ............             0
RECORDS RELOCATED ...........             0
```

RELOAD Statistics Report - Section 2

The following example shows additional statistics produced by REORG when reloading a database. The values reflect the processing done by the current RELOAD1 or RELOAD2 task.

```
 RELOAD STATS
RCDS READ ....................        4,380
RCDS STORED IN DATABASE ......        4,137
RCDS STORED ON TARGET PAGE ...        3,998
RCDS STORED TARGET PERCENT ...           96
RCDS WRITTEN TO OVERFLOW .....          243
RCDS CACHED ..................          357
RCDS STORED FROM CACHE .......          139
RCDS OVERFLOW FROM CACHE .....          218
CACHE STORAGE USED ...........       16,384
CACHE STORAGE LIMIT ..........       16,384
```

The fields reported are the following:

RCDS READ

The total database records read from a work file. This does not include pointer and other work records

RCDS STORED IN DATABASE

The number of database records stored in the database.

RCDS STORED ON TARGET PAGE

The number of database records stored on the intended target page.

RCDS STORED TARGET PERCENT

The percentage of database records stored on their target page.

RCDS WRITTEN TO OVERFLOW

The number of database records written to a SYSOF2 file, to be processed by the RELOAD2 overflow task.

RCDS CACHED

The number of database records written to the memory cache.

RCDS STORED FROM CACHE

The number of database records that were written to memory cache and then stored in the database.

RCDS OVERFLOW FROM CACHE

The number of database records that were written to memory cache and then written to the overflow work file.

CACHE STORAGE USED

The amount of allocated cache storage at the end of task. If this is the same as the storage limit, the cache reached its limit and records may have been written to the overflow file.

CACHE STORAGE LIMIT

The maximum storage allowed for the overflow cache. This can be changed using the OVERFLOW CACHE option.

DBKEYS File Layout

This section describes the contents of the record written to a DBKEYS cross reference file. The layout matches the DBKEYS file generated by CA IDMS/DB Reorg.

Field Name	Element Description	Explanation
OLDDBKEY	PIC S9(8) COMP	Old db-key
NEWDBKEY	PIC S9(8) COMP	New db-key
SR3DBKEY	PIC S9(8) COMP	SR3 db-key if the old record was relocated
RECORDID	PIC S9(4) COMP	Record ID in the source subschema
SR3RECID	PIC S9(4) COMP	Original record ID if relocated

The logical record length is 16 bytes. The BLKSIZE is determined by what is coded in the JCL or matching CREATE DSMODEL statement.

Each task that loads data generates a DBKEYS file for the data it loads. Depending on their intended use, these files may need to be merged together and sequenced before being used by another application.

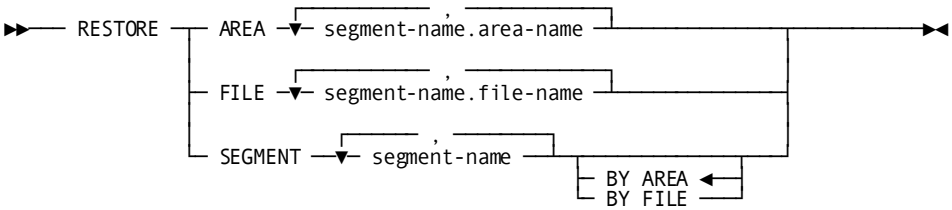
RESTORE

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

RESTORE Syntax



RESTORE Parameter

AREA

Directs the RESTORE utility to restore one or more areas.

segment-name

Specifies the name of the segment associated with the area.

area-name

Specifies the name of the area.

FILE

Directs the RESTORE utility to restore one or more files.

segment-name

Specifies the name of the segment associated with the file.

file-name

Specifies the name of the file.

SEGMENT *segment-name*

Specifies 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.

Usage

How to submit the RESTORE statement

You submit the RESTORE statement 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.

JCL Considerations

When you submit a RESTORE statement 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.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

Restoring a database by area

The following listing is generated after the statements in the "Restoring a Database by Area" example are successfully executed.

```
IDMSBCF  nn.n                                CA IDMS Batch Command Facility                    mm/dd/yy   PAGE 2


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

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
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 the statements in the "Restoring a Database by File" example shown previously are successfully executed.

```
IDMSBCF  nn.n                                CA IDMS Batch Command Facility                    mm/dd/yy   PAGE 4


  RESTORE FILE EMPDEMO.EMPDEMO,
               EMPDEMO.ORGDEMO,
               EMPDEMO.INSDEMO;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
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 the statements in the "Restoring a Database by Segment" example are successfully executed.

```
IDMSBCF  nn.n                                CA IDMS Batch Command Facility                    mm/dd/yy   PAGE 6


  RESTORE SEGMENT EMPDEMO;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
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
```

Note: For more information about restoring a database, see the *CA IDMS Database Administration Guide*.

RESTRUCTURE

The RESTRUCTURE utility modifies record occurrences to match new schema specifications.

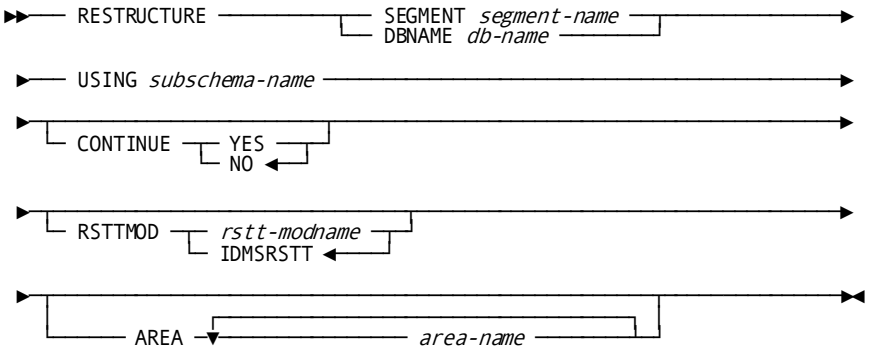
Using the RESTRUCTURE utility, you can:

- 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

RESTRUCTURE Syntax



RESTRUCTURE Parameter

db-name

Identifies the name of the database 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. The parameter *segment-name* qualifies which area the changes need to be made in

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.

Usage

How to submit the RESTRUCTURE statement

You submit the RESTRUCTURE statement only through the batch command facility. When submitting RESTRUCTURE statements, you must run the batch command facility in local mode.

The base restructuring table

The RESTRUCTURE 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.

Note: For more information and a description about the IDMSRSTT macro statements, see IDMSRSTT Macro Statements (see page 607).

The spill file

If you specify a *set-name* in the SETPTR statement of IDMSRSTT during RESTRUCTURE 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 processing to connect the new prior pointers.

Database keys of restructured records

During RESTRUCTURE 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 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 processing.

Modifying CALC and sort keys

During a restructure operation, you should not modify CALC and sort keys.

Note: For more information about modifying database components, see the *CA IDMS Database Administration Guide*.

Native VSAM data sets

You cannot use RESTRUCTURE 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 (for example, 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, follow 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 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.**

JCL Considerations

When you submit a RESTRUCTURE statement 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 40 with a maximum size of 32,760.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples and Sample Output

Adding set pointers

The following example directs the RESTRUCTURE 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

```
RESTRUCTURE empdemo using restr01
     rsttmod lrdkrstt continue yes;
```

Output from RESTRUCTURE

When the RESTRUCTURE operation is completed, the following listing is generated.

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   PAGE 1

   RESTRUCTURE EMPDEMO USING RESTR01
       RSTTMOD LRDKRSTT CONTINUE YES;
0UT000038 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
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  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                    mm/dd/yy   PAGE 1

   RESTRUCTURE CONNECT SEGMENT EMPDEMO USING EMPSS01
        RSTTMOD LRDKRSTT CONTINUE YES;
0UT000038 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
0UT000038 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
```

More Information

- For more information about changing a database definition, see the *CA IDMS Database Administration Guide*.
- For more information about CA IDMS/DB physical database characteristics, see the *CA IDMS Database Administration Guide*.
- For more information about coding IDMSRSTT macro statements, see Appendix B, IDMSRSTT Macro Statements.

Callable Restructure Utility

The Callable Restructure Utility (IDMSCRSU) is a version of the CA IDMS RESTRUCTURE 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.

Note: For more information about 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.

Considerations

IDMSCRSU does not support changes to pointers or calling database procedures. The assumption is that 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.

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.

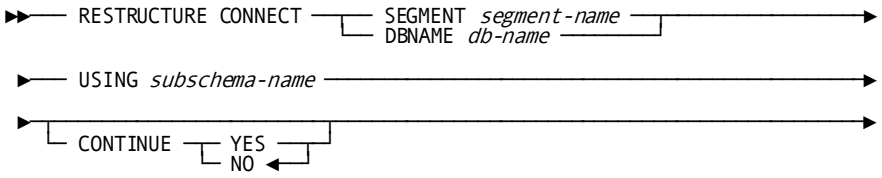
RESTRUCTURE CONNECT

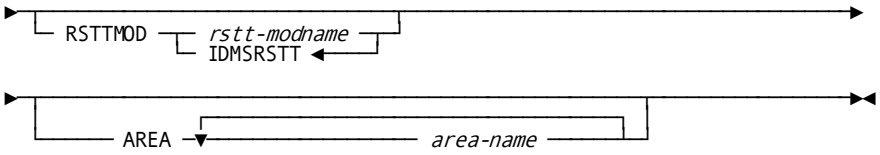
The RESTRUCTURE CONNECT utility statement connects new prior and owner pointers in existing sets. You execute the RESTRUCTURE CONNECT statement after executing the RESTRUCTURE statement as part of a database restructure operation. The RESTRUCTURE CONNECT statement uses the specifications in the base restructuring table and the information in the spill file generated by the RESTRUCTURE 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

RESTRUCTURE CONNECT Syntax





RESTRUCTURE CONNECT Parameter

db-name

Identifies the database name whose segments include the area or areas 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 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.

Usage

How to submit the RESTRUCTURE CONNECT statement

You submit the RESTRUCTURE CONNECT statement 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.

JCL Considerations

When you submit a RESTRUCTURE CONNECT statement 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. The size of the spill file should be a multiple of 40 with a maximum block size of 32,760.
- Sort files to sort the spill file in database key sequence.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples and Sample Output

To illustrate the use of the RESTRUCTURE CONNECT utility, the RESTRUCTURE 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 following example directs the RESTRUCTURE 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

```
RESTRUCTURE empdemo using restr01
     rsttmod lrdkrstt continue yes;
```

Output from RESTRUCTURE

When the RESTRUCTURE operation is completed, the following listing is generated.

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   PAGE 1

   RESTRUCTURE 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  Ldel=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  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                    mm/dd/yy   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
```

More Information

- For more information about changing database components, see the *CA IDMS Database Administration Guide*.
- For more information about the RESTRUCTURE utility, see RESTRUCTURE (see page 298).

ROLLBACK

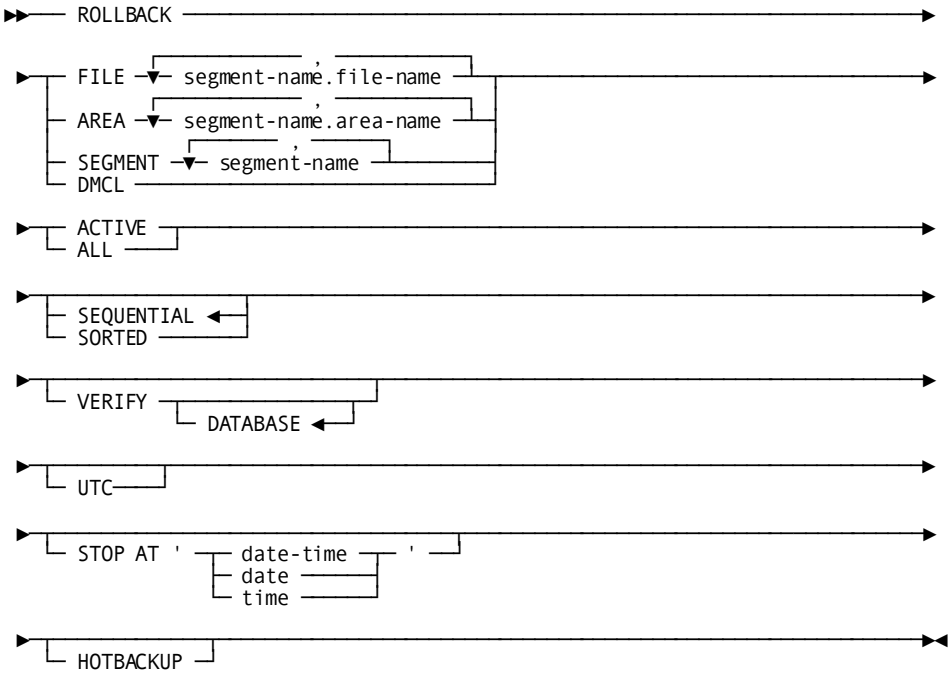
The ROLLBACK utility restores all or part of a database to a previous state by applying before images from the journal file.

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

ROLLBACK Syntax



ROLLBACK Parameter

FILE

Rolls back one or more files. Recovery by file does not unlock any areas associated with the files.

segment-name

Specifies the segment associated with the file.

file-name

Specifies 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

Specifies the name of the area.

SEGMENT

Rolls back and unlocks all areas associated with the specified segments.

segment-name

Specifies 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 appliedonly for transactions that remain open at the end of the journal file. Before images are not applied for incomplete distributed transactions whose state is InDoubt at the end of the journal file (unless a manual recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).
- 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. Before images are not applied for incomplete distributed transactions whose state is InDoubt at the stop time (unless a manual recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).

ALL

- If you do not specify a STOP time, before images are appliedfor all transactions in 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.

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. 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 SORTEDoption. 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 previously. 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.

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

HOTBACKUP

Forces ROLLBACK to also restore images for aborted run units.

UTC

Specifies that Start and Stop times are interpreted as UTC times instead of local times.

Usage

How to submit the ROLLBACK statement

You submit the ROLLBACK statement 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 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.

Note: For more information about when and how to use the ROLLBACK and ROLLFORWARD utilities, see the *CA IDMS Database Administration Guide*.

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 that can 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 UTC 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=(13,16,BI,D)
```

Note:

- In CMS, you do not need to use the ROLLBACK3490 parameter, but you must presort the journal images as described previously regardless of whether the journal files reside on disk or tape.
- 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 z/OS that you are using and whether the tape volumes are labeled, you might 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 to 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.

Note: For more information about CA IDMS/DB and native VSAM files, see the *CA IDMS Database Administration Guide*.

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 terminates 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 terminates with an error.

VERIFY is ignored if you specify SORTED.

Recovering large numbers of files

For z/OS 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. For more information about hot backup, see the *CA IDMS Database Administration Guide*.

ROLLBACK and Distributed Transactions

ROLLBACK reports on distributed transactions and supports the use of an input manual recovery control file. The input recovery control file is used to complete InDoubt distributed transactions. For more information, see JCL Considerations (see page 318) and the "Common Facilities for Distributed Transactions" appendix.

Note: For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Guide*.

JCL Considerations

When you submit a ROLLBACK statement 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

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. The format of this file is fixed block with a record length of 80.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

When the ROLLBACK operation is completed, the following listing is generated.

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   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.
```

More Information

- For more information about journal files, see the *CA IDMS Database Administration Guide*.
- For more information about database recovery, see the *CA IDMS Database Administration Guide*.
- For more information about using DBNAME for utility use, see the *CA IDMS Database Administration Guide*.

ROLLFORWARD

The ROLLFORWARD utility restores a backup copy of all or part of a database to a subsequent condition by applying after images from the journal file.

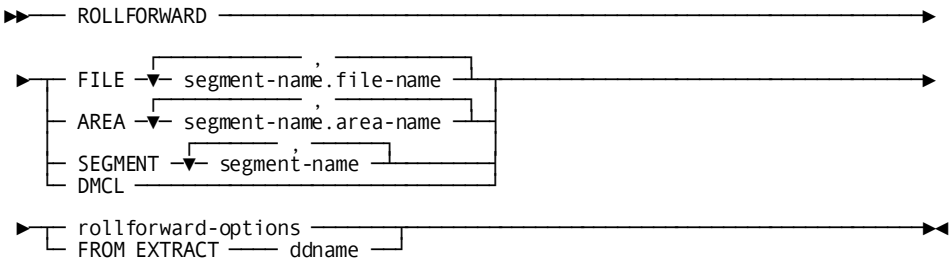
If requested, the ROLLFORWARD statement verifies the before images in the journal file against the contents of the database before applying the after images.

Authorization

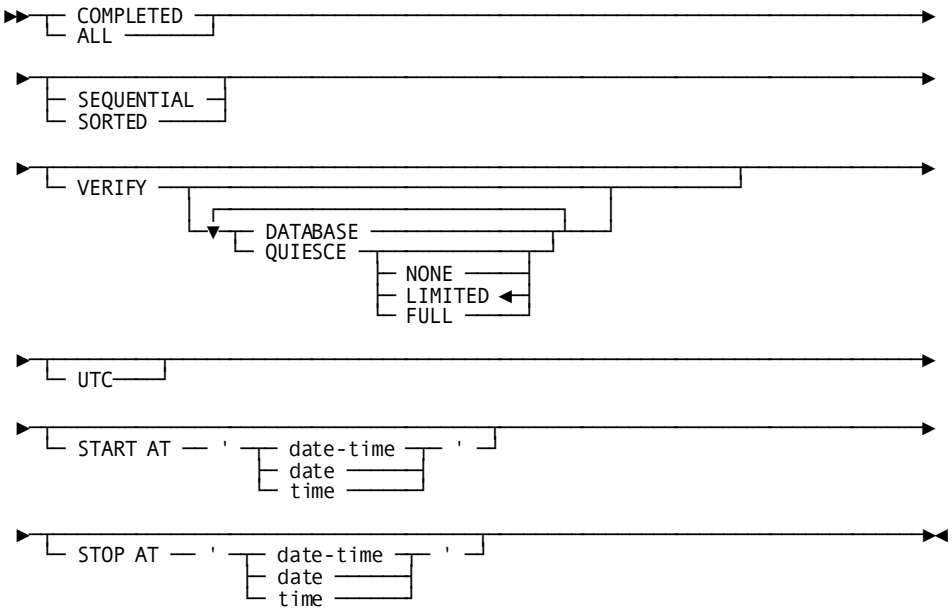
To	You Need This Privilege	On
Roll an area forward	DBAWRITE	The area
Roll a segment forward	DBAWRITE	All affected areas of the segment

To	You Need This Privilege	On
Roll a file forward	DBAWRITE	All affected areas associated with the file
Roll forward by DMCL	DBAWRITE	All affected areas within the DMCL
Roll forward FROM EXTRACT	DBAWRITE	All areas being recovered

ROLLFORWARD Syntax



Expansion of rollforward-options



ROLLFORWARD Parameter

FILE

Specifies that one or more files is to be restored. Areas associated with the specified files are not unlocked.

segment-name

Specifies the segment associated with the file.

file-name

Specifies the name of the file.

AREA

Specifies that one or more areas are to be restored and unlocked.

segment-name

Identifies the segment associated with the area.

area-name

Specifies the name of the area.

SEGMENT

Specifies to restore and unlock all areas associated with the specified segments.

segment-name

Specifies the name of the segment.

DMCL

Restores all areas defined by the DMCL specified in the SYSIDMS parameter file.

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. After images are applied for incomplete distributed transactions whose state is InDoubt at the end of the journal file (unless a manual recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).
- 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. After images are applied for incomplete distributed transactions whose state is InDoubt at the stop time (unless a manual recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).

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 z/VM. 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 z/VM 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.
  - NONE—Specifies that no quiesce point verification should be done. If "START AT" is specified, NONE is treated as LIMITED.
  - 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 previously. 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.

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

UTC

Specifies that Start and Stop times are interpreted as UTC times instead of local times.

Usage

How to submit ROLLFORWARD

You submit the ROLLFORWARD statement 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, as well as all relevant journals.

Note: For more information about when and how to use the ROLLBACK and ROLLFORWARD utilities, see the *CA IDMS Database Administration Guide*.

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 that can 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 can 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 about recovering from a hot backup, see the *CA IDMS Database Administration Guide*.

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. To perform a correct recovery, you must begin the recovery operation at a quiesce point for the portion of the database being recovered. 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 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 be used only 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 z/OS 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.

ROLLFORWARD and Distributed Transactions

ROLLFORWARD reports on distributed transactions and supports the use of input and output manual recovery control files. Unless ALL is specified, the input manual recovery control file is used to complete InDoubt distributed transactions. If an output manual recovery control file is included in the JCL, unless FROM EXTRACT is specified, an entry is written for each incomplete distributed transaction encountered. For more information, see JCL Considerations (see page 328) and the "Common Facilities for Distributed Transactions" appendix.

Note: For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Guide*.

JCL Considerations

When you submit a ROLLFORWARD statement 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

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a manual recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

The ROLLFORWARD utility generates the following listing after executing the statement in the first example.

```
IDMSBCF                                        IDMS Batch Command Facility                        mm/dd/yy   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.
```

More Information

- For more information about database recovery, see the *CA IDMS Database Administration Guide*.
- For more information about using DBNAME for utility use, see the *CA IDMS Database Administration Guide*.
- For more information about when to use the ROLLFORWARD utility, see the *CA IDMS Database Administration Guide*.

SYNCHRONIZE STAMPS

The SYNCHRONIZE STAMPS utility manipulates SQL synchronization stamps in the following ways:

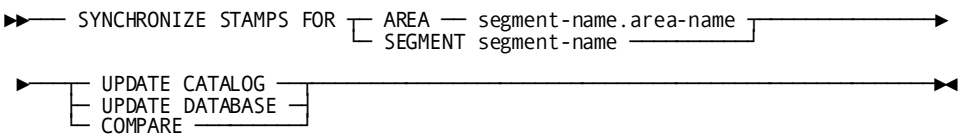
- Displays and compares the stamps in the catalog and the data area(s), issuing a warning if stamps are inconsistent
- Updates the catalog with the stamps from the data area(s)
- Updates the data area(s) with the stamps from the catalog

This utility provides an alternate mechanism for taking snapshot copies of identically defined databases and also as an aid in recovery situations in which either the catalog or a data area must be restored independently of each other.

Authorization

To	You Need This Privilege	On
Use SYNCHRONIZE STAMPS	DBAWRITE	Every area processed by SYNCHRONIZE STAMPS

SYNCHRONIZE STAMPS Syntax



SYNCHRONIZE STAMPS Parameter

AREA

Specifies the area in which to synchronize stamps.

segment-name

Specifies the name of the segment associated with the area.

area-name

Specifies the name of an area included in the DMCL module.

SEGMENT

Specifies the segment whose areas will have their stamps synchronized.

segment-name

Specifies the name of a segment included in the DMCL module.

UPDATE CATALOG

Specifies that the catalog is to be updated with the stamps from the data area(s).

UPDATE DATABASE

Specifies that the area(s) are to be updated with stamps from the catalog.

COMPARE

Displays and compares the stamps in the catalog and the data area(s) and issues a warning if the stamps are inconsistent.

Usage

How to submit the SYNCHRONIZE STAMPS statement

You submit the SYNCHRONIZE STAMPS statement using either the batch command facility or the online command facility.

Use caution when updating stamps

By using the SYNCHRONIZE STAMPS utility to update stamps in a catalog or data area, you are asserting that the definition in the catalog accurately describes data in the area. You should be sure that this is true before updating stamp values. No data validation is performed by the utility.

JCL Considerations

When you submit a SYNCHRONIZE STAMPS statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the following:

- File(s) that map to the area to be processed
- Dictionary that defines the table(s) in the area
- Journal file(s) if backups are not taken

To run under central version, a SYSCTL statement is needed.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Examples

The following example directs the SYNCHRONIZE STAMPS utility to compare the synchronization stamps in the USERDB.EMP_AREA with those in the catalog.

```
synchronize stamps for area userdb.emp_area compare;
```

Sample Output

The following report sample is produced by the SYNCHRONIZE STAMPS utility.

