

ClearPath Enterprise Servers

Enterprise Database Server for ClearPath MCP Utilities Operations Guide

ClearPath MCP 19.0

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product or related information described herein is only furnished pursuant and subject to the terms and conditions of a duly executed agreement to purchase or lease equipment or to license software. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Notice to U.S. Government End Users: This software and any accompanying documentation are commercial items which have been developed entirely at private expense. They are delivered and licensed as commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Use, reproduction, or disclosure by the Government is subject to the terms of Unisys' standard commercial license for the products, and where applicable, the restricted/limited rights provisions of the contract data rights clauses.

Contents

Section 1. Introduction

Documentation Updates	1-2
What's New?	1-2
Utility Tasks	1-2
Initializing the Database.	1-3
Running Enterprise Database Server Utilities	1-3
Controlling the Database	1-3
Maintaining the Database	1-3
Verifying the Database	1-4
Translating Messages	1-4
Using the Remote Database Backup Facility	1-5
Using the Open Distributed Transaction Processing Product	1-5

Section 2. Control File

Control File Provisions.	2-1
Control File Structure	2-2
Table of Contents	2-2
Text Directory/Records	2-4
Guard File Directory/Records	2-5
Structure Directory	2-5
Structure Records.	2-5
Partition Records	2-6
Control File Functions	2-6
Controlling Database Interlock	2-7
Storing Information	2-7
Checking Compatibility	2-8
Verifying Interfile Version Compatibility.	2-8
Handling Discontinuities	2-9
Handling Audit Block Serial Number (ABSNS) Rollover	2-10
Control File Interface with Database Software	2-10
DASDL Compiler.	2-10
Accessroutines Program	2-10
Database Recovery	2-11
DMUTILITY Program.	2-13
REORGANIZATION Program	2-16
Database Certification Program	2-17

Section 3. Using the DMSUPPORT Library

Entry Points 3-1
Entry Point Declarations 3-2
Example Programs 3-3

Section 4. Using DMUTILITY

DMUTILITY Commands 4-1
Running DMUTILITY 4-3
Dump Media: Tape Dump Versus Disk Stream Dump 4-5
 Tape Dumps 4-5
 Disk Stream Dumps 4-6
 Operator Interface to DMUTILITY for Tape Dumps 4-8
 Operator Interface to DMUTILITY for Disk Stream
 Dumps 4-8
 Restarting DMUTILITY for Tape and Disk Stream
 Dumps 4-8
Continuing DMUTILITY 4-10
DMUTILITY Error Handling 4-11
 Tape Input/Output Errors During Dump 4-11
 Disk Stream Input/Output Errors During Dump 4-12
 Tape Input/Output Errors During Load 4-12
 Disk Stream Input/Output Errors During Load 4-13
 Database Disk Input/Output Errors 4-13
 DMUTILITY Warnings During Dump 4-16

Section 5. Initializing and Maintaining

Initializing and Maintaining the Control File 5-1
 Running DMCONTROL 5-1
DMCONTROL Statement 5-3
 Potential Problems with RECOVER UPDATE 5-26
 Control File Recovery 5-27
 Control File Recovery and Change of Family 5-28
 Control File Recovery and Change of Population
 Control Attributes 5-29
DMUTILITY CANCEL Statement 5-29
Initializing Database Files 5-30
 Rules for Initialization 5-31
DMUTILITY INITIALIZE Statement 5-32
REDISTRIBUTE Command 5-34
MIGRATEDB Command 5-36

Section 6. Backing Up a Database

Tools Available for Creating and Managing Database Backups 6-2

Understanding the Database Backup Process 6-3

Tasks Related to Creating and Managing Database Backups 6-6

DUMP and APPEND Commands (DMUTILITY) 6-9

 Dump Option 6-11

 OFFLINE Option 6-16

 INCREMENTAL Option 6-18

 ACCUMULATED Option 6-19

 Dump Clause 6-20

 Dump List Clause 6-22

 Dump Selector Clause 6-24

 Portion Selector Clause 6-25

 BY FAMILYINDEX Option 6-26

 Dump Tape Specification 6-27

 Multidump Tape Specification 6-31

 Dump Disk Specification 6-31

 DUMP Command Examples Where the Backup
 Medium Is Single Dump Tape 6-34

 DUMP and APPEND Examples Where the Backup
 Medium Is Multidump Tape 6-37

 DUMP Command Examples Where the Backup
 Medium Is Disk 6-40

 DUMP Command Examples Where the Backup
 Medium Is Both Single Dump Tape and Disk 6-43

VERIFYDUMP Command (DMUTILITY) 6-44

Copying Database Backups 6-45

COPYDUMP Command (DMUTILITY) 6-46

DUPLICATEDUMP Command 6-50

TAPEDIRECTORY Command (DMUTILITY) 6-53

TAPESET DIRECTORY Command (DMUTILITY) 6-54

Cataloging the Information in Database Backups 6-57

DMDUMPDIR Program 6-60

 ENABLE Command 6-61

 DISABLE Command 6-62

 ADD Command 6-62

 DELETE Command 6-62

 LIST and WRITE Commands 6-63

BUILDDUMPDIRECTORY Command (DMUTILITY) 6-68

Recovering Database Backup Catalog Information 6-69

Quick-Reference Information 6-69

Section 7. Reorganizing the Database

Understanding Types of Reorganization 7-2
 Garbage Collection 7-2
 File Format Conversion 7-3
 Record Format Conversion 7-4
Understanding the Database Reorganization Process 7-5
Understanding the Reorganization Algorithm 7-15
Running the BUILDREORG Utility 7-20
 Running Through a Batch Job 7-21
Syntax for the BUILDREORG Utility 7-24
 Using the BUILDREORG UPDATE Option. 7-27
 Using an Alias Name. 7-28
 Using the Central Data Set GENERATE Statement 7-29
 Using the GENERATE Statement. 7-29
 Using the Central Data Set Sequence Statement 7-51
 Using the Reorg Global Control Statement. 7-53
Running the REORGANIZATION Program. 7-66
 Preparing to Reorganize 7-67
 Understanding the Phases of Reorganization 7-67
 Starting the REORGANIZATION Program 7-68
 Reorganizing a Nonusercoded Database 7-69
 Using the Transaction Processing System (TPS) During
 Reorganization 7-69
 Availability of Structures During Reorganization. 7-70
 Updating During Generation: The Fixup Process 7-73
 Finishing the Reorganization Process 7-75
 Reorganization Status Report. 7-77
 Terminating and Reccessing the REORGDB
 Reorganization Process 7-78
 Restarting a Reorganization 7-79
 Rebuild Recoveries and Reorganizations 7-80
 Rollback Recoveries and Reorganizations 7-83
 Row Recoveries and Reorganizations. 7-83
 Reorganization I/O Errors. 7-83
 Reorganization Data Errors 7-84
 Displaying Reorganization Status 7-86
 Enhancing Reorganization Performance 7-87
 Disk Storage Requirements. 7-89
 Limitations of Database Reorganization 7-91

Section 8. Recovering the Database

RECOVER Statement (DMUTILITY) 8-1
 Partial Database Recovery. 8-6

Designating How to Recover	8-8
Reconstructing Rows Using the Quickfix Process	8-12
Designating What to Recover and Where	8-12
Designating a Backup Dump	8-14
Partial Database Recovery of Partition Files	8-18
Whole Database Recovery	8-18
Recovery Methods	8-18
Database Recovery Using Incremental and Accumulated Dumps	8-39
Running Recovery	8-40
Visible Recovery Commands	8-42
ALLOWEDCORE = <integer> Command	8-43
OVERLAYGOAL = <decimal value> Command	8-44
WRITEDELAYFACTOR = <decimal value> Command	8-44
STATUS Command	8-46
STATISTICS and STATISTICS CLEAR Commands	8-49
COPY Statement (DMUTILITY)	8-51
Tape Dumps	8-62
Disk Dumps	8-63
DMUTILITY TAPECLONE Statement	8-64
STRUCTURECLONE Statement (DMUTILITY)	8-64

Section 9. Copying Audit Files

Why Copy Audit Files?	9-2
Facilities Provided by the COPYAUDIT Program	9-3
Initiating the COPYAUDIT Program	9-3
Checking the Results of a COPYAUDIT Run	9-5
Methods for Copying Audit Files	9-6
Using the QUICKCOPY Command	9-11
Using the COPY Command	9-23
Using the DIRECTORY Command to Display Audit File Tape Directories	9-26
Using the VERIFY Command to Verify Audit File Contents	9-27
Quick-Reference Information	9-29

Section 10. Printing, Viewing, and Extracting Audit Information

PRINTAUDIT Program Overview	10-1
Initiating the PRINTAUDIT Program	10-2
Overview of PRINTAUDIT Commands	10-6
Basic Command Syntax	10-6
Designating Intervals	10-9
Designating a Time Interval	10-10
Designating a Serial Number Interval	10-13
Designating a Relative Block Interval	10-14

Selecting Audit Data	10-15
Selecting Records by Stack Number	10-15
Selecting Records by Program Mix Number	10-16
Selecting Records by Program Identifier	10-17
Selecting Records by Structure Identifier or Block Number	10-18
Specifying an Alias Name in a Structure Identifier	10-19
Selecting Records by Field	10-20
Selecting Records by Record Type	10-21
Generating a Customized Version of the PRINTAUDIT Program	10-26
Developing the ALGOL Code	10-28
Using the SELECT Statement with the ALGOL Code	10-31
Examples of PRINTAUDIT Commands	10-32
Quick-Reference Information	10-39

Section 11. Checking Integrity and Performance

Database Certification	11-1
Running Database Certification	11-2
DBCERTIFICATION Command	11-5
ONLINE Command	11-6
HELP Command	11-6
INTERNAL FILES Command	11-7
OPTIONS Command	11-8
SORT Command	11-8
UPPERCASE Command	11-9
LOWERCASE Command	11-9
QUIT Command	11-10
CERTIFY Command	11-10
CERTIFY Options for Structure Types	11-14
Data Sets	11-14
Sets and Subsets	11-22
DMUTILITY DBDIRECTORY Statement	11-25
DMUTILITY DISABLE/ENABLE Statement	11-28
DMUTILITY LIST/WRITE Statement	11-30

Section 12. Communicating with the Database

Entering Visible DBS Commands	12-1
Visible DBS Commands	12-2
Errors and Warnings	12-2
DBS STATUS Command	12-2
DBS CHANGE Command	12-5
AUDIT ANALYZE AFN Command	12-9

AUDIT PROCESSOR TIMES Command	12-10
AUDIT CLOSE Command	12-11
AUDIT SCRATCHPOOL Command	12-13
AUDIT QUICKCOPY MAXFILESPEPTAPE Command	12-14
AUDIT QUICKCOPY SYNCTAPESET Command	12-15
CPSTATS Command	12-16
GARBAGE COLLECT Command	12-17
Alternative to a Reorganization	12-18
Disk Storage Requirements	12-18
How the GARBAGE COLLECT Command Works	12-18
After the GARBAGE COLLECT Operation	12-19
LOCKSTATISTICS Command	12-21
SNAPSHOT Command	12-21
STATISTICS Command	12-23
STATUS STRUCTURE Command	12-24
STRUCTURE CHANGE Command	12-27
STATUS HISTORY Command	12-34
STATUS MIX Command	12-39
STATUS RDB Command	12-42
STATUS REORG Command	12-43
SUPERCP RESTOREDBFILES Command	12-44
USEREORGDB TERMINATE Command	12-45
USEREORGDB DISCARD Command	12-45
DIAGNOSTICS Command	12-46

Section 13. Maintaining Databases Containing Large Objects

Tank Sizes Available for LOBS	13-1
LOBANALYZE Command (DMUTILITY)	13-2
LOBCLEANUP Command (DMUTILITY)	13-3
LOBCOMBINE/LOBSQUASH Command (DMUTILITY)	13-3
Interpreting the LOBANALYZE Report	13-4

Section 14. Using a Quiesce Database

Tasks Related to Quiesce Databases	14-1
QUIESCE Command (DMUTILITY)	14-2
RESUME Command (DMUTILITY)	14-5
QUIESCE QDC Command (DMUTILITY)	14-6
CREATE QDC Command (DMCONTROL)	14-12
RESTORE FROM QDC Command (DMCONTROL)	14-19
Creating Incremental/Accumulated Dumps from a Quiesce Database	14-21
Using a Quiesce Database Copy as a Recovery or a Copy Source	14-22

High Availability QUIESCE	14-26
QUIESCE HISTORY Option of the WRITE Command	14-30
CFRESTORE Command (DMUTILITY)	14-31
Quick-Reference Information	14-31
Section 15. Using Database Tape Encryption	
Architecture	15-1
Encryption Algorithms	15-2
DASDL Syntax	15-3
DMUTILITY Syntax	15-3
COPYAUDIT Syntax	15-4
DASDL Example	15-4
DMUTILITY Examples	15-5
COPYAUDIT Examples	15-7
Section 16. Using Permanent Directory Databases	
Creating a Permanent Directory Database	16-1
Reorganizing a Permanent Directory Database	16-3
Working with Dumps	16-4
Section 17. Loading and Dumping Conventional Files	
Steps for Using LOADDUMP	17-2
LOADDUMP	17-3
COBOL74 or COBOL85 MOVE Algorithm	17-12
Compiler Control Options for LOADDUMP	17-12
Section 18. Compiling Software	
Compilation WFL Job Parameters	18-1
DMSUPPORT	18-3
RECONSTRUCT	18-3
DMINTERPRETER	18-4
RMSUPPORT	18-4
Section 19. Controlling Partitioned Records	
Partition Directory Overview	19-1
Partition Directory Details	19-2
Audit and Recovery Considerations	19-3

Section 20. Using the Audit Reader Library Interface

Audit Reader Library Overview 20-1

Using the ALGOL Interfaces 20-2

 ALGOL Array Reference AUDIT_INFO [0]. 20-3

 Linkage and Operational Information 20-3

 Logical Audit File Information. 20-8

 Internal Buffer Information. 20-10

 Audit Section Information 20-10

 Block List Information 20-11

 ALGOL Array Reference AUDIT_BUFFERS [0, 0]. 20-12

Entry Points 20-12

 AUDIT_OPEN Entry Point Parameters 20-13

 AUDIT_CLOSE Entry Point Parameters 20-15

 AUDIT_NEXT_ABSN Entry Point Parameters 20-16

 AUDIT_RANDOM_ABSN Entry Point Parameters 20-17

 AUDIT_NEXT_RECORD Entry Point Parameters 20-18

Error Results. 20-20

Section 21. Database Events Management

Events Management Overview 21-1

 Event Log Files 21-2

 Using the Programmatic Interface 21-5

Section 22. Logging Data Access

System Logging Options 22-2

DASDL LOGACCESS Option 22-2

Enabling the LOGACCESS Option 22-5

Changing the LOGACCESS DMVERB List 22-8

 Enabling LOGACCESS Option with Visible DBS 22-8

 Using the DMCONTROL LOGACCESS Command 22-9

LOGACCESS Analysis 22-9

Section 23. Database Encryption

Database Encryption Components and Interdependencies 23-1

Using Database Encryption 23-4

Error Handling 23-11

Performance Impact and Best Practices 23-14

Section 24. Troubleshooting

Events That Cause Halt/Load Recoveries to Fail 24-1

Handling I/O Errors During a Halt/Load Recovery 24-1
Enterprise Database Server Errors During a Halt/Load Recovery . . . 24-2
DATAENCRYPT Option Error during a DASDL Compilation 24-2
REQUIRES *PK DISK Error during a Reorganization 24-2

Appendix A. Common Syntactic Items

Appendix B. Interpreting Database Statistics

Header B-1
Buffer Statistics B-2
Input/Output (I/O) Statistics. B-4
VSS2 Optimization B-9
VSS3 Optimization B-9
Database Usage Statistics B-9
Structure Lock Statistics B-11
Audit Statistics (First Part). B-12
Audit Statistics (Second Part) B-13
Transaction Statistics B-15
Global Lock Statistics B-16
Control Point Buffer Statistics B-17
Using Statistics B-18

Appendix C. COPYAUDIT Error Messages

COPYAUDIT Errors C-2
 COPYAUDIT Fatal Errors C-2
 COPYAUDIT Nonfatal Errors C-22

Appendix D. Using Mirrored Disks for Disaster Recovery

Which Enterprise Database Server Files to Mirror? D-1
Environment Considerations D-2
Backup Procedures D-2
Recovery Procedures D-3
 Control File Integrity D-3
 Data File Integrity D-3
 Audit File Integrity D-3

Appendix E. Understanding Railroad Diagrams

Railroad Diagram Concepts E-1
 Paths E-1
 Bold Faced Words E-2

Constants and Variables E-2
Constraints E-3
Following the Paths of a Railroad Diagram E-5
Railroad Diagram Examples with Sample Input E-6

Index 1

Figures

6-1.	Dump Tape Directory	6-57
7-1.	Creation of a New Database Description File	7-6
7-2.	Creation of a Reorganization Description File (Scenario 1)	7-10
7-3.	Creation of a Reorganization Description File (Scenario 2)	7-11
7-4.	Creation of a REORGANIZATION Program	7-12
8-1.	Standard Buffer Access/Write Operation Scenario	8-45
8-2.	Buffer Access/Write Operation Scenario with WRITEDELAYFACTOR Effect.	8-46
8-3.	Sample Visible Recovery Status Display.	8-47
8-4.	Sample Visible Recovery Statistics Report	8-51
14-1.	QUIESCE Command in a Database System Environment	14-4
14-2.	Single Server, One Live Database, Two Quiesce Database Copies.	14-9
14-3.	Single Server, One Live Database, Two Quiesce Database Copies, Live Database Backed Up from QDC ON BACKUPPK.	14-11
14-4.	Single Server, One Live Database, Two Quiesce Database Copies.	14-14
14-5.	Single Server, One Live Database, Two Quiesce Database Copies, Live Database Backed Up from QDC ON BACKUPPK.	14-16
14-6.	Rebuild of a Live Database Using a Quiesce Database Copy	14-21
14-7.	Using a Quiesce Database Copy as a Recovery Source.	14-23
17-1.	LOADDUMP Components	17-2
23-1.	Compile-Time Database Encryption Configuration.	23-2
23-2.	Run-Time Database Encryption Configuration.	23-3

Tables

4-1.	DMUTILITY Commands	4-1
4-2.	Elements of RETRYIO Messages	4-14
6-1.	Tasks Related to Creating and Managing Database Backups	6-2
7-1.	Control File Handling for Rebuild Recoveries	7-83
8-1.	Recover Specification and Source for Tape Dumps	8-28
8-2.	Recover Specification and Source for Disk Dumps	8-29
9-1.	Tasks That Can Be Accomplished by Using the COPYAUDIT Program	9-3
10-1.	Record Type Mnemonics	10-22
10-2.	PRINTAUDIT File Equations	10-27
10-3.	Variables Available to the USERPROCEDURE and USERWRAPUP Procedures	10-28
10-4.	Audit Data Block Information	10-29
10-5.	Stopper Pattern Information	10-30
14-1.	Tasks Related to Quiesce Databases or Quiesce Database Copies	14-1
18-1.	Compilation WFL Job Parameter Keywords	18-2
20-1.	Error Results	20-21
22-1.	Tasks Related to Using the LOGACCESS specification	22-1
22-2.	DMVERB Access	22-6

Section 1

Introduction

The Enterprise Database Server is a software system that enables you to create databases and maintain relationships between database elements. The Enterprise Database Server utility programs and the tasks they enable you to perform are the subject of this manual.