```
IDMSBCF  nn.n                    CA IDMS Batch Command Facility                  mm/dd/yy                PAGE 1
SYNCHRONIZE STAMPS FOR AREA USERDB.EMP_AREA COMPARE;
*+ Status = 0        SQLSTATE = 00000
                        *** Current Stamps Report ***
  Area:USERDB.EMP_AREA                        Table Stamping
      Catalog Stamp:  <null>

      Database Stamp: <null>

          Table ID:1024   Table:DEMO.EMPL
              Catalog Stamp:  1993-03-08-14.50.01.952955
              Database Stamp: 1993-03-08-14.50.01.952955

          Table ID:1025   Table:DEMO.POSITION
              Catalog Stamp:  1993-03-08-14.50.01.668147
              Database Stamp: 1993-03-08-14.50.01.668147

          Table ID:1026   Table:DEMO.MANAGERS
              Catalog Stamp:  1993-03-08-14.50.01.952955
              Database Stamp: 1993-03-08-14.50.01.952955

          Table ID:1027   Table:INV.PART
              Catalog Stamp:  1996-06-18-10.40.51.839925
              Database Stamp: 1996-06-18-10.40.51.839925

          Table ID:1028   Table:INV.COMPONENT
              Catalog Stamp:  1996-06-18-10.40.51.839925
              Database Stamp: 1996-06-18-10.40.51.839925

          Table ID:1029   Table:EMP.T5
              Catalog Stamp:  2001-11-05-09.31.31.638046
              Database Stamp: 2001-11-05-09.31.31.638046

          Table ID:1030   Table:LRD.EMPL
              Catalog Stamp:  2002-06-28-15.11.18.494317
              Database Stamp: 2002-06-28-15.11.18.494317

          Table ID:1031   Table:JPD.T5
              Catalog Stamp:  1999-06-21-13.04.08.968700
              Database Stamp: 1999-06-21-13.04.08.968700
```

Note: For more information about synchronization stamps, see the *CA IDMS Database Administration Guide* and the *CA IDMS SQL Reference Guide*.

TUNE INDEX

The TUNE INDEX utility performs the following functions:

- Adopts orphans in an index structure. An *orphaned indexed record* is a record whose index pointer does not point back to the index record (SR8) that contains the record's index entry. Orphans occur as the result of splitting an existing SR8 into two records to accommodate a new entry. As part of the split, some of the entries are moved to a new SR8, but the index pointer in their associated records is not adjusted to reflect the change, resulting in "orphaned" records. By eliminating orphans, runtime database performance is improved when traversing from an indexed record to its associated index entry.

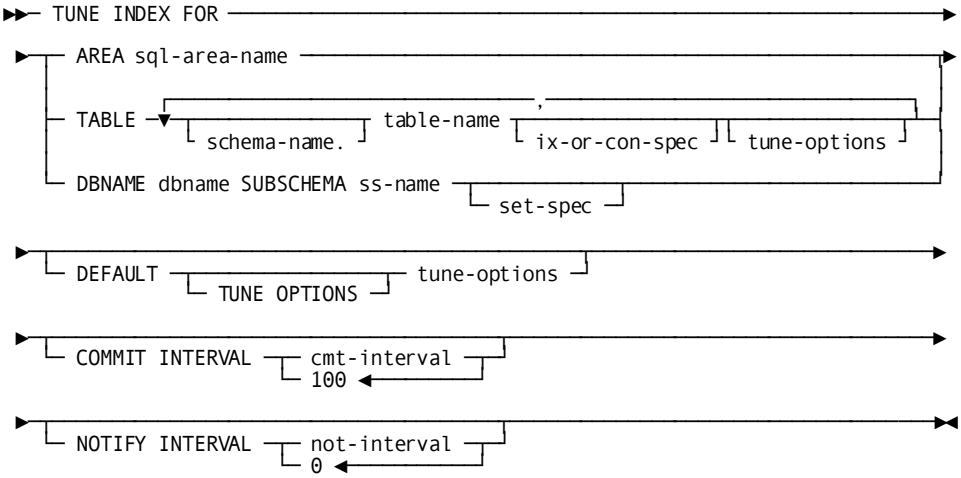
- Moves the top level SR8 to its optimal location.
- Optionally rebalances the index structure. Rebalancing ensures that the resulting index structure is a balanced tree and has a minimal number of levels and SR8's. You can temporarily override the index block contains value of the index and the page reserve value of the area that contains the index structure. Using these overrides allows tuning the index while allowing for future growth.
- Optionally resequences the index structure. Resequencing puts the SR8 records in physical sequence. By resequencing the index structure, database performance is improved when accessing the index structure sequentially at the bottom level.

Note: Under some circumstances the process of moving the top level SR8 to its optimal location might cause some degree of resequencing to be performed even if RESEQUENCE NO is specified, or when the RESEQUENCE option is allowed to default to NO.

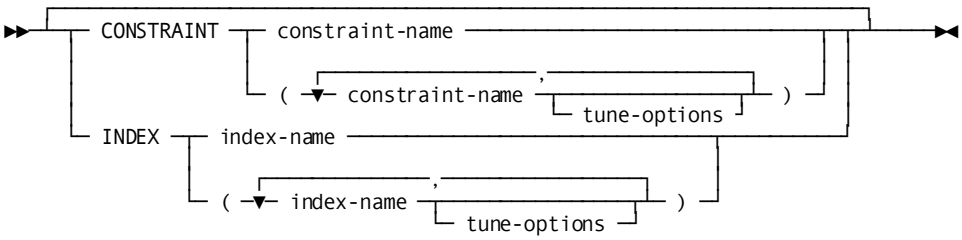
Authorization

A user must have DBAWRITE authority on all areas processed by the utility.

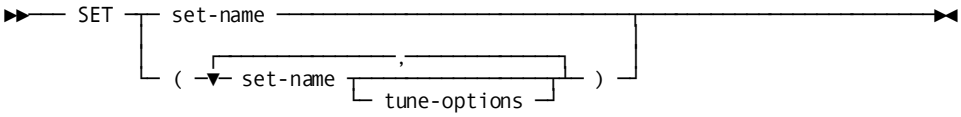
TUNE INDEX Syntax



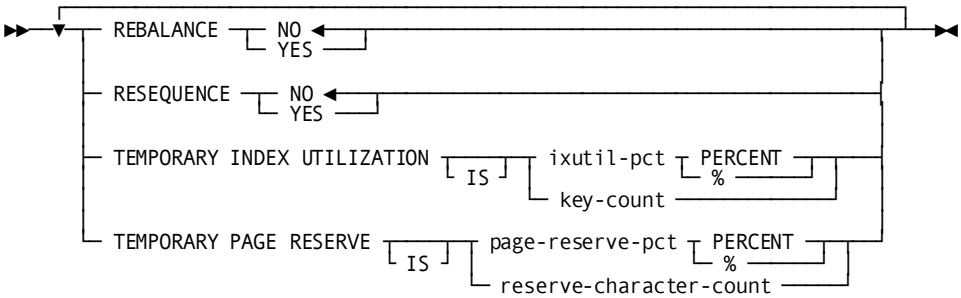
Expansion of *ix-or-con-spec*



Expansion of *set-spec*



Expansion of *tune-options*



TUNE INDEX Parameter

schema-name

Identifies the schema that will qualify the table name.

If omitted, the current session associated with the user session schema is used. Schema-name is required if there is no current session schema.

table-name

Identifies the table that is constrained by an indexed constraint.

ix-or-con-spec

Identifies constraints and indexes on the current table that is being tuned.

If omitted, all indexed constraints and indexes 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 length of the notify interval in minutes. After each interval expires, a message is issued stating how far the job has progressed. If the notify interval is specified as 0 or allowed to default to 0, no notify messages are created.

constraint-name

Identifies an indexed constraint on the current table that is to be tuned.

index-name

Identifies an index on the current table that is to be tuned.

DEFAULT TUNE OPTIONS

The tune options to be used during the processing of an index if tune options have not been specifically specified.

REBALANCE

Specifies whether to rebalance the index. A well-balanced index has the minimum number of index levels and best performance if the index is frequently accessed vertically from top to bottom.

YES

Rebalances the index.

Note: Rebalancing an index can be resource-intensive.

NO

No rebalancing is done.

RESEQUENCE

Specifies whether to resequence the index. A properly sequenced index is important only if the index is frequently accessed sequentially at the bottom level.

YES

Resequences the index for optimum performance.

Note: Resequencing an index can be resource-intensive.

NO

No resequencing is done.

TEMPORARY INDEX UTILIZATION

Specifies a temporary override for the operation. If not specified, the current run-time value for INDEX BLOCK CONTAINS is used and index blocks are used at 100%.

ixutil-pct

Specifies the percentage of the maximum number of entries that each index block should contain after tuning is complete. *ixutil-pct* is an integer in the range 10 through 100. The number of entries of an index block is computed as *index-block-contains* * *ixutil-pct* / 100.

key-count

Specifies the maximum number of entries that each index block should contain after tuning is complete. *key-count* is an integer in the range 3 through 8180.

Note: If the specified value exceeds the current run-time value of the INDEX BLOCK CONTAINS, the key-count value is ignored and the INDEX BLOCK CONTAINS value will be used.

TEMPORARY PAGE RESERVE

Specifies a temporary override of the page reserve for the area in which the index resides. If not specified or specified as NULL, the page reserve of the area in which the index resides is used.

page-reserve-pct

Specifies the percentage of each page to leave as free space if it contains a portion of an index being tuned. *page-reserve-pct* is an integer in the range 0 through 30. The page reserve of the area is computed as *area-page-size* * *page-reserve-pct* / 100.

reserve-character-count

Specifies the number of characters to reserve on each page to accommodate increases in the length of records or rows stored on the page if it contains a portion of an index being tuned. *reserve-character-count* is an integer with a value not larger than 30% of the page size.

Usage

General Considerations

The TUNE INDEX utility has the following usage considerations:

- To use the TUNE INDEX utility, you must specify 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 further improves performance.
- Index tuning is a resource-intensive operation consisting mostly of CPU and I/O.

Operating modes

You can execute the TUNE INDEX utility both online (through the online command facility) and in batch through central version or batch local. When index tuning is executed by a central version, TUNE INDEX tries to minimize impact on other online tasks as follows:

- When a record or area lock conflict occurs with other applications, TUNE INDEX takes the following actions:
  - For record lock conflicts, TUNE INDEX commits the updates done so far.
  - For area lock conflicts, TUNE INDEX finishes its database session and starts a new one.
- TUNE INDEX lowers its priority to one below its normally assigned priority. If a deadlock occurs, the default deadlock selection algorithm selects the task with the lowest priority as the deadlock victim. The TUNE INDEX utility can recover from a deadlock by restarting the index tuning process of the current index occurrence.

Commit interval

You can specify a commit interval that determines the frequency with which the utility will commit. The interval specifies the number of updates that can take place before a commit is issued. You can disable committing and automatic restart by specifying a 0-commit interval. Regardless of the commit interval specified, the utility always issues a COMMIT ALL at the end of the tune process of an index occurrence to release all record locks. It also issues a COMMIT ALL if it detects that another task is waiting on a record lock that it holds and it issues a FINISH if it detects that another task is waiting on an area lock that it holds.

Notify interval

You can specify a time interval in minutes. Each time this interval expires, a message is written indicating the index tuning progress. The message is written to the job log and the operator's console if TUNE INDEX runs in local mode; otherwise, it is written to the IDMS LOG and console. You can disable notification by specifying a 0-notify interval.

TUNE OPTIONS Usage

When processing multiple indexes within a single execution of the TUNE INDEX utility the tune options to be used are determined by the following hierarchy:

1. Options specified for a particular index.
2. Options specified on a DEFAULT TUNE OPTIONS statement
3. The utility default values.

JCL Considerations

When you submit a TUNE INDEX utility through the batch command facility in local mode, the JCL to execute the facility must include statements to define the files containing the areas to be processed. To run under central version, a SYSCTL statement is needed.

Examples

The following example directs the TUNE INDEX utility to adopt orphaned index records in the EMPDEMO dbname using subschema EMPSS01:

```
tune index for dbname empdemo subschema empss01;
```

The following example directs the TUNE INDEX utility to adopt orphaned index records, rebalance and resequence the index. It also shows how to temporarily override the DMCL or subschema values for PAGE RESERVE and INDEX BLOCK CONTAINS.

```
TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
           SET (EMP-NAME-NDX)
           DEFAULT TUNE OPTIONS
              REBALANCE  YES
              RESEQUENCE YES
              TEMPORARY INDEX UTILIZATION IS 80 %
              TEMPORARY PAGE RESERVE IS 15 PERCENT
           NOTIFY INTERVAL 1000;
```

Sample Output

The following example directs the TUNE INDEX utility to adopt orphaned index records in the EMPDEMO dbname using subschema EMPSS01:

```
IDMSBCF  nn.n                          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  -  Indexes selected for processing:
DB002994 C0M333: IDMSTUNE  -  SKILL-EXPERTISE (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 C0M333: IDMSTUNE  -  EMP-NAME-NDX (IBC=40) in area EMPDEMO.EMP-DEMO-REGION (PGRSV=0)
DB002994 C0M333: IDMSTUNE  -  OFFICE-EMPLOYEE (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 C0M333: IDMSTUNE  -  SKILL-NAME-NDX (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 C0M333: IDMSTUNE  -  JOB-TITLE-NDX (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 C0M333: IDMSTUNE  -  Statistics for area EMP-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  Orphan adoption read 36 records (of which 8 SR8s)
DB002994 C0M333: IDMSTUNE  -  Orphan adoption adopted 20 index orphans
DB002994 C0M333: IDMSTUNE  -  Resequencing read 4 records
DB002994 C0M333: IDMSTUNE  -  Statistics for area ORG-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  Orphan adoption read 259 records (of which 63 SR8s)
DB002994 C0M333: IDMSTUNE  -  Orphan adoption adopted 0 index orphans
DB002994 C0M333: IDMSTUNE  -  Resequencing read 126 records
DB002994 C0M333: IDMSTUNE  -  425 total records read
DB002994 C0M333: IDMSTUNE  -  20 total index orphans 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
```

The following example directs the TUNE INDEX utility to adopt orphaned index records, rebalance and resequence the index. It also shows how to temporarily override the DMCL or subschema values for PAGE RESERVE and INDEX BLOCK CONTAINS.

```
TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
          SET (EMP-NAME-NDX)
          DEFAULT TUNE OPTIONS
             REBALANCE  YES
             RESEQUENCE YES
             TEMPORARY INDEX UTILIZATION IS 80 %
             TEMPORARY PAGE RESERVE IS 15 PERCENT
          NOTIFY INTERVAL 1000;
```

The following report sample is produced by the TUNE INDEX utility.

```
TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
           SET (EMP-NAME-NDX)
           DEFAULT TUNE OPTIONS
              REBALANCE  YES
              RESEQUENCE YES
              TEMPORARY INDEX UTILIZATION IS 80 %
              TEMPORARY PAGE RESERVE IS 15 PERCENT
           NOTIFY INTERVAL 1000;
Status = 0       SQLSTATE = 00000       Messages follow:
DB002994 C0M333: IDMSTUNE  -  processing started
DB002994 C0M333: IDMSTUNE  -  Indexes selected for processing:
DB002994 C0M333: IDMSTUNE  -  EMP-NAME-NDX (IBC=32) in area EMPDEMO.EMP-DEMO-REGION (PGRSV=644)
DB002994 C0M333: IDMSTUNE  -  Statistics for area EMP-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  Orphan adoption read 34 records (of which 6 SR8s)
DB002994 C0M333: IDMSTUNE  -  Orphan adoption adopted 20 index orphans
DB002994 C0M333: IDMSTUNE  -  Rebalancing read 65 records
DB002994 C0M333: IDMSTUNE  -  Resequencing read 35 records
DB002994 C0M333: IDMSTUNE  -  134 total records read
DB002994 C0M333: IDMSTUNE  -  20 total index orphans adopted
DB002994 C0M333: IDMSTUNE  -  1 indexes/sets processed
DB002994 C0M333: IDMSTUNE  -  processing completed
```

More Information

- For more information about indexed constraints, see the *CA IDMS SQL Reference Guide*.
- For more information about dbnames, subschemas, and sets, see the *CA IDMS Database Administration Guide*.

UNLOAD

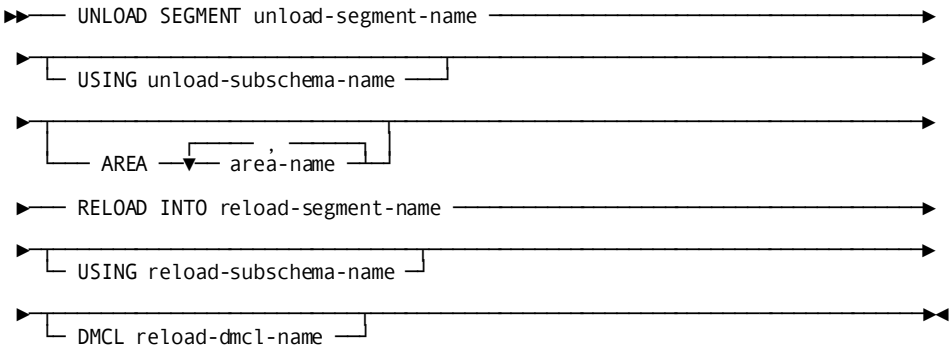
The UNLOAD utility unloads records from one or more areas of the database in preparation for reloading. The UNLOAD utility *does not* reload data.

Note: You cannot use UNLOAD to unload an SQL database that contains keys longer than 256 bytes. Instead, use the REORG utility instead of UNLOAD to reorganize a large key database.

Authorization

To	You Need This Privilege	On
Unload an area	DBAREAD	■ The area ■ Any area(s) with set connections to the area ■ Any area(s) containing an index defined on records in the area
Unload a segment	DBAREAD	All areas of the segment

UNLOAD Syntax



UNLOAD Parameter

unload-segment-name

Specifies the segment of the database to unload.

unload-subschema-name

Specifies the subschema that describes the database.

By default, if an *unload-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-name

Specifies an area in the unload segment to unload. The area must also exist in the reload segment.

By default, if no areas are specified, all areas in the unload segment that are also defined in the unload subschema will be unloaded.

reload-segment-name

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.

If an SQL-defined database is being unloaded, *segment-name* must be the same as the one specified in the SEGMENT clause.

reload-subschema-name

Specifies the subschema that describes the segment in which the data will be reloaded.

Do not specify a reload subschema if unloading an SQL-type segment.

This parameter is passed in a control record at the beginning of the RELDCTL file to be used by the RELOAD utility.

By default, if a reload subschema is not specified, the subschema that was used to unload the data is used to reload the data.

reload-dmcl-name

Specifies the DMCL that defines the segment into which the data will be reloaded. See "The Unload/Reload DMCL" in the "Usage" section 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.

By default, if a reload-dmcl-name is not specified, the DMCL used during the unload process is used. Unload uses the DMCL that is specified in the SYSIDMS parameter file. If no DMCL is specified in the SYSIDMS parameter file, the name IDMSDMCL is used.

Usage

How to submit the UNLOAD statement

You submit the UNLOAD utility 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 EXPAND PAGE (see page 84) for complete information about 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 and 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 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 or subschema must be available during the unload operation and during the reload operation.

You can make the following changes by using UNLOAD and RELOAD:

- Change area page ranges
- Change page size

  See EXPAND PAGE (see page 84) 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, see 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.

The UNLOAD utility cannot be used against native VSAM files.

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. The subschemas specified for the UNLOAD cannot contain areas that reside in page groups other than the page group for the segment being unloaded. If the environment has an EXIT34 installed that exit will be invoked. If the CA supplied EXIT34 is used (RHDCUX34) and has not been altered a message will be produced indicating that an unqualified FIND/OBTAIN DBKEY command has been issued. If the exit has been modified to abort the associated run-unit when this type of command is encountered, the UNLOAD utility will be abnormally terminated. You must use multiple invocations of the utility to process areas in different page groups.

UNLOAD and the DCMT VARY SEGMENT/AREA PERMANENT

If the UNLOAD utility is to be run with the purpose of changing an area's page range and that change includes changing the area's low page, it is recommended that none of the DCMT VARY SEGMENT/AREA commands using the PERMANENT option be issued against the original area(s). The PERMANENT feature is implemented by carrying the area's low-page number in the journals across cycles of the CV. Changing an area's low-page will prohibit future cycles of the CV to properly identify the area once the new page range is implemented.

If a DCMT VARY SEGMENT/AREA PERMANENT command is still in effect when the new page range is implemented, the area's usage-mode at startup will be determined by the value specified in the DMCL. The entry in the journals for the old area's page range will remain until the next format of the journals.

UNLOAD and the CLEANUP utility

If the area being unloaded contains logically deleted records, run the CLEANUP utility before running unload.

Note: For more information about making these kinds of changes, see the *CA IDMS Database Administration Guide*.

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.

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 next.

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.

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 constraint to tables being unloaded.

The Unload/Reload DMCL

The DMCL used in the unload and reload operations must include the segment whose areas are being unloaded.

The UNLOAD utility must establish access to DMCLs for the unloading of the original database and for determining the page ranges of the database areas to be used by the subsequent RELOAD utility. The DMCL(s) to be used by each step of the process is determined as follows.

Unloading the original database

The unload operation will use the DMCL specified in the SYSIDMS file for the UNLOAD utility jobstep. If no DMCL is specified in the SYSIDMS file, the DMCL named IDMSDMCL will be used.

Reloading the new database

The DMCL specified in the RELOAD clause of the UNLOAD utility parameter will be used. If a DMCL is not specified, the DMCL to be used by the UNLOAD operation will be used to determine record placement in the new DMCL and is passed on to the RELOAD utility.

If the area definitions between the original database and the reload database are different, the unload and reload DMCLs must have different names, or the area definition must reside in different segments within the same DMCL.

General Procedure for UNLOAD and RELOAD

Take the following steps to unload and reload a database:

1. Create the new subschema or DMCL reflecting the changes to be made.
2. Back up 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. Back up the same areas as in step 2.

This 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 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

JCL Considerations

When you submit an UNLOAD utility through the batch command facility, the JCL to execute the facility must include statements to define the following:

- Files containing the areas to be processed
- Journal file(s), which can be dummied
- Output files

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

The UNLOAD utility generates the following listing after successful completion of the previous example.

```
IDMSBCF                                        IDMS Batch Command Facility                        mm/dd/yy   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 -------- yy-mm-dd
         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_EMPL          DEPT               - OWNER
ORG_AREA           JOB_POS            JOB                - OWNER
UT004001 IDMSDBL1 RELEASE nn.n TAPE volser PROCESSING STARTED
         SUBSCHEMA NAME ------ UNLD$SQL
         COMPILE DATE -------- yy-mm-dd
         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 nn.n 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
```

Note: For more information about unloading and reloading a database, see the *CA IDMS Database Administration Guide*.

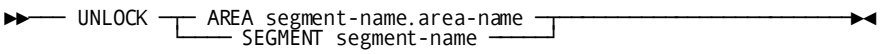
UNLOCK

The UNLOCK utility removes the external lock on an area.

Authorization

To	You Need This Privilege	On
Unlock an area	DBAWRITE	The area

UNLOCK Syntax



UNLOCK Parameter

area

Directs the UNLOCK utility to remove the lock, if present, on the specified area.

segment-name

Specifies the name of the segment associated with the area.

area-name

Specifies the name of the area.

SEGMENT *segment-name*

Specifies the name of the segment to be unlocked.

Usage

How to submit the UNLOCK statement

You submit the UNLOCK statement only through the batch command facility. When submitting UNLOCK statements, you must run the batch command facility in local mode.

JCL Considerations

When you submit an UNLOCK statement 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.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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;
```

Sample Output

Unlock by area

When UNLOCK processing is completed, the following listing is generated.

```
IDMSBCF                                          IDMS Batch Command Facility                          mm/dd/yy   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  nn.n          CA IDMS Batch Command Facility     mm/dd/yy    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
```

Note: For more information about area locks, see the *CA IDMS Database Administration Guide*.

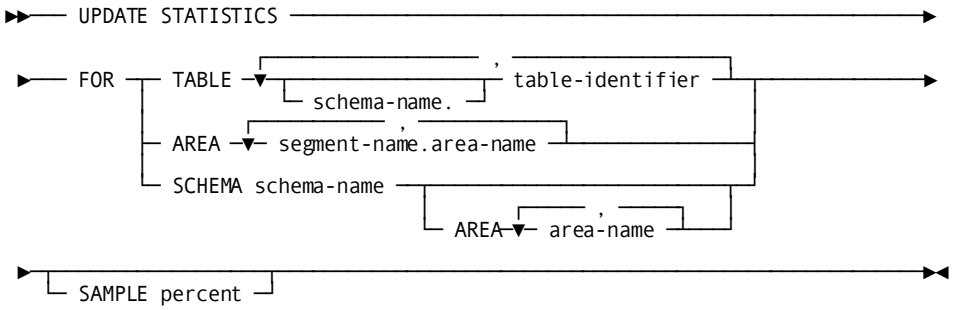
UPDATE STATISTICS

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

UPDATE STATISTICS Syntax



UPDATE STATISTICS Parameter

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

Specifies the name of the schema associated with the named table.

table-identifier

Specifies the identifier of a base table defined in the dictionary.

AREA

Updates statistics for all the tables in one or more SQL-defined areas. If one of the specified areas is non-SQL-defined, error message DB002316 is displayed which indicates that the area is NOT a relational area. In this case, specify the areas through the SCHEMA-clause of the UPDATE STATISTICS utility.

segment-name

Specifies the name of the segment associated with the area.

area-name

Specifies the name of the area.

SAMPLE

Specifies the percentage of the pages in an area that the UPDATE STATISTICS utility is to examine when calculating statistical information about one or more tables in the area.

percent

Specifies 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 the schema for which statistics are updated. The identified schema must be SQL-defined and can reference a non-SQL-defined schema.

AREA *area-name*

Identifies one or more areas of the identified schema for which statistics are to be updated.

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.

UPDATE STATISTICS for IDMSNTWK

When updating statistics for an SQL-defined schema that references the non-SQL-defined schema IDMSNTWK, the UPDATE STATISTICS utility will only process area DDLDML. If an area other than DDLDML is specified using the AREA clause, a warning will be issued indicating there are no tables to process.

UPDATE STATISTICS for native VSAM files

When updating statistics for an SQL-defined schema that references a non-SQL-defined schema containing native VSAM files, the job abends with error-message DB002300 and DBIO Error code 3077. The UPDATE STATISTICS module IDMSCOLS collects statistics of some information found on a BDAM DB page (for example, total space available count) that cannot be found on a native VSAM page. In this case, the DBIO Error code 3077 indicates that an attempt was made to run an invalid CA IDMS utility against a native VSAM file.

UPDATE STATISTICS for a non-SQL-defined schema

When updating statistics for an SQL-defined schema that references a non-SQL-defined schema, statistics are stored in the dictionary (DDLDML) where the non-SQL schema is defined. For a system-owned index key that does not contain group elements, statistics are kept for each index column. If the index key contains a group element, a DB003202 warning message is issued and only the statistics of the first index column are updated.

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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Example

Updating statistics for specified tables

The following UPDATE STATISTICS statement 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 following UPDATE STATISTICS statement updates statistics for all the tables in the SQLDEMO.EMPLAREA area.

```
update statistics
   for area sqldemo.emplarea;
```

Sample Output

After successful completion of the UPDATE STATISTICS statement on the SQLDEMO.EMPLAREA area shown previously, the following listing is produced.

```
IDMSBCF                                    IDMS Batch Command Facility                    mm/dd/yy   PAGE 1

  UPDATE STATISTICS FOR AREA SQLDEMO.EMPLAREA;
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

Note: For more information about updating statistics, see the *CA IDMS Database Administration Guide*.

VALIDATE

The VALIDATE utility checks linked and unlinked referential constraints for a referencing table, making sure that referenced tables exist and contain the appropriate column values.

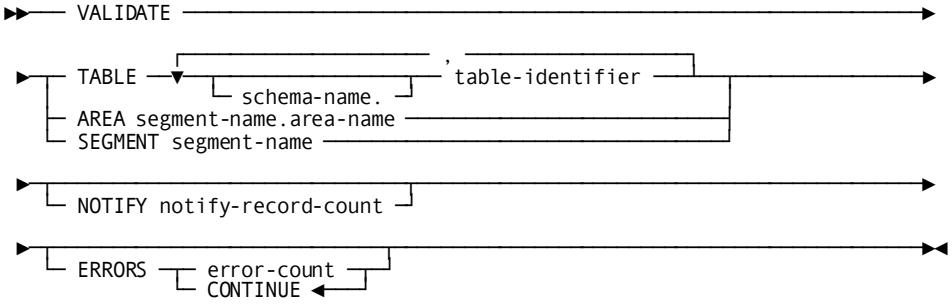
Type of VALIDATE	What it does
Complete VALIDATE	Runs both STEP1 and STEP2
Stepped VALIDATE	Runs one step at a time with intermediate file sorting required between STEP1 and STEP2

Authorization

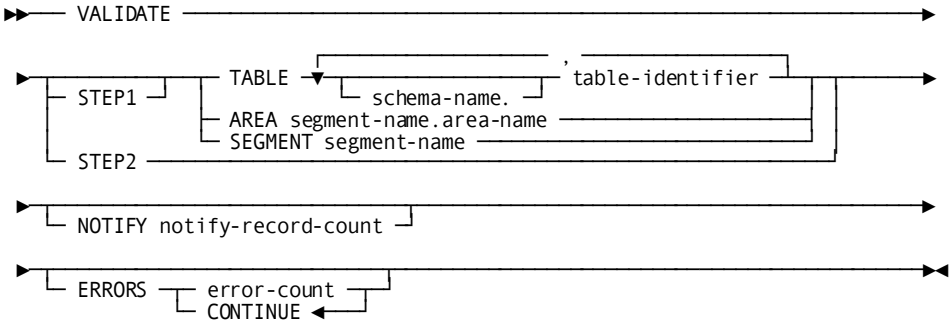
To	You need this privilege	On
Validate a table	SELECT	The table
Validate the tables in an area	SELECT	All tables in the area
Validate the tables in a segment	SELECT	All tables in the segment

VALIDATE Syntax

Syntax for complete VALIDATE



Syntax for stepped VALIDATE



Note: Only one LOAD, BUILD, or VALIDATE statement can be performed during one execution of the Batch Command Facility (IDMSBCF).

VALIDATE Parameter

TABLE

Identifies the referencing table to validate.

schema-name

Specifies the name of the schema that defines the table.

table-identifier

Specifies 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

Specifies the name of the segment containing the area.

area-name

Specifies 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

Specifies 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

Specifies 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

Specifies 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.

Usage

How to submit the VALIDATE statement

You submit the VALIDATE statement 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.

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.

Note: For more information about referential constraints, see the *CA IDMS Database Administration Guide*.

JCL Considerations

When you submit a VALIDATE statement through the batch command facility, the JCL to execute the facility must include statements to define the following:

- Dictionary containing table definitions
- Files containing the areas associated with the referencing tables to be processed
- Intermediate work files
- Sort work files are needed if doing a complete VALIDATE

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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter specific to your operating system.

Example

The following example 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;
```

Sample Output

The following listing was generated after validating sample tables M and M2 in the previous 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.DDLCAT;
Status = 1       Extended Reason Code = 2367     Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCAT was not locked.
UNLOCK AREA SYSSQL.DDLCATX;
Status = 1       Extended Reason Code = 2367     Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCATX was not locked.

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

VALIDATE TABLE LOAD.M,
               LOAD.M2
     ERRORS CONTINUE;
IDMSLOAD  -   volser     VALIDATE INDEXES STEP 1     yy-mm-dd-hh.mm.ss
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
```

More Information

- For more information about designing linked and unlinked referential constraints, see the *CA IDMS Database Design Guide*.
- For more information about defining linked and unlinked referential constraints, see the *CA IDMS Database Administration Guide*.

Chapter 6: Utility Programs

This section contains the following topics:

IDMSCALC (see page 365)
IDMSDBAN (see page 367)
IDMSDIRL (see page 385)
IDMSLOOK (see page 389)
IDMSRPTS (see page 403)
IDMSRSTC (see page 428)

IDMSCALC

The IDMSCALC utility is a subroutine that 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.

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 following table, then call IDMSCALC, passing the argument. IDMSCALC must be link edited with the calling program.

Calling the IDMSCALC Routine

The following example 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.

Note: For more information about how the CALC location mode works, see the *CA IDMS Database Administration Guide*.

IDMSDBAN

The IDMSDBAN database analysis utility 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

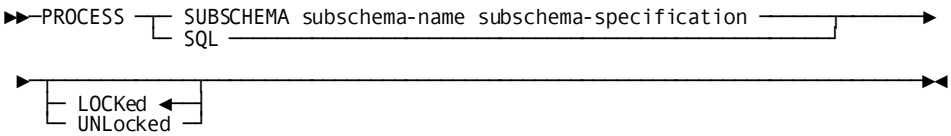
Authorization

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

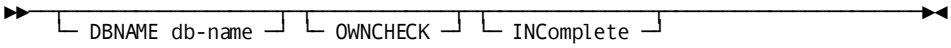
Syntax

PROCESS statement

Code on one line only.

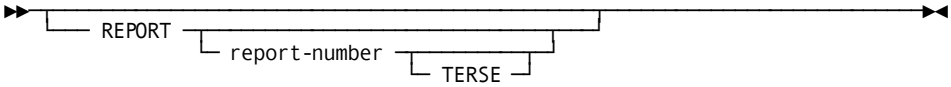


Expansion of subschema-specification



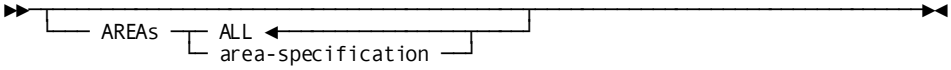
REPORT statement

Code on one line only.

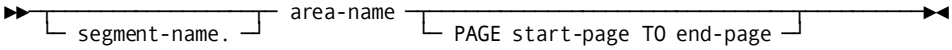


AREA statement

Code on one line only.

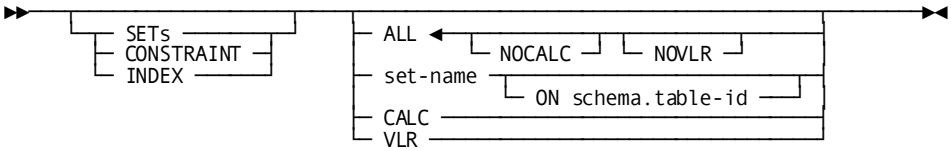


Expansion of area-specification



SET statement

Code on one line only.



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

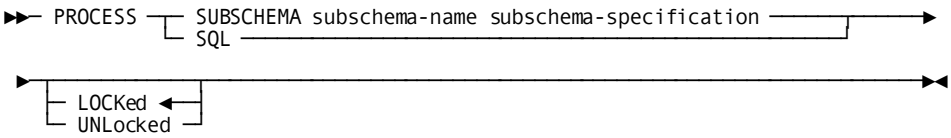
Statement	Required/ Optional	Description
PROCESS	Required	For non SQL-defined database: ■ Identifies subschema and DMCL module ■ Specifies whether areas are to be locked before processing For SQL-defined database: ■ Identifies that an SQL-defined database will be processed ■ Specifies whether areas are to be locked before processing
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

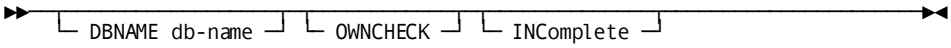
The PROCESS statement identifies the database to be processed and specifies whether areas are to be locked before processing begins. One PROCESS statement is required for each IDMSDBAN run.

Syntax

Code on one line only.



Expansion of subschema-specification



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.

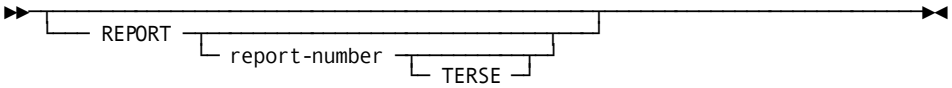
REPORT statement

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:

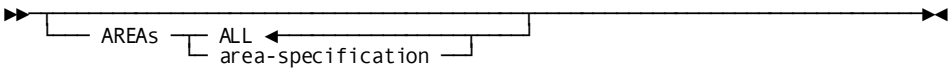
- TERSE causes **report 3** to be produced only for sets that have at least one non-empty set occurrence.
- TERSE causes **report 4** to be produced only for record types with at least one occurrence in the database.
- TERSE causes **report 5** to be produced only for sets with at least one occurrence in the database.

AREA statement

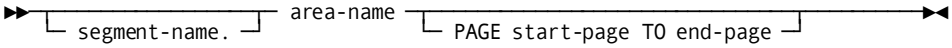
The AREA statement specifies the areas to be processed. If no AREA statement is coded, all areas in the named subschema or segment are processed.

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.

segment.:

For SQL-defined databases, identifies the name of the segment to be processed. You must specify *segment-name* if SQL was selected on the PROCESS statement.

When processing a non-SQL-defined database, do not specify *segment-name*.

Segment-name must be a 1 through 8-character value.

area-name:

Identifies the name of an area to be processed.

Area-name must be a 1 through 18-character value.

PAGE *start-page-n* TO *end-page-n:*

Specifies the range of pages to be processed within the named area. The specified page range must be included within the area page range.

By default, if you do not specify a page range, the entire area will be processed.

If you do specify a page range, be sure that the page range selected includes all database pages that can contain owner or member records of all sets to be processed. Otherwise, they won't be processed.

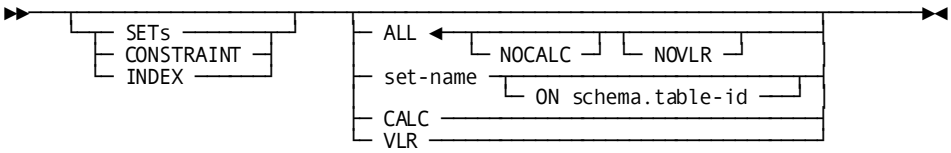
Start-page and *end-page* must contain numeric values and can be from 1 through 10-characters in length.

SET Statement

The SET statement specifies the sets, constraints, or indexes to be processed.

Syntax

Code on one line only.



Parameters

SETs/CONSTRAINT/INDEX

Identifies the sets, constraints, or indexes to process.

The keywords SET, CONSTRAINT, and INDEX are synonyms and can be used interchangeably.

All owners and members must be included in the area to be analyzed and page ranges must be identified in the AREA statement or no sets or constraints are processed.

ALL

Indicates that all of the following sets associated with a specified subschema or segment are to be processed for either a non-SQL or SQL-defined database:

- 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-id* 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).

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).

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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 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

IDMSDBN1 sweeps each specified area. For each area, IDMSDBN1:

- 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	■ Lists input parameters used in the run. ■ Lists all messages issued by IDMSDBN1. The messages: Report errors in the parameter input Trace the processing of areas Define any unexpected conditions detected by the program
#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

As a performance enhancement for the IDMSDBN2 step, the SIZE=E99999999 parameter has been added to the various sorts that are executed. However, some sort packages do not allow the usage of the SIZE= parameter on their SORT statements. If your sort package does allow the specification of the SIZE= parameter, you can reduce the run time of IDMSDBN2 by coding SORTSIZE=ON in the jobstep's SYSIDMS input file.

IDMSDBN2 does the following:

- 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: ■ Trace the processing of the chain file. ■ Report inconsistencies detected during chain processing.
#5: Set Analysis Information	Optional	Provides detailed statistics for each set type processed. ■ Chained sets—The report includes histograms of chain length, of page changes, and of pages used to store the set. ■ Sets of variable-length-record fragments—The report includes a histogram of pages used to store the set. ■ 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.)

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.
  - SYSLST is the destination for the reports the utility creates. If this DD points to a sequential file, pre-allocate the file before the IDMSDBN2 step and ensure the file has a disposition within the step of DISP=MOD. If you fail to do so, Report 1A will not be present on the file at step completion.

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

Example

No REPORT parameter specified

If you run IDMSDBAN with the following input parameters, 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 following input parameters, 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;
```

Sample Output

The following reports are generated when no REPORT parameters are specified as in the first example.

Report 1: Messages Phase I

```
IDMSDBAN - DATA BASE ANALYSIS                    REPORT 1:  MESSAGES                    DATE        TIME      PAGE
                                                                                      mm/dd/yy    20234050       1

PARAMETER CARD:    PROCESS SUBSCHEMA EMPSS01 DBNAME EMPDEMO UNLOCKED
PARAMETER CARD:    AREA EMP-DEMO-REGION
IDMSDBAN - DATA BASE ANALYSIS                    REPORT 1:  MESSAGES                    DATE        TIME      PAGE
                                                                                    mm/dd/yy    20234050       2
```

Report 2: Area

```
 598601 - AREA PROCESSING BEGINNING: EMPDEMO.EMP-DEMO-REGION

 IDMSDBAN - DATA BASE ANALYSIS                        REPORT 2:  AREA DATA                        DATE        TIME      PAGE
                                                                                                              mm/dd/yy    20234050
1

 AREA:      EMPDEMO.EMP-DEMO-REGION

 SUBSCHEMA: EMPSS01

 AREA PAGE RANGE - START PAGE:                                                  75,001
                   END PAGE:                                                    75,100
 PAGES ALLOCATED FOR AREA:                                                       100

 PAGE RANGE ANALYZED - START PAGE:                                              75,001
                       END PAGE:                                                75,100
 PAGES PROCESSED FOR AREA IN THIS RUN:                                           100

 NUMBER OF SMP WITHIN PAGES PROCESSED:                                             1
 PAGE SIZE IN BYTES:                                                           4,276
 MAXIMUM RECORDS ALLOWED PER PAGE:                                               255
 BYTES USED IN PAGES PROCESSED:                                                29,204
 BYTES AVAILABLE IN PAGES PROCESSED:                                          394,120
  NUMBER OF LOGICALLY FULL PAGES:                                                   0
 BYTES AVAILABLE ON LOGICALLY FULL PAGES:                                           0

 NUMBER OF DATA RECORDS IN PAGES PROCESSED:                                      346
 NUMBER OF PAGES WITH EMPTY CALC CHAINS:                                          58
 NUMBER OF RELOCATED RECORDS:                                                      0
 NUMBER OF LOGICALLY DELETED RECORDS:                                              0
 NUMBER OF VLR FRAGMENTS:                                                          0
 NUMBER OF LINE INDEXES FOR PHYSICALLY DELETED RECORDS:                            0
 SPACE AVAILABILITY HISTOGRAM


 PERCENT OF BYTES             PAGES         TOTAL      PERCENT OF     GRAPH
 AVAILABLE ON PAGE                       BYTES (000)  TOTAL PAGES    ...10...20...30...40...50...60...70...80...90..100

       100                      57             242         58        XXXXXXXXXXXXXXXXXXXXXXXXXXX
    90 - 99                     22              86         22        XXXXXXXXXX
    80 - 89                     14              51         14        XXXXXX
    70 - 79                      2               7          2        X
    60 - 69                      2               5          2        X
    50 - 59                      0               0          0
    40 - 49                      1               2          1        /
    30 - 39                      1               1          1        /
    20 - 29                      0               0          0
    10 - 19                      0               0          0
     1 -  9                      0               0          0
  LESS THAN 1                    0               0          0

 AVERAGE BYTES AVAILABLE PER PAGE (ALL PAGES)                 3,981
 AVERAGE BYTES AVAILABLE PER PAGE (NON-EMPTY PAGES)           3,624
```

Report 2: Area

```
IDMSDBAN - DATA BASE ANALYSIS                       REPORT 2:  AREA DATA                          DATE        TIME      PAGE
                                                                                                              mm/dd/yy    20234050
2
0
AREA:      EMPDEMO.EMP-DEMO-REGION         (CONTINUED)

SUBSCHEMA: EMPSS01


DATA RECORDS PER PAGE HISTOGRAM

 DATA RECORDS                PAGES                    PERCENT OF     GRAPH
   PER PAGE                                           TOTAL PAGES    ...10...20...30...40...50...60...70...80...90..100

       0                       57                          58        XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
     1 - 25                    40                          40        XXXXXXXXXXXXXXXXXXXX
    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
                                                                                                               mm/dd/yy    20234050
1
0
SET:         EMP-EMPOSITION       MODE:  CHAIN        ORDER:  FIRST

SUBSCHEMA:   EMPSS01


                NAME         SET MEMBERSHIP   NUMBER OF      EMPTY SETS    UNCONNECTED     LOCATION MODE
                                  TYPE         RECORDS                       MEMBERS

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 4: Record Data

```
 IDMSDBAN - DATA BASE ANALYSIS                    REPORT 4:  RECORD DATA                    DATE        TIME      PAGE
                                                                                                mm/dd/yy    20234050
1
0
 RECORD:        EMPLOYEE          RECORD ID:    415          LOCATION MODE: CALC

 SUBSCHEMA:     EMPSS01
 AREA:          EMPDEMO.EMP-DEMO-REGION
 FLR/VLR:       FLR

 DATA LENGTH:                                                              116
 RECORD LENGTH (DATA AND PREFIX):                                          188

 PAGE RANGE FOR RECORD IN AREA - START PAGE:                            75,003
                     END PAGE:                                          75,100
 PAGES ALLOCATED FOR RECORD IN AREA:                                        98

 PAGE RANGE ANALYZED - START PAGE:                                      75,003
                       END PAGE:                                        75,100
 PAGES PROCESSED FOR RECORD IN AREA IN THIS RUN:                            98

 NUMBER OF SMP WITHIN PAGES PROCESSED:                                       0
 PAGE SIZE IN BYTES:                                                     4,276
 BYTES USED FOR THIS RECORD TYPE IN PAGES PROCESSED:                    10,976
 BYTES AVAILABLE IN PAGES PROCESSED:                                   392,736

 NUMBER OF RECORDS OF THIS TYPE IN PAGES PROCESSED:                         56

 NUMBER OF RELOCATED RECORDS:                                                0
 NUMBER OF LOGICALLY DELETED RECORDS:                                        0
 DATA RECORDS PER PAGE HISTOGRAM


  DATA RECORDS               PAGES                  PERCENT OF     GRAPH
    PER PAGE                                        TOTAL PAGES    ...10...20...30...40...50...60...70...80...90..100

        0                      57                       58        XXXXXXXXXXXXXXXXXXXXXXXXXXXX
      1 - 25                   41                       42        XXXXXXXXXXXXXXXXXXXX
     26 - 51                    0                        0
     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)                                    0
 AVERAGE DATA RECORDS PER PAGE (PAGES WITH THIS RECORD TYPE)                  1
```

Report 1A: Messages Phase II

```
 IDMSDBAN PHASE II - SET ANALYSIS                        REPORT 1A:  MESSAGES                    DATE        TIME      PAGE
                                                                                                             mm/dd/yy    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
                                                                                                          mm/dd/yy   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     GRAPH
                                                     TOTAL SETS     ...10...20...30...40...50...60...70...80...90..100

       0                                  0              0
       2                                 47             84       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
       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
```

More Information

- For more information about database structures analyzed by IDMSDBAN, see the *CA IDMS Database Administration Guide*.
- For more information about database tuning, see the *CA IDMS Database Design Guide*.

IDMSDIRL

The dictionary load utility, IDMSDIRL, loads into a dictionary the components required to describe the dictionary itself and the components that 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.

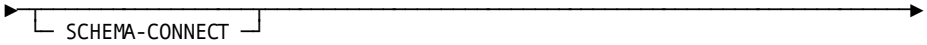
Syntax

DBL-QUOTE statement

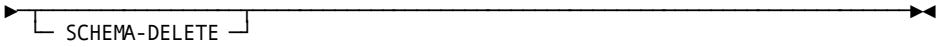
Code on one line only.

SCHEMA-CONNECT statement

Code on one line only.



SCHEMA-DELETE statement



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.

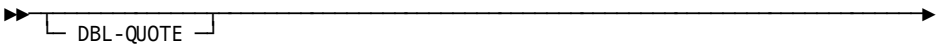
DBL-QUOTE statement

The DBL-QUOTE statement 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

The SCHEMA-CONNECT statement connects the CA-supplied internal schemas to the OOAK-S set.

By default, if you do not specify SCHEMA-CONNECT, the IDMSNTWK, IDMSSECS, and IDMSSECU schemas are not connected to the OOAK-S set.

Syntax

Code on one line only.

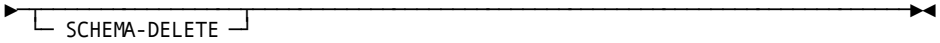
SCHEMA-DELETE statement

The SCHEMA-DELETE statement deletes any CA-supplied internal schemas (IDMSNTWK, IDMSSECS, and IDMSSECU) and associated subschemas from the dictionary.

Any orphaned records and elements are deleted.

The SCHEMA-DELETE statement does not result in a dictionary load.

Syntax



Usage

Input

Input to the IDMSDIRL utility consists of:

- The file containing the definitions to be loaded. This file is supplied on the CA IDMS/DB installation media.
- Optional statements to control the utility processing.

  Each of these statements must be coded on a separate line beginning in column 1.

  There is no end-of-statement terminator for these statements.

Output

The IDMSDIRL utility generates a message listing.

Batch operating mode

You can execute the IDMSDIRL utility either in local mode or under the central version. Local mode is recommended.

When to use IDMSDIRL

IDMSDIRL is typically used to load one data dictionary at a site.

The data dictionary is used by the following CA products:

- CA OLQ, when reporting on the dictionary
- CA Culprit, when producing one of the following reports:
  - CA ADS reports (AReports)
  - 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.

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

Note: For more information about the generic JCL used to execute the batch command facility, see the chapter for your operating system in this guide.

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
```

Sample Output

The IDMSDIRL utility program generates the following listing after processing the previous input.

```
IDMSDIRL - DATA DIRECTORY LOAD UTILITY    RELEASE nn.n volser

OOAK ALREADY EXISTS

END OF IDMS DIRECTORY LOAD - IDMSDIRL
```

More Information

- For more information about data dictionaries and the use of IDMSDIRL, see the *CA IDMS Database Administration Guide*.
- For more information about the records and sets included in the IDMSNTWK schema, see the *CA IDMS Dictionary Structure Reference Guide*.
- For more information about the IDMSDIRL input file, see the *CA IDMS Installation and Maintenance Guide—z/OS*.

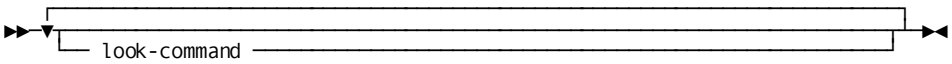
IDMSLOOK

The IDMSLOOK load module print utility reports on the contents of selected load modules. Using IDMSLOOK, you can report on:

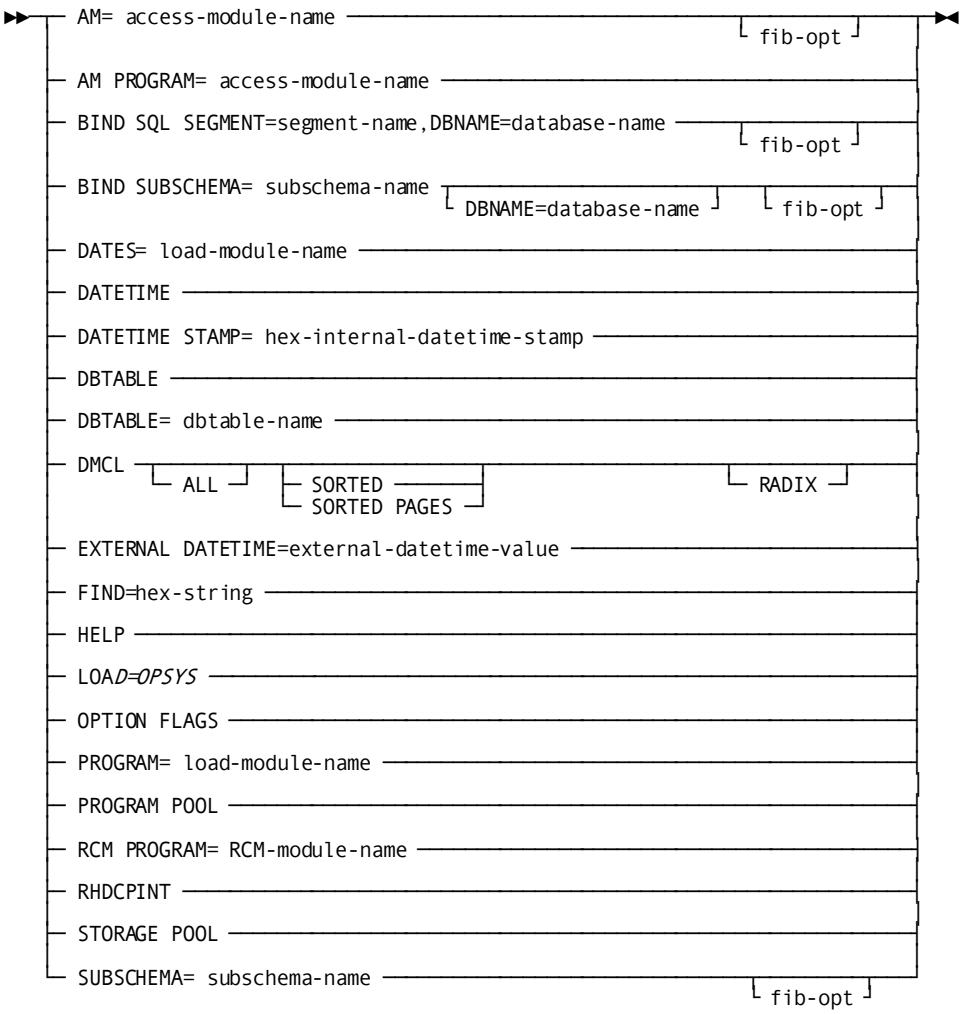
- The contents of a database name table module
- Area, record, and set information in a subschema load module
- Area, record, set, file, and physical characteristics for a subschema bound to a specific database
- File, area, journal, and buffer information in a DMCL load module together with its associated 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
- The contents of the product intent module RHDCPINT

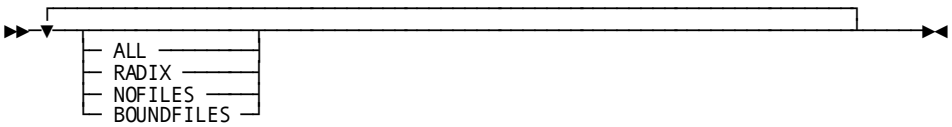
Syntax



Expansion of look-command



Expansion of fib-opt



Input Parameter Statements

ALL

Specifies that some commands display additional information about the DMCL or subschema entities reported. Currently supported by DMCL and FIB related reports.

AM=*access-module-name*

Displays the contents of the RCMs included in an ACCESS MODULE and the IB50 built into the ACCESS MODULE.

AM PROGRAM=

Displays a core dump of an ACCESS module.

BIND SQL SEGMENT=

Displays the logical and physical attributes for areas, tables, constraints, and indexes for a segment of an SQL-defined database. The output is similar to that of the BIND SUBSCHEMA function.

segment-name

Specifies the name of the segment that contains the SQL database areas.

DBNAME=*database-name*

Specifies the name of the database that contains the segment where the catalog for the SQL definitions reside.

BIND SUBSCHEMA=

Displays the logical and physical attributes of the subschema.

subschema-name

Specifies the name of a subschema load module.

DBNAME=

Required unless you are binding to an originally built release 10.x subschema that is being converted to Release 12.0 format.

database-name

Specifies the name of a database.

BOUNDFILES

Specifies that only files connected to bound areas are displayed. This option is available for FIB-related reports.

DATES=

Displays the DATE/TIME stamps of the components of a specified load module.

load-module-name

Specifies the name of a load module.

DATETIME

Displays the current Date/Time.

DATETIME STAMP=

Displays the external value of an internal date/time stamp.

hex-internal-datetime-stamp

The 16 hexadecimal digits that make up the internal representation of the date/time stamp.

DBTABLE

Displays the contents of the default DBNAME table that is associated with the DMCL.

DBTABLE=

Displays the contents of the named DBNAME table.

dbtable-name

Specifies the name of a DB table load module

DMCL

Reports the contents of the current DMCL module.

ALL

Optionally, produces the following information in addition to the standard information 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.

RADIX

Specifies that each area displays the number of bits reserved for a dbkey line number in hex. This replaces the Data Sharing flag value.

EXTERNAL DATETIME=*external-datetime-value*

The 26 characters that make up the external representation of the date/time stamp. The format is yyyy-mm-dd-hh.mm.ss.ffffff.