This section covers the

- Purpose, audience, and conventions for this guide
- Documentation update information
- Descriptions of the tasks you can perform using Enterprise Database Server programs



This manual documents the features of the Enterprise Database Server for ClearPath. It includes the features for scalability, capacity, and availability in addition to the traditional features. An XE icon in the left margin designates an explanation for one of these features.

For information about Enterprise Database Server for ClearPath MCP capabilities, refer to the *Enterprise Database Server for ClearPath MCP Getting Started and Installation Guide*.

Conventions



Throughout this manual, you can locate information about XE features by looking for

- An XE icon in the left margin
- Boldfaced words in railroad diagrams and examples used to illustrate syntax

An explanation of railroad diagrams is in [Appendix E, Understanding Railroad Diagrams](#).

Documentation Updates

This document contains all the information that was available at the time of publication. Changes identified after release of this document are included in problem list entry (PLE) 19223051. To obtain a copy of the PLE, contact your Unisys representative or access the current PLE from the Unisys Product Support website:

<http://www.support.unisys.com/all/ple/19223051>

Note: If you are not logged into the Product Support site, you will be asked to do so.

What's New?

The following table identifies new and revised information for this release.

New or Revised Information	Location in Guide
Added the <supercp restoredbfiles command> to the list of Visible DBS Commands	SUPERCP RESTOREDBFILES Command
Updated the rules for initializing database files	Initializing Database Files
Added new information on the encryption key set	Using Database Encryption
Added information on the RESTARTSORT option	Running the BUILDREORG Utility
Add the BUSAIO record type mnemonic	Table 10-1
Added information on the PUBLICIO current event	Section 12, Communicating with the Database Section 21, Database Events Management

Utility Tasks

The Enterprise Database Server utility programs work together to enable you to perform the following tasks:

- Initialize database files.
- Wholly or partially dump database files.
- Reorganize database files to minimize space or reformat records.
- Wholly or partially recover databases manually or automatically.
- Copy, print, or display audit files.
- Verify the integrity of database files, monitor database performance, and dynamically adjust performance parameters.
- Load and dump to and from conventional files.

The following is a brief description about the Enterprise Database Server programs.

Initializing the Database

Each database must have a control file associated with it. The control file is used to store database parameters and other control information. SYSTEM/DMCONTROL is the control file maintenance program. The control file is initialized by using the DMCONTROL statement INITIALIZE. This statement creates a new control file using information from the database description file. Once the control file is initialized, the database files can then be initialized using the SYSTEM/DMUTILITY *INITIALIZE* statement.

Running Enterprise Database Server Utilities

You can run the Enterprise Database Server utility programs by using the unique command syntax specific to each task.

Each database has a DMSUPPORT library that is used by Enterprise Database Server software. User programs can also use the DMSUPPORT library to obtain additional exception information. The first user to invoke a database also invokes this code file; the file can then be used by multiple users working on the same database.

Controlling the Database

The Enterprise Database Server programs require up-to-date database information such as database parameters, audit control, and status information. This control information is located in a file called the control file. Each database must have a control file, and most Enterprise Database Server programs use information stored in this control file. It is very important that you become thoroughly familiar with the interaction between the control file and Enterprise Database Server programs.

Maintaining the Database

Maintenance involves dumping, reorganizing, and recovering the database; copying and printing audit files; and verifying database integrity.

Databases can be dumped to tape as a means of backing them up. DMUTILITY enables you to dump all or part of a database, dump online or offline, maintain a Dump Directory, and print the directory of a tape.

The Enterprise Database Server REORGANIZATION program is used when physical changes must be made in the structure of the database files. There are three types of reorganization. The first, and most often used, is called garbage collection. It allows you to consolidate unused space and return this space to the system. The second type, file format conversion, allows you to change those attributes of database files that do not affect record formats. The third type, record format conversion, allows you to change the format of an existing data set, global data, set, or automatic subset record.

The Enterprise Database Server generally recovers itself after a system crash. However, when manual recovery is needed, the DMUTILITY, DMRECOVERY, DMDATARECOVERY, and RECONSTRUCT programs work together to provide either partial or whole database recovery.

Partial database recovery allows you to recover rows or structures of a database using backup dumps and the audit trail to reconstruct the damaged areas. A convenient *quick fix* recovery method uses a reverse audit scan process to correct rows locked by write errors. The database can be used during partial database recovery.

Whole database recovery uses a REBUILD process, a ROLLBACK process, or both processes. The REBUILD process uses the database files from a backup dump and applies audit trail afterimages to bring the database forward in time. If this is unsuccessful, or if the database is an unaudited database, the DMUTILITY *COPY* statement can be used to recover a corrupted database. The ROLLBACK process is used to move the database backward in time.

SYSTEM/COPYAUDIT and SYSTEM/PRINTAUDIT are used as audit tools. COPYAUDIT copies audit files from one medium to another. PRINTAUDIT prints or displays whole or partial audit files.

Verifying the Database

Several pieces of Enterprise Database Server software help to verify the integrity of database files. Database Certification ensures the integrity of Enterprise Database Server data structures. The dbaTOOLS product analyzes the logical and physical structure of an Enterprise Database Server database.

For more information about dbaTOOLS, refer to the *Software Product Catalog*.

Translating Messages

All of the Enterprise Database Server utility programs use the MultiLingual System (MLS) to obtain text for their messages. This enables output messages to be translated into different natural languages using the MLS utility program SYSTEM/MSGTRANS. The MLS program allows you to access output messages, translate them into one or more languages, and store the messages and translations for future use. The natural language available in the released products is English. In addition, the Enterprise Database Server utilities handle the CONVENTION task attribute for internationalization. This attribute allows users to specify a desired convention definition at run time.

For more information on MLS, refer to the *MultiLingual System Administration, Operations, and Programming Guide*.

Using the Remote Database Backup Facility

Remote Database Backup is a product you can purchase to create and manage an online backup of a database. Remote Database Backup can form an integral part of the disaster recovery plan for your organization.

For information on Remote Database Backup, refer to the *Remote Database Backup Operations Guide*.

Using the Open Distributed Transaction Processing Product

The Open Distributed Transaction Processing product enables the use of global transactions, that is, transactions that affect more than one database. To make an Enterprise Database Server database capable of participating in global transactions, you must add the OPENOLTP option to the DASDL source file, and optionally, add an RMSUPPORT library specification. When you update your database after adding the OPENOLTP option, two data sets are automatically added to your database description file: RX-GLOBAL-TR and RX-SIBDESCS. These two inquiry-only data sets are used internally by the Open Distributed Transaction Processing software to manage global transactions.

For more information on the Open Distributed Transaction Processing product, refer to the *Open Distributed Transaction Processing Programming Guide* and the *Open Distributed Transaction Processing Installation and Administration Guide*.

Section 2

Control File

All Enterprise Database Server databases include a system-maintained file that contains database control information. This file, titled <database name>/CONTROL, must be present whenever the database is used.

The control file provides a convenient means of controlling the operational aspects of a database by performing the following functions:

- Checking compatibility between tailored software and database files
- Verifying that all database data files are at the same level of update
- Storing audit control information, dynamic database parameters, and other information
- Controlling database interlock

Control File Provisions

The provisions provided by the database control file include the following:

- The control file provides a place to store the database in-use bit across halt/loads. This enables the Accessroutines to protect the integrity of unaudited databases by giving an OPENERRORE if the database was OPEN UPDATE and a halt/load occurred, or if the database was discontinued. (In this case, the only general recovery technique is to reload the control file and database data files from a backup dump and reprocess the input.)
- The control file provides a place to store the file version timestamps. These timestamps are checked against the version timestamps in the data-file headers to guarantee that all database files are at the same level of update.
- The control file provides a place to store the format timestamps for each structure in the database. All tailored software verifies that the format timestamp for each structure in the control file matches the format timestamp in the database description file against which it was compiled. This guarantees that the tailored software is compatible with the control file and data files.
- Programs such as DMUTILITY and DMRECOVERY open the control file first, thereby preventing possible deadlock conditions.
- The interlock control is used when the entire database must be used exclusively by one program (REORGANIZATION, for example). Opening the control file exclusively, and sometimes writing a special value to it, allows for database interlock control.
- The control file helps to handle file discontinuities introduced by DMUTILITY *INITIALIZE* and *REORGANIZATION* in database files.

Control File

- The control file provides a place to store dynamic database parameters such as SYNCPOINT, CONTROLPOINT, and ALLOWEDCORE.
- The control file contains the name of the DMSUPPORT code file. The Enterprise Database Server software uses this name when invoking the DMSUPPORT library.
- The control file stores information about active partitions.

Control File Structure

The control file has a fixed block size, and the last word of each block is a software checksum word.

The control file consists of several sections, each of which begins on a block boundary. In order, the sections of the control file are as follows:

1. Table of contents
2. Text directory
3. Text records
4. Guard file directory
5. Guard file records
6. Structure directory
7. Structure records
8. Partition records

These control file sections are described on the following pages.

Table of Contents

The table of contents is in block zero and contains descriptors to the other sections of the control file and miscellaneous control information. These are described in the following paragraphs.

Control File Format Level

The control file format level indicates the format level of the control file. The control file program sets this word equal to a constant when it creates the control file. The value of the constant might change between releases. If so, the control file program will be able to convert the control files of the previous two releases to the current format.

Database Timestamp

The database timestamp is stored in the control file by the control file program when it creates the control file. The control file program obtains the database timestamp from the Data and Structure Definition Language (DASDL) description file. All tailored software checks the database timestamp in the control file to ensure that the software was compiled against the same DASDL description file.

Update Level

The DASDL update level is stored in the control file by the control file program. The control file program obtains the DASDL update level from the DASDL description file. All tailored software checks the update level to ensure it is compatible with the control file.

State Variables

State variables include the database in-use bit, the REBUILD and ROLLBACK flags, the exclusive lock word, and the restart timestamp.

Dynamic Database Parameters

Database parameters include the current values of SYNCPOINT, CONTROLPOINT, and ALLOWEDCORE, among others.

Audit Control

Audit control information includes the current audit file number, the current audit block serial number (ABSN), and the limits and current values for the primary and secondary audit tape serial numbers (when auditing to designated serial numbers) or scratch pool, if specified.

Structure Directory Descriptor

The structure directory descriptor identifies the starting block, the number of blocks, and the number of entries in the structure directory.

Partition Directory Descriptor

The partition directory descriptor identifies the starting block, the number of blocks, and the number of entries in the partition directory part of the control file.

Last Assigned Partition Number

The last assigned partition number gives the unique integer last assigned for a partition number. Normally, it is equal to the number of partition entries, because no provision exists to delete partitions. A value of -1 indicates that the partition directory needs to be initialized. (This occurs when the control file was lost and recovered, but the partition directory has not yet been recovered.)

Text Directory Descriptor

The text directory descriptor identifies the starting block, the number of blocks, and the number of entries in the text directory. If the number of entries is zero, then no text section is present.

Guard File Directory Descriptor

The guard file directory descriptor identifies the starting block, the number of blocks, and the number of entries in the guard file directory. If the number of entries is zero, then no guard file section is present.

Transaction Processing System (TPS) Information

Information to synchronize the database with the transaction processing system is maintained in the control file.

Tape Directory Flag

If the database has a Dump Tape Directory, this information is indicated in the control file.

Audit File Attributes

The physical attributes of the audit trail are kept in the control file. Audit control information also includes TPS information. TPS information consists, in part, of the current transaction file number, transaction block number, and transaction offset.

Text Directory/Records

The text directory points to the various texts stored in the control file. These texts include the DMSUPPORT title, subsystem ID, and primary and secondary COPYAUDIT Work Flow Language (WFL) file titles.

Directory entries point to the text records section, where each entry is a word containing the length of the string, followed by the characters of the string.

Guard File Directory/Records

The guard file directory points to the various guard file titles for the database and any logical databases. The entries point to the guard file records section, where each entry is a word containing the length of the string, followed by the characters of the string.

Structure Directory

The structure directory contains one word for each database structure that maps into a file. This word contains the structure number and the block address and word offset of the structure record for that structure.

Structure Records

This section of the control file contains a record for each structure in the database. Each structure record contains the information described in the following paragraphs.

Creation Timestamp

The creation timestamp is set to the time-intrinsic function TIME(6) during a DMUTILITY *INITIALIZE* or *REORGANIZATION* whenever a new file is created. The creation timestamp is set to the invalid timestamp when the structure entry is first created by the control file program.

Version Timestamps

Each structure record contains two version timestamps that are used for file update version compatibility checking. The time-intrinsic function TIME(6) is used for the version timestamps. All software that updates the database also updates the version timestamp of the data files when the data file is first opened.

Format Timestamp

The format timestamp is set to the time-intrinsic function TIME(6) whenever the physical file format of a structure changes as a result of a DASDL update. All tailored software verifies that the format timestamp for the structure matches the format timestamp in the database description file against which it was compiled. If the structure format changes, *REORGANIZATION* must be run before the new structure can be used. If the record format of the structure does not change, applications do not have to be recompiled.

Structure State Flag

The structure state flag identifies the current state of the file. Values for the structure state flag are NORMAL STATE and CREATION DATE-TIME STAMP NEEDS TO BE AUDITED.

Family Name

Family name identifies the pack family on which the file resides.

Structure Attributes

The attributes kept in the control file include the structure name, buffer and reblocking specifications, number of areas, type (for example, data set), subtype (for example, direct), and format level.

The state information and the current settings for POPULATIONINCR and POPULATIONWARN are also included. State information reflects the most recent automatic action that pertains to the given option.

Partition Records

This section of the control file contains a record for every active database partition. The relative record number corresponds to the unique partition number in the PARTITIONINFO data set record for the partition. Some of the fields in the partition record are the same as for structure records. These fields include the following:

- Creation timestamp
- Version timestamp
- Format timestamp
- Structure state flag
- Family name

In addition, the control file record for each partition includes the partition identifier and the structure number, as explained in the following subsections. Duplicating the partition identifier and the structure number from the PARTITIONINFO ORDERED data set makes the partition directory conveniently accessible to DMUTILITY, even while the database is being updated.

Partition Identifier

The partition identifier is the alphanumeric ID used as the last identifier of the file title.

Structure Number

The structure number is needed to uniquely qualify the partition identifier.

Control File Functions

The database control file functions are described on the following pages.

Controlling Database Interlock

Because some tailored software can operate on database files without actually opening the database—for example, REORGANIZATION—the EXCLUSIVE file attribute is used to coordinate control file updates. When a function must maintain exclusive control of the database across halt/loads, a special value is written in the lock word of the control file. When the function is complete, the control file is unlocked. Exclusive control of the database using the database interlock control is described in the following paragraphs.

Using Recovery Needed Flag

If a database is OPEN UPDATE and a halt/load occurs or the database is discontinued, the database might have lost integrity and can no longer be used safely. A state flag to this effect must be left on disk because all information in main storage has been lost. The control file is used to hold the database in-use bit for both audited and unaudited databases. The database in-use bit prevents the Accessroutines or DMUTILITY from accessing the database when the database was OPEN UPDATE and a halt/load occurred or the database was discontinued.

For audited databases, the database in-use bit causes the Accessroutines to automatically initiate halt/load recovery, and locks out all other software until halt/load recovery completes.

In general, the only safe course for unaudited databases is to reload the database files and control file from a backup dump and reprocess the input.

Using Exclusive Functions

REBUILD, ROLLBACK, and some forms of REORGANIZATION are examples of functions that must have exclusive use of the entire database. Special words in the control file identify the exclusive user of the database and cause other functions to be locked out.

Preventing DMUTILITY or Recovery Deadlock

Halt/load recovery and abort recovery open the database files exclusively to guarantee that the recovery processes can set the end-of-file (EOF) pointer for the database files. Exclusivity is necessary because of the difficulty in adjusting the EOF pointer in any other file information block (FIB) that might have the file open. This also prevents a deadlock condition with DMUTILITY. Either recovery waits for the DMUTILITY program to complete, or DMUTILITY will wait for a running recovery to complete.

Storing Information

The information in the control file controls audit, dynamic database parameters and TPS synchronized recovery.

Audit Control

The current audit file number, audit block serial number, and database in-use bit are stored in the control file. In addition, when auditing to a designated range of tape serial numbers, the range and current serial number (for both primary and secondary audit files) are stored in the control file.

Dynamic Database Parameters

Through the Visible DBS mechanism, several global database parameters, such as ALLOWEDCORE, SYNCPOINT, and CONTROLPOINT, can be interrogated and changed. The control file provides a place to preserve these values when the database is not active.

TPS Synchronized Recovery

For access to databases by way of the transaction processing system (TPS), TPS information needed by the TPS software for synchronized recovery is stored in the control file.

Checking Compatibility

The database timestamp and the DASDL description update level are stored in the control file by the control file program. These values are also stored in the code files of the database software. The tailored database software checks these values to ensure that the software is compatible with the control file.

In addition, each structure has a format timestamp. The format timestamp helps handle cases where structures are added or deleted by way of a DASDL update.

Verifying Interfile Version Compatibility

The control file ensures that all database files are consistent. For example, it guarantees that a data set and all its sets are at the same level of update.

A version timestamp is stored in the header of every Enterprise Database Server data file. A corresponding timestamp is maintained in the control file for each data file. If all files match the control file, then they must match each other.

TIME(6) is used for the version timestamp because of the low probability of a false match. This timestamp helps you determine which version of the file was loaded when the version is wrong. In the file itself, the version timestamp is stored in the header using the DMTIMESTAMP attribute.

Because a halt/load can occur while the Enterprise Database Server is updating the timestamps, there are two file version timestamps in the control file for each database file. The database file matches the control file if it matches either of the version timestamps in

the control file. The two timestamps in the control file are unequal only when all three are being updated. A special *universal* file version timestamp exists which, by definition, matches any timestamp. It is used in special situations; for example, when recovering the control file.

All software that updates a database data file updates the version timestamps before any changes are made to the file. Nothing is done when the files are closed.

If the file version timestamps do not match when the file is opened, the software displays a message and then waits for an acceptance. The display and accept messages give the actual timestamp of the file, as well as the correct timestamp. The user can force the software to accept the file by entering *AX:OVERRIDE*.

To protect the consistency of the database, some internal locks are procured while checking the versions of the data files. This might prevent other users from accessing the database until a version mismatch has been resolved.

For example, when the Accessroutines updates the database, the first thing it does is update the version timestamps of all the files it opens. Assume that another copy of one of the database files is copied in from a backup dump. The file that the Accessroutines is using can automatically be removed from the disk directory, but the Accessroutines is still attached to the file until it closes it. After the Accessroutines finishes and closes all the files, the files on disk will have one inconsistent file. All database software will detect a version timestamp mismatch when opening the inconsistent file and refuse to use it. In this case, if the database is audited, you can rebuild the file using a backup dump and the audit files, or you can use a RECONSTRUCT RESTORE command to fix the version timestamp mismatch. If the database is unaudited, then the database files must be reloaded and the input reprocessed.