- yyyy specifies the year. yyyy must be an integer in the range 0001 through 9999.
- mm specifies the month within the year. mm must be an integer in the range 01 through 12.
- dd specifies the day within the month. dd must be an integer in the range 01 through 31.
- hh specifies the hour on a 24-hour clock. hh must be an integer in the range 00 through 23.
- mm specifies the number of minutes past the hour. mm must be an integer in the range 00 through 59.
- ss specifies the number of seconds past the minute. ss must be an integer in the range 00 through 59.
- ffffff specifies the number of millionths of a second past the specified second.

FIND=*hex-string*

Displays the program name and offset into the program where the address was found. Hex-string is 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

Displays the parameters supported by the LOOK task.

LOAD=OPSYS

Specifies that load modules are 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.

NOFILES

Suppresses the display of DMCL files, buffers, journals, and DBTABLE data. This option is available for FIB-related reports.

OPTION FLAGS

Displays all the optional APARs that have been activated in the current RHDCOPTF module.

PROGRAM=

Displays the DATE/TIME stamp of all the components that make up the load module. Also provides a core dump of the load module.

load-module-name

Specifies the name of the load module.

PROGRAM POOL

Displays the contents of the PROGRAM POOL. Shows the program name, entry point address, load address, use count, and size of the program.

RADIX

Specifies that commands that display DMCL area information display the number of bits in a dbkey in hex that is reserved for a dbkey. This replaces the data sharing flag status.

RCM PROGRAM=

Displays a core dump of an RCM MODULE.

RHDCPINT

Displays a list of all products you can license along with the product intent status if a RHDCPINT module is found in the library concatenation.

STORAGE POOL

Displays the contents of the STORAGE POOL. Shows the storage address, storage size, task number that acquired the storage, owner of the storage, and storage type.

SUBSCHEMA=

Displays the logical attributes of the subschema.

subschema-name

Specifies the name of a subschema.

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 can be executed online through the DC task code LOOK.

Note: For more information on using DC tasks, see the *CA IDMS System Tasks and Operator Commands Guide*.

JCL Considerations

For more information about the JCL used to execute IDMSLOOK, see the chapter for your operating system in this guide.

Examples

Requesting subschema information

The following SUBSCHEMA statement 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 following statement directs IDMSLOOK to report all information on a DMCL (page range, size, file mappings, and so on, as well as date and time history) and to sort the information by area name.

```
dmcl all sorted
```

Requesting a hexadecimal dump

The following PROGRAM statement 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=rtpgr001
```

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=yyyy-mm-dd  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
Chain Sorted-> CALC                                    Next,Prior
Owner -------> SR1                                     Next=00  Prior=04
Member ------> SR6                                     Next=00  Prior=04
Member ------> SR7                                     Next=00  Prior=04
                 Ckey  Offset=16   Length=16   Data Type=Character
Member ------> SKILL                                   Next=00  Prior=04
                 Ckey  Offset=20   Length=4    Data Type=Numeric (Unsigned)
Member ------> OFFICE                                  Next=00  Prior=04
                 Ckey  Offset=16   Length=3    Data Type=Character
Member ------> JOB                                     Next=00  Prior=04
                 Ckey  Offset=28   Length=4    Data Type=Numeric (Unsigned)
Member ------> INSURANCE-PLAN                          Next=00  Prior=04
                 Ckey  Offset=8    Length=3    Data Type=Character
Member ------> EMPLOYEE                                Next=00  Prior=04
                 Ckey  Offset=72   Length=4    Data Type=Numeric (Unsigned)
Member ------> DEPARTMENT                              Next=00  Prior=04
                 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
                 Ckey  Offset=86   Length=15   Data Type=Character
                 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                         Next=00  Prior=04  Owner=08
                Ckey  Offset=20   Length=2    Data Type=Character

Index Sorted-> EMP-NAME-NDX                      SR8Next,SR8Prior
Owner -------> SR7                               SR8Next=08  SR8Prior=12
Member ------> EMPLOYEE                          SR8Next=20
                Ckey  Offset=86   Length=15   Data Type=Character
                Ckey  Offset=76   Length=10   Data Type=Character
```

Report requested by DMCL

Note: For data sharing installations, the "Shr" column in the generated report can have one of the following values:

- blank—no data sharing enabled for area.
- D—data sharing has been defined in the DMCL definition of the area, but it is not currently being shared.
- R—data sharing is currently active for the identified area.
- The RADIX option replaces the "Shr" column with a radix value.

```
IDMSLOOK  -  OPSYS=z/OS         Release XX.X Service pack X   tape XXXXXX
DMCL ALL

             DMCL=S74DMCL               Runtime Size---> 13008  (77,832 Bytes)
    This DMCL uses dbtable R120DBTB    Compiled Size---> 0BD7C  (48,508 Bytes)
Date Last Critical Change=2008-04-30  20.34.06    The Operating System is z/OS
Date Created=2007-05-25  10.40.48      Date Last Updated=2008-04-30  18.00.39
    Dynamic File Allocation - on
    Data Sharing is active - on connectivity no abend
    Data Sharing Lock Entries - 4,096
    Data Sharing Members - 4
    Data Sharing Default Shared Cache - IDMSCACHE00002
    Memory Cache Location ANYWHERE Storage Limit OPSYS

                              Page      Low         High      Page
     Area Name            Shr Group     Page        Page      Size    DDNAME
----------------------    --- -----  ----------  ----------  ------  --------

DBCR.BRNCHTEL                    15     680,001     685,012   4,000  BRANCHA
                                                                     BRANCHB
                                                                     BRANCHC
                                                                     BRANCHD
  NETWORK area
  On STARTUP go Update                    On WARMSTART use current status
  Definition date last critical change=2007-05-25  10.40.48
  Page Reserve size      0      Space Management Page Interval       1,984
  Max Records Per Page                                                  255
  Page Range Symbolic is BRNCHTEL           Value is      680,001--->685,012
DBCR.ACCTHIST                    15     690,001     740,040   2,932  ACCOUNTA
                                                                     ACCOUNTB
                                                                     ACCOUNTC
                                                                     ACCOUNTD
                                                                     ACCOUNTE
  NETWORK area
  On STARTUP go Update                    On WARMSTART use current status
  Definition date last critical change=2007-05-25  10.40.48
  Page Reserve size      0      Space Management Page Interval       1,450
  Max Records Per Page                                                  255
  Page Range Symbolic is ACCTHIST           Value is      690,001--->740,040

EMPDEMO.EMP-DEMO-REGION           0      75,001      75,100   4,276  EMPDEMO
  NETWORK area
  On STARTUP go Update                    On WARMSTART use current status
  Definition date last critical change=2007-05-25  10.40.48
  Page Reserve size      0      Space Management Page Interval       2,122
  Max Records Per Page                                                  255
  Page Range Symbolic is EMP-DEMO-REGION    Value is       75,001--->75,100
EMPDEMO.INS-DEMO-REGION           0      75,101      75,150   4,276  INSDEMO
  NETWORK area
  On STARTUP go Update                    On WARMSTART use current status
  Definition date last critical change=2007-05-25  10.40.48
  Page Reserve size      0      Space Management Page Interval       2,122
  Max Records Per Page                                                  255
  Page Range Symbolic is INS-DEMO-REGION    Value is       75,101--->75,150
EMPDEMO.ORG-DEMO-REGION           0      75,151      75,200   4,276  ORGDEMO
  NETWORK area
  On STARTUP go Update                    On WARMSTART use current status
  Definition date last critical change=2007-05-25  10.40.48
  Page Reserve size      0      Space Management Page Interval       2,122
  Max Records Per Page                                                  255
  Page Range Symbolic is ORG-DEMO-REGION    Value is       75,151--->75,200

.
.
.

                                        Mem.- Shared
   File Name                 DDNAME  Type Cache Cache   Buffer Name
--------------------------- -------- ---- ----- ----- ------------------

DBCR.ACCOUNTA               ACCOUNTA BDAM   No    Yes  DBCR_ACCT_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
```

```
DBCR.ACCOUNTB               ACCOUNTB BDAM   No    Yes  DBCR_ACCT_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
DBCR.ACCOUNTC               ACCOUNTC BDAM   No    Yes  DBCR_ACCT_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
DBCR.ACCOUNTD               ACCOUNTD BDAM   No    Yes  DBCR_ACCT_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
DBCR.ACCOUNTE               ACCOUNTE BDAM   No    Yes  DBCR_ACCT_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
DBCR.BRANCHA                BRANCHA  BDAM   No    Yes  DBCR_BRCH_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
DBCR.BRANCHB                BRANCHB  BDAM   No    Yes  DBCR_BRCH_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
DBCR.BRANCHC                BRANCHC  BDAM   No    Yes  DBCR_BRCH_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
DBCR.BRANCHD                BRANCHD  BDAM   No    Yes  DBCR_BRCH_BUFFER

      Definition date last critical change=2007-05-25  10.40.48
EMPDEMO.EMPDEMO             EMPDEMO  BDAM   No    No  DEFAULT_BUFFER
      Definition date last critical change=2007-05-25  10.40.48
EMPDEMO.INSDEMO             INSDEMO  BDAM   No    No  DEFAULT_BUFFER
      Definition date last critical change=2007-05-25  10.40.48
EMPDEMO.ORGDEMO             ORGDEMO  BDAM   No    No  DEFAULT_BUFFER
      Definition date last critical change=2007-05-25  10.40.48

.
.
.

           DMCL Journals         Page Size      # of Pages
           -------------         ----------     ----------

           SYSJRNL1                   2,932         30,000
             Definition date last critical change=2007-05-25  10.40.48
           SYSJRNL2                   2,932         30,000
             Definition date last critical change=2008-04-30  17.57.03
           SYSJRNL                   19,068        Archive

           Journal Buffers      Buffer Size   # of Buffers
           ---------------      -----------   ------------

           JNL_BUFFER                 2,932             80
             Definition date last critical change=2007-05-25  10.40.48


                     Buffer    CV      CV   Total CV     Local  Local  Total Local
DMCL Buffers          Size   Buffers  Type    Size      Buffers  Type    Size
------------         ------   ------   --   ----------   ------   --   ----------

DBCR_BRCH_BUFFER      4,000    5,000   OS   20,000,000    1,000   OS    4,000,000
DBCR_ACCT_BUFFER      2,932    5,000   OS   14,660,000    1,000   OS    2,932,000
LOG_BUFFER            1,076        5   OS        5,380        5   DC        5,380
SCRATCH_BUFFER        2,676        5   OS       13,380        5   DC       13,380
DEFAULT_BUFFER        9,076       30   OS      272,280       20   OS      181,520

                0 Bytes used for CV buffers in DC storage
         94,951,040 Bytes used for CV buffers in OS storage
         94,951,040 Bytes used for CV DMCL Buffers

             18,760 Bytes used for LOCAL buffers in DC storage
         19,113,520 Bytes used for LOCAL buffers in OS storage
         19,132,280 Bytes used for LOCAL DMCL Buffers



      Dbtable=R120DBTB             Compiled Date=2007-05-16  17.18.23
                The DEFAULT Dictionary is TSTDICT
```

```
DBNAME is *DEFAULT  match on subschema is OPTIONAL
    Subschema IDMSNWK? maps to IDMSNWK? using DBNAME ---> TSTDICT
    Subschema EMPSS??? maps to EMPSS??? using DBNAME ---> EMPDEM2
    Subschema EV?????? maps to EV?????? using DBNAME ---> VSAMTDB
    Subschema ET?????? maps to ETSTSUBS using DBNAME ---> ETOTDB
    Subschema DBCR???? maps to DBCR???? using DBNAME ---> DBCR

DBNAME is DBCR      match on subschema is OPTIONAL
             Include SEGMENT ---> DBCR                          0 BIND COUNT

DBNAME is EMPDEM2   match on subschema is OPTIONAL
             Include SEGMENT ---> EMPDEMO                       0 BIND COUNT

.
.
.


IDMSLOOK  -   1 Selection Card  Processed
```

Display Component Module Information

The following LOOK command displays information about the component modules of program IDMSCHDC.

```
LOOK DATES=IDMSCHDC
IDMSLOOK  -  OPSYS=z/OS        Release 18.0 Service pack 0   tape GJI00B
DATES=IDMSCHDC

         was loaded From CDMSLIB DSN --> IDMSNDV.MOTM.IDMS.BASE.P2.LOADLIB
Entry Point Offset +0      -  Reentrant     -  AMODE 31  -  RMODE ANY
       31,016 Bytes in Load Module IDMSCHDC loaded at 38D1CA00

            Module    Offset   Date   Time    FMID      RMID

           IDMSFSED   +18     100111  1645  CAGJI00   CAGJI00
           IDDSFEDC   +6080   100111  1637  CAGJI00   CAGJI00
           IDMSCHPT   +7408   100111  1640  CAGJI00   CAGJI00
           IDMSDATE   +7878   100111  1712  CAGJI00   CAGJI00
```

More Information

- For more information about using DCMT DISPLAY facilities for DMCL, DBTABLE, subschema, and program modules, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about defining DMCLs, DBTABLEs, and subschemas, see the *CA IDMS Database Administration Guide*.

IDMSRPTS

The IDMSRPTS data dictionary reports utility 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.

Full Report Name	Syntax Name	Description
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.

Full Report Name	Syntax Name	Description
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

Full Report Name	Syntax Name	Description
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.

Full Report Name	Syntax Name	Description
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.

Full Report Name	Syntax Name	Description
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

To	You Need This Privilege or Authority
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

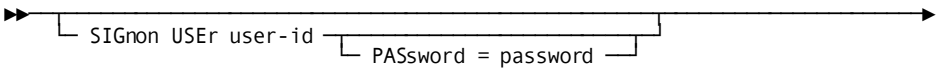
Non-database, schema, and subschema reports

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

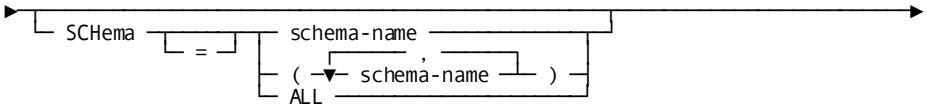
To	You Need This Privilege	On
Run schema and subschema reports	DISPLAY or higher	Database entity on which you want to report if SECURITY FOR IDMS IS ON
Run all reports except User Listing and physical database definition reports	DISPLAY or higher	Entity occurrence on which you want to report if the PUBLIC ACCESS IS NONE

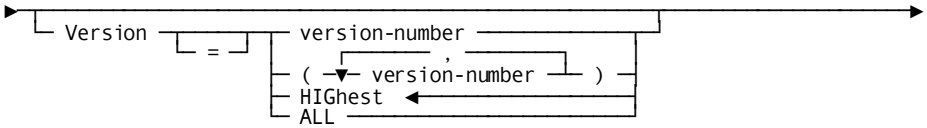
Syntax

SIGNON USER statement

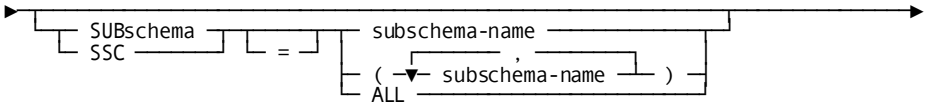


SCHEMA statement

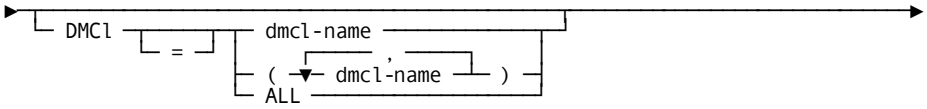




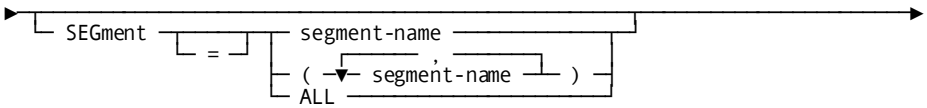
SUBSCHEMA statement



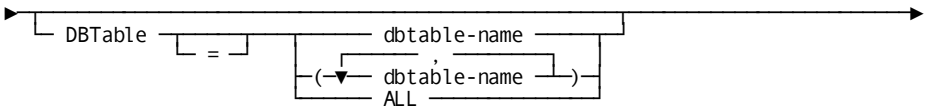
DMCL statement



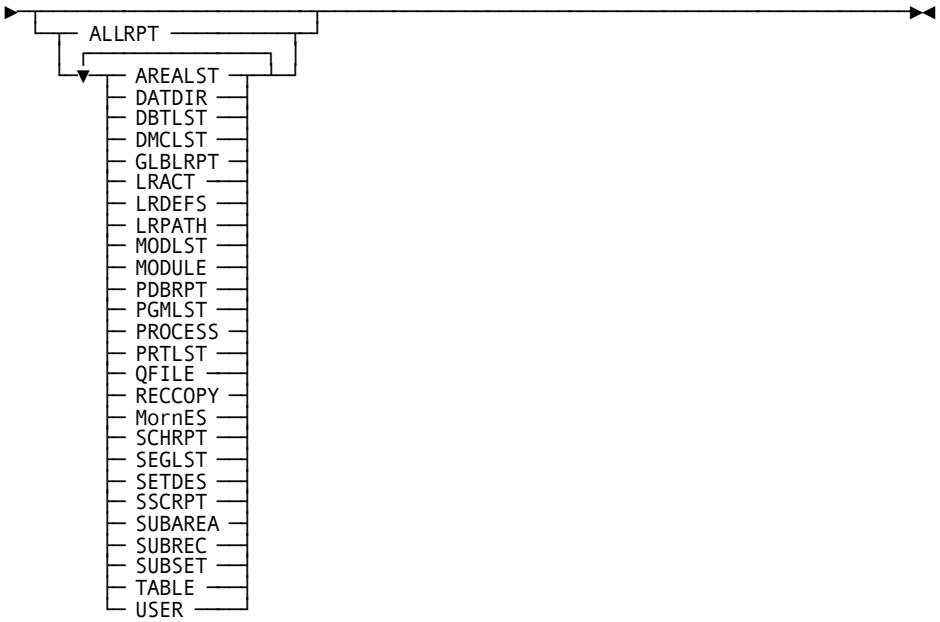
SEGMENT statement



DBTABLE statement



REPORTS SELECTION statement



Statements

Statement descriptions

IDMSRPTS utility processing is controlled by the input statements listed in the following table.

Note: When specifying multiple input statements (other than SIGNON and REPORTS SELECTION), you must specify them in the order listed in the following table.

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.

Statement	Description
DBTABLE	Specifies the database name table(s) to report on.
REPORTS SELECTION	Specifies the report(s) to generate.

SIGNON USER statement

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.

Note: Your external user ID (from the job card) is always used for physical database definition security checking.

If you do not specify a SIGNON USER statement, your external user ID is also used for IDD security checking.

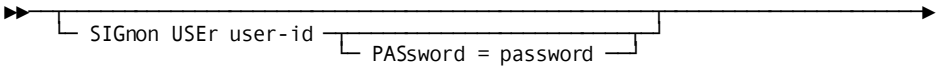
Use the SIGNON USER statement, specifying the user ID and password of an authorized user, if your external user ID is:

- Not defined in the dictionary and IDD SIGNON IS ON
- Is defined in the dictionary but does not have authority to sign on or access the entity-types and 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/DB 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



Parameters

user-id

Identifies a user in the dictionary.

PASsword = ***password***

Specifies 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

The SCHEMA statement identifies the schema(s) to report on.

A schema statement is required if the reports selection includes any of the following:

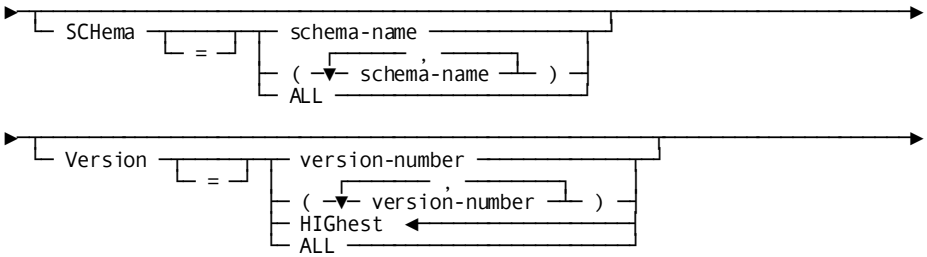
- SCHRPT
- AREALST
- PGMLIST
- RECDES
- SETDES

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

Specifies 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

The SUBSCHEMA statement identifies the subschema(s) to report on.

Both a schema and subschema statement are required if the reports selection includes any of the following:

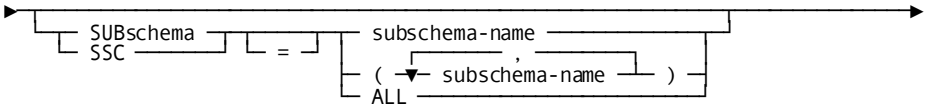
- SSCRPT
- DATDIR
- LRACT
- LRDEFS
- LRPATH
- SUBREC
- SUBSET

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 can be used interchangeably.

subschema-name

Specifies 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 *subschema-name* 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

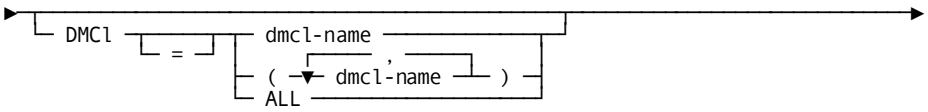
The DMCL statement identifies the DMCL(s) to report on.

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

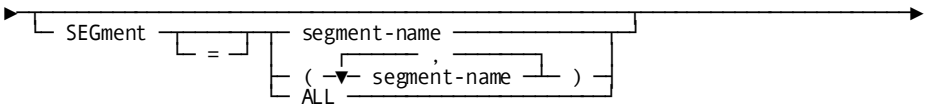
The SEGMENT statement identifies the segment(s) to report on.

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

Specifies 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

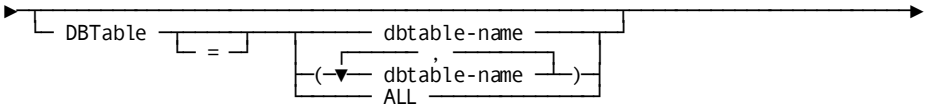
The DBTABLE statement identifies the database name table(s) to report on.

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).

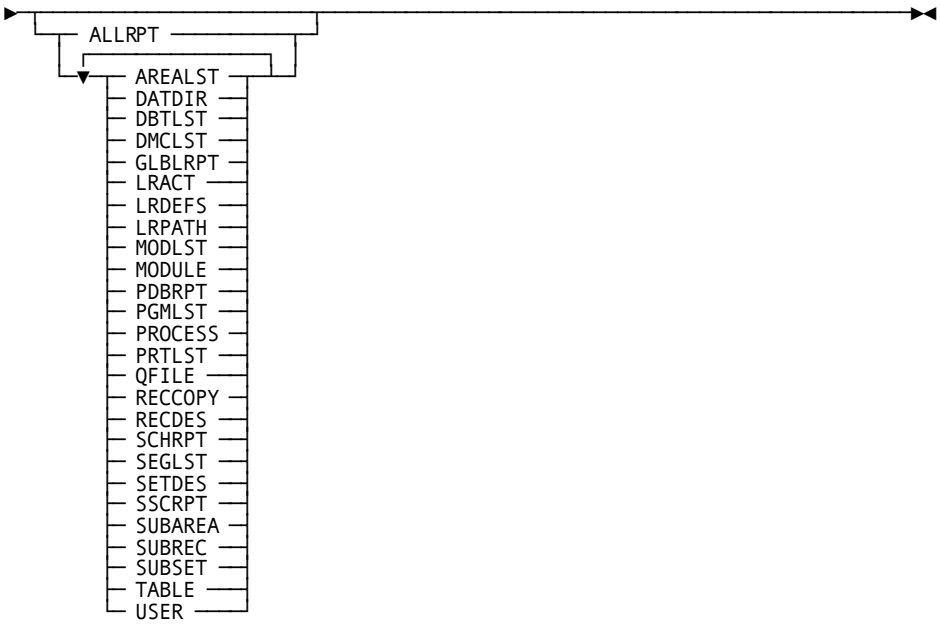
ALL

Directs IDMSRPTS to report on all database name tables defined in the catalog segment of the dictionary.

REPORTS SELECTION statement

The REPORTS SELECTION statement 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.

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.

Note: For a complete description of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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.

JCL Considerations

For more information about the JCL used to execute IDMSRPTS, see the chapter for your operating system in this guide.

Examples

Protocol Listing

The following input parameters direct IDMSRPTS to generate the Protocol Listing.

```
prtlst
```

DMCL, SEGMENT, DBTABLE Listing

The following input parameters 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=idmsdmcl
segment=all
dbtable=all
pdbrpt
```

Schema Record and Set Description Listing

The following input parameters 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 following input parameters 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 EMPSCHM.