Handling Discontinuities

A *discontinuity* means that a change has occurred in a structure and the structure no longer matches its audit image. Database recovery is affected by this mismatch.

Discontinuities affecting audit and recovery are introduced by DMUTILITY *INITIALIZE* and *REORGANIZATION*. The control file manages discontinuities with a state flag and a creation timestamp for each structure. Each time a new file is made by DMUTILITY *INITIALIZE* or *REORGANIZATION*, the structure state flag is assigned the value NEED TO AUDIT NEW CREATION DATE-TIME STAMP and the creation timestamp is changed. After any new file is created, the first program to open the database update causes a structure discontinuity audit record to be audited with the new creation timestamp, and the structure state flags are reset. This makes the discontinuity visible in the audit. DMRECOVERY can then handle the valid situations and give errors for invalid cases.

When a structure no longer exists, or exists in a different format, REBUILD can ignore audit records appropriately. In the case of a reorganization run, REBUILD can wait until it reaches a point in the audit that matches the version of the file loaded from the backup dump. When a file is initialized with DMUTILITY, all subsequent changes are reflected in the audit, and REBUILD can repeat the action of the INITIALIZE and then apply the audit. Thus, discontinuity arising from DMUTILITY initialization can be crossed.

REBUILD can start rebuilding from any time and stop rebuilding at any time. The backup dumps used must be selected so that the creation timestamps of the files loaded from the dumps match the creation timestamps in the audit at the point REBUILD is to stop. When REBUILD stops, all the files must be in phase or concurrent. Because REBUILD can simulate REORGANIZATION, the dumps need not have been taken after the REORGANIZATION. REBUILD can then begin applying audit images at the proper time.

When performing reconstruction, DMDATARECOVERY always goes to the end of the audit. Therefore, the dump tapes selected for this function must contain files that have not been reorganized since the dumps were made. (This rule applies only to the files that are actually being reconstructed.)

ROLLBACK starts with the current database data files and cannot back out through a discontinuity. ROLLBACK stops automatically if a structure discontinuity (STRDC) audit record is encountered.

Handling Audit Block Serial Number (ABSN) Rollover

The audit block serial number (ABSN) in the audit file increases automatically. The ABSN limit is 4,294,967,294. When the ABSN reaches this limit, the Accessroutines opens a new audit file and resets the ABSN to 1.

Control File Interface with Database Software

Many software programs depend upon information in the control file. The ways in which these programs use the control file are described in the following paragraphs.

DASDL Compiler

The control file program is processed by the DASDL compiler as a dependent process after the database description file is locked. If the control file program fails, the DASDL compiler emits a warning. DASDL does not process the control file program if the DASDL compiler control option DMCONTROL is changed to RESET, or when compiling for syntax. SYSTEM/DMCONTROL is initiated by default. If the compiler control option DMCONTROL is RESET, you must run the control file program.

Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information about the DASDL specifications for the control file.

Accessroutines Program

The Accessroutines is the program through which all Enterprise Database Server inquiries and updates are executed. The Accessroutines uses the control file to check the database continuity timestamp, the DASDL update level, and the structure format timestamp for each structure opened.

The following host language commands open a database.

OPEN INQUIRY Command

OPEN INQUIRY does not cause the database in-use bit to be turned on or the audit to be opened. OPEN INQUIRY checks the file version timestamps but does not update them.

OPEN UPDATE Command

The first program that executes OPEN UPDATE causes the database in-use bit to be set, the audit to be opened, and the file version timestamps to be updated. If any structure state flags are initialized, special audit records are written for the creation timestamps of these structures. The state flags are then reset.

All modified data and audit buffers are flushed to disk when all OPEN UPDATE programs close the database even if the database has been opened with the OPEN INQUIRY command.

Database Recovery

For complete file version compatibility protection whenever a database data file is opened for update, the file version timestamps in the control file must be checked and updated. Therefore, all forms of recovery check and update the file version timestamps when the files are opened. The structure format timestamps are also checked. The following processes utilize the control file to check the timestamps. This interaction with the control file is explained on the following pages.

DMDATARECOVERY Program

DMDATARECOVERY is a coroutine of the RECONSTRUCT program. It aids in row recovery by reading the audit trail and applying audit images.

DMDATARECOVERY always reconstructs to the current end of the audit. Because bringing a database file across a discontinuity in the audit is logically impossible, the creation timestamps of the files to be reconstructed from a backup dump must be equal to their creation timestamps in the current control file.

ABORT and Halt/Load Recovery Program

ABORT and halt/load recovery always start with the current database files. Therefore, there are no problems with crossing discontinuities in the audit.

Rollback Process

The rollback recovery process requires the final audit file number in the control file to be correct when it begins; otherwise, it could start with the wrong audit file, and the database could lose its integrity. Starting with the correct audit file is not normally a problem unless the control file is recovered separately (for example, with the control file program) before the rollback process is started. In this case, special care should be taken by the user to ensure that the final audit number is correct.

The rollback process cannot roll the database back across a discontinuity in the audit and stops automatically if it encounters a discontinuity.

Rebuild Process

The rebuild process rebuilds an entire database using previous dumps of the database and subsequent audit files. The rebuild process normally terminates in the final portion of the audit; that is, where the DASDL update level is equal to the current level. In this case, the current database software (DMSUPPORT) and the current control file should be used.

If you want to rebuild to a point in the audit where the DASDL update level is less than the current level, you must be aware of any DASDL update changes that occurred after the rebuild process stopping point in the audit. If any DASDL update changes have occurred that affect the format or even the existence of structures, an old control file and old database software of what will be the new DASDL update level must be used to perform the rebuild process successfully.

You can perform a rebuild process even though a DASDL update or reorganization exists in the audit. The rebuild process allows the merging of selected files from several sets of dump tapes in order to load a complete set of compatible database files. Take care that all partial dumps are included in the DMUTILITY statement so that a complete set of database files are available for loading; otherwise, DMRECOVERY does not run due to either missing database files or partial files.

When a reorganization is performed on a structure, only that structure is invalidated on prior dump tapes. The dump can still be used for structures not affected by the reorganization.

A rebuild process must have as its stopping point a time when all database files are in phase. Because the rebuild process can simulate a reorganization, the dumps need not have been taken after the reorganization. The creation timestamps of the files loaded from the dumps must equal the creation timestamps for the time in the audit at which the rebuild process is to stop.

The rebuild process uses the creation timestamps to determine when to apply audit images to the file. At the end of its run, the rebuild process checks to ensure that the creation timestamps are correct. If they are not, the rebuild process terminates with an error, thus indicating that the dump tapes specified in the RECOVER statement did not contain data files with creation timestamps equal to the creation timestamps in the audit file where the rebuild process stopped.

A rebuild process can handle any existing DMUTILITY initializations by repeating the initialization and continuing to apply audit images. To prevent needless application of audit images and extra input/output, a dump of the structure at the latest possible time should be used.

A rebuild is often invoked when a large portion of the database is lost. In this event, the control file might have been lost or damaged also; therefore, a valid current control file must be obtained before the rebuild process is started. When a valid control file is

obtained (perhaps by using the control file RECOVER UPDATE option in conjunction with a backup dump of the control file), special care should be taken to ensure that the control file contains the correct audit file number. An incorrect final audit file number in the control file can affect a rebuild that uses an online dump.

The final audit file must contain records for all audit trail regions, because the audit trail can be divided into three consecutive regions in relation to an online dump. The following information describes these regions:

- The first region consists of all audit records prior to two control points before the start of the dump. All changes in this region of the audit are reflected in the dump, because modified database buffers are written to disk at least every other control point.
- The second region (middle region) consists of audit records that might not be reflected in the dump.
- The third region (final region) consists of all audit records after the first audit record with a timestamp greater than the date and time at the end of the dump. All changes in this region of the audit are not reflected in the dump because no modified database buffer is ever written to disk until the corresponding buffer is successfully written to the audit.

A rebuild cannot normally stop safely in the middle region; if it does, the database might not be in phase. The only time a rebuild can stop in the middle region is if the rebuild is to the end of the audit, and the audit ends in the middle region. This would happen if the dump finished after the audit was closed for the final time. (The dump might have been started while the database was still open.)

DMRECOVERY checks to see that it is not stopping in the middle region incorrectly; that is, if DMRECOVERY has not seen an audit record with a timestamp greater than that in any of the blocks of the DMUTILITY dump, it checks to see that it has rebuilt to the end of the audit. For this test to be valid, the final audit file number must be correct in the control file before the rebuild is started. For all other cases, DMRECOVERY still works properly, even if the final audit file number is incorrect. However, it is advisable to always ensure that the final audit file number in the control file is correct before starting the rebuild.

Note: *DMUTILITY passes the parameters to the recovery programs by way of the files <dbname>/RECONSTRUCTINFO (for row recovery), <dbname>/REBUILDINFO (for rebuild) and <dbname>/ROLLBACKINFO (for rollback). The layout of these files can be found in DATABASE/PROPERTIES.*

DMUTILITY Program

The DMUTILITY program uses the control file to obtain the DMSUPPORT library title, to check and update the file version timestamps and for interlock control. The DMUTILITY statements and their interaction with the control file are described on the following pages.

Detailed explanations on the use of the DMUTILITY statements are provided later in this guide. Explanations for performing database backups using the DUMP statement and for performing dump directory maintenance are provided in [Section 6, Backing Up a Database](#).

CANCEL Statement

The CANCEL statement causes DMUTILITY to

- Open the control file exclusively; that is, the control file is opened, but no other users can access the file.
- Unlock the control file and hence make the file generally accessible.

Use the CANCEL statement under any of the following circumstances:

- If DMUTILITY fails to complete properly when attempting to perform an offline dump or offline copy
- If Database Certification fails during an offline certification
- If an INITIALIZE request is not successful

COPY Statement

The COPY statement must be used with extreme care; otherwise, the integrity of the database could be destroyed. For example, if only an old control file is copied in, the database could become unusable by causing file version timestamp mismatches with all the database files. Furthermore, the tailored database software would refuse to run because the DASDL update level had changed.

Among the valid uses of COPY are

- When the control file is lost or damaged, COPY can be used to get a file from a backup dump that the control file program can use as a base to recover the current control file.
- When the entire database has been dumped using the offline DUMP construct, COPY can be used to reload the entire database, including the control file. The database will be in the state that it was at the time of the dump. The user can now access the database.

If the entire file was copied from the dump tape, the COPY statement sets the file version timestamp (F.DMTIMESTAMP) to the value in the control file that was dumped to the dump tape. If less than the entire file was copied, it sets the file version timestamp to the current value of TIME(6).

For audited databases, the user must take care to avoid creating undetected audit discontinuities. Even when the entire database is copied, the audit trail must be taken into consideration and maintained with the database. The end of the audit must be restored to the position of the audit when the database was dumped. Thus, when dumping the database, the user must keep the last audit file with the dump in case the database is to be copied back at a later time. Copying from an online dump is not recommended.

DBDIRECTORY/TAPEDIRECTORY Statements

When DMUTILITY executes a DBDIRECTORY or TAPEDIRECTORY statement, it accesses the current control file to check and list the timestamps of the designated structures.

If you use the AX OVERRIDE statement to override timestamp mismatches that occur while executing a DBDIRECTORY or TAPEDIRECTORY statement, the control file is not updated.

[Section 6, Backing Up a Database](#), describes dump directory maintenance in detail.

DUMP Statement

The control file must always be present to execute the DUMP statement, and it is always dumped to cycle one, version one, of the dump tape. The version and format timestamps of each structure dumped are checked, as well as the database timestamp and DASDL update level.

To guarantee that a dump is made while the database is not being updated, OFFLINE DUMP must be specified.

When an offline dump is executed, DMUTILITY opens the control file exclusively, writes a special value in the lock word, closes the control file, and reopens the control file nonexclusively. Upon completion of the dump, DMUTILITY unmarks the control file. If DMUTILITY fails to complete the offline dump and does not unmark the control file, the CANCEL statement described previously can be used to unmark the control file.

For unaudited databases, all dumps are offline dumps and OFFLINE need not be specified. For audited databases, OFFLINE must be specified if an offline dump is desired; otherwise, an online dump occurs.

Online dumps can be performed while the database is being updated. Online dumps are permitted only for audited databases because such dumps must be used in conjunction with the audit.

[Section 6, Backing Up a Database](#), describes the dump process in detail and provides the DUMP statement syntax.

LIST and WRITE Statements

The contents of the current control file can be listed using the LIST or WRITE statements in DMUTILITY.

For other database files, the timestamps are printed using the current control file but are not checked.

RECOVER Statement

The RECOVER statement is used to initiate all manual forms of recovery for audited databases; that is, all forms of recovery except for halt/load and ABORT recovery, which are initiated automatically.

DMDATARECOVERY is not an exclusive function; that is, it can be run while the database is in use by the Accessroutines.

Control File

The rebuild and rollback processes require exclusive use of the entire database. DMUTILITY opens the control file exclusively and writes a special mark in it so that the function initiated is the only one allowed to run on the database.

The rebuild process can override any other exclusive function that has been discontinued, except reorganization. For example, if DMUTILITY fails while performing an offline dump, and fails to unmark the control file, DMUTILITY can be run designating REBUILD. DMUTILITY then overrides the mark in the control file for the offline dump. DMUTILITY cannot override a reorganization because of the nebulous state of the database description file, control file, and database files.

At the conclusion of the DMUTILITY phase of a rebuild or rollback process, DMUTILITY marks the control file as being in exclusive use by DMRECOVERY, closes the control file, and then initiates DMRECOVERY. If the DMRECOVERY part of the rebuild or rollback process is interrupted, restart the recovery of the database by running SYSTEM/DMRECOVERY with the following statement:

```
RUN SYSTEM/DMRECOVERY("DB = <database name>")
```

[Section 8, Recovering the Database](#), describes the recovery process in detail and provides the RECOVER statement syntax.

REORGANIZATION Program

When an update reorganization starts, the control file program updates the control file. During the reorganization, the GENERATE and FIXUP tasks update timestamps for structures in the control file. After the control file has been updated, the REORGANIZATION program reorganizes the database.

If the EXCLUSIVE option is specified for a database reorganization, the control file is marked as being in exclusive use by the REORGANIZATION program. In addition, if the format of the structure is to be changed, the DASDL update level in the structure directory is appropriately updated. The control file remains in this state throughout the entire reorganization run. User programs are able to access the database until the reorganization begins. However, once the reorganization begins, the control file is marked as being in exclusive use by the REORGANIZATION program. All user programs are not allowed to access the database until the reorganization is finished. Any attempt to access the database during the actual reorganization run results in an OPENERROR on the control file.

After all the data sets have been reorganized, the REORGANIZATION program unlocks the control file from the IN EXCLUSIVE USE BY REORGANIZATION state.

If the reorganization process fails, the database—including the control file, description file, the old DMSUPPORT library, and other tailored software—must be reloaded from backup dumps.

Database Certification Program

If an aborted run of Database Certification is an exclusive run and the program is not going to be restarted, the `DMUTILITY CANCEL` statement must be used to unlock the control file.

Section 3

Using the DMSUPPORT Library

The DMSUPPORT library is a flexible piece of software that accommodates special user requirements. It is also used by the Enterprise Database Server software. The library provides entry points that allow you to obtain error codes. These codes consist of error type, error message, and the identity of the structure on which the error occurred. This section describes how the entry points to the DMSUPPORT library are used.

Entry Points

Entry points are procedures in the DMSUPPORT library program that can be called by another program. The following entry points are available to user programs. The table accompanying each entry point description shows the parameter types and procedure result type in each of the host languages.

DMEXCEPTIONNAME (<exception> , <text>)

Returns in <text> the name of the exception category (for example, "NOTFOUND") found in <exception>. "NO EXCEPTION" is returned if no exception occurred.

Language	Returned Result	<exception>	<text>
ALGOL	BOOLEAN	BOOLEAN	STRING
COBOL	None	01 (COMP-2 GP)	01 DISPLAY
COBOL74/COBOL85	None	01 DISPLAY	01 DISPLAY

DMSTRUCTURENAME (<exception> , <text>)

Returns the name of the structure found in <exception>. Blanks (COBOL74 and COBOL85) or null strings (ALGOL) are returned if the exception is not structure related.

Language	Returned Result	<exception>	<text>
ALGOL	BOOLEAN	BOOLEAN	STRING
COBOL	None	01 (COMP-2 GP)	01 DISPLAY
COBOL74/COBOL85	None	01 DISPLAY	01 DISPLAY

DMEXCEPTIONTEXT (<exception> , <text>)

Examines <exception> and returns in <text> the category name, structure name and number (if structure related), and text describing the exception subcategory. "NO EXCEPTION" is returned if <exception> indicates no exception occurred.

Language	Returned Result	<exception>	<text>
ALGOL	BOOLEAN	BOOLEAN	STRING
COBOL	None	01 (COMP-2 GP)	01 DISPLAY
COBOL74/COBOL85	None	01 DISPLAY	01 DISPLAY

The <exception> is a standard Enterprise Database Server exception word for ALGOL. For COBOL74 and COBOL85, <exception> is a bit pattern representing the DMSTATUS register contents and generated by the DMSTATUS(DMRESULT) construct. (Refer to the specific user language manual for additional details.)

For COBOL74 and COBOL85, the DMRESULT is not a valid syntax; the exception can be obtained by combining the results of DMSTRUCTURE, DMCATEGORY, DMERRORTYPE, and DMERROR. Even though typed, the ALGOL entry points always return FALSE.

All entry points return EBCDIC character data in the text parameter. For COBOL74 and COBOL85, text is truncated or padded with trailing blanks as necessary.

Entry Point Declarations

To ease the process of writing programs to use the DMSUPPORT library, DATABASE/DMSUPPORT contains declarations of the library and its entry points for ALGOL. COBOL74 and COBOL85 do not require declaration of libraries. The appropriate sequence range can be included for the language being used. The library identifier is DMSUPPORT, and the entry point identifiers and parameter types are as shown in the preceding text.

The included ranges declare all entry points. If you wish to rename DMSUPPORT entry points or declare only a subset of the entry points, you must declare the entry points themselves using the following table.

Language	Sequence Range
ALGOL	21000000-21999999

The entry point declarations you can include rename actual exported entry points so that the same function can be called by the same name in different languages. Because library entry points cannot be renamed in COBOL74 and COBOL85, the *DM* prefixed entry points are used for these languages. The renaming facility in ALGOL is used to produce DM entry points in each of these languages. The majority of the actual work is performed by the

ALGOL entry points. After translating the parameters that are not compatible with ALGOL to ALGOL-compatible parameters, COBOL74 and COBOL85 entry points call the ALGOL entry points to perform the actual request against the database.

The following table shows the names of exported entry points, and the corresponding ones available to each language. You can use this table to construct your own DMSUPPORT entry point declarations.