```
schema=empschm subschema=all
subarea datdir
```

Sample Output

Protocol Listing

```
                        GLOBAL                          PRTLST
IDMSRPTS nn.n                               ---------- PROTOCOL LISTING ----------                    DATE     TIME   PAGE
PRTLST                                            DICTIONARY SYSTEM   OF NODE DEFAULT                     mm/dd/yy  180854     1

PROTOCOL NAME .. BATCH
PROTOCOL VERSION 0001
DATE CREATED ... mm/dd/yy
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 nn.n                          ------------ DMCL LISTING ------------                    DATE     TIME   PAGE
DMCLST                                   DICTIONARY SYSTEM   OF NODE DEFAULT                    mm/dd/yy  181203     1
                                                    DMCL CVDMCL13

                                      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

    FILE ..... APPLCAT        APPLCAT          1      1000                                          4276

        AREA ..... DDLCAT                      1      1000      0    304001     305000     305000     4276         0

    FILE ..... APPLCATX       APPLCATX         1       500                                          4276

        AREA ..... DDLCATX                     1       500      0    306001     306500     306500     4276         0

    FILE ..... APPLCATL       APPLCATL         1       500                                          4276

        AREA ..... DDLCATLOD                   1       500      0    308001     308500     308500     4276         0

SEGMENT ..... APPLNWK                                                   0

    FILE ..... APPLDML        APPLDML          1      1000                                              4276

        AREA ..... DDLDML                      1      1000      0    300001     301000     301000     4276         0

    FILE ..... APPLLOD        APPLLOD          1       500                                          4276

        AREA ..... DDLDCLOD                    1       500      0    302001     302500     302500     4276         0

SEGMENT ..... ASFNWK                                                    0

    FILE ..... ASFDML         ASFDML           1      1000                                          4276

        AREA ..... DDLDML                  1      1000          0    400001     401000     401000     4276         0

    FILE ..... ASFLOD         ASFLOD           1       500                                          4276

        AREA ..... DDLDCLOD                    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
                          ...
IDMSRPTS nn.n                          ----------- SEGMENT LISTING -----------                    DATE     TIME   PAGE
SEGLST                                   DICTIONARY SYSTEM   OF NODE DEFAULT                    mm/dd/yy  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 ..... DDLCATLOD                                       0    308001     308500     308500     4276         0

        WITHIN FILE .... APPLCATL APPLCATL         1       500          308001                308500     4276
```

```
                        ...
IDMSRPTS nn.n         ----------- DBTABLE LISTING ------------       DATE     TIME   PAGE
DBTLST                  DICTIONARY SYSTEM   OF NODE DEFAULT        mm/dd/yy  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   SYSMSG

    DBNAME ...... ASFDICT    MATCH ON SUBSCHEMA OPTIONAL

        SEGMENT   ASFNWK
        SEGMENT   SYSMSG

    DBNAME ...... DIRLDICT   MATCH ON SUBSCHEMA OPTIONAL

        SEGMENT   DIRLNWK
        SEGMENT   SYSMSG

    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   SYSMSG
        SEGMENT   SYSTEM
                     ...
```

Schema Record and Set Description Listing

```
IDMSRPTS nn.n                          -- SCHEMA RECORD DESCRIPTION LISTING ---                    DATE     TIME   PAGE
RECDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                    mm/dd/yy  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... COVERGE                        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
 .
 .
 .
IDMSRPTS nn.n                          -------- SET DESCRIPTION LISTING -------                    DATE     TIME   PAGE
SETDES                                   DICTIONARY APPLDICT OF NODE DEFAULT                    mm/dd/yy  181426     1
                                             SCHEMA EMPSCHM  VERSION    1

SET..... CALC             MODE CHAIN             ORDER SORTED

OWNER.... SR1              0001 NEXT PRIOR
MEMBER... SYSTEM           0007 NEXT PRIOR       MANDATORY AUTO
MEMBER... DEPARTMENT       0410 NEXT PRIOR       MANDATORY AUTO                                  NAT DUP NOT ALLOW
  IN AREA ORG-DEMO-REGION                             CALC KEY   DEPT-ID-0410                 ASC
MEMBER... EMPLOYEE         0415 NEXT PRIOR       MANDATORY AUTO                                  NAT DUP NOT ALLOW
  IN AREA EMP-DEMO-REGION                             CALC KEY   EMP-ID-0415                   ASC
MEMBER... INSURANCE-PLAN   0435 NEXT PRIOR       MANDATORY AUTO                                  NAT DUP NOT ALLOW
  IN AREA INS-DEMO-REGION                             CALC KEY   INS-PLAN-CODE-0435            ASC
MEMBER... JOB              0440 NEXT PRIOR       MANDATORY AUTO                                  NAT DUP NOT ALLOW
  IN AREA ORG-DEMO-REGION                             CALC KEY   JOB-ID-0440                   ASC
MEMBER... OFFICE           0450 NEXT PRIOR       MANDATORY AUTO                                  NAT DUP NOT ALLOW
  IN AREA ORG-DEMO-REGION                             CALC KEY   OFFICE-CODE-0450              ASC
```

```
MEMBER... SKILL            0455 NEXT PRIOR       MANDATORY AUTO                                  NAT DUP NOT ALLOW
  IN AREA ORG-DEMO-REGION                              CALC KEY   SKILL-ID-0455                 ASC

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   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 nn.n          -- SUBSCHEMA DATA DIRECTORY LISTING --        DATE     TIME   PAGE
DATDIR                  DICTIONARY APPLDICT OF NODE DEFAULT        mm/dd/yy  181529     1
              SUBSCHEMA EMPHTL1  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 nn.n          -- SUBSCHEMA DATA DIRECTORY LISTING --        DATE     TIME   PAGE
DATDIR                  DICTIONARY APPLDICT OF NODE DEFAULT        mm/dd/yy  181529     2
              SUBSCHEMA EMPHTL1  OF SCHEMA EMPSCHM  VERSION  100

RECORD:      COVERGE                           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'
  COVFAMLY                         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'
 .
 .
 .
```

Note: For more information about CA reports produced by IDMSRPTS, see the *CA IDMS Reports Guide*.

IDMSRSTC

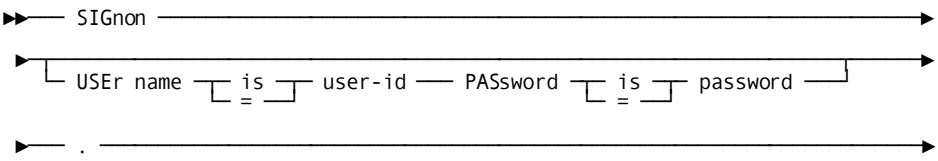
The IDMSRSTC schema compare utility generates IDMSRSTT macro statements for use in a database restructure operation. IDMSRSTC generates the statements by comparing two schemas:

- **An old schema** that describes the database before restructuring
- **A new schema** that describes the database after restructuring

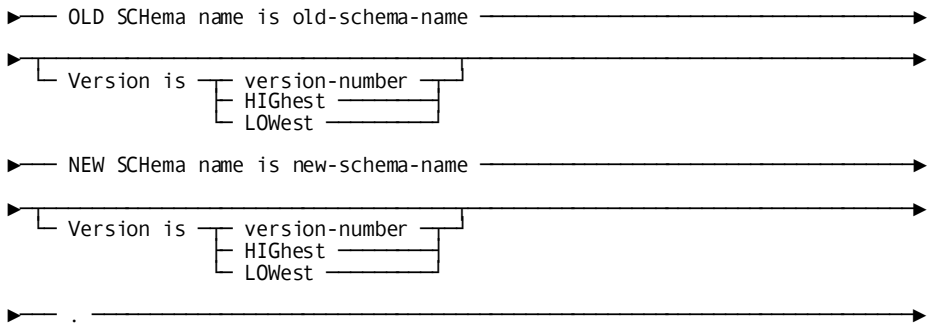
The utility reads the schema definitions from the data dictionary.

Syntax

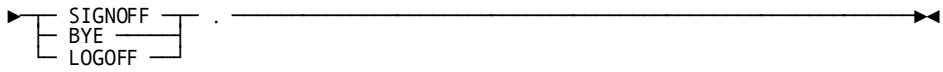
SIGNON statement



SCHEMA statement



SIGNOFF statement



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.

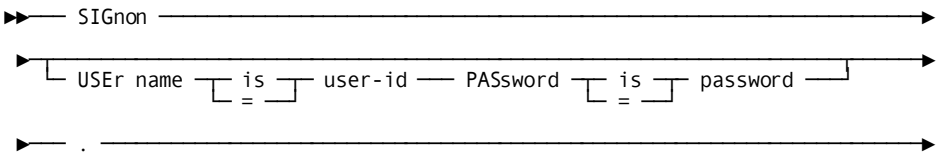
SIGNOFF statement

The SIGNOFF statement must be the last parameter statement.

SIGNON statement

The SIGNON statement initiates IDMSRSTC processing and optionally specifies a user identifier and password.

Syntax



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.

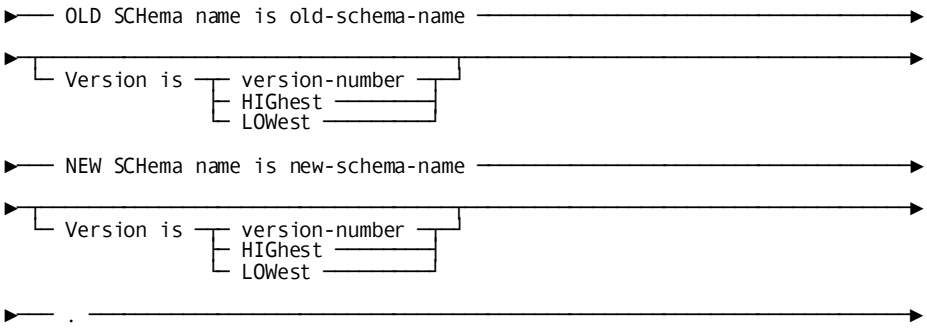
You must include the PASSWORD parameter if the user specified in the USER parameter is associated with a password in the dictionary, unless the USER ID is the same as the user known to the execution environment. In this case, the password is ignored.

IS and = are synonyms and can be used interchangeably.

SCHEMA statement

The SCHEMA statement 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.

LOWest

Directs IDMSRSTC to use the lowest version number assigned to the named schema.

NEW SCHema name is *new-schema-name*

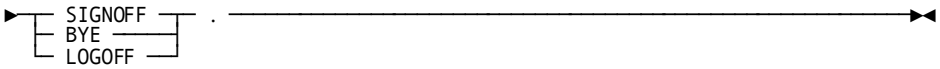
Specifies the name of a schema that describes the database after restructuring. *New-schema-name* must identify a schema defined in the data dictionary.

SIGNOFF statement

The SIGNOFF statement terminates IDMSRSTC processing.

SIGNOFF, BYE, and LOGOFF are synonyms and can be used interchangeably.

Syntax



Usage

Input

Input to the IDMSRSTC utility consists only of statements to control the utility processing.

Output

The IDMSRSTC utility generates:

- A card-image file containing the IDMSRSTT macro statements with comments and error messages
- A formatted listing that duplicates the information in the card-image file

Execution mode

You can execute the IDMSRSTC utility either in local mode or under the central version.

Note: For more information about the JCL used to execute IDMSRSTC, see the chapter for your operating system in this guide.

Specifying dictionary name and nodename

To override the default dictionary name or dictionary nodename that the IDMSRSTC utility will access, use the SYSIDMS DICTNAME and DICTNODE parameters. For a complete description of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

Review the IDMSRSTT macro statements

You should review the IDMSRSTT macro statements generated by IDMSRSTC before assembling them into a base restructuring table. Informational and warning messages included in the IDMSRSTC listing indicate statements that might require modification. You will also need to modify the macro statements under the following circumstances:

Circumstance	Modification
A database procedure not included in the new schema is to be executed during the run of the RESTRUCTURE 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.

Note: 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 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 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.

Example

If you execute IDMSRSTC with the input statements shown next, 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.
```

Sample Output

The IDMSRSTC utility generates the following report after executing the parameters in the previous example.

```
 IDMSRSTC  nn.n                                 CA                                DATE           TIME          PAGE
 volser                        RESTRUCTURE SCHEMA COMPARE ACTIVITY LIST          mm/dd/yy       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                                                                              WORD  2
 IDMSRSTC  nn.n                                   CA                              DATE           TIME          PAGE
 volser                        RESTRUCTURE SCHEMA COMPARE ACTIVITY LIST          mm/dd/yy       16411366       0002

              ** TRANSACTION SUMMARY **
0ENTITY               ADD MODIFY REPLACE DELETE DISPLAY
 .................... ... ...... ....... ...... .......

 SCHEMA                 0      0       0      0       1

 NO ERRORS OR WARNINGS ISSUED FOR THIS COMPILE
```

More Information

- For more information about the IDMSRSTT macro statements, see Appendix B, IDMSRSTT Macro Statements.
- For more information about restructuring a database, see RESTRUCTURE (see page 298) and RESTRUCTURE CONNECT.

Chapter 7: z/OS JCL

This section contains the following topics:

Overview (see page 437)
Batch Command Facility (see page 437)
Utility Statements (see page 440)
Utility Programs (see page 475)

Overview

This chapter presents sample z/OS JCL used to run CA IDMS utility statements and programs.

Statements common to utilities that use the Batch Command Facility are presented first. Additional required statements to run each utility statement and program are presented next, alphabetically, by utility.

Batch Command Facility

The following z/OS JCL sample is used to execute the Batch Command Facility (IDMSBCF) for CA IDMS.

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.

Note: For more information about the Command Facility, see the *CA IDMS Common Facilities Guide*.

Local mode IDMSBCF (z/OS)

```
//         EXEC PGM=IDMSBCF,REGION=2048K
//STEPLIB  DD  DSN=idms.dba.loadlib,DISP=SHR
//         DD  DSN=idms.custom.loadlib,DISP=SHR
//         DD  DSN=idms.cagjload,DISP=SHR
//dcmsg    DD  DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,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 might be needed for the user catalog and the system dictionary depending on your security implementation.

Note: DD statements for database and native VSAM files can be omitted if data set 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.custom.loadlib	Data set name of the load library containing customized CA IDMS system software modules
idms.cagjload	Data set name of the load library containing vanilla 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
sysjrnl	DDname of the tape journal file, if one is defined in the DMCL

idms.tapejrnl	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. **Note:** For a complete description of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

Central version IDMSBCF (z/OS)

```
//          EXEC PGM=IDMSBCF,REGION=2048K
//STEPLIB   DD  DSN=idms.dba.loadlib,DISP=SHR
//          DD  DSN=idms.custom.loadlib,DISP=SHR
//          DD  DSN=idms.cagjload,DISP=SHR
//SYSCTL    DD  DSN=idms.sysctl,DISP=SHR
//dcmsg     DD  DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl   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 data set 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.custom.loadlib	Data set name of the CA IDMS/DB load library containing customized load modules
idms.cagjload	Data set name of the CA IDMS/DB load library containing vanilla load modules
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area

sysjrnl	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. **Note:** For a complete description of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

Utility Statements

ARCHIVE JOURNAL

DD statements for the batch command facility (z/OS)

```
//j1jrnl   DD  DSN=idms.j1jrnl,DISP=SHR
//j2jrnl   DD  DSN=idms.j2jrnl,DISP=SHR

Additional journal file assignments, as required

//archjrnl DD  DSN=idms.archive,DISP=(NEW,KEEP),
//            UNIT=tape,VOL=SER=nnnnnn
//*NOTE: The blksize and lrecl are set by the
//*      blksize that is defined in the DMCL.
//*      The file record format is fixed.

Additional archive journal file assignments, as required
```

j1jrnl	DDname of the first disk journal file, as defined in the DMCL
idms.j1jrnl	Data set name of the first disk journal file
j2jrnl	DDname of the second disk journal file, as defined in the DMCL
idms.j2jrnl	Data set name of the second disk journal file
archjrnl	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

ARCHIVE LOG

DD statements for the batch command facility (z/OS)

```
//dlogdb     DD  DSN=idms.dlogdb,DISP=SHR

//dmsgdb     DD  DSN=idms.dmsgdb,DISP=SHR
//sysjrnl    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)

ARCHIVE TRACE

The CA IDMS installation media contains the following sample JCL (see page 23) (intended for demonstration purposes only):

- ATRCJCL—Contains a skeleton of the JCL required to run the ARCHIVE TRACE utility.

BACKUP

DD statements for the batch command facility (z/OS)

```
//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: ■ The size of the largest page being backed up, plus 8 ■ 32,760

BUILD

DD statements for the batch command facility (z/OS)

```
//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 LOAD (see page 136)
user.build *	Data set name of the output (SYS003) file; for sizing information see BUILD (see page 59)
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.

CLEANUP

DD statements for the batch command facility (z/OS)

```
//sysjrnl  DD  DSN=idms.tapejrnl,DISP=SHR

//userdb   DD  DSN=user.userdb,DISP=SHR

Additional database file assignments, as required
```

sysjrnl	DDname of the tape journal file
idms.tapejrnl	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

Central version

To execute CLEANUP SEGMENT under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

CONVERT CATALOG

DD statements for the batch command facility (z/OS)

```
//ddlcat    DD        DSN=ucat.ddlcat,DISP=OLD
//ddlcatx   DD        DSN=ucat.ddlcatx,DISP=OLD
//j1jrnl    DD        DSN=tape.j1jrnl,(NEW,disp)

Additional journal file assignments, as required.
```

ddlcat	DDname of the database file containing the DDLCAT area of the dictionary to be converted
ucat.ddlcat	Data set name of the database file containing the DDLCAT area of the dictionary to be converted
ddlcatx	DDname of the database file containing the DDLCATX area of the dictionary to be converted
ucat.ddlcatx	Data set name of the database file containing the DDLCATX area of the dictionary to be converted
j1jrnl	DDname of the first journal file, as defined in the DMCL
tape.j1jrnl	Data set name of the first journal file
disp	Disposition of the first journal file

Central version

To execute CONVERT CATALOG under the central version, modify the previous JCL as follows:

- Optionally remove any journal and dictionary DD statements.
- Insert the following statement:

```
//SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

CONVERT PAGE

DD statements for the batch command facility (z/OS)

```
//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

EXPAND PAGE

DD statements for the batch command facility (z/OS)

```
//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

EXTRACT JOURNAL

DD statements for the batch command facility (z/OS)

```
//SYS001   DD  DSN=&.&archive.,DISP=(OLD)

//extract   DD  DSN=jrnl.extract,DISP=(NEW,CATLG),
//              DCB=(RECFM=VB,BLKSIZE=bbbbb),
//              UNIT=nnnn,VOL=SER=nnnnnn
//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
bbbbb	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.

FASTLOAD

DD statements for the batch command facility (z/OS)

```
//SYS001    DD  DSN=&.&sortunld.,DISP=SHR

//SYS002    DD  DSN=user.dbl001,DISP=(OLD,DELETE)
//SYS004    DD  DSN=user.dbl004,DISP=(NEW,PASS),UNIT=tape004,
//              DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004)
//SYS005    DD  DSN=user.dbl005,DISP=(NEW,PASS),UNIT=tape005,
//              DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS009    DD  DSN=user.dbl009,DISP=(NEW,PASS),UNIT=tape009,
//              DCB=(RECFM=VB,LRECL=rrr009,BLKSIZE=bbb009)
//SYS010    DD  DSN=user.dbl010,DISP=(NEW,PASS),UNIT=tape010,
//              DCB=(RECFM=VB,LRECL=rrr010,,BLKSIZE=bbb010)
//SYS011    DD  DSN=user.dbl011,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=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    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 FASTLOAD (see page 97)
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 FASTLOAD (see page 97)
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

FIX ARCHIVE

DD statements for the batch command facility (z/OS)

```
//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

FIX PAGE

DD statements for the batch command facility (z/OS)

```
//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

Central version

To execute FIX PAGE under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

  ```
  //SYSCTL    DD  DSN=idms.sysctl,DISP=SHR
  ```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

FORMAT

DD statements for the batch command facility (z/OS)

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:

```
//j1jrnl    DD  DSN=idms.j1jrnl,DISP=(NEW,disp),
//                    UNIT=disk,VOL=SER=nnnnnn,
//                    SPACE=(space)
```

To reformat an existing disk journal file:

```
//j1jrnl    DD  DSN=idms.j1jrnl,DISP=(OLD,PASS)
```

```
Additional database and journal file assignments, as required
```

To format a new SYSTRK file:

```
//anydd    DD  DSN=user.systrkn,DISP=(NEW,disp),
//                    UNIT=disk,VOL=SER=nnnnnn,
//                    SPACE=(space)
```

To reformat an existing SYSTRK file:

```
//anydd    DD  DSN=user.systrkn,DISP=(OLD,PASS)
```

userdb	DDname of the database file
user.userdb	Data set name of the database file
disp	Disposition of the new file (CATLG, KEEP, or PASS)
disk	Symbolic device name of the file being formatted
nnnnnn	Volume serial number of the file being formatted
space	Space allocation for the file being formatted
j1jrnl	DDname of the disk journal file
idms.j1jrnl	Data set name of the disk journal file

anydd	Target DDname specified in the FORMAT SYSTRK statement
user.systrkn	Data set name of the SYSTRK file

Central version

To execute FORMAT AREA and FORMAT SEGMENT under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

INSTALL STAMPS

Local mode DD statements for the batch command facility (z/OS)

```
//userdb   DD  DSN=user.userdb,DISP=OLD

//userdict DD  DSN=user.userdict,DISP=SHR
//sysjrnl  DD  DSN=idms.sysjrnl,DISP=OLD
```

userdb	DDname of the database file
user.userdb	Data set name of the database file
userdict	DDname of the database file containing the dictionary with the table definitions
sysjrnl	DDname of the tape journal file
idms.sysjrnl	Data set name of the tape journal file

Central version

To execute INSTALL STAMPS under the central version, modify the JCL shown previously 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
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

LOAD

DD statements for the batch command facility (z/OS)

```
//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 LOAD (see page 136)
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 LOAD (see page 136)
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.

LOCK

DD statements for the batch command facility (z/OS)

```
//userdb    DD  DSN=user.userdb,DISP=SHR
```

userdb	DDname of the database file
user.userdb	Data set name of the database file

Central version

To execute LOCK AREA under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

MAINTAIN INDEX

DD statements for the batch command facility (z/OS)

```
//userdb   DD  DSN=user.olddb,DISP=SHR

Additional existing database files assignments, as required

//SYS003   DD  DSN=user.dbl003,DISP=(NEW,PASS),UNIT=tape003,

//             DCB=(RECFM=VB,LRECL=rrr003,BLKSIZE=bbb003)
//SYS004   DD  DSN=user.dbl004,DISP=(NEW,PASS),UNIT=tape004,
//             DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004)
//SYS005   DD  DSN=user.dbl005,DISP=(NEW,PASS),UNIT=tape005,
//             DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS006   DD  DSN=user.dbl006,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.dbl003	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 MAINTAIN INDEX (see page 157) for sizing information.
bbb003	Block size of the SYS003 intermediate work file
user.dbl004	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.dbl005	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 MAINTAIN INDEX (see page 157) for sizing information
bbb005	Block size of the SYS005 intermediate work file
user.dbl006	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.

MERGE ARCHIVE

DD statements for the batch command facility (z/OS)

```
//SYS001  DD  DSN=idms.tjrnlold,DISP=(OLD,PASS),UNIT=tapein
//SYS002  DD  DSN=idms.tjrnlfix,DISP=(NEW,PASS),UNIT=tapeout,
//            DCB=BLKSIZE=bbbb
//JRNM01  DD  DSN=idms.tmrgold,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.tmrgold	Data set name of the input tape merged file. If none exists yet, specify //JRNM01 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).

PRINT INDEX

DD statements for the batch command facility (z/OS)

```
//sysjrnl  DD  DUMMY

//userdb   DD  DSN=user.userdb,DISP=SHR
```

sysjrnl	DDname of the dummy journal file
userdb	DDname of the database file
user.userdb	Data set name of the database file

PRINT JOURNAL

DD statements for the batch command facility (z/OS)

```
//SYS001    DD  DSN=idms.archjrnl,DISP=SHR
```

idms.archjrnl	Data set name of the archive journal file

PRINT LOG

DD statements for the batch command facility (z/OS)

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
//sysjrnl    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
sysjrnl	DDname of the tape journal file defined in the DMCL module

PRINT PAGE

DD statements for the batch command facility (z/OS)

```
//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

Central version

To execute PRINT PAGE under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

  ```
  //SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
  ```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

PRINT SPACE

DD statements for the batch command facility (z/OS)

```
//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

Central version

To execute PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

PRINT TRACE

The CA IDMS installation media contains the following sample JCL (see page 23) (intended for demonstration purposes only):

- PTRCJCL – Contains a skeleton of the JCL required to run the PRINT TRACE utility.

PUNCH

Local mode DD statements for the batch command facility (z/OS)

```
//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
//sysjrnl   DD  DSN=DUMMY
```

usercat	DDname of the database file containing the DDLCAT area of the dictionary
user.ddlcat	Data set 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	Data set 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	Data set 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 previously 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

RELOAD

DD statements for the batch command facility (z/OS)

```
//SYS001   DD  DSN=&&sortunld,DISP=SHR
//SYS002   DD  DSN=user.dbl002,DISP=(OLD,DELETE)
//SYS003   DD  DSN=user.dbl003,DISP=(OLD,DELETE)
//SYS004   DD  DSN=user.dbl004,DISP=(NEW,PASS),UNIT=tape004,
//             DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004,)
//SYS005   DD  DSN=user.dbl005,DISP=(NEW,PASS),UNIT=tape005,
//             DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS006   DD  DSN=user.dbl006,DISP=(NEW,PASS),UNIT=tape006,
//             DCB=(RECFM=VB,LRECL=rrr006,BLKSIZE=bbb006)
//SYS007   DD  DSN=user.dbl007,DISP=(NEW,PASS),UNIT=tape007,
//             DCB=(RECFM=VB,LRECL=rrr007,BLKSIZE=bbb007)
//SYS008   DD  DSN=user.dbl008,DISP=(NEW,PASS),UNIT=tape008,
//             DCB=(RECFM=VB,LRECL=rrr008,BLKSIZE=bbb008)
//SYS009   DD  DSN=user.dbl009,DISP=(NEW,PASS),UNIT=tape009,
//             DCB=(RECFM=VB,LRECL=rrr009,BLKSIZE=bbb009)
//SYS010   DD  DSN=user.dbl010,DISP=(NEW,PASS),UNIT=tape010,
//             DCB=(RECFM=VB,LRECL=rrr010,BLKSIZE=bbb010)
//SYS011   DD  DSN=user.dbl011,DISP=(NEW,PASS),UNIT=tape011,
//             DCB=(RECFM=VB,LRECL=rrr011,BLKSIZE=bbb011)
//SYSPCH   DD  DSN=&&sortreld,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
```

&&sortunld	Data set name of the sort parameters created by UNLOAD
user.dbl002	Data set name of the file put out by UNLOAD as SYS002
user.dbl003	Data set name of the file put out by UNLOAD as SYS003
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 SYS002 used by UNLOAD
bbb004	Block size of the SYS004 intermediate work file; should be the same as the block size for SYS002 used by UNLOAD
user.dbl005	Data set name of the intermediate work file containing member descriptors for chained sets from IDMSDBL2
tape005	Symbolic device name of the SYS005 file
rrr005	Record size of the SYS005 file; for sizing information, see RELOAD (see page 231)
bbb005	Block size of the SYS005 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
user.dbl006	Data set name of the intermediate work file containing member descriptors for user owned index sets from IDMSDBL2
tape006	Symbolic device name of the SYS006 file
rrr006	Record size of the SYS006 file; for sizing information, see RELOAD (see page 231)
bbb006	Block size of the SYS006 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
user.dbl007	Data set name of the intermediate work file containing the output of SORT2
tape007	Symbolic device name of the SYS007 file
rrr007	Record size of the SYS007 file; should be the same as the larger of: ■ the record size for SYS003 used by UNLOAD ■ the record size for SYS006
bbb007	Block size of the SYS007 intermediate work file; must be at least as large as the larger of: ■ the block size for SYS003 used by UNLOAD ■ the block size for SYS006
user.dbl008	Data set name of the intermediate work file containing the reformatted index information from IDMSDBLX
tape008	Symbolic device name of the SYS008 file
rrr008	Record size of the SYS008 file; should be the same as rrr007

bbb008	Block size of the SYS008 intermediate work file; should be the same as bbb007
user.dbl009	Data set name of the intermediate work file containing the sorted index set descriptors from SORT3
tape009	Symbolic device name of the SYS009 file
rrr009	Record size of the SYS009 file; should be the same as the larger of: ■ the record size for SYS005 ■ the record size for SYS008
bbb009	Block size of the SYS009 intermediate work file; must be at least as large as the larger of: ■ the block size for SYS003 used by UNLOAD ■ the block size for SYS006
user.dbl010	Data set name of the intermediate work file containing prefix pointer information from IDMSDBL3
tape010	Symbolic device name of the SYS010 file
rrr010	Record size of the SYS010 file; for sizing information, see RELOAD (see page 231)
bbb010	Block size of the SYS010 intermediate work file; must be at least 44 bytes (the size of a pointer descriptor plus four bytes)
user.dbl011	Data set name of the intermediate work file containing sorted prefix 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
&&sortreld	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 must 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

REORG

DD statements for the batch command facility (z/OS)

When allocating the REORG control file or when running setup only

```
//RORGCTL  DD DSN=user.reldctl,DISP=(NEW,CATLG),
//            UNIT=disk,VOL=SER=nnnnnn,
//            DCB=(RECFM=FB,LRECL=8192,BLKSIZE=8192)
```

Manual creation of work files other than DBKEYS files

```
//wxnnnnn  DD DSN=user.workfile,DISP=(NEW,CATLG),
//            UNIT=disk,VOL=SER=nnnnnn,
//            DCB=(RECFM=VB,BLKSIZE=wrkbbb)
```

Manual creation of DBKEYS work files

```
//WDnnnnn  DD DSN=user.workfile,DISP=(NEW,CATLG),
//            UNIT=disk,VOL=SER=nnnnnn,
//            DCB=(RECFM=FB,LRECL=16,BLKSIZE=dbkbbb)
```

Manual references to work files

```
//wxnnnnn  DD DSN=user.workfile,DISP=SHR
```

File assignments when running the unload phase

```
//RORGCTL  DD DSN=user.reldctl,DISP=SHR
//RORGJCL  DD DSN=user.jclfile,DISP=SHR
//unlddb   DD DSN=user.unldfile,DISP=SHR
```

Additional database file assignments, as required if not using dynamic allocation

Manual references to work files if not using dynamic allocation

File assignments when running the reload phase

```
//RORGCTL  DD DSN=user.reldctl,DISP=SHR
//RORGJCL  DD DSN=user.jclfile,DISP=SHR
//relddb   DD DSN-user.reldfile,DISP=SHR

Additional database file assignments, as required if not using
dynamic allocation

Manual references to work files if not using dynamic
allocation

//SORTMSG  DD SYSOUT=A
//SORTWK01 DD UNIT=disk,SPACE=(CYL,(nnn,nnn))

Additional SORTWKnn files as necessary
```

user.reldctl	Data set name of the REORG control file containing control information
user.jclfile	Data set name of the file containing JCL for automatic job submission
wxnnnnn	DDname for a work file. It must match the name generated in the Unload/Reload Work File Summary report.
user.workfile	Data set name of a work file when manually allocating work files
wrkbbb	Blocksize for a variable blocked sequential work file
WDnnnnn	DDname for a DBKEYS file. It must match the name generated in the Unload/Reload Work File Summary report.
dbkbbb	Blocksize for a fixed-blocked, sequential DBKEYS file. Must be a multiple of 16.
disk	Symbolic unit name for a device
nnnnnn	Volume serial number
unlddb	DDname of the existing database file
user.unldfile	Data set name of the existing database file
relddb	DDname of the target database file
user.reldfile	Data set name of the target database file

RESTORE

DD statements for the batch command facility (z/OS)

```
//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
```

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
user.bkp	Data set name of the backup file

RESTRUCTURE

DD statements for the batch command facility (z/OS)

```
//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 40 with a maximum size of 32,760.