COBOL74/COBOL85	ALGOL
DMEXCEPTIONNAME	ALGOEXCEPTIONNAME
DMSTRUCTURENAME	ALGOLSTRUCTURENAME
DMEXCEPTIONTEXT	ALGOEXCEPTIONTEXT

Example Programs

The following program fragments illustrate the use of each DMSUPPORT entry point in ALGOL, COBOL74, and COBOL85. For COBOL74 and COBOL85, the DMSTATUS(DMRESULT) construct can be used only in a MOVE statement.

ALGOL Program Fragment

```

BEGIN
  $ INCLUDE DMSUPPORT = "DATABASE/DMSUPPORT" 21000000-21999999
  STRING MSG;
  BOOLEAN RESULT;
  DATABASE EXAMPLEDB;
  DMSUPPORT.TITLE := "DMSUPPORT/EXAMPLEDB.";

  OPEN EXAMPLEDB : RESULT;
  IF RESULT THEN
  BEGIN
    DMEXCEPTIONNAME (RESULT, MSG);
    DISPLAY ("EXCEPTION CATEGORY: " !! MSG);

    DMSTRUCTURENAME (RESULT, MSG);
    DISPLAY ("EXCEPTION STRUCTURE: " !! MSG);

    DMEXCEPTIONTEXT (RESULT, MSG);
    DISPLAY (MSG);
  END;
  ...
END;
```

COBOL74/COBOL85 Program Fragment

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
DATA-BASE SECTION.
DB EXAMPLEDB ALL.
WORKING-STORAGE SECTION.
01 MSG                PIC X(100).
01 RESULT             PIC X(6).
01 DMSUPPORT-NAME PIC X(19) VALUE "DMSUPPORT/EXAMPLEDB".
PROCEDURE DIVISION.
MAIN.
    CHANGE ATTRIBUTE TITLE OF "DMSUPPORT" TO DMSUPPORT-NAME.
    OPEN UPDATE EXAMPLEDB
    ON EXCEPTION
        PERFORM DMEXCEPT.
    . . .
    STOP RUN.
DMEXCEPT.
    MOVE DMSTATUS(DMRESULT) TO RESULT.
    CALL "DMEXCEPTIONNAME OF DMSUPPORT"
        USING RESULT, MSG.
    DISPLAY "EXCEPTION CATEGORY: " MSG.
    CALL "DMSTRUCTURENAME OF DMSUPPORT"
        USING RESULT, MSG.
    DISPLAY "STRUCTURE NAME: " MSG.
    CALL "DMEXCEPTIONTEXT OF DMSUPPORT"
        USING RESULT, MSG.
    DISPLAY MSG.
```

Section 4

Using DMUTILITY

DMUTILITY is a program that provides a convenient method of controlling the operational aspects of a database. DMUTILITY provides the following capabilities:

- Dumping and loading database files
- Printing data and control information for database files
- Initializing all or specific structures in a database
- Initiating nonautomatic forms of recovery for audited databases

DMUTILITY Commands

The commands given to DMUTILITY to perform tasks are briefly described in [Table 4–1](#). These commands are preceded by the <db statement>, which specifies the database name and the location of the control file. The database statement can be found in [Appendix A, Common Syntactic Items](#).

Table 4–1. DMUTILITY Commands

Command	Description
BUILDDUMPDIRECTORY	Causes one or more database dumps to be entered in the database Dump Tape Directory.
CANCEL	Clears the offline dump flag or exclusive certification flag in the control file following an unsuccessful offline dump or unsuccessful exclusive certification.
CFRESTORE	Restores the control file of a live database from its quiesce database copy.
COPY	Copies (loads) database files to disk, either from a tape dump or from a disk stream dump.
COPYDUMP	Generates a copy of a backup dump.
DBDIRECTORY	Prints the current status of rows in the database files.
DISABLE/ENABLE	Disables or enables access to a database.
DUMP/APPEND	Copies (dumps) database files from disk, either to tape or to disk.

Table 4–1. DMUTILITY Commands (cont.)

Command	Description
DUPLICATEDUMP	Generates a copy of a backup dump and ensures reel-for-reel or file-for-file compatibility between the original dump and the copy of the dump.
INITIALIZE	Initializes database data files.
LIST/WRITE	Prints the contents of database files.
LOBANALYZE	Produces a report of the disk usage of the large object (LOB) tank data sets.
LOBCLEANUP	Performs cleanup operations on deleted LOBS, or LOBS of deleted structures.
LOBCOMBINE	Merges adjacent spaces that have been left after LOBS have been deleted.
LOBSQUASH	Performs an intensive defragmentation of empty space in the LOB tanks.
MIGRATEDB	Adds or deletes disjoint data sets and its sets from an active database.
QUIESCE	Creates a coherent copy of an online database.
QUIESCE QDC	Starts the creation of a quiesce database copy.
RECOVER	Initiates the non-automatic forms of recovery for audited databases.
REDISTRIBUTE	Redistributes a database file of a disjoint data set across the family pack members of a multifamily pack.
RESUME	Returns a database to its normal state following a QUIESCE command.
TAPEDIRECTORY	Prints information about rows in the database files that were copied to tape with the dump statement.
TAPESET	Lists the contents or recreates the fast access directory file for a multidump tape or set of multidump tapes.
VERIFY	Verifies that an audit file can be used by the data management software without error.
VERIFYDUMP	Checks that dumps are free from errors such as block checksum errors and I/O errors.

These statements are used in combination with various Enterprise Database Server tasks. The syntax for these statements is fully illustrated and described in the respective sections. Because the <db statement> is used often, it is described in [Appendix A, Common Syntactic Items](#).

Running DMUTILITY

DMUTILITY uses as input a quoted string to specify the action to be performed by DMUTILITY. Use either of the following general formats to run the DMUTILITY program for traditional databases:

```
RUN SYSTEM/DMUTILITY("<db statement> <utility statement>");
```

or

```
BEGIN JOB; RUN SYSTEM/DMUTILITY ("*")
```

```
DATA
  <db statement> <utility statement>
? END JOB.
```

Use either of the following general formats to run the DMUTILITY program for permanent directory databases:

```
RUN SYSTEM/DMUTILITY("<db statement> <utility statement>");
  DATAPATH = *DIR/TEST ON TESTPACK;
```

or

```
BEGIN JOB; RUN SYSTEM/DMUTILITY ("*");
DATAPATH = *DIR/TEST ON TESTPACK;
```

```
DATA
  <dbstatement> <utility statement>
? END JOB.
```

The <db statement> communicates the database name and location of its control file to the DMUTILITY program. Because DMUTILITY is not a tailored program, it needs this user-supplied information to locate the database files, the DMSUPPORT library, and the control file. The syntax for this statement can be found in [Appendix A, Common Syntactic Items](#). For databases that reside within permanent directories, the <db statement> should not contain an asterisk (*) or (usercode) prefix, or a family name.

The DMUTILITY program uses the tailored support library whose default title is DMSUPPORT/<database name> or <permanent directory>/DMSUPPORT/<database name>/<level>. The DMSUPPORT library is compiled in combination with the other tailored database software. When this library is not present, the DMUTILITY program displays a DMOPENERROR 66 and the program cannot complete its task successfully.

When dumping or copying a dump, and recovering a database that has the DMDUMPDIR option enabled, you might need to specify an appropriate file equation in order to locate the database description file. The following is an example of the DMUTILITY command that might be used:

```
RUN $SYSTEM/DMUTILITY(<db statement><utility statement>);
FILE DASDL = DESCRIPTION/<database name>;
```

Using DMUTILITY

If DMUTILITY is initiated from a remote terminal and passed an input string consisting of an asterisk (*), then DMUTILITY automatically requests its input parameters from the remote terminal.

The success of a DMUTILITY run can be determined in the Work Flow Language (WFL) by examining the task attribute TASKVALUE of the DMUTILITY task at the completion of the run. The task attribute TASKVALUE when used in WFL is called VALUE.

When a user job runs DMUTILITY multiple times and uses the same TASK variable for each run of DMUTILITY, you need to initialize the TASK variable before reusing it again. If the TASK variable is not initialized, DMUTILITY creates a debug listing that can impact performance. Limited debug output is also produced when you set the task attribute SW8 to TRUE.

The values of TASKVALUE have the following meanings:

Value	Meaning
0	Error (fatal or nonfatal)
1	OK
2	Warning given

When DMUTILITY detects an error of any kind, an error message is displayed. This message generally indicates that DMUTILITY has detected an error in the user input, such as a syntax error or some rows not specified for a REBUILD. (The printer listing should be examined for a detailed explanation.) Similarly, a warning message is displayed when a warning has been issued.

DMUTILITY does not reset the task attribute TASKVALUE from 0 to 1 if a nonfatal error is successfully recovered. This allows the user to trap errors that might have occurred during the DMUTILITY run, even if DMUTILITY goes to the normal end of task. In order to differentiate between fatal and nonfatal errors in the WFL, it is suggested that TASKVALUE be used in conjunction with the WFL Boolean primary COMPLETEDOK. For more information on the WFL Boolean primary, refer to the *WFL Reference Manual*.

Continuing and Restarting DMUTILITY Operations

Under certain circumstances DMUTILITY operations can be continued or restarted. To restart a DMUTILITY operation, you must supply a negative task value. Limited debug output is also produced when you set the task attribute SW8 to TRUE.

For more information on restarting DMUTILITY dumps, refer to "Restarting DMUTILITY for Tape and Disk Stream Dumps" later in this section. For more information on continuing DMUTILITY operations, refer to "Continuing DMUTILITY" later in this section.

Dump Media: Tape Dump Versus Disk Stream Dump

DMUTILITY allows dumps to tape (tape dump) or to disk (disk stream dump).

Tape dumps remain the default medium for protecting an Enterprise Database Server database in a production environment. DMUTILITY supports the following functions for all types of dumps:

- Building a dump directory
- Copying files and copying backup dumps
- Dumping
- Recovering
- Getting directory information
- Verifying backup dumps

Tape Dumps

Dumps to tape can be backed up to either single dump tapes or multiple dump (multidump) tapes. Single dump tapes contain a single database backup from a single database. Multidump tapes can contain many database backups from one or multiple databases.

Multidump Tapes

Multidump tapes can be created only on tape drives that have fast access capability. To support the multidump feature, DMUTILITY also creates a fast access directory that contains information about each dump contained on the tape (including the tape address). The directory is created at the location specified by your system DL LIBMAINTDIR command and has the usercode of the task that created the first backup dump on the tape. If no LIBMAINTDIR specification exists, the directory is created at a location based on the rules for family substitution.

Multidump tapes do not have continuation reels. As a result, the value of VERSION is always 1. If the total dump does not fit on a reel, the dump contents that has been written on the reel is deleted and the operator is requested to initiate a new DUMP command rather than an APPEND command.

If the multidump tape is used to hold backups from more than one usercode, ensure that all of the usercodes have visibility to the fast access directory. One method to accomplish this task is to give the fast access directory PUBLIC security.

If the fast access directory is missing, a new directory is automatically created when the next dump is appended to the tape, or it can be manually re-created by using the TAPASET DIRECTORY CREATE command. In either case, the usercode initiating the command must be able to create files under the usercode of the first dump on the tape; otherwise, the operating system emits a security violation error.

To access a database backup stored on a multidump tape, the usercode under which DMUTILITY runs must have visibility to the database, and if guardfiles are employed, the usercode must have the correct permissions to read or recover the database.

Disk Stream Dumps

A disk stream dump is a quick and convenient alternative to a tape dump. Instead of using tape, a disk stream dump resides on disk. In contrast to several reels of tape for a tape dump, a disk stream dump resides on one disk pack. Disk stream dumps may be used by both DMUTILITY and DMDUMPDIR. Enterprise Database Server application users can find disk stream dumps a useful tool in shortening development and testing time.

DMUTILITY accesses disk stream dumps like regular tape dumps, because all information is preserved in a disk stream dump.

To specify disk instead of tape, DMUTILITY requires that you use the following syntax wherever a tape name is required:

```
<file name> ON <family name>
```

Example

In the following example, the TAPEDIRECTORY statement specifies a disk stream dump. The disk stream file is BACKUP/TEST and resides on the disk pack DMS. The disk stream file, BACKUP/TEST, is under the usercode that DMUTILITY runs under.

```
TAPEDIR BACKUP/TEST ON DMS
```

Omitting Tape Options and Tape Specifications

Tape options, such as WORKERS and FORWARD COMPARE, are ignored during disk stream processing. If you include tape options in copy, dump, or recovery specifications when a disk stream dump is used, a warning message is displayed.

If you include a serial number specification in a disk stream dump request, DMUTILITY terminates with a syntax error.

Limitations on Disk Stream Dumps

Disk stream dumps can take up a lot of disk space. It is therefore recommended that you perform disk stream dumps only on a scheduled basis so that you can ensure that the required disk resources are available.

The following information identifies the limitations of disk stream dumps:

Limitation	Explanation
Disk size	<p>Tape dumps have no fixed size limits because tapes can be readily mounted to accommodate large dumps. A disk stream dump, however, is bounded by the amount of available space left on the specified disk pack. A disk stream dump, because it is a disk file, cannot grow beyond the available space on the disk pack.</p> <p>A manual calculation of available disk space on the specified disk pack must be done prior to an offline disk stream dump. As a rough guide, the available space on disk must be greater than twice the disk sector size of the DMCONTROL file plus the sum of disk sector sizes for each DMS data file to be dumped.</p> <p>Note: <i>There is a physical disk file limitation of 1000 areas. To create a multiple-file disk stream dump, use the FILES specification.</i></p>
Disk resource contention	<p>To avoid a deadlock between DMUTILITY and other programs, an offline disk stream dump should be done on a disk pack that is not used by other programs.</p> <p>In particular, care must be taken when dumping disk stream files to the system resource pack or to the halt/load unit. A deadlock on the available disk space between the Master Control Program (MCP) and DMUTILITY freezes the entire system.</p> <p>The purpose for keeping backups is defeated if the disk stream dumps reside on the disk pack containing the database files or the audit trail.</p>
Online disadvantages	<p>Avoid performing online disk stream dumps if the volume of transactions performed against the database is either unknown, volatile, or can grow beyond bounds. Online disk stream dumps can run into the disk size limitations or disk resource contention problems described previously.</p>
Usercode	<p>Usercodes cannot be explicitly specified in a disk stream file name. Disk stream files are found under the usercode that ran DMUTILITY. Note that disk stream dumps executed from the system operator display terminal (ODT) are found under the asterisk (*) usercode.</p> <p>For example, DMUTILITY, running under usercode X, cannot access disk stream dumps under usercode Y, even if X is a privileged user.</p>
Tape incompatibility	<p>A library maintenance tape copy of a DMUTILITY disk stream dump cannot be used in place of a DMUTILITY tape dump.</p> <p>You can use the DMUTILITY command COPYDUMP to copy a disk stream dump to tape or to copy a tape dump to disk.</p> <p>Also, DMUTILITY does not allow disk stream file names of the following form:</p> <p style="text-align: center;"><file name> ON TAPE</p> <p>The word TAPE in the family name is interpreted by DMUTILITY as a tape specification (KIND = TAPE).</p>

Operator Interface to DMUTILITY for Tape Dumps

You can change the number of workers for a single dump tape while DMUTILITY is running by entering:

```
<mix no> AX WORKERS = <integer>
```

This message can be entered even if the WORKERS construct was not used in the original input. The <mix no> is the mix number of the main DMUTILITY stack.

The WORKERS construct is not valid for multidump tapes.

Operator Interface to DMUTILITY for Disk Stream Dumps

If you use the WORKERS option with a disk stream dump, DMUTILITY returns a warning. The following messages illustrate the input you might supply and the DMUTILITY response:

User Input

```
8832 AX WORKERS = 3
```

DMUTILITY Response

```
8832 TAPE OPTION 'WORKERS' CANNOT BE SET DURING DISK DUMPS
```

If the requested disk stream file already exists, DMUTILITY responds with the message

```
<mix no> AX '<FILENAME> on <FAMILY>' or 'QUIT DMUTILITY'
```

In this case, you have the following choices:

- Request another disk stream file name. If the name is valid and does not already exist, DMUTILITY processes the request.
- Overwrite the existing file. First a library maintenance copy of the existing file should be made. Once the copy has been made, the existing file can be removed and the desired disk stream file name is resubmitted.
- Terminate DMUTILITY by transmitting

```
QUIT DMUTILITY
```

DMUTILITY terminates and the following message is displayed:

```
QUITTING DMUTILITY FOR A DISK STREAM DUMP ATTEMPT
```

Restarting DMUTILITY for Tape and Disk Stream Dumps

You can restart DMUTILITY *DUMP COPY*, *DUPLICATEDUMP*, *COPYDUMP*, *VERIFYDUMP*, and *RECOVER* operations if the operation was discontinued internally because of a fatal input/output error while reading or writing to tape or disk, or following a halt/load.

If the operation that was discontinued does not use a backup dump, you can restart the operation by running SYSTEM/DMRECOVERY as follows:

```
RUN SYSTEM/DMRECOVERY("DB = <database name>")
```

If the operation does use a backup dump, then you can restart the DMUTILITY operation by supplying the job number of the failed operation as a negative task value. The following statement illustrates a restart request:

```
RUN SYSTEM/DMUTILITY (<db statement>);  
VALUE=<negative job number>
```

Note: Operations that write to a tape that is specified by a <multidump tape specification> must be restarted and must use the negative task value. If the operations are not restarted, an incomplete dump remains on the tape.

A DMUTILITY run can be restarted without a negative task number. This happens automatically when the following occurs: the WFL job initiates a DMUTILITY run, the WFL job aborts, and the WFL job is restarted.

DMUTILITY restarts using the HLDUMPINFO files created by the original run of DMUTILITY. If the HLDUMPINFO is not present, DMUTILITY does not restart. The HLDUMPINFO file is titled

```
<database name>/HLDUMPINFO/<job number>
```

In addition, each tape, disk file, or dump worker that was active at the time the program was discontinued has its own HLDUMPINFO file. The worker HLDUMPINFO file must be present in order for the worker to restart from where it stopped. If the worker HLDUMPINFO file is not present, the program restarts from the beginning. The worker HLDUMPINFO file is titled

```
<database name>/HLDUMPINFO/<job number>/<worker number>
```

The DMRECONFILTER program uses its parameter file <database name>/RECONFILTERINFO as its restart information file. As each audit is processed, the restart information is updated. If a halt/load occurred or the program discontinued, rerunning the DMRECONFILTER restarts each worker back to the audit file the worker was processing at the time of termination.

Following the successful completion of the DMUTILITY run, all HLDUMPINFO files created during the DMUTILITY run are automatically removed.

Because the job number is part of the title of the DMUTILITY halt/load restart file, never run more than one copy of DMUTILITY within a single WFL job deck.

Continuing DMUTILITY

DMUTILITY runs can be continued for those options of the RECOVER statement that require tape or disk input; that is, a rebuild request or row recovery where USING BACKUP is specified. If the original request did not load the complete set of files or rows because of input/output errors or user oversight in specifying the recover source, the continued request allows the missing files and rows to be added before DMRECOVERY or RECONSTRUCT is initiated.

To continue a DMUTILITY run, you must supply a task value of 8 and an unchanged RECOVER request. Only the RECOVER list or RECOVER source statements can be changed to add the necessary files. If you supply a task value other than 8, the printer backup file generated by the operation might contain internal diagnostic debugging information.

Depending on the input request, DMUTILITY uses the RECONSTRUCTINFO or REBUILDINFO file created by the previous DMUTILITY run and simply adds to it. The RECONSTRUCTINFO or REBUILDINFO file must be present at the time of the continuation run. These files are titled as follows:

```
<database name>/RECONSTRUCTINFO  
<database name>/REBUILDINFO
```

As many continuation runs as necessary can be performed to load the needed files or rows.

Example

The following example illustrates the use of a continuation request.

Assume the database was dumped to four tapes: T1, T2, T3, and T4. Further assume that the following REBUILD request was specified:

```
RUN SYSTEM/DMUTILITY  
 ("<db statement> RECOVER  
 (REBUILD THRU AUDIT 5) FROM T1,T2,T3");
```

Because REBUILD must load the entire database, DMUTILITY would notice that some of the database files were not loaded and would prevent the REBUILD. Using the continuation request, you could load the missing tape without reloading the other dump tapes. This is accomplished as follows:

```
RUN SYSTEM/DMUTILITY  
 ("<db statement> RECOVER  
 (REBUILD THRU AUDIT 5) FROM T4");  
VALUE = 8;
```

DMRECOVERY would be initiated normally following the successful loading of all database files.

DMUTILITY Error Handling

DMUTILITY can encounter input/output errors during the dumping or loading of a database or partial database. How DMUTILITY handles these errors is described on the following pages.

Tape Input/Output Errors During Dump

Single Dump Tapes

DMUTILITY handles both hard input/output errors and timeouts that occur while writing to tape. An appropriate error message is displayed, and the tape on which the error occurred is labeled *BADTAPE* and closed. The following message is then displayed:

```
<error message> AX 'OK' FOR RETRY OR NEW TAPE
```

You can enter *AX OK* or discontinue the job using the DS system command. Entering *AX OK* causes DMUTILITY to restart the dump for the tape that failed. If a failure occurs, try using any of the following techniques to resolve the problem:

- Clean the tape drive.
- Switch reels.
- Clean the tape that has the error.
- Change tape drives.

Other DMUTILITY workers proceed with their dumping while this worker is waiting on the accept message.

Multidump Tapes

Because this style of dump tape contains multiple backup files that have already been created successfully, the tape cannot be labeled as *BADTAPE* without also losing all the previous work.

When a hard I/O error or timeout occurs while writing to a multidump tape, DMUTILITY follows one of two courses of action:

- When the error occurs while writing to the first dump on the tape, the events are the same as if the tape were a single dump tape.
- For creation of another dump on the tape, DMUTILITY returns to the end of the previous good dump and marks the tape as being closed. This action preserves the previous data while assuring that new backups do not encounter the bad area again. DMUTILITY then displays the following message:

```
<error message> AX 'OK' FOR RETRY OR NEW TAPE
```

You can enter *AX OK* or discontinue the job with the DS system command. Entering *AX OK* causes DMUTILITY to create a new multidump tape and restart the dump for the tape that failed. The title of the fast access directory created for the new tape contains the current date and time.

Disk Stream Input/Output Errors During Dump

Input/output errors that occur while DMUTILITY is reading from disk are fatal. DMUTILITY cannot successfully complete a dump unless all rows specified in the dump list are dumped. The row on disk that caused the error must be reconstructed before it can be dumped. In this case, DMUTILITY discontinues itself, leaving its restart files present so it can restart the dump where it stopped.

The same error conditions are detected by DMUTILITY during a disk stream dump as during a tape dump. The error handling description under "Tape Input/Output Errors during Dump" also applies to disk stream I/O errors. Keep in mind that all references to tape must be translated for disk.

Disk I/O error messages are prefixed by DISKSTREAM to distinguish them from tape dump I/O error messages and database disk I/O error messages.

DMUTILITY attempts to mark (retile) the bad disk stream file with the name BADDISKSTREAMFILE. The following accept message is then displayed:

```
DISKSTREAM FILE: <error message> - AX 'OK' FOR RETRY OR NEW DUMP
```

You can enter *AX OK* or discontinue the job using the DS system command. Entering *AX OK* causes DMUTILITY to restart the dump for the disk stream dump file that failed.

Unlike tape dump I/O errors, a disk stream dump I/O error indicates a fault with the disk pack, rather than a particular file.

On a retry attempt, the bad disk stream file remains resident while DMUTILITY retries the disk stream dump. At the system ODT, reserve disk areas on the bad disk stream file as needed; then remove the bad disk stream file.

Tape Input/Output Errors During Load

If DMUTILITY encounters hard input/output errors or timeouts while loading rows from a dump (DMUTILITY commands RECOVER and COPY), an error message is printed.

When I/O errors occur during a RECOVER command, the following message is displayed:

```
AX 'RETRY' OR 'SKIP ROW' OR 'QUIT RECOVER'
```

When I/O errors occur during a COPY command, the following message is displayed:

```
AX 'RETRY' or 'SKIP ROW' OR 'QUIT COPY'
```

If *RETRY* is entered, DMUTILITY starts over at the beginning of the tape and attempts to read it again.

If *SKIP ROW* is entered, DMUTILITY tries to pass over the row on the tape that has the input/output error. If DMUTILITY skips a row in this manner while a RECOVER command is in progress, DMRECOVERY is not automatically initiated. This action enables you to reload the row by using a DMUTILITY continuation run before initiating DMRECOVERY.

If DMUTILITY attempts a RETRY or a SKIP ROW that is unsuccessful, one or more error messages are displayed.

QUIT RECOVER or QUIT COPY causes the rest of the rows on tape and any subsequent reels in the tape CYCLE to be skipped. If there are other input tapes, DMUTILITY proceeds to load those tapes. As with the SKIP ROW response, DMRECOVERY is not initiated automatically if a RECOVER command was in progress. This allows the missing rows to be loaded from other tapes.

Disk Stream Input/Output Errors During Load

The same error conditions are detected by DMUTILITY during a disk stream load as during a tape load. The error handling description under "Tape Input/Output Errors During Load" also applies to disk stream I/O errors. Keep in mind that all references to tape must be translated for disk.

Disk I/O error messages are prefixed by DISKSTREAM to distinguish them from tape load I/O messages and database disk I/O error messages.

Database Disk Input/Output Errors

If DMUTILITY encounters input/output errors (either hardware or CHECKSUM/ADDRESSCHECK) while accessing database files on disk, the procedure RETRYIO is called. RETRYIO displays information regarding the error and, if so requested, retries the input/output operation and displays the results of the retry. If the retry fails, RETRYIO discontinues the worker that caused the error.

However, if a number of bad rows occurred for a particular DMUTILITY worker during a dump, the operator can try the LOCKBADROWS option to lock all rows with the read errors. The operator can then obtain a report from a previous printer file and review the bad rows assigned to that worker.

DMUTILITY detects ADDRESSCHECK and CHECKSUM errors after successful reads; therefore, no result descriptor is displayed for these errors. All successful writes are reread and compared against the original before they are considered acceptable.

RETRYIO Error Messages

When RETRYIO is entered, two messages are displayed. The following text describes the messages.

RETRYIO Message 1

The first message indicates whether the input/output operation was a read or a write, for which structure the input/output operation was intended, and the name of the file in which the error occurred. Examples of the first message follow:

```
DISPLAY:***READ ERROR ON STR #2, FILE:(DMSII)DB/D/DATA ON DMS.  
DISPLAY:***WRITE ERROR ON STR #3, FILE:(DMSII)DB/D/S ON DMS.
```

RETRYIO Message 2

The second message gives detailed information about the input/output operation. Examples of this second message follow:

```

DISPLAY:***RSLT=CHECKSUM FAILED, FAMILYNAME=DMS, FAMILYINDEX=2,
ADDRESS=108355, ROW=3, BLOCK=42.
DISPLAY:***RSLT=400002860009 : UNIT NOT READY, FAMILYNAME=DMS,
FAMILYINDEX=1, ADDRESS=2917, ROW=1, BLOCK=1.
    
```

These examples show RETRYIO messages that give details of failed input/output operations. No result descriptor is displayed in the first example because DMUTILITY detects ADDRESSCHECK and CHECKSUM errors only after successful read operations.

The elements of the example are described in [Table 4-2](#).

Table 4-2. Elements of RETRYIO Messages

Message	Description	System Representation
RSLT	I/O result descriptor, followed by an explanation of the error.	<buffer>.IORESULT
FAMILYNAME	Name of the family on which the file resides.	<fid>.FAMILYNAME
FAMILYINDEX	Family index of the unit to which the I/O operation was attempted.	<fid>(<row>).FAMILYINDEX
ADDRESS	Hardware address of the block.	<fid>(<row>). ROWADDRESS.[47.08].
ROW	Row of the file on which the I/O operation was attempted.	<block> DIV <fid>.AREASIZE
BLOCK	Relative segment number of the first segment of the block of the file on which the I/O operation was attempted.	<buffer>.IORECORDNUM

After these messages are displayed, the I/O operation is automatically retried once. If this retry fails, RETRYIO displays a message asking the operator whether or not another retry should be attempted. If the operator requests another retry, the I/O operation is retried up to the number of times specified by the value of MAXRETRIES. In this case, if the retry fails a second time, RETRYIO repeats the DISPLAY messages and the prompt to the operator. If the operator requests that no more retries are to be performed, the worker that caused the error is discontinued, and DMUTILITY can be restarted to recover the work that the worker left unfinished.

Examples of RETRYIO Messages

Example 1

```
DISPLAY:***READ ERROR ON STR #19, FILE:(DMSII)DB/D/E/DATA ON DMS.
DISPLAY:***RSLT=CHECKSUM FAILED, FAMILYNAME=DMS, FAMILYINDEX=3,
ADDRESS=671124, ROW=26, BLOCK=263.
DISPLAY:***READ FOR STR #19 ON DMS WAS RETRIED 1 TIMES BEFORE
SUCCEEDING.
```

In this example, a CHECKSUM error is detected and corrected after one retry.

Example 2

```
DISPLAY:***READ ERROR ON STR #4, FILE:(DMSII)DB/D/E/DATA
ON DMS...
***RSLT=007080000000 : CHECKSUM FAILED,
FAMILYNAME=DMS,,
FAMILYINDEX=1, ADDRESS=469846, ROW=0, BLOCK=0.
DISPLAY:***READ FOR STR #4 ON DMS HAS BEEN RETRIED
1 TIME WITHOUT SUCCESS.
ACCEPT: R TO RETRY I/O FOR STR #4 ON DMS, QUIT TO
ABORT THIS WORKER
OR LOCKBADROWS TO LOCK ALL BAD ROWS ASSIGNED TO
THIS WORKER BEFORE ABORTING.
```

In this example, a CHECKSUM error is detected and is not corrected after one retry. The operator could use the option LOCKBADROWS to lock all the bad rows assigned to this worker, as follows:

```
OPERATOR ENTERED: ?AX<MIX #> LOCKBADROWS
DISPLAY:** FATAL ERROR : A WRITEVOLUME HAS BEEN QUIT
- SOME SPECIFIED ROWS HAVE NOT BEEN DUMPED.
#P-DS
```

Example 3

```
DISPLAY:***WRITE FOR STR #3 ON DMS HAS BEEN RETRIED 1 TIMES
WITHOUT SUCCESS.
ACCEPT:R TO RETRY I/O FOR STR #3 ON DMS, OR 'SKIP ROW' OR
'QUIT COPY'.
```

This example shows the prompt displayed after an unsuccessful retry attempt. Any operator response to this prompt that begins with R is interpreted to mean RETRY.

Example 4

```
DISPLAY:***WRITE FOR STR #3 ON DMS WAS RETRIED 14 TIMES BEFORE
SUCCEEDING.
```

This example shows the message displayed after a successful retry attempt. All successful write operations are reread and compared against the original before they are considered acceptable.

Example 5

```
DISPLAY:***WRITEERROR ON STR #63, FILE:(DMSII)DB/D/ISSET ON DMS.
DISPLAY:***RSLT=400010B20501 : WRITE LOCK OUT, FAMILYNAME=DMS,
      FAMILYINDEX=4, ADDRESS=495092, ROW=4, BLOCK=4015.
DISPLAY:***WRITE FOR STR #63 ON DMS WAS RETRIED 1 TIMES BEFORE
      SUCCEEDING.
```

In this example, a transient write operation error is detected and corrected on the first retry.

Example 6

```
DISPLAY:***WRITE ERROR ON STR #1, FILE:(DMSII)DB/DATA ON DMS.
DISPLAY:***RSLT=400007840801 : MPX OR CONTROLLER ERROR,
      FAMILYNAME=DMS, FAMILYINDEX=1, ADDRESS=493688, ROW=0, BLOCK=0.
DISPLAY:***WRITE FOR STR #1 ON DMS HAS BEEN RETRIED 1 TIMES
      WITHOUT SUCCESS.
ACCEPT:R TO RETRY I/O FOR STR #3 ON DMS, OR 'SKIP ROW' OR
      'QUIT RECOVER'.
```

In this example, a write operation error occurred during an attempted recovery of the database. After the operator has diagnosed and corrected the error, the following message is displayed:

```
OPERATOR ENTERED: AX:R.
DISPLAY:***WRITE FOR STR #1 ON DMS WAS RETRIED 2 TIMES BEFORE
      SUCCEEDING.
```

If *R* is designated, the write operation is retried. If *SKIP ROW* is designated, the row that gets the I/O error is omitted. However, if *QUIT TAPE* is entered, the remaining rows that are to be loaded by the READVOLUME that handles the I/O error are skipped.

DMUTILITY Warnings During Dump

The database disk I/O errors must not be confused with the warning messages DMUTILITY issues during a dump. The following warning message appears if the row was marked by the Access routines as having an error in a read operation:

```
** WARNING:  THIS ROW HAS A NON-FATAL READ ERROR AND THE
      ROW'S READ ERROR BIT HAS BEEN SET BY THE ACCESSROUTINES.
```

If you specified AXREADERROR in the dump option, and this read operation error occurs, DMUTILITY stops and displays a message asking you to enter one of the following commands:

- OK to continue
- SKIP ROW to skip the row
- TERMINATE to quit the DMUTILITY run

The following warning message appears if the row was locked out by an error in a write operation:

```
**WARNING:  THIS ROW LOCKED OUT (NOT DUMPABLE).
```

If this write operation error occurs, the row must be reconstructed before it can be dumped.

The following warning message appears if the row was locked out by DMUTILITY during an unsuccessful reconstruction:

```
**  WARNING:  UNRECONSTRUCTED ROW  (NOT DUMPABLE) .
```

If this lockout warning message appears, the row must be reconstructed before it can be dumped.

Section 5

Initializing and Maintaining

Both the control file and the database data files must be initialized before any actions can be performed on a database. The first part of this section describes the initialization and maintenance of the control file. The second part describes the initialization of database files.

Note: *The tasks identified in this section can be initiated through Database Operations Center.*

Initializing and Maintaining the Control File

All Enterprise Database Server databases depend on the system-maintained control file for their control information. This file, titled <database name>/CONTROL, must be present whenever a database is used. The control file is used for database interlock control (exclusive use), storage of audit control information and dynamic database parameter values, and file compatibility checking. Because the control file is used for all database operations, it is important that you are thoroughly familiar with the control file information contained in [Section 2, Control File](#).

The control file is created and maintained by the control file program, SYSTEM/DMCONTROL. The control file program requires exclusive use of the control file to perform the functions described in this section.

Running DMCONTROL

There are two ways to run DMCONTROL. The first method requires that the Data and Structure Definition Language (DASDL) description file be file-equated to the database description file. The run statement for this method is as follows:

```
RUN SYSTEM/DMCONTROL ("<control file parameter>");  
FILE DASDL = DESCRIPTION/<database name>;
```

In some cases, an existing control file, CFOLD, must be used as input. It might be necessary to file-equate CFOLD. Input commands are submitted to DMCONTROL with a string parameter.

The second method to run DMCONTROL does not require file equation. For this method, the run statement is as follows:

```
RUN SYSTEM/DMCONTROL ("<dmcontrol statement>")
```

Initializing and Maintaining

If both forms of the run statement are used in combination, the second method takes precedence over the first method.

If the DASDL compiler control option DMCONTROL is SET, the DASDL compiler automatically initiates the control file program as a dependent process to create or update the control file. DASDL does not process the control file program if the compiler control option DMCONTROL is RESET, or when compiling for syntax. DMCONTROL is SET by default.

If you are updating the DASDL, you should remember the following points about the control file:

- Because the control file is a dynamic extension of the database description file, the control file program must be run after every nonsyntax DASDL run. When the database is to be reorganized, the control file is handled in a special fashion by the REORGANIZATION program.
- The database cannot be in use when the control file program updates the control file.
- Because an old DMSUPPORT library does not accept a newly created control file (their update levels do not match), it is advisable to run a DASDL update with the compiler option DMCONTROL equaling RESET to avoid updating the control file until it is time for the new DMSUPPORT library to be put into production. (Note also that the new DMSUPPORT library does not accept an old control file.)
- Because there might not be any other way to conveniently get back to the old update level, the old control file and old database description file should be saved before every DASDL update.
- When executing a DASDL update on a database in which a partitioned structure has been deleted, it is imperative that the DMSUPPORT library successfully complete compilation before the control file update. The control file update requires the use of the newly compiled DMSUPPORT library. The DASDL compiler option DMCONTROL should be RESET to prevent execution of SYSTEM/DMCONTROL.

The control file program is also used to recover the control file in the event it is lost. Much of the information in the control file is recovered from the DASDL description file; however, some of the information is restored with the aid of the user, for example, the current audit file number.

SYSTEM/DMCONTROL can also be used to change the pack families upon which the audit trail, the database structures, and the Enterprise Database Server code files reside without having to do a DASDL update. However, if you can do a DASDL update, this capability is not recommended as a replacement. If SYSTEM/DMCONTROL is used, a family change bit in the control is turned on. Any subsequent DASDL update does not return the family designations to the pre-SYSTEM/DMCONTROL change specification as long as the family change bit is 1. The family change bit can be changed using the OVERRIDE FAMILY command.

The LOCKEDFILE file attribute is set for the control file if the LOCKEDFILE option is specified in DASDL. When SYSTEM/DMCONTROL is running, it automatically unlocks or locks the control file as needed. However, the LOCKEDFILE file attribute is always set

according to the DASDL specifications once SYSTEM/DMCONTROL has finished. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information about the LOCKEDFILE option.

The success of a DMCONTROL run can be determined in the Work Flow Language (WFL) by examining the task attribute TASKVALUE of the DMCONTROL task at the completion of the run. The task attribute TASKVALUE when used in WFL is called VALUE.

The TASKVALUE task attribute has the following meanings.

Value	Meaning
0	OK
1	Error (fatal or nonfatal)
2	Warning given

When DMCONTROL detects an error of any kind, an error message is displayed. This message generally indicates that DMCONTROL has detected an error in the user input, such as a syntax error or an incompatibility in the control and description file update levels. (The printer listing should be examined for a detailed explanation.) Similarly, a warning message is displayed when a warning has been issued.

DMCONTROL Statement

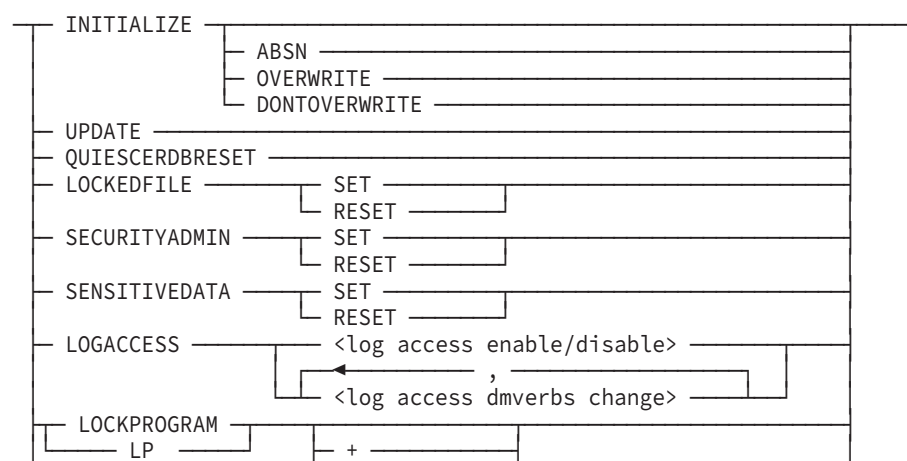
The control file program syntax is illustrated and explained on the following pages.

Syntax

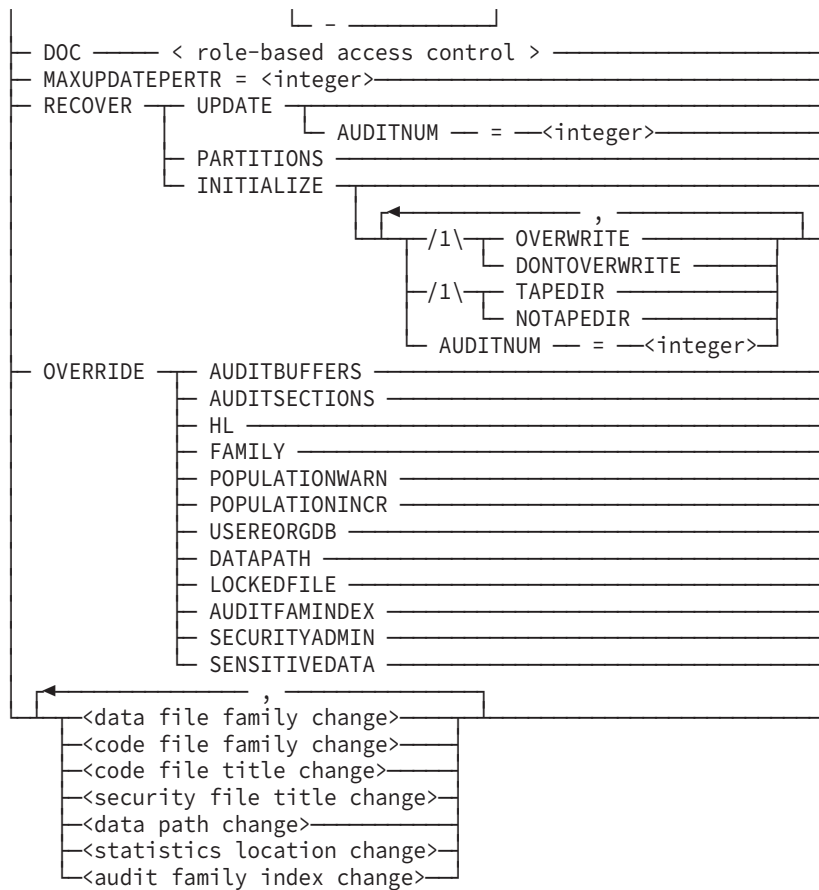
<dmcontrol statement>



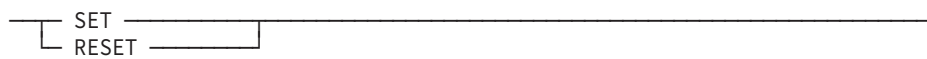
<control file parameter>



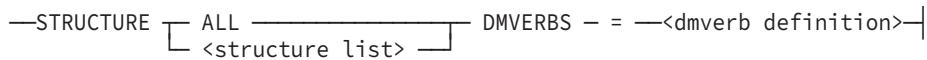
Initializing and Maintaining



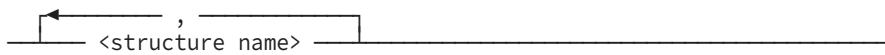
<log access enable/disable>



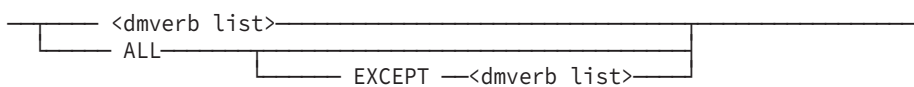
<log access dmverb change>



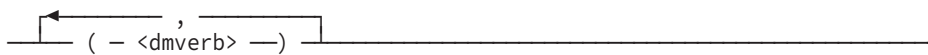
<structure list>



<dmverb definition>



<dmverb list>



<dmverb>

ASSIGN
ASSIGNLOB
CREATESTORE
DELETE
DELETELOB
FIND
FINDLOB
FREE
GENERATE
INSERT
LOCK
LOCKSTORE
REMOVE
SECURE

<data file family change>

STRUCTURE <structure name>	FAMILY = <family name>
FAMILY <packfamily1> = <packfamily2>	AUDITFAMILY = <family name>
ALTERNATE	SECAUDITFAMILY = <family name>
ALTERNATE	

<code file family change>

ACCESSROUTINES	FAMILY = <family name>
DMSUPPORT	DEFAULT
RECOVERY	
DATARECOVERY	
DMCONTROL	
DMUTILITY	
RECONSTRUCT	
LOBUTILITY	
REORGANIZATION	
RMSUPPORT	
COPYAUDITPRIWFL	
COPYAUDITSECWFL	

<code file title change>

ACCESSROUTINES	TITLE = <file title>
DMSUPPORT	
RECOVERY	
DATARECOVERY	
DMCONTROL	
DMUTILITY	
RECONSTRUCT	
LOBUTILITY	
REORGANIZATION	
RMSUPPORT	
COPYAUDITPRIWFL	
COPYAUDITSECWFL	

<security file family change>

```

/1\- DBGUARDFILE _____ FAMILY = _____
/1\- CONTROLGUARDFILE _____ |
/1\- PRIAUDITGUARDFILE _____ |
/1\- SECAUDITGUARDFILE _____ |
<structure security change> _____ |
<logical database change> _____ |
    
```

<security file title change>

```

/1\- DBGUARDFILE _____ TITLE = _____
/1\- CONTROLGUARDFILE _____ |
/1\- PRIAUDITGUARDFILE _____ |
/1\- SECAUDITGUARDFILE _____ |
<structure security change> _____ |
<logical database change> _____ |
    
```

<structure security change>

```

- STRUCTURE -<structure name>- STRSECURITYGUARD _____
    
```

<logical database change>

```

- LOGICALDB -<logical database name>- GUARDFILE _____
    
```

<data path change>

```

- DBPATH = <path name> _____
    
```

<statistics location change>

```

- STATISTICSLOC = _____
  | _____
  | PRINTER _____
  | _____
  | <packname> _____
  | _____
    
```

<audit family index change>

```

- AUDIT = _____
  | _____
  | SETFAMINDEX _____
  | _____
  | RESETFAMINDEX _____
  | _____
    
```

<role-based access control>

```

| _____
| <role> _____
| _____
| <permission> _____
| _____
| <user id> _____
| _____
    
```

<role>

```

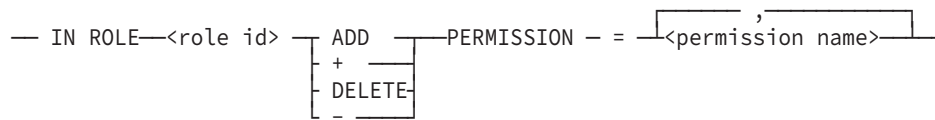
| _____
| ADD _____
| + _____
| _____
| DELETE _____
| _____
| _____
    
```

<role id>

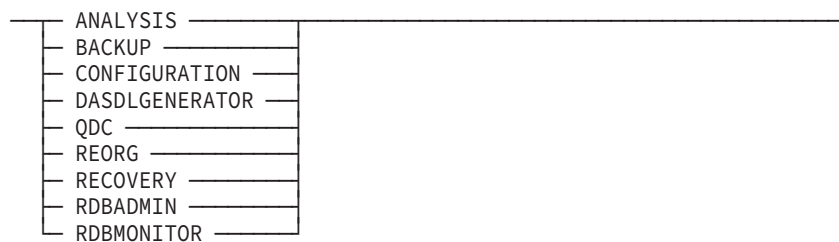
```

- < identifier > _____
    
```

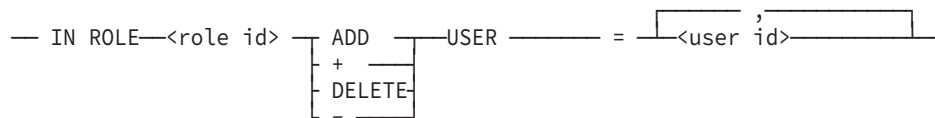
<permission>



<permission name>



<user id>



<user id>



<db statement>

Communicates the database name and location of its description file to the DMCONTROL program. Because DMCONTROL is not a tailored program, it needs this user-supplied information to locate the description file.

A complete description of the database statement can be found in [Appendix A, Common Syntactic Items](#).

<control file parameter>

Parameters can be expressed using the statements described in the following text.

INITIALIZE

Creates the initial control file for a new database from the information in the description file. The state flags are initialized; the audit information is initialized; the partition information is assigned the value NULL; all structure state flags are assigned the value INITIALIZATION REQUIRED; and the version and creation timestamps for each structure are marked as INVALID.

Initializing and Maintaining

This function invalidates any existing database files. Therefore, if a control file is resident at the time of the run, an acceptance message is displayed, and a response from the user is required to continue the run. This prevents the accidental removal of a good control file.

The INITIALIZE function can also be used to create a temporary control file for use with the DMUTILITY COPY command. If the control file is lost, an old copy must be obtained from a DMUTILITY backup dump. However, DMUTILITY requires a control file in order to perform the COPY. The INITIALIZE function can be used to create a temporary control file, as the COPY overwrites it with the one from the backup dump.

To ensure a level of continuity between the newly copied control file and the other data management software, run RECOVER UPDATE on the newly copied file.

The Transaction Processing System (TPS) information in the new control file is assigned the value 0.

If the RECORDCOUNT option is set in the DASDL, the structure record count is no longer valid after this operation. A garbage collection with the CHECKRECORDCOUNT option should be performed to recalculate the number of records in a structure. A warning message, such as the following, is displayed if the structure is touched by the first opener:

```
WARNING:
RECORD COUNT function for <structure name> is temporarily
deactivated due to the previous INITIALIZE, RECOVER INITIALIZE
or RECOVER UPDATE operation. Perform a garbage collection
reorganization with the CHECKRECORDCOUNT option to reactivate
with the correct record count.
```

ABSN

Resets the audit block serial number (ABSN).

Caution

This option is no longer required or recommended because the ABSN in the audit file increases automatically. Refer to "Handling Audit Block Serial Number (ABSN) Rollover" in Section 2 for additional information.

OVERWRITE and DONTOVERWRITE

If neither OVERWRITE nor DONTOVERWRITE is specified and a control file exists, DMCONTROL waits and issues a message asking if you wish to continue. If OVERWRITE is specified and a control file for the database exists, DMCONTROL overwrites the existing control file. If DONTOVERWRITE is specified and a control file for the database exists, DMCONTROL returns an error.

UPDATE

Causes a new control file to be created from the existing control file and the new DASDL description file after a DASDL update run.

The DASDL update level must be greater than 1. The update level in the old control file must be no more than one update level less than that in the database description file. If the old control file format level is greater than the current level, an error results. If the old control file format level is less than the current format level, it is still accepted. The new control file has the update level of the database description file, and the current control file format level.

If a reorganization is in progress, the control file program terminates with an error; otherwise, the state flags are transferred from the old control file to the new control file.

The dynamic database parameters are transferred from the new database description file. The ranges for audit to designated serial numbers are also transferred from the description file. The dynamic audit information is transferred from the old control file. If the current tape serial number is not within the new range of designated serial numbers, it is changed to the start of the new range. Deleted structures are dropped out of the control file. Partitioned records relating to a deleted partitioned structure are marked as being deleted within the control file. Added structures are marked as INITIALIZATION REQUIRED.

The scratch pool name for the audit tape is also transferred.

For a reorganized structure, the reorganization alters the creation timestamp when it reorganizes the structure. In addition, the reorganization alters the format timestamp appropriately if it changes the format of the structure. Because partitions cannot be reorganized, the partition records section of the control file remains unaltered.

If you alter pack assignments using SYSTEM/DMCONTROL, the family change bit in the control file is set to 1. As a result, any future DASDL updates or reorganizations do not affect the family specifications in the control file. If you want the DASDL update or the reorganization to update pack assignments, then before performing the DASDL update or reorganization reset the family change bit in the control file by using SYSTEM/DMCONTROL with the OVERRIDE FAMILY option.

After a file or a record format reorganization completes, the description file contains the new reorganization information as well as the old information. After the reorganization, if you try to update the control file without first recompiling the DASDL source file, the control file update fails with a format timestamp mismatch on the reorganized structure.

To avoid this situation

- Remove any reorganization statements from the DASDL source file after a file or a record format reorganization.
- Recompile the DASDL source file with the UPDATE option to remove old information from the description file.

Once the old reorganization information is removed from the description file, DMCONTROL updates can process successfully.

QUIESCERDBRESET

Disables the Remote Database Backup capability of a database copy in the state of quiesce. This operation is required before the copy can be used for processing.

LOCKEDFILE

Sets or resets the LOCKEDFILE file attribute of the database files (including the control file) to TRUE or FALSE. It allows the LOCKEDFILE attribute setting to be specified without performing a DASDL update.

Specifying LOCKEDFILE SET, sets the LOCKEDFILE file attribute of all the database files to TRUE.

Specifying LOCKEDFILE RESET, resets the LOCKEDFILE file attribute of all the database files to FALSE.

The modification of the LOCKEDFILE attribute will take effect on the next created audit file.

DMUTILITY sets the LOCKEDFILE file attribute on all dump files when LOCKEDFILE is set in the control file.

SECURITYADMIN

This option sets the SECURITYADMIN file attribute of the control file, data files, new audits, and backup files. It allows the SECURITYADMIN attribute to be SET or RESET without performing a DASDL update. Refer to the *DASDL Programming Reference Manual* for SECURITYADMIN restrictions.

Specifying SECURITYADMIN SET sets the SECURITY ADMIN file attribute of all the database files, including the control, audit, and backup files to TRUE. Specifying SECURITYADMIN RESET resets the SECURITY ADMIN file attribute of all of the database files, including the control, audit, and database backup files to FALSE. By default, this attribute is RESET.

The SECURITYADMIN attribute for data files and the control file is modified immediately. The SECURITYADMIN attribute for audit files is modified at the next audit switch for the new audit file. The modified SECURITYADMIN attribute is applied when new backup files are created.

When SECURITYADMIN is set for a database, it places restrictions on how a database is maintained. For example, only a security administrator (someone who has a usercode designated as PU (privilege user) and has SECADMIN privileges) can copy files with the SECURITYADMIN file attribute set. In addition, only a security administrator can perform a backup procedure such as DMUTILITY DUMP. For additional information, refer to the *Security Overview and Implementation Guide*.

The `OVERWRITE SECURITYADMIN` option causes the database override `SECURITYADMIN` change bit in the control file to be `RESET`. Subsequent `DASDL` updates can then set the `SECURITYADMIN` designation to the values specified in the `DASDL` source.

SENSITIVEDATA

Sets the `SENSITIVEDATA` file attribute of the database files (including the control, audit and backup files) to `TRUE` or `FALSE`. Use this option to specify the `SENSITIVEDATA` attribute setting without performing a `DASDL` update.

Specifying `SENSITIVEDATA SET` sets the `SENSITIVEDATA` file attribute of all the database files (including the control, audit and backup files) to `TRUE`.

Specifying `SENSITIVEDATA RESET` resets the `SENSITIVEDATA` file attribute of all the database files (including the control, audit and backup files) to `FALSE`. By default, this attribute is `RESET`.

The `SENSITIVEDATA` file attribute of the data and control files is modified immediately. The `SENSITIVEDATA` file attribute of the audit files is modified at the next audit switch for the new audit file. The modified `SENSITIVEDATA` attribute is applied when the new backup files are created.

Refer to the *File Attributes Programming Reference Manual* for additional information about the `SENSITIVEDATA` file attribute.

LOGACCESS

Enables or disables all `LOGACCESS`-capable structures in the database or changes the `DMVERB` list for a `LOGACCESS`-capable structure. The `LOGACCESS` command can only be performed when the database is closed, and the changes take effect on the next database open.

Specifying `LOGACCESS SET` enables all `LOGACCESS`-capable structures in the database.

Specifying `LOGACCESS RESET` disables all `LOGACCESS`-capable structures in the database.

The `<dmverb list>` is the list of permitted verbs.

Specifying the `<log access dmverb change>` option allows changing the `DMVERB` list for a `LOGACCESS`-capable structure without performing a `DASDL` update. The `<structure name>` is a data set name and changes to the data set are applied to its associated sets and or subsets.. The data set must be `LOGACCESS`-capable, and duplicate structure specification causes a syntax error.

When specifying the `EXCEPT <dmverb list>` option, the `<dmverb list>` construct following the word `EXCEPT` indicates the `DMVERBS` that are to be removed from the list of `DMVERBS` that were permitted by the `ALL` option

When specifying ALL as the structure name, the change applies to all LOGACCESS-capable data sets in the database.

The ALL option in <dmverb definition> specifies all DMVERBS. While the DMVERB list displayed in the control file listing for the structure includes FINDLOB, ASSIGNLOB and DELETEDLOB, these DMVERBS only apply to XE structures.

Examples

The following are LOGACCESS examples.

Example 1

The following example changes the DMVERB list for all LOGACCESS-capable structures in the database so only the DELETE and LOCKSTORE commands can be logged.

```
R$SYSTEM/DMCONTROL ("DB=MYDB LOGACCESS STRUCTURE ALL DMVERBS =
                    (DELETE, LOCKSTORE)");
```

Example 2

The following example changes the DMVERBS list for structure D1 and D2 to perform logging on the LOCK DMVERB and for structure D3 to perform logging on the LOCKSTORE DMVERB.

```
R$SYSTEM/DMCONTROL ("DB=MYDB LOGACCESS
                    STRUCTURE D1, D2 DMVERBS=(LOCK),
                    STRUCTURE D3 DMVERBS = (LOCKSTORE)");
```

Note: This function can only be performed when the database is closed; the changes take effect on the next database open. On the next database open, a DMS ACCESS log entry which shows the mix number and BOT timestamp of the last DMCONTROL LOGACCESS run, is written to the system log.

Example 3

The following log entry example shows the information on the last DMCONTROL LOGACCESS RESET statement.