RESTRUCTURE CONNECT

DD statements for the batch command facility (z/OS)

```
//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
```

userdb	DDname of the database file
user.userdb	Data set name of the database file
idms.spill	Data set name of the spill file.
bbbb	Block size of the spill file; it should be a multiple of 40 with a maximum size of 32,760

ROLLBACK

DD statements for the batch command facility (z/OS)

```
//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

ROLLFORWARD

DD statements for the batch command facility (z/OS)

```
//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=jrnl.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
```

userdb	DDname of the user database file.
user.userdb	Data set name of the user database file.
&.&archive.	Data set name of the complete archive or tape journal file.
tape	Symbolic device name of the archive or tape journal file.
extract	DDname of the extract journal file. If not specified in the command it will default to SYS002.
jrnl.extract	Data set name of the extract journal file.

SYNCHRONIZE STAMPS

Local mode DD statements for the batch command facility (z/OS)

```
//userdb    DD  DSN=user.userdb,DISP=OLD

//userdict DD  DSN=user.userdict,DISP=SHR
//sysjrnl   DD  DSN=idms.sysjrnl,DISP=OLD
```

userdb	DDname of the database file
user.userdb	Data set name of the database file

userdict	DDname of the database file containing the dictionary with the table definitions
sysjrnl	DDname of the tape journal file
idms.sysjrnl	Data set name of the tape journal file

Central version

To execute SYNCHRONIZE STAMPS under the central version, modify the JCL shown previously 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

TUNE INDEX

DD statement for the batch command facility (z/OS)

```
//userdb   DD   DSN=user.userdb,DISP=SHR
```

userdb	DDname of the database file
user.userdb	Data set name of the database file

Central version

To execute TUNE INDEX under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

UNLOCK

DD statements for the batch command facility (z/OS)

```
//userdb    DD  DSN=user.userdb,DISP=SHR
```

userdb	DDname of the database file
user.userdb	Data set name of the database file

UNLOAD

DD statements for the batch command facility (z/OS)

```
//userdb    DD  DSN=user.olddb,DISP=SHR

Additional existing database files assignments, as required

//sysjrnl   DD  DUMMY
//SYS002    DD  DSN=user.dbl002,DISP=(NEW,PASS),UNIT=tape002,
//              DCB=(RECFM=VB,BLKSIZE=bbb002)
//SYS003    DD  DSN=user.dbl003,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
sysjrnl	DDname of the dummy journal file
user.dbl002	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.dbl003	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.

UPDATE STATISTICS

DD statements for the batch command facility (z/OS)

```
//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
//j1jrnl    DD  DSN=tape.j1jrnl,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
j1jrnl	DDname of the first journal file, as defined in the DMCL
tape.j1jrnl	Data set name of the first journal file
disp	The disposition of the first journal file

Central version

To execute UPDATE STATISTICS under the central version, modify the previous JCL as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
```

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

VALIDATE

DD statements for the batch command facility (z/OS)

```
//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 VALIDATE (see page 359)
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 VALIDATE (see page 359)
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.

Utility Programs

IDMSDBAN

Local mode IDMSDBAN (z/OS)

```
//DBN1     EXEC PGM=IDMSDBN1,REGION=region-size
//STEPLIB  DD  DSN=idms.dba.loadlib,DISP=SHR
//         DD  DSN=idms.custom.loadlib,DISP=SHR
//         DD  DSN=idms.cagjload,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.custom.loadlib,DISP=SHR
//         DD  DSN=idms.cagjload,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.custom.loadlib	Data set name of the CA IDMS/DB load library containing customized load modules, including customized subschemas
idms.cagjload	Data set name of the CA IDMS/DB load library containing vanilla load modules and delivered subschemas
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

IDMSDIRL

Local mode IDMSDIRL (z/OS)

```
//DIRL     EXEC PGM=IDMSDIRL,REGION=512K
//STEPLIB  DD  DSN=idms.dba.loadlib,DISP=SHR
//         DD  DSN=idms.custom.loadlib,DISP=SHR
//         DD  DSN=idms.cagjload,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,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.custom.loadlib	Data set name of the CA IDMS/DB load library containing customized load modules
idms.cagjload	Data set name of the CA IDMS/DB load library containing vanilla load modules
sysjrnl	DDname of the tape journal file

idms.tapejrnl	Data set name of the tape journal file
tape	Symbolic device name of the tape journal file
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 media)
tapein	Symbolic device name of the IDMSDIRL input file

Central version

To execute IDMSDIRL under the central version, modify the JCL shown previously 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.

IDMSLOOK

Local mode IDMSLOOK (z/OS)

```
//EXEC     PGM=IDMSLOOK,REGION=1024K
//STEPLIB  DD  DSN=idms.dba.loadlib,DISP=SHR
//         DD  DSN=idms.custom.loadlib,DISP=SHR
//         DD  DISP=idms.cagjload,DISP=SHR
//dcmsg    DD  DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,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.custom.loadlib	Data set name of the CA IDMS/DB load library containing customized load modules
idms.cagjload	Data set name of the CA IDMS/DB load library containing vanilla load modules

IDMSRPTS

Local mode IDMSRPTS (z/OS)

```
//RPTS      EXEC PGM=IDMSRPTS,REGION=256K
//STEPLIB   DD  DSN=idms.dba.loadlib,DISP=SHR
//          DD  DSN=idms.custom.loadlib,DISP=SHR
//          DD  DSN=idms.cagjload,DISP=SHR
//sysjrnl   DD  DSN=idms.tapejrnl,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 might 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.custom.loadlib	Data set name of the CA IDMS/DB load library containing customized load modules
idms.cagjload	Data set name of the CA IDMS/DB load library containing vanilla load modules
sysjrnl	DDname of the tape journal file
idms.tapejrnl	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 *dictionary-name* resides

Central version

To execute IDMSRPTS under the central version, modify the JCL shown previously 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.

IDMSRSTC

Local mode IDMSRSTC (z/OS)

```
//RSTC     EXEC PGM=IDMSRSTC,REGION=512K
//STEPLIB  DD  DSN=idms.dba.loadlib,DISP=SHR
//         DD  DSN=idms.custom.loadlib,DISP=SHR
//         DD  DSN=idms.cagjload,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,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 might 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.custom.loadlib	Data set name of the CA IDMS/DB load library containing customized load modules
idms.cagjload	Data set name of the CA IDMS/DB load library containing vanilla load modules
sysjrnl	DDname of the tape journal file
idms.tapejrnl	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 *dictionary-name* resides

Central version

To execute IDMSRSTC under the central version, modify the JCL shown previously 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.

IDMSRSTT

Assemble and link an IDMSRSTT module

```
//ASMCL    EXEC HLASMCL,
//             PARM.C='OBJECT,NODECK,RENT'
//C.SYSLIB   DD DISP=SHR,DSN=idms.cagjsrc
//C.SYSIN    DD *

     Put IDMSRSTT macro statements here

//L.SYSLMOD  DD DISP=SHR,DSN=idms.custom.loadlib
//L.SYSIN    DD *

         Optionally add an INCLUDE statement for each procedure named
         in the NUPROCS= clause here. If not included, any procedures
         will be dynamically loaded.

  ENTRY IDMSRSTT
  NAME idmsrstt(R)
/*
```

idms.cagjsrc	Library name containing source IDMS macros
idms.custom.loadlib	Library name into which to save the new IDMSRSTT load module
idmsrstt	Module name of the IDMSRSTT table

Chapter 8: z/VSE JCL

This section contains the following topics:

Overview (see page 483)
=COPY Facility (see page 483)
IDMSLBLS Procedure (see page 484)
SYSIDMS Parameter File (see page 485)
Batch Command Facility (see page 485)
Utility Statements (see page 487)
Utility Programs (see page 520)
IDMSLBLS Procedure (see page 529)

Overview

This chapter includes sample z/VSE JCL to execute CA IDMS utilities. Additionally, a description of the =COPY facility, IDMSLBLS procedure, and the SYSIDMS parameter file, as they relate to the utilities, is included.

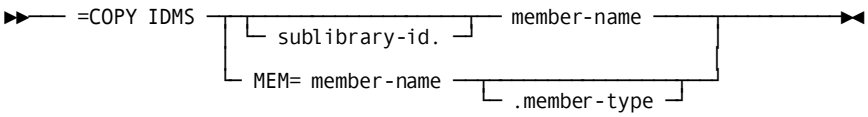
=COPY Facility

The =COPY IDMS statement is used to copy the library member into the job stream. Under z/VSE, some or all of the utility statements to be submitted to the Batch Command Facility (IDMSBCF) can be stored as a member in a source statement library.

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



Parameters

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.

member-type

Identifies the type of the member that contains the input parameter statements to be submitted to IDMSBCF. The default is A.

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.

IDMSLBLS Procedure

IDMSLBLS is a procedure provided during a CA IDMS z/VSE installation. It contains file assignments for CA IDMS dictionaries, sample databases, disk journal files, and the SYSIDMS parameter file.

A copy of the IDMSLBLS procedure appears at the end of this chapter.

File assignments for the CA IDMS files in the IDMSLBLS 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 is an EXEC statement for the IDMSLBLS procedure. The sample JCL assumes you will use either the IDMSLBLS procedure or another procedure that you create.

The files required to run each utility are identified in individual chapters under, "JCL Considerations".

SYSIDMS Parameter File

SYSIDMS is a parameter file used in the execution of 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 z/VSE environment. For example, there is a BLKSIZE parameter that allows you to override the blocksize for a file and a DEVADDR=SYS*nnn* 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 IDMSLBLS procedure so that parameters can be passed through SYSIPT.

SYSIDMS is referenced only in the generic JCL for the Batch Command Facility. You should include SYSIDMS parameters, as appropriate, in the JCL stream for each utility.

Note: On VSE systems, the default block size of the intermediate work files used by the CA IDMS utilities is 6000 bytes. If an intermediate work file contains records larger than 6000 bytes, you must use SYSIDMS to override the blocksize.

For example, in the UNLOAD utility, the record size of the SYS002 file is 24 bytes plus the size of the data portion of the record occurrence. If the total is greater than 6000 bytes, you can use the following SYSIDMS parameter as an override:

```
FILENAME=SYS002,BLKSIZE=32767,FILETYPE=D
```

Batch Command Facility

The following sample z/VSE JCL is used to execute the 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 (z/VSE)

```
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLS
// DLBL   idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL   SYSnnn,'idms.sysjrnl',,nnnnnn,,f
// ASSGN  SYSnnn,x'cuu'

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 neededfor the user catalog and the system dictionary depending on your security implementation.

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, and SYSIDMS parameter file. **Note:** For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idmslib	Dtfname of the CA IDMS library
idms.library	Data set name of CA IDMS 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 libraries	The CA IDMS libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file **Note:** For a complete listing of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

Central version

To run IDMSBCF under the central version, add a SYSCTL file and remove the tape journal file assignment.

```
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLS
// DLBL    idmslib,'idms.library'
// EXTENT  SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    sysctl,'idms.sysctl',1999/365,SD
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR

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

Utility Statements

ARCHIVE JOURNAL

File assignments for the batch command facility (z/VSE)

```
// TLBL    SYSnnn,'idms.archive',,nnnnnn,,f
// ASSGN   SYSnnn,x'cuu'

Additional archive journal file assignments, as required
```

idms.archive	File-ID of the tape archive file as defined in the DMCL

ARCHIVE LOG

File assignments for the batch command facility (z/VSE)

```
// DLBL    dclog,'idms.system.ddldclog',,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL    SYS002,'idms.archive',,nnnnnn,,f
// ASSGN   SYS002,x'cuu'
```

dclog	File-ID of the database log file
idms.archive	File-ID of the archive log file

ARCHIVE TRACE

The CA IDMS installation media contains the following sample JCL (see page 23) (intended for demonstration purposes only):

- ATRCJCL—Contains a skeleton of the JCL required to run the ARCHIVE TRACE utility.

BACKUP

File assignments for the batch command facility (z/VSE)

```
// DLBL    userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR

Additional file assignments, as required

// TLBL    SYS001,'user.bkpfile',,nnnnnn,,f
// ASSGN   SYS001,x'cuu'
```

userdb	Filename of the user file
user.userdb	File-ID of the user file
user.bkpfile	File-ID of the tape backup file

BUILD

File assignments for the batch command facility (z/VSE)

```
// 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  SYSnnn,nnnnnn,,,ssss,llll
// 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.load	File-ID of the input (SYS002) file; for sizing information see discussion of SYS003 in LOAD (see page 136).
user.build	File-ID of the output (SYS003) file; for sizing information see BUILD (see page 59).

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.

CLEANUP

File assignments for the batch command facility (z/VSE)

```
// 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

Central version

To execute CLEANUP SEGMENT under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

CONVERT CATALOG

DD statements for the batch command facility (z/VSE)

```
//DLBL      ddlcat,'ucat.ddlcat',,DA
//EXTENT    SYSnnn,nnnnnn
//ASSIGN    SYSnnn,DISK,VOL=vvvvvv,SHR
//DLBL      ddlcatx,'ucat.ddlcatx',,DA
//EXTENT    SYSnnn,nnnnnn
//ASSIGN    SYSnnn,DISK,VOL=vvvvvv,SHR
```

ddlcat	Filename of the file containing the DDLCAT area of the dictionary to be converted
ucat.ddlcat	File-ID of the file containing the DDLCAT area of the dictionary to be converted
ddlcatx	Filename of the file containing the DDLCATX area of the dictionary to be converted
ucat.ddlcatx	File-ID of the file containing the DDLCATX area of the dictionary to be converted

Central version

To execute CONVERT CATALOG under the central version:

- Optionally remove any journal and database file statements
- Insert the following statement:

```
//DLBL SYSCTL,file-id,_,SD
//EXTENT SYSCTLnnn,vvvvvv,1,_,1,1
//ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

CONVERT PAGE

File assignments for the batch command facility (z/VSE)

```
//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

EXPAND PAGE

File assignments for the batch command facility (z/VSE)

```
// 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

EXTRACT JOURNAL

File assignments for the batch command facility (z/VSE)

```
// DLBL   SYS001,'archive',,SD
// EXTENT SYS001,nnnnnn
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
// DLBL   extract,'jrnl.extract',,SD
// EXTENT sysnnn,nnnnnn
// ASSGN  sysnnn,DISK,VOL=vvvvvv,SHR

Add sort work files

Add SYSIDMS parameters

FILENAME=SYSnnn,BLKSIZE=bbbbb
```

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.
jrnl.extract	Data set name of the extract journal file.
bbbbb	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.

FASTLOAD

File assignments for the batch command facility (z/VSE)

```
// DLBL   SORTWK1,'sort.work.file'
// EXTENT  SYSnnn,nnnnnn,,,ssss,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR

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,llll
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   SYS001,'sort.unload',,SD
// EXTENT SYS001,nnnnnn,,,ssss,llll
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
// DLBL   SYS002,'user.dbl001',,SD
// EXTENT SYS002,nnnnnn,,,ssss,llll
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// DLBL   SYS004,'dbl004.name',,SD
// EXTENT SYS004,nnnnnn,,,ssss,llll
// ASSGN  SYS004,DISK,VOL=vvvvvv,SHR
// DLBL   SYS005,'user.dbl005',,SD
// EXTENT SYS005,nnnnnn,,,ssss,llll
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// TLBL   SYS009,'user.dbl009',,nnnnnn,,f
// ASSGN  SYS009,x'cuu'
// DLBL   SYS010,'user.dbl010',,SD
// EXTENT SYS010,nnnnnn,,,ssss,llll
// ASSGN  SYS010,DISK,VOL=vvvvvv,SHR
// DLBL   SYS011,'user.dbl011',,SD
// EXTENT SYS011,nnnnnn,,,ssss,llll
// 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 the *CA IDMS Common Facilities Guide*.

user.dbl001	File-ID of the file generated by the format program
user.dbl004	File-ID of the intermediate work file containing the output from SORT1. Record and block size should be the same as SYS001.
user.dbl005	File-ID of the intermediate work file containing a control record and set membership information from IDMSDBL2.
user.dbl009	File-ID of the intermediate work file containing the sorted contents of SYS005.
user.dbl010	File-ID of the intermediate work file containing pointer information from IDMSDBL3.
user.dbl011	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.

FIX ARCHIVE

File assignments for the batch command facility (z/VSE)

```
// 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
f	File number of the tape journal file
idms.tjrnlfix	File-ID of the output tape journal file

FIX PAGE

File assignments for the batch command facility (z/VSE)

```
// 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

Central version

To execute FIX PAGE under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

FORMAT

File assignments for the batch command facility (z/VSE)

To format a new database file:

```
// DLBL    userdb,'user.userdb',,tt
// EXTENT  SYSnnn,nnnnnn,,,ssss,rrrr
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

To reformat an existing database file:

```
// DLBL    userdb,'user.userdb',,tt
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

To format a new disk journal file:

```
// DLBL    j1jrnl,'idms.j1jrnl',,SD
// EXTENT  SYSnnn,nnnnnn,,,ssss,rrrr
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

To reformat an existing disk journal file:

```
// DLBL    j1jrnl,'idms.j1jrnl',,SD
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional database and journal file assignments, as required

To format a new SYSTRK file:

```
// DLBL ffff.,'idms.systrk1',,xx
// EXTENT SYSnnn,nnnnnn,,,ssss,rrrr
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

To reformat an existing SYSTRK file:

```
// DLBL ffff,'idms.systrk1',,xx
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

userdb	Filename of the database file
user.userdb	File-ID of the database file
idms.j1jrnl	File-ID of the disk journal file
j1jrnl	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: ■ SD (sequential) if formatting by file or segment (unless it is a VSAM file) ■ DA (direct access) if reformatting by area ■ VSAM if formatting or reformatting a CA IDMS/DB VSAM file ■ Omit if reformatting by file
ffff	File name of the SYSTRK file

idms.systrk1	FILE-ID of the SYSTRK file
xx	File type of the file being formatted ■ SD (sequential) if INITIAL is specified ■ DA (direct access) if INITIAL is not specified ■ VSAM if formatting a VSAM file

Multiple extents

You can format a CA IDMS/DB VSAM file with multiple extents under z/VSE. 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.

Central version

To execute FORMAT AREA or FORMAT SEGMENT under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

INSTALL STAMPS

Local mode

File assignments for the batch command facility (z/VSE)

```
// DLBL    userdb,'user.userdb',,tt
// EXTENT  SYSnnn,nnnnnn,,,ssss,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 TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

LOAD

File assignments for the batch command facility (z/VSE)

```
// 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,llll
// ASSGN   SYS001,DISK,VOL=vvvvvv,SHR
// DLBL    SYS002,'user.loadin',,SD
// EXTENT  SYS002,nnnnnn,,,ssss,llll
// ASSGN   SYS002,DISK,VOL=vvvvvv,SHR
// DLBL    SYS003,'user.loadout',,SD
// EXTENT  SYS003,nnnnnn,,,ssss,llll
// ASSGN   SYS003,DISK,VOL=vvvvvv,SHR
// DLBL    SYSnnn,'sort.load',,SD
// EXTENT  SYSnnn,nnnnnn,,,ssss,llll
// 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,llll
// 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 LOAD (see page 136).
user.loadout	File-ID of the output SYS003 file; for sizing information see LOAD (see page 136).
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.

LOCK

File assignments for the batch command facility (z/VSE)

```
// DLBL    userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

user.userdb	File-ID of the user database file

Central version

To execute LOCK AREA under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

MAINTAIN INDEX

File assignments for the batch command facility (z/VSE)

```
// DLBL    SORTWK1,'sort.work.file'
// EXTENT  SYSnnn,nnnnnn,,,ssss,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR

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,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    SYS003,'user.dbl003',,SD
// EXTENT  SYS003,nnnnnn,,,ssss,llll
// ASSGN   SYS003,DISK,VOL=vvvvvv,SHR
// DLBL    SYS004,'user.dbl004',,SD
// EXTENT  SYS004,nnnnnn,,,ssss,llll
// ASSGN   SYS004,DISK,VOL=vvvvvv,SHR
// DLBL    SYS005,'user.dbl005',,SD
// EXTENT  SYS005,nnnnnn,,,ssss,llll
// ASSGN   SYS005,DISK,VOL=vvvvvv,SHR
// DLBL    SYS006,'user.dbl006',,SD
// EXTENT  SYS006,nnnnnn,,,ssss,llll
// 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. Block size of the RELDCTL file should be a multiple of 60 with a maximum size of 32,760. Block size is controlled through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.
user.dbl003	File-ID of the intermediate work file containing index descriptors from IDMSTABX.
user.dbl004	File-ID of the intermediate work file containing the output from SORT3.
user.dbl005	File-ID of the intermediate work file containing pointers for user-owned index sets from IDMSDBL3.

user.dbl006	File-ID of the intermediate work file containing sorted pointers for user-owned index sets from SORT4.

PRINT INDEX

File assignments for the batch command facility (z/VSE)

```
// 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

PRINT JOURNAL

File assignments for the batch command facility (z/VSE)

```
// DLBL    SYS001,'idms.archjrnl',,SD
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

SYS001	Filename of the archive journal file
idms.archjrnl	File-ID of the archive journal file

PRINT LOG

File assignments for the batch command facility (z/VSE)

To print from the DDLDCLOG area:

```
// DLBL    dclog,'idms.system.ddldclog',,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
```

To print from the archive log file:

```
// TLBL    SYS001,'idms.archive',,nnnnnn,,f
// ASSGN   SYS001,x'cuu'
```

idms.archive	File-ID of the archive log file

PRINT PAGE

File assignments for the batch command facility (z/VSE)

```
// 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

Central version

To execute PRINT PAGE under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

PRINT SPACE

File assignments for the batch command facility (z/VSE)

```
// 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

Central version

To execute PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

PRINT TRACE

The CA IDMS installation media contains the following sample JCL (see page 23) (intended for demonstration purposes only):

- PTRCJCL – Contains a skeleton of the JCL required to run the PRINT TRACE utility.

PUNCH

Local mode

File assignments for the batch command facility (z/VSE)

```
// DLBL    usercat,'user.ddlcat',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    usercatl,'user.ddlcatl',,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'cuu'

Additional dummied journals, as required
```

user.ddlcat	File-ID of the dictionary file containing the DDLCAT area of the dictionary
user.ddlcatl	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.

RELOAD

File assignments for the batch command facility (z/VSE)

```
// DLBL   SORTWK1,'sort.work.file'
// EXTENT  SYSnnn,nnnnnn,,,ssss,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR

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,llll
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   SYS001,'sort.unload',,SD
// EXTENT SYS001,nnnnnn,,,ssss,llll
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
// DLBL   SYS002,'user.dbl002',,SD
// EXTENT SYS002,nnnnnn,,,ssss,llll
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// DLBL   SYS003,'user.dbl003',,SD
// EXTENT SYS003,nnnnnn,,,ssss,llll
// ASSGN  SYS003,DISK,VOL=vvvvvv,SHR
// DLBL   SYS004,'user.dbl004',,SD
// EXTENT SYS004,nnnnnn,,,ssss,llll
// ASSGN  SYS004,DISK,VOL=vvvvvv,SHR
// DLBL   SYS005,'user.dbl005',,SD
// EXTENT SYS005,nnnnnn,,,ssss,llll
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// DLBL   SYS006,'user.dbl006',,SD
// EXTENT SYS006,nnnnnn,,,ssss,llll
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// DLBL   SYS007,'user.dbl007',,SD
// EXTENT SYS007,nnnnnn,,,ssss,llll
// ASSGN  SYS007,DISK,VOL=vvvvvv,SHR
// DLBL   SYS008,'user.dbl008',,SD
// EXTENT SYS008,nnnnnn,,,ssss,llll
// ASSGN  SYS008,DISK,VOL=vvvvvv,SHR
```

```
// DLBL   SYS009,'user.dbl009',,SD
// EXTENT SYS009,nnnnnn,,,ssss,llll
// ASSGN  SYS009,DISK,VOL=vvvvvv,SHR
// DLBL   SYS010,'user.dbl010',,SD
// EXTENT SYS010,nnnnnn,,,ssss,llll
// ASSGN  SYS010,DISK,VOL=vvvvvv,SHR
// DLBL   SYS011,'user.dbl011',,SD
// EXTENT SYS011,nnnnnn,,,ssss,llll
// ASSGN  SYS011,DISK,VOL=vvvvvv,SHR
```

sort.unload	File-ID of the sort parameters created by UNLOAD
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 the *CA IDMS Common Facilities Guide*.
user.dbl002	File-ID of the file generated by UNLOAD as SYS001.
user.dbl003	File-ID of the file generated by UNLOAD as SYS003.
user.dbl004	File-ID of the intermediate work file containing the output from SORT1.
user.dbl005	File-ID of the intermediate work file containing member descriptors for chained sets from IDMSDBL2.
user.dbl006	File-ID of the intermediate work file containing member descriptors for user-owned index sets from IDMSDBL2.
user.dbl007	File-ID of the intermediate work file containing the output of SORT2.
user.dbl008	File-ID of the intermediate work file containing the reformatted index information from IDMSDBLX.
user.dbl009	File-ID of the intermediate work file containing the sorted index set descriptors from SORT3.
user.dbl010	File-ID of the intermediate work file containing prefix pointer information from IDMSDBL3.
user.dbl011	File-ID of the intermediate work file containing sorted prefix pointer information from SORT4.
userdb	Filename of the database file.
user.userdb	File-ID of the database file.

REORG

DD statements for the batch command facility (z/VSE) Defining the REORG control file, all jobs and job steps

```
// DLBL RORGCTL,'user.rorgctl',,SD
// EXTENT SYSnnn,vvvvvv,,,sssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Defining the REORG JCL file, required when submitting jobs

```
// DLBL RORGJCL,'user.jclfile',,SD
// EXTENT SYSnnn,vvvvvv,,,sssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Manual definition of work files, when not using DSMODELs

```
// DLBL wxnnnnn,'user.workfile',,SD llll
// EXTENT SYSnnn,vvvvvv,,,sssss,
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional work file definitions, as required

DB File definitions when running the unload phase, if not using dynamic allocation

```
// DLBL unlddb,'user.unlddb',,DA
// EXTENT SYSnnn,vvvvvv
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional unload database file definitions, as required

DB File definitions when running the reload phase, if not using dynamic allocation

```
// DLBL relddb,'user.relddb',,DA
// EXTENT SYSnnn,vvvvvv
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional reload database file definitions, as required

SORT work file assignments when running the reload phase

```
// DLBL SORTWK1,'sort.work.file'
// EXTENT SYSnnn,vvvvvv,,,sssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

Additional sort work file definitions, as required

user.reldctl

File-ID of the REORG control file containing control information. The block size is 8192 bytes, and must be specified in SYSIDMS using the FILENAME=RORGCTL RECFM=F BLKSIZE=8192 parameters.

Note: For more information about SYSIDMS parameters, see the *CA IDMS Common Facilities Guide.*

user.jclfile

File-ID of the file containing JCL for automatic job submission. The block size must be a multiple of 80 bytes, and must be specified in SYSIDMS using the FILENAME=RORGJCL RECFM=F BLKSIZE= parameters.

wxnnnnn

File name of the DLBL for a work file. It must match the name generated in the Unload/Reload Work File Summary report.

user.workfile

File-ID of a work file when manually allocating work files.

unlddb

File name of the DLBL for an unload database file.

user.unlddb

File-ID of the unload database file, this is source database file.

relddb

File name of the DLBL for a reload database file.

user.relddb

File-ID of the reload database file, this is the target database file.

RESTORE

File assignments for the batch command facility (z/VSE)

```
// 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'
// ASSGN   SYS001,x'cuu'

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

RESTRUCTURE

File assignments for the batch command facility (z/VSE)

```
// 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 RESTRUCTURE CONNECT (see page 306).

RESTRUCTURE CONNECT

File assignments for the batch command facility (z/VSE)

```
// 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,nnnnnn,,,ssss,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR

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 RESTRUCTURE CONNECT (see page 306).

ROLLBACK

File assignments for the batch command facility (z/VSE)

```
// DLBL    userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR

Additional database file assignments, as required

// TLBL    SYS001,'archive'
// ASSGN   SYS001,x'cuu'

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

ROLLFORWARD

File assignments for the batch command facility (z/VSE)

```
// 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'
// ASSGN   SYSnnn,X'cuu'

If recovering from a journal extract file

// DLBL    extract,'jrnl.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.
jrnl.extract	File-ID of the extract journal file.

SYNCHRONIZE STAMPS

Local mode

File assignments for the batch command facility (z/VSE)

```
// DLBL    userdb,'user.userdb',,tt
// EXTENT  SYSnnn,nnnnnn,,,ssss,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 SYNCHRONIZE STAMPS under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

TUNE INDEX

File assignments for the batch command facility (z/VSE)

```
//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

Central version

To execute TUNE INDEX under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

UNLOCK

File assignments for the batch command facility (z/VSE)

```
// DLBL     userdb,'user.userdb',,DA
// EXTENT   SYSnnn,nnnnnn
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR
```

user.userdb	File-ID of the user database file

UNLOAD

File assignments for the batch command facility (z/VSE)

```
// DLBL   SORTWK1,'sort.work.file'
// EXTENT  SYSnnn,nnnnnn,,,ssss,llll
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR

Additional sort files, as required

// DLBL   RELDCTL.'user.reldctl',,SD
// EXTENT SYSnnn,nnnnnn,,,ssss,llll
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   SYS002,'user.dbl002',,SD
// EXTENT SYS002,nnnnnn,,,ssss,llll
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// DLBL   SYS003,'user.dbl003',,SD
// EXTENT SYS003,nnnnnn,,,ssss,llll
// ASSGN  SYS003,DISK,VOL=vvvvvv,SHR
// DLBL   IJSYSPCH,'sort.unload'
// EXTENT SYSPCH,nnnnnn,,,ssss,llll
   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 the *CA IDMS Common Facilities Guide*.
user.dbl002	File-ID of the SYS002 output file.
user.dbl003	File-ID of the SYS003 output file.
sort.unload	File-ID of the file containing sort parameters created by UNLOAD.

UPDATE STATISTICS

File assignments for the batch command facility (z/VSE)

```
// DLBL    userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL    SYSnnn,'idms.tapejrnl',,nnnnnn,,f
// ASSGN   SYSnnn,x'cuu'
```

userdb	Filename of the user file

Central version

To execute UPDATE STATISTICS under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

file-id	Data set name of the sysctl file
SYSnnn	Logical unit of the volume for which the extent is effective
vvvvvv	Volume serial number

VALIDATE

File assignments for the batch command facility (z/VSE)

```
// 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,llll
// 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,llll
// 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 VALIDATE (see page 359).
user.valout	File-ID of the SYS003 output file; for sizing information see VALIDATE (see page 359).
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.

Utility Programs

IDMSDBAN

IDMSDBAN (z/VSE)

```
// DLBL   idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLS
// 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. **Note:** For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idmslib	Dtfname of the CA IDMS library
idms.library	Data set name of CA IDMS 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 libraries	The CA IDMS libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file **Note:** For a complete listing of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
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
cuu	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

IDMSDIRL

Local mode IDMSDIRL (z/VSE)

```
// DLBL   idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLS
// TLBL   SYSnnn,'idms.tapejrnl',,nnnnnn,,f
// ASSGN  SYSnnn,x'cuu'
// DLBL    dictdb,'idms.dictdb',,DA
// EXTENT SYS005,nnnnnn
// ASSGN  SYS005,DISK,VOL=nnnnnn,SHR
// TLBL   SYS001,'idms.dirldata',,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. **Note:** For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idmslib	Dtfname of the CA IDMS library
idms.library	Data set name of CA IDMS 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 libraries	The CA IDMS libraries, as established during installation

SYSIDMS	Filename of the SYSIDMS parameter file **Note:** For a complete listing of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
idms.tapejrnl	File-ID of the tape journal file
f	File number of the tape file
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 and ASSGN statements for the tape journal file from the previous JCL
- Add a SYSCTL file

IDMSLOOK

IDMSLOOK (z/VSE)

```
// DLBL   idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL   SYSnnn,'idms.tapejrnl',,nnnnnn,,f
// ASSGN  SYSnnn,x'cuu'
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC   PROC=IDMSLBLS
// 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. **Note:** For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idmslib	Dtfname of the CA IDMS library
idms.library	Data set name of CA IDMS 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 libraries	The CA IDMS libraries, as established during installation
SYSIDMS	Filename of the SYSIDMS parameter file **Note:** For a complete listing of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

IDMSRPTS

Local mode

IDMSRPTS (z/VSE)

```
// DLBL   idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLS
// TLBL   SYSnnn,'idms.tapejrnl',,nnnnnn,,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. **Note:** For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idmslib	Dtfname of the CA IDMS library
idms.library	Data set name of CA IDMS 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 libraries	The CA IDMS libraries, as established during installation

SYSIDMS	Filename of the SYSIDMS parameter file **Note:** For a complete listing of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
SYS009	Logical unit assignment of the tape journal file
idms.tapejrnl	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 and ASSGN statements for the tape journal file from the previous JCL.
- Add a SYSCTL file

IDMSRSTC

Local mode

IDMSRSTC (z/VSE)

```
// DLBL   idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLS
// TLBL   SYSnnn,'idms.tapejrnl',,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,llll
   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. **Note:** For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.
idmslib	Dtfname of the CA IDMS library
idms.library	Data set name of CA IDMS 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 libraries	The CA IDMS libraries, as established during installation

SYSIDMS	Filename of the SYSIDMS parameter file **Note:** For a complete listing of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
sys009	Logical unit assignment of the tape journal file
idms.tapejrnl	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
cuu	Physical device assignment of the card-image file

Central version

To execute IDMSRSTC under the central version:

- Remove the TLBL and ASSGN statements. for the tape journal from the previous JCL.
- Add a SYSCTL file

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

/*

          Optionally add an INCLUDE statement for each procedure
          named in the NUPROCS= clause here. If not included, any
          procedures will be dynamically loaded.

// DLBL userlib
// EXTENT ,nnnnnn
// LIBDEF PHASE,CATALOG=(userlib.idmslib)
// EXEC LNKEDT
```

userlib	Filename of the user library
nnnnnn	Volume serial number of the library
userlib.idmslib	File identifier of the CA IDMS sublibrary
idmsrstt	Phase name of the IDMSRSTT table

IDMSLBLS Procedure

IDMSLBLS is a procedure provided during a z/VSE installation. It contains file definitions for the following CA IDMS components. 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, see the *CA IDMS Installation and Maintenance Guide—z/VSE*.

Tailor the IDMSLBLS procedure to reflect the filenames and definitions in use at your site and include this procedure in z/VSE JCL job streams.

The sample z/VSE JCL provided in this document includes the IDMSLBLS procedure. Therefore, individual file definitions for CA IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

IDMSLBLS 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    dclog,'idms.system.ddldclog',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    empldem,'idms.sqldemo.empldemo',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.projseg.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.ddldml',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.ddlcatl',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    asfdml,'idms.asfdict.ddldml',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    j1jrnl,'idms.j1jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    j2jrnl,'idms.j2jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    j3jrnl,'idms.j3jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL    SYSIDMS,'#SYSIPT',0,SD
/+
/*
```

idmslib.sublib	Name of the sublibrary within the library containing CA IDMS modules
user.sublib	Name of the sublibrary within the library containing user modules
idmslib	Filename of the file containing CA IDMS modules
idms.library	File-ID associated with the file containing CA IDMS modules
SYS*nnn*	Logical unit of the volume for which the extent is effective
nnnnnn	Volume serial identifier of appropriate disk volume
ssss	Starting track (CKD) or block (FBA) of disk extent
dccat	Filename of the system dictionary catalog (DDLCAT) area
idms.system.dccat	File-ID of the system dictionary catalog (DDLCAT) area
dccatl	Filename of the system dictionary catalog load (DDLCATLOD) area
idms.system.dccatlod	File-ID of the system dictionary catalog load (DDLCATLOD) area

dccatx	Filename of the system dictionary catalog index (DDLCATX) area
idms.system.dccatx	File-ID of the system dictionary catalog index (DDLCATX) area
dcdml	Filename of the system dictionary definition (DDLDML) area
idms.system.ddldml	File-ID of the system dictionary definition (DDLDML) area
dclod	Filename of the system dictionary definition load (DDLDCLOD) area
idms.system.ddldclod	File-ID of the system dictionary definition load (DDLDCLOD) area
dclog	Filename of the system log (DDLDCLOG) area
idms.system.ddldclog	File-ID of the system log (DDLDCLOG) area
dcrun	Filename of the system queue (DDLDCRUN) area
idms.system.ddldcrun	File-ID of the system queue (DDLDCRUN) area
dcscr	Filename of the system scratch (DDLDCSCR) area
idms.system.ddldcscr	File-ID of the system scratch (DDLDCSCR) area
dcmsg	Filename of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	File-ID of the system message (DDLDCMSG) area
dclscr	Filename of the local mode system scratch (DDLOCSCR) area
idms.sysloc.ddlocscr	File-ID of the local mode system scratch (DDLOCSCR) area
dirldb	Filename of the IDMSDIRL definition (DDLDML) area
idms.sysdirl.ddldml	File-ID of the IDMSDIRL definition (DDLDML) area
dirllod	Filename of the IDMSDIRL definition load (DDLDCLOD) area
idms.sysdirl.dirllod	File-ID of the IDMSDIRL definition load (DDLDCLOD) area
empdemo	Filename of the EMPDEMO area
idms.empdemo1	File-ID of the EMPDEMO area
insdemo	Filename of the INSDEMO area
idms.insdemo1	File-ID of the INSDEMO area
orgdemo	Filename of the ORGDEMO area

idms.orgdemo1	File-ID of the 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 (DDLDML) area
dloddb	Filename of the application dictionary definition load area
idms.appldict.ddldclod	File-ID of the application dictionary definition load (DDLDCLOD) area
sqldd	Filename of the SQL catalog (DDLCAT) area
idms.syssql.ddlcat	File-ID of the SQL catalog (DDLCAT) area
sqllod	Filename of the SQL catalog load (DDLCATL) area
idms.syssql.ddlcatl	File-ID of SQL catalog load (DDLCATL) area
sqlxdd	Filename of the SQL catalog index (DDLCATX) area
idms.syssql.ddlcatx	File-ID of the SQL catalog index (DDLCATX) area
asfdml	Filename of the asf dictionary definition (DDLDML) area
idms.asfdict.ddldml	File-ID of the asf dictionary definition (DDLDML) area
asflod	Filename of the asf dictionary definition load (ASFLOD) area
idms.asfdict.asflod	File-ID of the asf dictionary definition load (ASFLOD) area
asfdata	Filename of the asf data (ASFDATA) area

idms.asfdict.asfdata	File-ID of the asf data (ASFDATA) area
ASFDEFN	Filename of the asf data definition (ASFDEFN) area
idms.asfdict.asfdefn	File-ID of the asf data definition (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 9: CMS Commands

This section contains the following topics:

Overview (see page 537)
Batch Command Facility (see page 537)
Utility Statements (see page 539)
Utility Programs (see page 566)

Overview

This chapter presents sample CMS commands to run CA IDMS/DB utility statements and programs.

Common commands used to run the Batch Command Facility are presented first. Additional commands required to run each utility statement and program are presented next, alphabetically, by utility.

Batch Command Facility

The following CMS commands are used to execute the 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 sysjrnl 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
```

Important! Additional file assignments might be needed for the user catalog and the system dictionary.

Note: When executing local mode or batch-to-CV mode execs in CMS, filedefs for CDMSLIB loadlibs are optional. However, if omitted, and multiple EXECOS OSRUN statements are coded in the same EXEC, you must code the following CMS statement prior to the first EXECOS OSRUN statement in the EXEC:

```
SET STORECLR ENDCMD
```

You must also code the following CMS statement following the last EXECOS OSRUN statement:

```
SET STORECLR ENDSVC
```

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
sysjrnl	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. **Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
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 sysjrnl 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

Utility Statements

ARCHIVE JOURNAL

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF j1jrnl DISK j1jrnl jrnlfile fm (RECFM F LRECL lll XTENT nnn
FILEDEF j2jrnl DISK j2jrnl jrnlfile fm (RECFM F LRECL lll XTENT nnn

Additional journal file assignments, as required

FILEDEF archjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL rrrr BLKSIZE bbbb
```

j1jrnl	DDname of the first disk journal file, as defined in the DMCL
j1jrnl jrnlfile fm	File identifier of the first disk journal file
j2jrnl	DDname of the second disk journal file
j2jrnl jrnlfile fm	File identifier of the second disk journal file
lll	Page size of the journal file
nnn	Number of pages in the journal file
archjrnl	DDname of the tape archive file, as defined in the DMCL
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

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 sysjrnl DUMMY
FILEDEF SYS002 DISK archive dbfile fm (RECFM VB LRECL 280 BLKSIZE bbbb
```

dlogdb	DDname of the system log area
idms dlogdb fm	File identifier of the system log area
ppp	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)

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 llll 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

llll	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: ■ The size of the largest page being backed up, plus 8 or ■ 32,760

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 lll 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 LOAD
SYS003	DDname of the output (SYS003) file; for sizing information see 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.

CLEANUP

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF sysjrnl DISK tapejrnl jrnlfile fm
FILEDEF userdb DISK userdb dbfile fm

Additional database file assignments, as required
```

sysjrnl	DDname of the tape journal file
tapejrnl jrnlfile 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

Central version

To execute CLEANUP SEGMENT under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by CLEANUP SEGMENT using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

CONVERT CATALOG

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF ddlcat DISK sysdict ddlcat f(RECFM F LRECLppp BLKSIZE ppp XTENT nnn
FILEDEF ddlcatx DISK sysdict ddlcatx f(RECFM F LRECLppp BLKSIZE ppp XTENT nnn
FILEDEF j1jrnl DISK j1jrnl jrnlfile e (RECFM F LRECLppp BLKSIZE ppp XTENT nnn
```

ddlcat	DDname of the database file containing the DDLCAT area of the dictionary to be converted
ucat ddlcat f	File identifier of the database file containing the DDLCAT area of the dictionary to be converted
ddlcatx	DDname of the database file containing the DDLCATX area of the dictionary to be converted
ucat ddlcatx f	File identifier of the database file containing the DDLCATX area of the dictionary to be converted

j1jrnl	DDname of the first journal file, as defined in the DMCL
j1jrnl jrnlfile e	File identifier of the first journal file

Central version

To execute CONVERT CATALOG under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements
- Identify the DC/UCF system to be accessed using the CVMACH and CVNUM SYSIDMS parameters

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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

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

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.
bbbbb	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.

FASTLOAD

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 DISK tblu sortparm fm
FILEDEF SYS002 DISK fmtpgm file fm
FILEDEF SYS004 TAP1 SL VOLID dbl004 (RECFM VB LRECL rr04 BLKSIZE bb04
FILEDEF SYS005 TAP2 SL VOLID dbl005 (RECFM VB LRECL rr05 BLKi1ZE 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
```

tblu sortparm fm	File identifier of the file containing sort parameters created by IDMSTBLU
fmtpgm file fm	File identifier of the file generated by the format program
dbl004	Volume serial number of the intermediate work file containing the output from SORT1
rr04	Record size of the SYS004 file; should be the same as the record size for SYS001
bb04	Block size of the SYS004 intermediate work file; should be the same as the block size for SYS001
dbl005	Volume serial number of the intermediate work file containing a control record and set membership information from IDMSDBL2
rr05	Record size of the SYS005 file; for sizing information, see FASTLOAD
bb05	Block size of the SYS005 intermediate work file; must be at least rr05 plus four bytes
dbl009	Volume serial number of the intermediate work file containing the sorted contents of SYS005
rr09	Record size of the SYS009 file; should be the same as rr05
bb09	Block size of the SYS009 intermediate work file; should be the same as bb05
dbl010	Volume serial number of the intermediate work file containing pointer information from IDMSDBL3

rr10	Record size of the SYS010 file; for sizing information, see 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

FIX ARCHIVE

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 TAP1 SL VOLID tjrnlold (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF SYS002 TAP2 SL VOLID tjrnlfix (RECFM VB LRECL lll BLKSIZE bbbb
```

tjrnlold	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

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

Central version

To execute FIX PAGE under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by FIX PAGE using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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 lll XTENT nnn

Additional file assignments, as required
```

To format a new or existing disk journal file:

```
FILEDEF j1jrnl DISK j1jrnl jrnlfile fm (RECFM F LRECL lll XTENT nnn

Additional journal file assignments, as required
```

To format a new or existing SYSTRK file:

```
FILEDEF ffff DISK idms systrk1 fm (RECFM F LRECL lll 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
j1jrnl	DDname of the first disk journal file, as defined in the DMCL
j1jrnl jrnlfile fm	DDname of the first disk journal file
ffff	DDname of the SYSTRK file
idms systrk1 fm	File identifier of the SYSTRK file

Central version

To execute FORMAT AREA or FORMAT SEGMENT under the central version, modify the CMS commands shown previously as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by FORMAT AREA or FORMAT SEGMENT using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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 sysjrnl DISK sysjrnl jrnlfile 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
sysjrnl	DDname of the tape journal file
sysjrnl jrnlfile fm	File identifier of the tape journal file

Central Version

To execute INSTALL STAMPS under the central version, modify the JCL shown previously, 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

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 needed only 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 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 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.

LOCK

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

Central version

To execute LOCK AREA under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by LOCK AREA using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

MAINTAIN INDEX

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK usrolddb dbfile fm
FILEDEF SYS003 TAP1 SL VOLID dbl003 (RECFM VB LRECL rr03 BLKSIZE bb03
FILEDEF SYS004 TAP2 SL VOLID dbl004 (RECFM VB LRECL rr04 BLKSIZE bb04
FILEDEF SYS005 TAP3 SL VOLID dbl005 (RECFM VB LRECL rr05 BLKSIZE bb05
FILEDEF SYS006 TAP4 SL VOLID dbl006 (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
dbl003	Volume serial number of the intermediate work file containing index descriptors from IDSMTABX
rr03	Record size of the SYS003 file; see MAINTAIN INDEX (see page 157) for sizing information
bb03	Block size of the SYS003 intermediate work file
dbl004	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
dbl005	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 MAINTAIN INDEX (see page 157) for sizing information
bb05	Block size of the SYS005 intermediate work file
dbl006	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

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 JRNM01 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

PRINT INDEX

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF sysjrnl DUMMY
FILEDEF userdb DISK userdb dbfile fm
```

sysjrnl	DDname of the dummy journal file
userdb	DDname of the database file
userdb dbfile fm	File identifier of the database file

PRINT JOURNAL

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF SYS001 DISK archjrnl jnlfile fm
```

archjrnl	DDname of the archive journal file
archjrnl jnlfile fm	File identifier of the archive journal file

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 *sysjrnl* 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
sysjrnl	DDname of the tape journal file defined in the DMCL module named in the IDMSNWKS subschema

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

Central version

To execute PRINT PAGE under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by PRINT PAGE using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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	age size of the database file
nnnn	Number of pages in the database file

Central version

To execute PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT under the central version, modify the CMS commands shown previously as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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 sysjrnl 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 ddlcatx fm	File identifier of the DDLCATX area of the dictionary
sysjrnl	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 previously, 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

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
```

sortunld parms a	File identifier of the sort parameters created by UNLOAD
user dbl002 a	File identifier of the file generated by UNLOAD as SYS001
user dbl003 a	File identifier of the file generated by UNLOAD as SYS003
tap4	Symbolic device name of the SYS004 file containing the output from SORT1
rrr004	Record size of the SYS004 file; should be the same as the record size for SYS002 used by UNLOAD
bbb004	Block size of the SYS004 intermediate work file; should be the same as the block size for SYS002 used by UNLOAD
tap5	Symbolic device name of the SYS005 file containing member descriptors for chained sets from IDMSDBL2
rrr005	Record size of the SYS005 file; for sizing information, see RELOAD

bbb005	Block size of the SYS005 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
tap6	Symbolic device name of the SYS006 file containing member descriptors for user-owned index sets from IDMSDBL2
rrr006	Record size of the SYS006 file; for sizing information, see RELOAD
bbb006	Block size of the SYS006 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)
tap7	Symbolic device name of the SYS007 file containing the output of SORT2
rrr007	Record size of the SYS007 file; should be the same as the larger of: ■ the record size for SYS003 used by UNLOAD ■ the record size for SYS006
bbb007	Block size of the SYS007 intermediate work file; must be at least as large as the larger of: ■ the block size for SYS003 used by UNLOAD ■ the block size for SYS006
tap8	Symbolic device name of the SYS008 file containing the reformatted index information from IDMSDBLX
rrr008	Record size of the SYS008 file; should be the same as rrr007
bbb008	Block size of the SYS008 intermediate work file; should be the same as bbb007
tap9	Symbolic device name of the SYS009 file containing the sorted index set descriptors from SORT3
rrr009	Record size of the SYS009 file; should be the same as the larger of: ■ the record size for SYS005 ■ the record size for SYS008
bbb009	Block size of the SYS009 intermediate work file; must be at least as large as the larger of: ■ the block size for SYS003 used by UNLOAD ■ the block size for SYS006

tap10	Symbolic device name of the SYS010 file containing prefix pointer information from IDMSDBL3
rrr010	Record size of the SYS010 file; for sizing information, see RELOAD
bbb010	Block size of the SYS010 intermediate work file; must be at least 44 bytes (the size of a pointer descriptor plus four bytes)
tap11	Symbolic device name of the SYS011 file containing the sorted prefix pointer information from SORT4
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
user reldctl a	File identifier of the RELDCTL file containing control information created during the UNLOAD
sortreld parms a	File identifier of the SYSPCH file
bbbctl	Blocksize of the RELDCTL file; it must be a multiple of 60 with a maximum size of 32,760
sortwk01 work a3	File identifier of temporary sort work file (if needed)
userdb1	DDname of the first database file
user userdb1 f	File identifier of the first database file

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
```

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
vvvvvv	Volume id of the tape file

llll	Record length of the tape backup file
bbbb	Block size of the tape backup file

RESTRUCTURE

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 40 with a maximum size of 32,760

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
```

userdb	DDname of the database file
user userdb f	File identifier of the database file
idms spill f	File identifier of the spill file
bbbb	Size of the spill file should be a multiple of 40 with a maximum size of 32,760
sortwk01 work a3	File identifier of temporary sort work file (if needed)

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
```

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

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 jrnl.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
```

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.
jrnl.ext ft fm	File identifier of the journal extract file

SYNCHRONIZE STAMPS

Local Mode FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK userdb dbfile fm
FILEDEF userdict DISK userdict dbfile fm
FILEDEF sysjrnl DISK sysjrnl jrnlfile 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
sysjrnl	DDname of the tape journal file
sysjrnl jrnlfile fm	File identifier of the tape journal file

Central Version

To execute SYNCHRONIZE STAMPS under the central version, modify the JCL shown previously, 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

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

Central version

To execute TUNE INDEX under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by TUNE INDEX using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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

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 sysjrnl 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
sysjrnl	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 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 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

UPDATE STATISTICS

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK user userdb 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 j1jrnlDISK j1jrnl jrnlfile e (RECFM F LRECLppp BLKSIZE ppp XTENT nnn

Additional journal file assignments, as required
```

userdb	DDname of the database file
user userdb f	File identifier of the database file
ddlcat	DDname of the database file containing the area of the system dictionary with the table definitions
sysdict ddlcat f	File identifier 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 f	File identifier of the database file containing the area of the system dictionary with indexes
j1jrnl	DDname of the first journal file, as defined in the DMCL
j1jrnl jrnlfile e	File identifier of the first journal file

Central version

To execute UPDATE STATISTICS under the central version, modify the CMS commands shown previously as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by UPDATE STATISTICS using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

VALIDATE

FILEDEF commands for the batch command facility (CMS)

```
FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE ppp XTENT nnnn
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 needed only
      when performing a complete VALIDATE.

FILEDEF SORTMSG PRINTER
FILEDEF sortwknn DISK sort worknn a3

Additional sort files, as required
```

userdb	DDname of the database file
user userdb f	File identifier of the database file
*tap2**	Symbolic device name of the input SYS002 file; for sizing information see VALIDATE
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 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.

Utility Programs

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
```

Important! Additional file assignments might be needed for the user catalog and the system dictionary.

userdb	DDname of the user database file
user userdb b	File identifier of the user database file
pppp	Page size of the user database file
nnnn	Number of pages in the user database file
nnnnnn	Volume serial number of the chain file generated by IDMSDBN1
SYSIDMS	DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters. **Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

sysidms input a	File identifier of the file containing SYSIDMS parameters if applicable
dban input a	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.

IDMSDIRL

Local mode IDMSDIRL (CMS)

```
FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF dictdb DISK idms dictdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSLST PRINTER
FILEDEF SYS001 TAP2 SL VOLID nnndrl (RECFM VB LRECL 600 BLKSIZE 5992
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK dirl input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSDIRL
```

Important! User catalog and the system dictionary depending on your security implementation.

sysjrnl	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
nnndrl	Volume serial number of the IDMSDIRL input file (on the installation media)
SYSIDMS	DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters. **Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
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 previously, 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.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

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
```

Important! Additional file assignments might be needed for the user catalog and the system dictionary.

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. **Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
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

IDMSRPTS

Local mode IDMSRPTS (CMS)

```
FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll 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
```

Important! Additional file assignments might be needed for the user catalog and the system dictionary.

sysjrnl	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. **Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
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 previously, 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.

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

IDMSRSTC

Local mode IDMSRSTC (CMS)

```
FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll 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
```

Important! Additional file assignments might be needed for the user catalog and the system dictionary.

sysjrnl	DDname of the tape journal file
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
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. **Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.
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 previously, as follows:

- Remove the SYSJRNL and DICTDB FILEDEF commands.
- Identify the DC/UCF system to be accessed by IDMSRTC using the CVMACH and CVNUM SYSIDMS parameters.

IDMSRSTT

Assemble and link an IDMSRSTT module

```
GLOBAL MACLIB idmslib
ASMAHL rsttmac (RENT

   Optionally add the rstt module into a txtlib.

TXTLIB ADD userlib idmsrstt

   Optionally link the rstt module into a loadlib.

FILEDEF idmsrstt DISK idmsrstt TEXT A

   If linking NUPROCS= procedures from TEXT files
   add a FILEDEF for each procedure.  If linking from
   a TXTLIB add a filedef for the txtlib.

   Note: linking procedures is not required.
   They will be dynamically loaded if not linked.

FILEDEF SYSLMOD DISK idmsload LOADLIB A (RECFM V LRECL 1024 BLKSIZE 1024
LKED linkctl (RMODE ANY LIST MAP XREF PRINT

Linkage editor control statements in linkctl file:

INCLUDE idmsrstt

   Optionally add an INCLUDE statement for each procedure
   named in the NUPROCS= clause here.
   If not included any procedures will be dynamically loaded.

NAME idmsrstt(R)
```

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
idmslib	Filename of the library containing the IDMSRSTT macro
idmsload	Filename of the loadlib to store the rstt module
linkctl	Filename of TEXT file used as input to LKED

Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Guide*.

Appendix A: FASTLOAD Format Program Sample Listing

This appendix shows a listing of a format program that can be used with IDMSDBLU to load the sample Commonweather database provided during the CA IDMS/DB 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.                  CA
*                               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  O1-CARD.
  04  O1-KEYFIELDS.
   05  FILLER                 PIC  X(4).
   05  O1-OFFICE-CODE         PIC  9(3).
  04  O1-DATAFIELDS.
   05  O1-OFFICE-ADDRESS      PIC  X(56).
   05  FILLER                 PIC  X(17).
 03  O2-CARD.
  04  O2-KEYFIELDS.
   05  FILLER                 PIC  X(4).
   05  O2-OFFICE-CODE         PIC  9(3).
  04  O2-DATAFIELDS.
   05  O2-OFFICE-PHONE        PIC  9(7) OCCURS 3 TIMES.
   05  O2-OFFICE-AREA         PIC  9(3).
   05  O2-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).
   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-DO-DEPARTMENT THRU 1090-DD-EXIT
                   ELSE
           IF  TSA-EMPLOYEE
               PERFORM 1510-DO-EMPLOYEE THRU 1590-DE-EXIT
                   ELSE
           IF  TSA-JOB
               PERFORM 2010-DO-JOB THRU 2090-DJ-EXIT
                   ELSE
           IF  TSA-EMPOSITION
               PERFORM 2510-DO-EMPOSITION THRU 2590-DEM-EXIT
                   ELSE
           IF  TSA-EXPERTISE
               PERFORM 3010-DO-EXPERTISE THRU 3090-DEX-EXIT
                   ELSE
           IF  TSA-SKILL
               PERFORM 3510-DO-SKILL THRU 3590-DS-EXIT
                   ELSE
           IF  TSA-OFFICE
               PERFORM 4510-DO-OFFICE THRU 4590-DO-EXIT
                   ELSE
           IF  TSA-STRUCTURE
               PERFORM 5010-DO-STRUCTURE THRU 5090-DS-EXIT
                   ELSE
           IF  TSA-INS-PLAN-CODE
               PERFORM 5510-DO-INSURANCE THRU 5590-DI-EXIT
                   ELSE
           IF  TSA-COVERAGE
               PERFORM 6010-DO-COVERAGE THRU 6090-DC-EXIT
                   ELSE
           IF  TSA-DENTAL
               PERFORM 6510-DO-DENTAL THRU 6590-DDN-EXIT
                   ELSE
           IF  TSA-HOSPITAL
               PERFORM 7010-DO-HOSPITAL THRU 7090-DH-EXIT
                   ELSE
               PERFORM 7510-DO-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 'O1' AND AN 'O2' 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 = 'O1'
               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 O1-CARD.

    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

    IF CI-CARD-TYPE = 'O2'
        AND CI-OFFICE-CODE = O1-OFFICE-CODE
        MOVE CI-DATA-IMAGE     TO O2-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-DO-OFFICE.

************************************************
*    THIS ROUTINE STORES A NEW OFFICE RECORD      *
************************************************

*****BUILD OCCURRENCE OF OFFICE RECORD*************************

     MOVE O1-OFFICE-CODE      TO  OFFICE-CODE-0450.
     MOVE O1-OFFICE-ADDRESS   TO  OFFICE-ADDRESS-0450.
     MOVE O2-OFFICE-AREA      TO  OFFICE-AREA-CODE-0450.
     MOVE O2-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-DO-EXIT.
     EXIT.
```

```
4615-OFFICE-PHONE.
    IF O2-OFFICE-PHONE (I-CTRL) IS NOT NUMERIC
        MOVE ZEROS TO OFFICE-PHONE-0450 (I-CTRL)
            ELSE
        MOVE O2-OFFICE-PHONE (I-CTRL)
            TO OFFICE-PHONE-0450 (I-CTRL).
4615-OP-EXIT.
    EXIT.

5010-DO-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.
    PERFORM DBLU-STATUS.

7090-DH-EXIT.
    EXIT.

7510-DO-NONHOSP.

*****************************************************************
*    THIS ROUTINE STORES A NEW NON-HOSP-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 NON-HOSP-CLAIM RECORD*****************

    MOVE N1-NC-CLAIM-DATE     TO  CLAIM-DATE-0445.
    MOVE N1-NC-PATIENT-NAME   TO  PATIENT-NAME-0445.
    MOVE N1-NC-PATIENT-DOB    TO  PATIENT-BIRTH-DATE-0445.
    MOVE N1-NC-SEX            TO  PATIENT-SEX-0445.
    MOVE N1-NC-REL-TO-EMP     TO  RELATION-TO-EMPLOYEE-0445.
    MOVE N1-NC-PHYS-NAME      TO  PHYSICIAN-NAME-0445.
    MOVE N2-NC-PHYS-ADDRESS   TO  PHYSICIAN-ADDRESS-0445.
    MOVE N2-NC-PHYS-ID        TO  PHYSICIAN-ID-0445.
    MOVE N3-NC-DIAGNOSIS      TO  DIAGNOSIS-0445 (1).
    MOVE N4-NC-DIAGNOSIS      TO  DIAGNOSIS-0445 (2).
    MOVE I-CTRL               TO  NUMBER-OF-PROCEDURES-0445.
```

```
7520-STORE-NONHOSP.
    MOVE 'NON-HOSP-CLAIM'     TO  RECORD-SR-NAME.
    MOVE 445                  TO  RECORD-ID.
    MOVE NON-HOSP-CLAIM       TO  RECORD-DATA.
    CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
                          OWNER-DESCRIPTOR-ONE.
    PERFORM DBLU-STATUS.

7590-DN-EXIT.
    EXIT.

*****************************************************************
*    CLOSE OUT LOAD PROGRAM OPERATIONS HERE.                    *
*                                                               *
*    DISPLAY APPROPRIATE RUN-TIME STATISTICS FROM PROGRAM       *
*    AND DATABASE SYSTEM;  THEN CALL IDMSDBLU WITH A -1         *
*    IN RECORD-ID TO CLOSE HIS FILES AND PUT OUT A CONTROL      *
*    RECORD.                                                    *
*****************************************************************

9999-END.
    DISPLAY SUM-CARDS-IN ' CARDS'.
    DISPLAY SUM-TRANSACTIONS '  TRANSACTIONS'.

    MOVE -1 TO RECORD-ID.
    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

This section contains the following topics:

Overview (see page 607)
IDMSRSTT BUFSIZE (see page 609)
IDMSRSTT RECNAME (see page 610)
IDMSRSTT SETPTR (see page 615)
IDMSRSTT FIELD (see page 617)
IDMSRSTT END (see page 622)
END (see page 622)

Overview

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
- **A 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 and RESTRUCTURE CONNECT utility statements.

Note:

- For more information about generating the IDMSRSTT macro statements automatically, see IDMSRSTC (see page 428).
- For more information on restructuring a database, see RESTRUCTURE (see page 298) and 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

Statement	Description
IDMSRSTT SETPTR	For each record type being modified: ■ One for each pointer in the record if any pointers are being changed ■ One that specifies ALL if no pointers are being changed
IDMSRSTT FIELD	For each record type being modified: ■ 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 ■ One that specifies ALL if the record is fixed length and no fields are being changed
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.

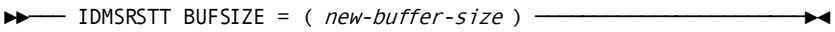
Note: For more information about the IDMSRPTS reports, see IDMSRPTS (see page 403).

IDMSRSTT BUFSIZE

The IDMSRSTT BUFSIZE statement specifies the sizes of the record buffers to be used during RESTRUCTURE utility processing.

te for the *old-buffer-size* and *new-buffer-size* parameters. For example, round the calculated record length up to the nearest multiple of 100.

IDMSRSTT BUFSIZE Syntax



IDMSRSTT BUFSIZE Parameter

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

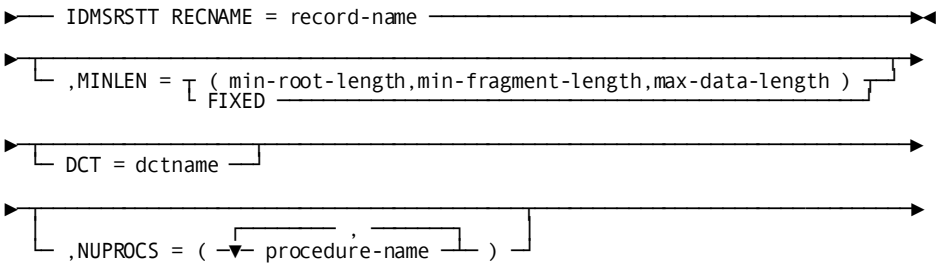
Estimating buffer-size values

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.

IDMSRSTT RECNAME

The IDMSRSTT RECNAME statement 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



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 the RESTRUCTURE utility uses a MODIFY statement to rewrite the restructured record to the database.

procedure-name:

Specifies 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 RESTRUCTURE utility 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 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 =

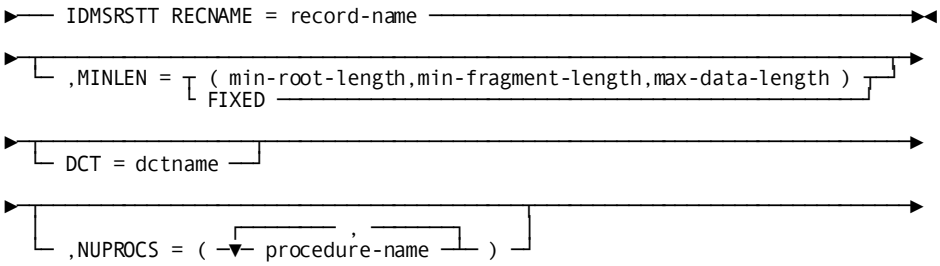
List all procedures to be executed for a record during the restructure process **before a MODIFY statement**, as follows:

- 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.

IDMSRSTT RECNAME Syntax



IDMSRSTT RECNAME Parameter

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 the RESTRUCTURE utility uses a MODIFY statement to rewrite the restructured record to the database.

procedure-name:

Specifies 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 RESTRUCTURE utility 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 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.

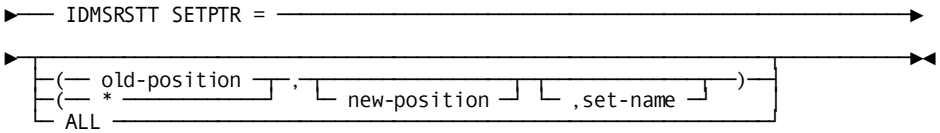
- A procedure that is not specified in the new schema should be included if the procedure is to be executed before a MODIFY statement during the restructure process; such procedures must be added manually to the IDMSRSTT macro statements generated by IDMSRSTC. For example, a procedure that initializes a new data item specified in an IDMSRSTT FIELD statement for the record should be included.

IDMSRSTT SETPTR

The IDMSRSTT SETPTR statement 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

IDMSRSTT SETPTR Syntax



IDMSRSTT SETPTR Parameter

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

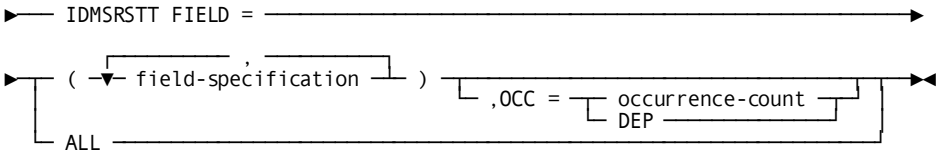
When you add a new pointer or copy an existing pointer to a position that already contains a pointer, the pointer in the position is overwritten.

IDMSRSTT FIELD

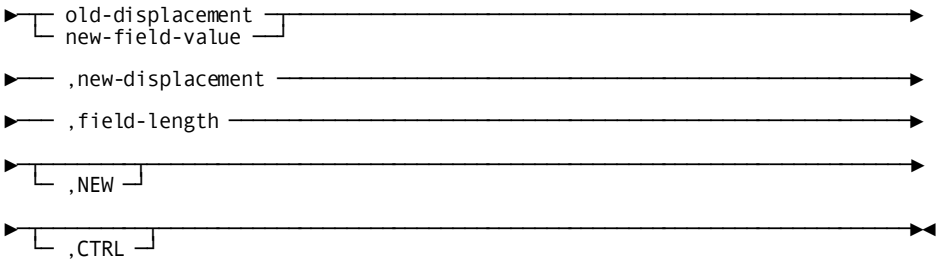
The IdMSRSTT FIELD statement identifies the data fields in a restructured record. Using the IDMSRSTT FIELD statement, you can:

- Add new data fields
- Delete existing data fields
- Move existing data fields to new positions

IDMSRSTT FIELD Syntax



Expansion of field-specification



IDMSRSTT FIELD Parameter

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 following expanded parameters.

,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 can 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" in this section.

,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. Therefore, *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.

IDMSRSTT FIELD 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. Therefore, *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'

Usage	Initial value
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

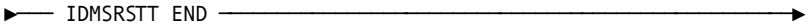
The restructure utility as a general rule cannot expand a field to the right of a decimal point. To do this, the standard procedure would be to first add the needed bytes to the left of the field. After the RESTRUCTURE has been performed, a user-written program must modify each affected record by multiplying the expanded fields by the proper factor to realign the decimal point to the new position.

IDMSRSTT END

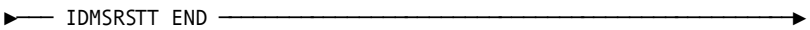
The IDMSRSTT END statement terminates the restructuring information.

IDMSRSTT END must be the next to last macro statement coded for each base restructuring table you assemble.

Syntax



IDMSRSTT END Syntax

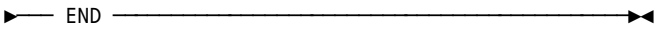


END

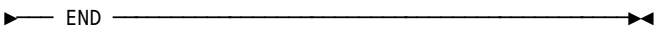
The END statement terminates the assembler input.

END must be the last macro statement coded for each base restructuring table you assemble.

Syntax



END Syntax



Appendix C: Common Facilities for Distributed Transactions

This section contains the following topics:

Overview (see page 625)
Reporting on Distributed Transactions (see page 625)
Manual Recovery Input Control File (see page 627)
Manual Recovery Output Control File (see page 629)
Execution JCL Changes (see page 629)

Overview

This appendix discusses the recovery utilities that report on distributed transactions and that support the use of a manual recovery control file for use with the two-phase commit feature. This appendix describes these common enhancements as they apply to the following recovery utility statements:

- EXTRACT JOURNAL
- FIX ARCHIVE
- MERGE ARCHIVE
- PRINT JOURNAL
- ROLLBACK
- ROLLFORWARD

Reporting on Distributed Transactions

A distributed transaction journal record consists of a fixed portion and up to three variable arrays of data. The fixed portion contains the distributed transaction identifier (DTRID) and a local branch ID (BID), which identifies an individual branch of the distributed transaction. The fixed portion can be followed by any of the following:

- A list of local transaction identifiers (LIDs), one for each branch of the transaction that made local database changes
- A list of external transaction identifiers if the transaction is known externally by another identifier, such as an XA XID or an RRS URID
- A list of interests that other resource or transaction managers have in the distributed transaction

The recovery utilities report some or all of the above information in their detailed report and list distributed transactions that were incomplete at stop time in their summary report.

The following example shows the output that is produced by PRINT JOURNAL REPORT FULL when it encounters a typical DCOM record. If the REPORT TERSE option is specified, neither external transaction identifiers nor resource manager interests are included. Other recovery utilities show similar information.

```
NODE SYSTEM74 DTRID-BID   SYSTEM74::01650C9509CE38A3-01650C90A4CDA0BD       DCOM
  LOC_ID    10016  PGM_ID PROCDISM
  RRS URID  B8DEBCA57E84B6700000000D01020000                               *...v=d.........              *
  RM NAME   SYSTEM74::RRS_RMI   TYPE RRS    ROLE SDSRM STATE InDoubt      FLG1/2 0001 EXITS 40 0034000000000000
            D9D9E240C2404040404040404040404040B8DEBCA57E84B6700000000D01020000  *RRS b             ...v=d........*
            18C1E3D94BC2F8F9F0F9F7F8F6C1F2C1F8C1C1F4F04BC9C2D4                 *.ATR.B8909786A2A8AA40.IBM      *
  RM NAME   SYSTEM73::DSI_CLI   TYPE IDMS   ROLE CRM   STATE InDoubt      FLG1/2 0000 EXITS 76 0000000000000000
            E2E8E2E3C5D4F7F301650C90A4CDA90400000000080000001650C2E949172E101 *SYSTEM73....u.z............mj...*
            650c9509cE38A380000000000000000E2E8E2E3C5D4F7F303D9C51D340C4C3   *..n..t..........SYSTEM.73REAL DC*
            400000000000000D5C4E2E3C5D4F7F3000000000000000000000000          * .........NDSYSTEM73............*
            00000000                                                         *....
```

A brief description of the report's contents follows. For an in-depth discussion of the meaning of this report, see *CA IDMS Database Administration Guide*.

- *Node SYSTEM74* identifies the name of the system that produced the journal entry, in this case, *SYSTEM74*.
  - *DTRID-BID SYSTEM74::01650C9509CE38A3-01650C90A4CDA0BD* identifies the DTRID and the BID of the top-level branch of the distributed transaction for which the DCOM record was written. The DTRID is *SYSTEM74::01650C9509CE38A3* and the BID is *01650C90A4CDA0BD*.
  - *DCOM* is the type of distributed transaction journal record that is being reported.
- *LOC_ID 10016* identifies the work done by a local transaction branch that is included in the distributed transaction. In this case, the local identifier is *10016*.
  - *PGM_ID PROCDISM* identifies the name of the application program that started the local transaction branch. In this case, the program is *PROCDISM*.
- *RRS URID B8DEBCA57E84B6700000000D01020000* identifies the transaction, as it is known externally.
- *RM NAME SYSTEM74::RRS_RMI* identifies a resource manager that has registered an interest in the distributed transaction. In this case, the resource manager is *RRS*.
  - *TYPE RRS* indicates that the RM type is *RRS*.
  - *ROLE SDSRM* indicates that this interest is the controlling interest for the transaction, and therefore RRS is the transaction's coordinator.
  - *STATE InDoubt* indicates the interest's state. In this case, the interest is in an *InDoubt* state.

- *FLG1/2 0001* displays flags that are used to restart the transaction following a system failure.
- *EXITS 40 0034000000000000* shows the exits that have been registered by the resource manager and the responses returned by the exits that have already been called during the life of the transaction.
- *D9D9E240C2...* shows the data (in hex and character format) that the resource manager wishes preserved should it be necessary to restart the transaction following a system failure. This information will vary depending on the resource manager that registered the interest.

- *RM NAME SYSTEM73::DSI_CLI* identifies a resource manager that has registered an interest in the distributed transaction. In this case, the resource manager is a CA IDMS system named *SYSTEM73*.
  - *TYPE IDMS* indicates the type of the resource manager.
  - *ROLE CRM* indicates that this interest is not a controlling interest for the transaction. Therefore, the associated resource manager *(SYSTEM73)* is a participant in the transaction.
  - *FLG1/2 0000* displays flags that are used to restart the transaction following a system failure.
  - *EXITS 76 0000000000000000* shows the exits that have been registered by the resource manager and the responses returned by the exits that have already been called during the life of the transaction.
  - *E2E8E2E3C5...* shows the date (in hex and character format) that the resource manager wishes to have preserved if it is necessary to restart the transaction following a system failure. This information will vary depending on the resource manager that registered the interest.

Manual Recovery Input Control File

A manual recovery input control file can be used to explicitly specify whether an InDoubt distributed transaction should be committed or backed out. Its use is optional, but if included in a utility's execution JCL, it will be used as input to the following recovery operations:

- EXTRACT JOURNAL (unless ALL is specified)
- FIX ARCHIVE
- MERGE ARCHIVE (if COMPLETE is specified)
- PRINT JOURNAL
- ROLLBACK
- ROLLFORWARD (unless ALL is specified)

The Input Control file only affects InDoubt transactions. It will have no effect on transactions that were in a Commit or BackOut state; or forgotten.

If the Input Control File is not used to commit or back out an InDoubt transaction, its state will remain in doubt. Its updates will still be applied or remain applied on the database and no journal records will be generated for that transaction. The updates will be complete and the database intact, that is, no broken pointers. Provided the database is not used and updated, the InDoubt transactions can still be backed out or formally committed in a later recovery job.

The file contains 80-byte records whose format is:

```
<DTrid><Action>
```

Where <Dtrid> is a 26-character display-format DTRID and <Action> is either COMMIT or BACKOUT. If more than one record specifies the same DTRID value, all but the last one are ignored.

The following example specifies that the transaction identified by DTRID SYSTEM74::01650C9509CE38A3 should be backed out:

```
SYSTEM74::01650C9509CE38A3 BACKOUT
```

If manual control input entries are used in a recovery operation that creates an output journal file (FIX ARCHIVE, EXTRACT JOURNAL, and MERGE ARCHIVE), then additional distributed transaction journal records will be written to the output file to complete the transaction in the specified way.

The following is a sample of the report generated by FIX ARCHIVE. It lists entries in the manual recovery input control file and shows the effect of those entries in its summary report. In this example, the distributed transaction identified by CICSCICS::B8AD18E5A9BF0F41 is committed by the generation of new DCOM and DFGT journal records.

```
Input Control Records:
   CICSCICS::B8AD18E5A9BE1300   BACKOUT

   CICSCICS::B8AD18E5A9BF0F41   COMMIT

   . . .
Incomplete Distributed Transactions At Stop Time:
     NODE     DTRID-BID                                    STATE   ACTION
**** SYSTEM74 CICSCICS::B8AD18E5A9BF0F41-016507A67C2E6D53 InDoubt Commit
*GEN SYSTEM74 DTRID-BID  CICSCICS::B8AD18E5A9BF0F41-016507A67C2E6D53       DCOM
              LOC_ID         28 PGM_IE CICSDML1
*GEN SYSTEM74 DTRID-BIT  CICSCICS::B8AD18E5A9BF0F41-016507A67C2E6D53       DFGT
```

Manual Recovery Output Control File

Since a manual recovery control file is an 80-byte card image file, you can create it with a text editor. A prototype control file can also be optionally created by any of the following recovery operations if a manual recovery output control file is included in its execution JCL:

- EXTRACT JOURNAL
- FIX ARCHIVE
- MERGE ARCHIVE
- PRINT JOURNAL
- ROLLFORWARD (unless FROM EXTRACT is specified)

When a control file is generated, an entry is created for every distributed transaction whose final state is InDoubt. Automatically generated entries always specify that the transaction should be backed out. It is expected that the resulting file will be edited prior to using it as input to a recovery operation.

Execution JCL Changes

Manual recovery control files are optional, so no execution JCL changes are necessary unless their use is desired.

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a manual recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.