```
DMS    7808  (PROD)OBJECT/HLI/TESTDB ON TESTPK.      (STACK 09C1)
          DATABASE ACCESS      : DISABLE LOGACCESS
          USERCODE              : PROD
          SOURCENAME            : US-USER1/LPROD/263808/CANDE/1
          DATABASE NAME        : (PROD) TESTLOBDB      (MIX 7809,
                                                    STACK 09C1)
          STRUCTURE NAME       : *
          DMCONTROL MIX        : 7807
          DMCONTROL BOT        : 08/14/2013 14:39:52
```

Example 4

The following log entry example shows the information on the last DMCONTROL LOGACCESS statement which changed the DMVERB list on structures.

```
DMS      7669 (PROD)OBJECT/HLI/TESTDB ON TESTPK.      (STACK 0862)
          DATABASE ACCESS   : CHANGE VERBLIST
          USERCODE          : PROD
          SOURCENAME        : US-USER1/LPROD/263808/CANDE/1
          DATABASE NAME     : (PROD) TESTLOBDB (MIX 7670,
                               STACK 0862)
          DMCONTROL MIX     : 7668
          DMCONTROL BOT     : 08/14/2013 14:17:26
```

Refer to the “Logging Data Access” section later in this guide for additional information.

LOCKPROGRAM

Initiates the database stack, DMSUPPORT, and RDBSUPPORT as locked processes by making use of the LP (Lock Program) MCP system command. This prevents these three processes from being discontinued. For additional information, refer to LP (Lock Program) in the *System Commands Reference*.

By default, the LOCKPROGRAM option is reset.

The LOCKPROGRAM option can be changed either by a DASDL update or by performing a DMCONTROL operation. The value of the option is listed in the control file content.

DOC

Role-based access control enables administrative access to a database from Database Operations Center (DOC) in order to prevent unintended or accidental access by any user for a given Database Operations Center-supported function. It allows the owner of an Enterprise Database Server database defined in the DASDL source file to provide additional restrictions for Database Operations Center users. These restrictions include accessing the database as well as accessing certain operations of the database. Database Operations Center role-based access control allows a database owner to define a role that permits certain predefined permissions of Database Operations Center task groups. After defining the roles, the owner can assign roles to each usercode.

Database Operations Center facilitates the policy setting of role-based access control for a database. Only the user who is defined as the owner of the DASDL-defined database can use role-based access control to perform the policy management. Once a policy is created, it stays in the control file of the database. Database Operations Center users have rights to perform database tasks as defined in the security policies.

The following are valid permissions and the permissions contained in each of them:

ANALYSIS

- List Audit File Contents (Print Audit)
- List Database Contents (DMUTILITY LIST)
- List the Status of Database files
- Monitor Database and Status

- Check Database Integrity (DBCertification)
- Verify Database backup
- List Backup Directory
- Verify Audit Files
- View Multi-Dump Tape Directories
- View Audit File Tape Directory

BACKUP

- Copy a Dump from disk
- Create Database backup
- Duplicate Database backup
- Create Multi-Dump Tape Directories
- Copy Audit files
- Create Audit file Tape Directory
- Verify Database backup
- List Backup Directory
- Verify Audit Files
- View Multi-Dump Tape Directories
- View Audit File Tape Directory
- Maintain Catalog backup

CONFIGURATION

- Control File Override Functions
- Unlock Control File
- Update Control File
- Set/Reset LOCKEDFILE Option
- Set/Reset Audit Family Index Option
- Change File Titles/Family
- Change Data File Family
- Change Security File Title
- Change Data Path
- Change Statistics Location
- LOB Maintenance
- Maintain Catalog backup

DASDL GENERATION

- Compile Database from DASDL
- Create new Control File
- Initialize Database

QDC

- Suspend Database
- Resume Database
- Create Quiesce database Copy
- Disable/Enable Database Access

REORGANIZATION

- Defragment Database
- Reorganize Database (Buildreorg/Run Reorg)
- Reorganization Wizard

RECOVERY

- Partial/Whole Database Recovery
- Recover Database
- Recover Control File
- Restore Control File from QDC
- Restore from Quiesce Database Copy
- Copy Database from Backup
- Reset Database In-use bit

RDBADMIN

- Initialize the Remote Auditing Control File
- Modify or View Host Information
- Modify the Audit File Transmission Mode
- Enable the Remote Auditing Capability
- Clone the DB
- Clone Specific Structures
- Perform a Host Takeover or Type Change
- Acknowledge the Manual Transfer of Audit file
- Cancel the Remote Auditing Capability

RDBMONITOR

- View Remote Audit Status
- View Remote Audit Statistics

Notes:

- *Users with privileged usercodes can use CANDE, but not Database Operations Center, to define or redefine the Database Operations Center role-based access control for the database.*
- *Once Database Operations Center role-based access control is set-up, no user is permitted to access the database from Database Operations Center unless authorized.*
- *Once Database Operations Center role-based access control is set-up, the database owner is added to the SUPERUSER role. The SUPERUSER role allows users unrestricted access to all forms in Database Operations Center for the specific database. The user is assigned all permissions of Database Operations Center task groups, and these permissions cannot be removed by any privileged user codes until the database owner is changed.*

Examples

The following is an example of how to create a role:

```
RUN $SYSTEM/DMCONTROL ("DB=MYDB DOC ADD ROLE = GROUP1ROLE ")
```

The following example assigns permissions to a role:

```
RUN $SYSTEM/DMCONTROL("DB = MYDB DOC IN ROLE GROUP1ROLE  
ADD PERMISSION = ANALYSIS")
```

The following example assigns users to a role:

```
RUN $SYSTEM/DMCONTROL("DB = MYDB DOC IN ROLE GROUP1ROLE ADD  
USER = MYUC1,MYUC2")
```

The following example deletes permissions from a role:

```
RUN $SYSTEM/DMCONTROL("DB = MYDB DOC IN ROLE GROUP1ROLE  
DELETE PERMISSION = ANALYSIS")
```

The following example deletes users from a role:

```
RUN $SYSTEM/DMCONTROL("DB = MYDB DOC IN ROLE GROUP1ROLE  
DELETE USER=MYUC1")
```

The following example deletes a role:

```
RUN $SYSTEM/DMCONTROL("DB = MYDB DOC DELETE ROLE =GROUP1ROLE ")
```


MAXUPDATEPERTR

MAXUPDATEPERTR allows you to control the maximum number of updates per transaction. The value of MAXUPDATEPERTR must be greater than 0 and not exceed 50,000. The update value of MAXUPDATEPERTR is affected at the next syncpoint. When the limit is exceeded, all of the previous updates are backed-out and the program receives a LIMITERROR 8. This feature can only be turned-off by performing a DASDL update.

RECOVER UPDATE

If the current control file is lost or corrupted, the RECOVER UPDATE function can be used to create a good control file from an old copy of the control file, the current database description file, and user input.

The control file used for recovery should be copied from the most recent DMUTILITY dump. For a tape dump, the control file is always dumped by DMUTILITY to cycle one, version one, of the dump tape. For a disk stream dump, the control file is always present on the disk stream file. For both tape dumps and disk stream dumps, the control file is present even if the control file was not specified in the dump list when the dump was taken.

For proper recovery of the control file, file discontinuities must not have occurred. Recovering from a control file that was dumped prior to a file discontinuity can invalidate dumps or cause incorrect recovery of database files. For this reason, it is mandatory that a dump of all affected structures be taken whenever a discontinuity occurs in any structure.

Note: *Using the RECOVER UPDATE option disables the Remote Database Backup capability of a database. For more information on recovering lost and corrupted database files in a Remote Database Backup environment, refer to the Remote Database Backup Operations Guide.*

A check is performed for RECOVER UPDATE to ensure that the description file and the old control file have compatible update levels. If the description file is marked as requiring reorganization, the update levels must be identical. If the description file is not marked as requiring reorganization, then the update level of the old control file must be the same as, or one less than, the description file update level.

If the update levels are not compatible, the RECOVER UPDATE function is not allowed and the following error message is displayed:

```
GIVEN CONTROL FILE TOO OLD FOR PROPER CONTROL FILE RECOVERY
```

When this level of incompatibility occurs and a proper control file is not available, RECOVER INITIALIZE must be used to recover the control file.

During control file recovery, version timestamps are obtained from the database files, and the creation timestamps are taken from the old control file. The database description file is used to initialize the rest of the control file.

You are asked to supply information which cannot be correctly recovered from either the current database description file or the old control file.

If the database is audited, you are asked for the current audit file number. You can specify the current audit file number through the `AUDITNUM = <integer>` syntax. The database in-use bit is turned on, forcing `DMRECOVERY` to run. `DMRECOVERY` then assigns the next audit block serial number in the control file.

The `RECOVER UPDATE` function should be used when recovering a control file that contains preallocated direct data set information. Preallocated direct data sets contain a format level of 1 in the control file, indicating that word 0 (zero) of block 0 (zero) contains a `DATAEOF` value. If a control file is used which indicates a conventional direct data set (format level 0) when the direct data set is actually preallocated (format level 1), then all software functions correctly, except that `DMUTILITY` dumps both the conventional and preallocated rows of the file.

If the transaction file number in the TPS information of the old control file is greater than zero, you are asked for the current transaction file number. In addition, if you input a nonzero value for the transaction file number, you are asked for the current transaction block number and current transaction offset. If you input the value 0 for the current transaction number, the transaction block number and offset are automatically assigned the value 0 by `DMCONTROL`.

If you encounter difficulties while using the `RECOVER UPDATE` option, refer to “Potential Problems with `RECOVER UPDATE`” later in this section.

If any settings associated with either the `POPULATIONINCR` or `POPULATIONWARN` option have been changed through the `VDBS` interface, you can reenter the settings the next time the database is active. The preferred alternative is to make the changes by performing a `DASDL` update. This action has the advantage of also updating the description file.

If the `POPULATIONINCR` option is defined in the schema, performing a `RECOVER UPDATE` resets the `AREAS` value to the value contained in the description file. If an automatic population increase took place, the new control file is adjusted to match the physical data file the first time an update program opens the affected structures.

If the `RECORDCOUNT` option is set in the `DASDL`, the structure record count is no longer valid after this operation. A garbage collection with the `CHECKRECORDCOUNT` option should be performed to recalculate the number of records in a structure. A warning message, such as the following, is displayed if the structure is touched by the first opener:

```
WARNING:
RECORD COUNT function for <structure name> is temporarily
deactivated due to the previous INITIALIZE, RECOVER INITIALIZE
or RECOVER UPDATE operation. Perform a garbage collection
reorganization with the CHECKRECORDCOUNT option to reactivate
with the correct record count.
```

RECOVER PARTITIONS

Used only when `RECOVER UPDATE` or `RECOVER INITIALIZE` fails to recover the partition information in the control file. This situation occurs if the `PARTITIONINFO` data set needs to be reconstructed. The `PARTITIONINFO` data set should be reconstructed, and then `RECOVER PARTITIONS` used to finish recovery of the control file.

RECOVER INITIALIZE

Used to create a usable control file when the control file is lost or corrupted, and a control file cannot be loaded from any DMUTILITY dump tape. This function creates a new control file from the database description file and user input. The creation timestamp of all structures is assigned the value 0. Therefore, you should immediately dump the entire database after using this function. The TPS information in the new control file is also assigned the value 0.

The control file should be recovered using the RECOVER INITIALIZE function only as a last resort. RECOVER UPDATE is the preferred recovery method.

If any settings associated with either the POPULATIONINCR or POPULATIONWARN option have been changed through the Visible DBS interface, you can reenter the settings the next time the database is active. The preferred alternative is to make the changes by performing a DASDL update. This action has the advantage of also updating the description file.

If neither OVERWRITE nor DONTOVERWRITE is specified and a control file exists, DMCONTROL waits and issues a message asking if you wish to continue. If OVERWRITE is specified and a control file for the database exists, DMCONTROL overwrites the existing control file. If DONTOVERWRITE is specified and a control file for the database exists, DMCONTROL returns an error.

You have the option to specify the current audit file number through the AUDITNUM = <integer> syntax. If you do not, DMCONTROL prompts you for the current audit file number. You also have the option to specify through the TAPEDIR or NOTAPEDIR syntax whether a tape directory is present. If you do not, DMCONTROL prompts you for information on the current directory.

If you do not want DMCONTROL to prompt you, you must specify information for all of the following syntax: OVERWRITE or DONTOVERWRITE, AUDITNUM, and TAPEDIR or NOTAPEDIR.

If the RECORDCOUNT option is set in the DASDL, the structure record count is no longer valid after this operation. A garbage collection with the CHECKRECORDCOUNT option should be performed to recalculate the number of records in a structure. A warning message, such as the following, is displayed if the structure is touched by the first opener:

```
WARNING:
RECORD COUNT function for <structure name> is temporarily
deactivated due to the previous INITIALIZE, RECOVER INITIALIZER
or RECOVER UPDATE operation. Perform a garbage collection
reorganization with the CHECKRECORDCOUNT option to reactivate
with the correct record count.
```

OVERRIDE AUDITBUFFERS

Causes the database OVERRIDE AUDITBUFFERS change bit in the control file to be turned off so that the subsequent DASDL update returns the audit buffers designations to the audit buffers specified in the DASDL source and changes the audit buffers value of the code files to the value specified in the DASDL source.

If the audit buffers change bit is 1, the audit buffers specification in the new control file remains as it was in the old control file.

If you decide to change the value for audit buffers by way of a DASDL update after you change the value by using the Visible DBS command `AUDIT BUFFERS`, you must specify `OVERRIDE AUDITBUFFERS`. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified with the Visible DBS command `AUDIT BUFFERS`.

OVERRIDE AUDITSECTIONS

Causes the database `OVERRIDE AUDITSECTIONS` change bit in the control file to be turned off so that the subsequent DASDL update returns the audit sections designations to the audit sections specified in the DASDL source and changes the audit sections value of the code files to the value specified in the DASDL source.

If the audit sections change bit is 1, the audit sections specification in the new control file remains as it was in the old control file.

If you decide to change the value for audit sections by way of a DASDL update after you change the value by using the Visible DBS command `AUDIT SECTIONS`, you must specify `OVERRIDE AUDITSECTIONS`. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified with the Visible DBS command `AUDIT SECTIONS`.

OVERRIDE HL

Causes the database in-use bit (also referred to as the halt/load bit) in the control file to be changed to zero. This option is only valid for unaudited databases.

In general, when the database in-use bit is on, questions are raised about the integrity of the database; therefore, `OVERRIDE HL` is intended for use only in very limited circumstances when you have more knowledge about the state of the database than does the control file. It is not an alternative to a normal recovery with reloading and reprocessing. Using this function requires the utmost care; it must not be used when there is any doubt about the state of the database.

Error messages are produced if the database is audited, or if the database in-use bit is initially off.

The control file program requires exclusive use of the control file in order to perform this function.

OVERRIDE FAMILY

Causes the database family change bit in the control file to be turned off so that the subsequent DASDL update returns the family designations to the family names specified in the DASDL source and changes the name texts of the code files to the names specified in the DASDL source.

If the family change bit is 1, the update of the control file is done directly on the current control file in order to retain the family specifications.

If you decide to change the family name by way of a DASDL update after you have changed the name by using a DMCONTROL operation, you must perform a control file override. The override enables the system to recognize that the family name in the DASDL update takes precedence over the name you specified in the DMCONTROL operation.

OVERRIDE GUARDFILETITLE

Causes the database security file change bit in the control file to be turned off so that a subsequent DASDL update returns the security file designations to the titles specified in the DASDL source.

When the database security file change bit is set, the DASDL changes to the security file titles stored in the control file are ignored in order to retain the security specifications modified through a DMCONTROL operation.

If you decide to change any of the security file titles by way of a DASDL update after you have changed a title by using a DMCONTROL operation, you must perform a control file override. The override enables the system to recognize that the security title in the DASDL update takes precedence over the title you specified in a previous DMCONTROL operation.

OVERRIDE POPULATIONWARN

Causes the database population warning protection bit in the control file to be reset so that the next DASDL update changes the POPULATIONINCR settings to the values specified in the DASDL source.

If you decide to change the value for POPULATIONWARN by way of a DASDL update after you change the value by using the Visible DBS command POPULATIONWARN, you must specify OVERRIDE POPULATIONWARN. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified with the Visible DBS command POPULATIONWARN.

OVERRIDE POPULATIONINCR

Causes the database population increment protection bit in the control file to be reset so that the next DASDL update changes the POPULATIONINCR settings to the values specified in the DASDL source.

If you decide to change the value for POPULATIONINCR by way of a DASDL update after you change the value by using the Visible DBS command POPULATIONINCR, you must specify OVERRIDE POPULATIONINCR. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified with the Visible DBS command POPULATIONINCR.

OVERRIDE USEREORGDB

Discontinues a USEREORGDB option that has been recessed and that you no longer want to complete. You can also use this option to reset the USEREORGDB status when a REORGDB reorganization fails.

OVERRIDE DATAPATH

Causes the database override datapath change bit in the control file to be reset. Subsequent DASDL updates can then set the DBPATH designation to the values specified in the DASDL source.

Refer to [Section 16, Using Permanent Directory Databases](#).

OVERRIDE LOCKEDFILE

Causes the database override LOCKEDFILE change bit in the control file to be reset. Subsequent DASDL updates can then set the LOCKEDFILE designation to the values specified in the DASDL source.

OVERRIDE SECURITYADMIN

Causes the database override SECURITYADMIN changed bit in the control file to be reset. Subsequent DASDL updates can set the SECURITYADMIN designation to the values specified in the DASDL source

OVERRIDE SENSITIVEDATA

Causes the database override SENSITIVEDATA changed bit in the control file to be reset. Subsequent DASDL updates can set the SENSITIVEDATA designation to the values specified in the DASDL source.

OVERRIDE AUDITFAMINDEX

Causes the database override famindex bit in the control file to be reset. Subsequent DASDL updates can then set either SETFAMINDEX or RESETFAMINDEX designation to the values specified in the DASDL source.

<data file family change>

Allows changes to the pack family specifications of structures, the primary audit, and the secondary audit, as described in the following text.

To ensure the DMCONTROL statement changes the family in the correct control file when the database is a quiesce database copy or has been used to create a quiesce database copy, CF and CFOLD must be file-equated. If file equation is not used, a syntax error is issued, and the DMCONTROL statement does not update the control file.

For more information about quiesce database copies, refer to [Section 6, Backing Up a Database](#).

STRUCTURE

Changes the pack family designation of structure <structure name> to family <family name>. A structure name can be a data set, set, or subset name. It also can be the alias name of a data set, set, or subset. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for additional information about alias names.

Only structures that have physical data (data sets and sets) can have their family designation changed. Structures that do not have physical data, such as Access structures, are treated as though they could not be found, and an appropriate message is returned.

If a structure is added to an existing database, the new structure is created on the default pack or on the pack designated for the structure in the DASDL source file. Any previous runs of SYSTEM/DMCONTROL that altered pack designations have no effect on the new structure. To alter the pack designation for the new structure, run SYSTEM/DMCONTROL again.

In the following example, the family of the structure INVENTORY is changed to the family INVENTORYDATA. When the database is opened, the Access routines opens the INVENTORY structure on the pack family INVENTORYDATA:

```
STRUCTURE INVENTORY FAMILY = INVENTORYDATA
```

FAMILY

Changes the pack family designation of all structures that reside on the family <packfamily1> to the family <packfamily2>. If no structures reside on <packfamily1>, a warning is returned when this function is attempted.

If a structure is added to an existing database, the new structure is created on the default pack or on the pack designated for the structure in the DASDL source file. Any previous runs of SYSTEM/DMCONTROL that altered pack designations have no effect on the new structure. To alter the pack designation for the new structure, run SYSTEM/DMCONTROL again.

In the following example, all structures that resided on the family INVENTORYDATA are opened on the family DBDATA:

```
FAMILY INVENTORYDATA = DBDATA
```

AUDITFAMILY

Changes the designation of the pack family on which the audit trail resides to the family <family name>. If the database is not audited or the audit is being directed to tape, a warning message is returned when this function is attempted.

In the following example, the audit trail is created on the family AUDITPACK:

```
AUDITFAMILY = AUDITPACK
```

Note: If you set the DASDL audit trail attribute *COPY*, then the disk audits are copied to tape automatically. If you set this DASDL attribute and change the primary or secondary audit from one pack to another, then the last disk audit written to the old destination is not copied automatically to tape. Instead, you must manually initiate the *COPYAUDIT* job to back up the last disk audit on the old pack.

ALTERNATE AUDITFAMILY

Changes the designation of the pack family on which the alternate audit trail resides to the family <family name>. If the database is not audited or the alternate audit is being directed to tape, a warning message is returned when this function is attempted. The semantics for this function are similar to the *AUDITFAMILY* command.

SECAUDITFAMILY

Changes the designation of the pack family on which the audit trail is duplicated to the family <family name>. If the database specification does not include duplicated audit, or if the duplicated audit trail is being directed to tape, the command is not processed and a warning message is returned.

In the following example, the duplicated audit trail is created on the family *DUPAUDITPACK*:

```
SECAUDITFAMILY = DUPAUDITPACK
```

ALTERNATE SECAUDITFAMILY

Changes the designation of the pack family on which the alternate audit trail is duplicated to the family <family name>. If the database is not audited or the alternate audit is being directed to tape, a warning message is returned when this function is attempted. The semantics for this function are similar to the *SECAUDITFAMILY* command.

<code file family change>

Changes the family of the designated code file or *COPYAUDIT* WFL job. You can either designate a family or use the keyword *DEFAULT*. The *DEFAULT* keyword causes the code file or WFL job to have no family designation. Therefore, the *FAMILY DISK* specification of the user opening the database determines the location of the code file.

To ensure the *DMCONTROL* statement changes the code file title in the correct control file when the database is a quiesce database copy or has been used to create a quiesce database copy, *CF* and *CFOLD* must be file-equated. If file equation is not used, a syntax error is issued, and the *DMCONTROL* statement does not update the control file.

For more information about quiesce database copies, refer to [Section 6, Backing Up a Database](#).

Use the COPYAUDITPRIWFL option to change the family of the COPYAUDIT WFL job used to copy the primary audit file. Use the COPYAUDITSECWFL option to change the family of the COPYAUDIT WFL job used to copy the secondary audit file.

In the following example, the pack family of the DMSUPPORT code file is changed to DBDATA:

```
DMSUPPORT FAMILY = DBDATA
```

<code file title change>

Changes the title of the designated code file. The title can include a usercode, code file, and family. To override the title change, use the OVERRIDE FAMILY command. This command enables you to update the code file title in the control file without performing a DASDL update.

In the following example, the title of the DMSUPPORT code file has been changed to (PROD) DMSUPPORT/TESTDB ON DBDATA:

```
DMSUPPORT TITLE = (PROD) DMSUPPORT/TESTDB ON DBDATA
```

<security file title change>

Changes the title of the designated security file. The title can include a usercode, code file, and family name. To override the title change, use the OVERRIDE GUARDFILETITLE command. This command enables you to update the security file titles in the control file without performing a DASDL update. For a security file title change to take effect, the SECURITYGUARD option must be specified in DASDL first. Once the guard file title is defined in DASDL, the title can then be changed using DMCONTROL.

In the following example, the title of the guard file for the secondary audit is changed to (PROD) XXX on DBDATA:

```
DB=TEST SECAUDITGUARDFILE TITLE = (PROD) XXX on DBDATA
```

<data path change>

Changes the permanent directory location. When making a data path change, you must move the data files, audit files, control file, and the tailored software to the new location.

Once you have moved a database to a location that differs from the specified DASDL source (a new DBPATH specification), equate the files CFOLD and CF when running DMCONTROL. For additional information, refer to “<data file family change>” and “<code file family change>” earlier in this topic.

Also, refer to [Section 16, Using Permanent Directory Databases](#).

<statistics location change>

Changes the location of the statistics report. If you choose to specify the packname, the statistics output is reported as follows:

```
<dbusercode>DBSTATS/<dbname>/YYYYMMDD/HHMMSS ON <packname>
```

If it is a permanent directory database, the DATAPATH is assumed and stored as the following, where YYYYMMDD and HHMMSS are the date and time when the statistics are generated:

```
<path name>/DBSTATS/<dbname>/YYYYMMDD/HHMMSS ON <packname>
```

<audit family index change>

SETFAMINDEX

Forces all rows of a sectioned audit file to be assigned to the same family.

RESETFAMINDEX

Enables the MCP to assign rows of a sectioned audit file using a round-robin method.

RESETCLONED option

The RESETCLONED option is intended for internal use only in the Remote Database Backup environment by clone WFL jobs generated by Database Operations Center. For additional information about the clone process in the Remote Database Backup environment, refer to the *Remote Database Backup Planning and Operations Guide*.

SETCLONED option

The SETCLONED option is intended for internal use only in the Remote Database Backup environment by clone WFL jobs generated by Database Operations Center. For additional information about the clone process in the Remote Database Backup environment, refer to the *Remote Database Backup Planning and Operations Guide*.

Potential Problems with RECOVER UPDATE

The following text lists problems that can occur when you use the RECOVER UPDATE option with the TPS RECOVERTPSINFO option. The text also identifies solutions for the problems.

Example 1

No TPS users exist but the CFTPSFLAG flag is corrupted and now has a nonzero value. RECOVERTPSINFO asks the nonTPS user to enter the transaction file number.

Solution

The solution is to transmit the number 0. All other TPS information is assigned the value 0, which is normal for a nonTPS user.

Example 2

TPS users exist but the CFTPSFLAG flag is corrupted and now has the value 0. RECOVERTPSINFO automatically sets all TPS information to the value 0. If the TPS TRHISTORY file has audited a transaction and the TPS user reinitiates the TPS TRHISTORY file, the user loses all transactions that would have been processed between the latest transaction in the TPS TRHISTORY file.

Solution

The solution is to manually correct the TPS information in the new control file after the RECOVER UPDATE run, or to perform a RECOVER UPDATE run again using the old control file.

Example 3

A TPS user mistakenly inputs the number 0 instead of a nonzero value for the CFTPSFLAG flag. There is no synchronized recovery.

Solution

The solution is either to perform another RECOVER UPDATE run using a copy of the old control file, or to manually update the TPS information in the new control file.

Example 4

A TPS user performs a RECOVER INITIALIZE run.

Solution

The problems and solutions are exactly the same as in the Example 2.

Control File Recovery

You can recover the control file by loading it from a backup dump and rebuilding the entire database. This process, however, has several drawbacks. It involves much processing, is time-consuming, risks input/output errors on the dump tapes and audit files, requires that the database software used be at the same update level as the control file, and might not be able to bring the database past a point in the audit where the update level was increased. However, upon successful completion of this process, the database is in phase.

As an alternative, the control file program can be run specifying one of the RECOVER options. The creation timestamps cannot be recovered exactly if the RECOVER INITIALIZE option is used, but they can be recovered if the RECOVER UPDATE option is used and the control file loaded is recent enough. If the creation timestamps are not recovered exactly, it is not generally possible to rebuild the database through any discontinuities in the audit.

For unaudited databases, the RECOVER functions are intended for use only under very limited circumstances. In all cases, you should have extra knowledge about the state of the database. In general, when a control file is lost for an unaudited database, the only safe action is to reload the database and reprocess the input. In some cases, creating a new control file is all that is necessary; for example, if the control file resides on a separate pack from the other files and that pack was corrupted. It is for such cases that the RECOVER functions are permitted for unaudited databases.

Control File Recovery and Change of Family

When you have changed the families of the database and code files with DMCONTROL rather than a DASDL update, a RECOVER UPDATE or RECOVER INITIALIZE statement might not rebuild all of the control file. The following paragraphs show how to recover a control file when a DMCONTROL family change has been made.

Nonpartitioned Databases

For nonpartitioned databases, do the following:

1. Mount the most current DMUTILITY dump tape and run DMUTILITY to copy the control file to disk.
2. Run DMCONTROL with a RECOVER option, preferably RECOVER UPDATE. DMCONTROL creates the new control file from the description file and from the control file copied from the dump tape.
3. Run DMCONTROL to change the families of the data and code files to the correct family.
4. An automatic recovery is performed when a user next opens the database.

Partitioned Databases

For partitioned databases, do the following:

1. Mount the most current DMUTILITY dump tape and run DMUTILITY to copy the control file to disk.
2. Run DMCONTROL with a RECOVER option, preferably RECOVER UPDATE. DMCONTROL creates the new control file from the description file and from the control file copied from the dump tape, locks the new control file, then performs a RECOVER PARTITIONINFO to recover the partition entries.

The PARTITIONINFO data set is read, causing database recovery. Recovery opens the data files that need to be recovered. If the data files were moved, a NO FILE condition occurs for each data file that could not be found. To allow the recovery to complete, do one of the following:

- Move the data files.
 - Terminate recovery and DMCONTROL by entering *DS*.
 - Relabel the pack.
3. Run DMCONTROL to change the families of the database and the code files.

If data files have been moved to allow database recovery to complete (the first alternative in step 2) and the families are changed with DMCONTROL (step 3), the data files need to be moved to reflect their designation in the control file.

If recovery and DMCONTROL were terminated (the second alternative in step 2), run DMCONTROL with a RECOVER PARTITIONS to perform the recovery. This causes database recovery to be performed and recovers the partition entries.

4. The RECOVER PARTITIONS might cause the partition data structure families to revert to the family specification of the description file. If the family specifications were changed with DMCONTROL, run DMCONTROL again to change the partition structures to the correct families.

Control File Recovery and Change of Population Control Attributes

When you have changed the population of the control attributes with DMCONTROL rather than a DASDL update, a RECOVER UPDATE or RECOVER INITIALIZE statement might not rebuild all of the control file. The following procedure shows how to recover a control file when a change of population control attributes has been made:

1. Mount the most current DMUTILITY dump tape and run DMUTILITY to copy the control file to disk.
2. Run DMCONTROL with a RECOVER option, preferably RECOVER UPDATE. DMCONTROL creates the new control file from the description file and from the control file copied from the dump tape.

An automatic recovery is performed the next time the database is opened.

3. Following the recovery, you can use the VDBS interface to change any population attributes that have been dynamically altered since the last DMUTILITY dump. The preferred alternative is to make the changes by performing a DASDL update. This action has the advantage of also updating the description file.

DMUTILITY CANCEL Statement

The DMUTILITY *CANCEL* statement unlocks the control file.

Syntax

— CANCEL ——————|

Explanation

The CANCEL statement can unlock the control file if it was locked for the following reasons:

- An offline dump fails.

To perform an offline dump, DMUTILITY must ensure that no programs are updating the database during the time of the dump. To guarantee that no programs are updating the database, DMUTILITY marks the database control file as being in exclusive use by DMUTILITY.

If a CANCEL statement is executed during an offline dump, update programs would be allowed to begin executing before the actual completion of the offline dump. This would result in a corrupted offline dump. In order to ensure the validity of an offline dump, the CANCEL function must have exclusive use of the control file to prevent any premature cancels from taking place.

If for any reason DMUTILITY is discontinued before the offline dump completes, the control file must be unlocked before any processing against the database (other than a restart of the offline dump) can proceed.

- An offline copy fails.

To perform an offline copy, DMUTILITY must ensure that no programs are accessing the database during the copy. To guarantee that no programs access the database, DMUTILITY marks the source control file as being in exclusive use by DMUTILITY. DMUTILITY also marks the destination control file if the copy is AS or ONTO another database. If DMUTILITY fails to complete the copy, use the CANCEL statement to unlock the source control file.

- An offline certification fails.

If the certification is not online, Database Certification marks the control file as being in exclusive use by Database Certification.

Initializing Database Files

A database must first be initialized before it can be used by any Enterprise Database Server program. The initialization procedure is handled by DMUTILITY.

The INITIALIZE statement initializes the entire database or specific structures within the database. This statement creates files with a security type of PRIVATE. To create files with a security type of GUARDED, use structure SECURITYGUARD files in the DASDL specifications.

Following a successful initialization, the file-state field in the control file for initialized structures is assigned the value CFAUDINZ for audited databases and CFFILENORMAL for unaudited databases.

Although DMUTILITY ensures that there are no dangling set references, a structure initialization only creates that structure and does not logically delete all records. Thus, AGGREGATE, POPULATION, COUNT item, and AA words (such as links that point to the structure being updated) are not updated. To delete all records in the Enterprise Database Server and appropriately update AGGREGATE, POPULATION, COUNT items, and pointers, each record of the data set must be deleted with a user program or with the Extended Retrieval with Graphic Output (ERGO) program.

Rules for Initialization

Initialization has the following rules:

- When a disjoint data set is initialized, all sets that refer to the data set and all embedded structures within the data set are automatically initialized.
- If you initialize a disjoint structure and there is a population item in the global data record for the structure, the population information in the global data record is not updated. As a result, the population information stored in the global data record and the actual number of records that are in the structure might not match. Despite the inconsistency in population information, all the records in the structure are accessible.
- If you initialize the global data record to update the population item of the initialized structure, then all the population items declared in the global data record are initialized. As a result, the population information in the global data record and the actual number of records in the structures might not match. Despite the inconsistency in population information, all the records in the structure are accessible.
- If you initialize a disjoint structure for which an aggregate item exists in the global data record, all disjoint structures having an aggregate item in the global data record must be initialized along with the global data record.
- If you initialize a structure containing a counted link, you must also initialize the structure referenced by the link.
- If the INITIALIZE statement fails for any reason, reissue the command to initialize the database structure or structures.

Note: *The CANCEL statement does not override a failed INITIALIZE statement.*

- If you initialize a structure that is referenced by an embedded manual subset, you should also initialize the owner of the embedded manual subset.
- You can initialize disjoint sets and automatic subsets only when you are also initializing the data sets they reference. You can initialize disjoint manual subsets even when you are not initializing the data sets they reference.
- You cannot initialize structures if the DASDL LOCKEDFILE option is set. Either update the database with the LOCKEDFILE option reset, or set the LOCKEDFILE file attribute for the structures you want to initialize to false.
- You cannot initialize structures if a DMUTILITY DUMP of the same database is in progress.
- If you are initializing a permanent directory database, refer to [Section 16, Using Permanent Directory Databases](#), for information.

Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on the LOCKEDFILE option.

Before initializing structures, DMUTILITY locks the control file to ensure that it has exclusive use of the database. Initialization can be performed in several steps. Different structures can be initialized in each step.

DMUTILITY INITIALIZE Statement

The following information illustrates the syntax required to run the initialization process:

Syntax

<initialize statement>

```

— INITIALIZE — = —————|
                    | ALL —————|
                    | , —————|
                    |<structure name>—<initialize options>—|
    
```

<online initialize statement>

```

— ONLINE — INITIALIZE —<structure name> —<initialize options> —|
    
```

<initialize options>

```

| ( — PREALLOCATE —<integer>— )—|
| ( REMOVEOLDFILE ) —————|
    
```

ALL or =

Initializes the entire database.

ONLINE INITIALIZE

The ONLINE INITIALIZE command allows the initialization of one structure at a time. DMUTILITY does not require exclusive use of the database control file during online initialization.

A structure can be ONLINE INITIALIZED if it meets the following criteria:

- The structure is a disjoint data set.
- The structure is not a restart data set.
- The structure is not linked to or from another data set.
- The structure is not partitioned.
- REAPPLYCOMPLETED and INDEPENDENTTRANS are SET.
- The structure cannot have a manual set or subset.

During the brief period that a structure is being initialized online, any application that executes a DMVERB for an affected structure encounters a DEADLOCK #4 exception.

During the online initialization process, the data set that is to be initialized and its corresponding sets will be saved as old files.

The old files for a data set are called:

OLD/<database name>/<data set name>/DATA

The old files for a set are called:

OLD/<database name>/<data set name>/<set name>

Use these saved files to restore the data set and its sets in the event the online initialization process is terminated abnormally. Use the REMOVEOLDFILE option to remove the old file at the end of the successful ONLINE INITIALIZE process.

<structure name>

Designates an identifier that represents the symbolic name of the structure as specified in DASDL. In the case of the global data record, it is recommended that the database name be used. In the case of the global data record of a modeled database, the name of the original database must be used. For ONLINE INITIALIZE, this must be a data set.

<initialize options>

Allows a fixed number of records to be preallocated when a direct data set is initialized by DMUTILITY. The <integer> construct specifies the number of records to be preinitialized. Any attempts to preallocate records for other than direct data sets are flagged as errors.



In addition, the PREALLOCATE option cannot be specified for sectioned direct data sets.

Direct data set preallocation reduces the overhead involved when new records are added to a direct data set that contains key values significantly larger than the last previous records. The overhead is saved by preallocating all of the required data blocks when convenient.

Specifically, to preallocate a direct data set, you must specify a fixed number of direct data set records to be preallocated when a direct data set is initialized using DMUTILITY. No run-time initialization is necessary when the Access routines add a record anywhere within the preallocated area. Also, when the structure is dumped through DMUTILITY, the preallocated areas are not dumped, but information is recorded that allows the structure to be reconstructed or rebuilt. Initialization can be repeated if interrupted by a halt/load or other failure.

With the ONLINE INITIALIZE command, the REMOVEOLDFILE option when specified, removes old files after a successful initialization. Note that during the initialization process, if the old file is present, the old files are renamed as follows:

For data set, they are renamed:

OLD/<database name>/DATA/<mmdd>/<hhmmssyy>

For set, they are renamed:

```
OLD/<database name>/<data set name>/<set name>/<mmdd>/<hhmmssyy>
```

When initialization is performed on a database or on structures within a database, all changes to the database are audited in the control file. When the database is next opened and updated, the changes are written to the audit file. DMRECOVERY can rebuild all specified structures in a database on which a DMUTILITY *INITIALIZE* has been performed, providing the audit file has been updated or the control file is not lost. If the control file is lost (because, for example, an old control file was reloaded) before the database is open updated, DMUTILITY *INITIALIZE* must be redone.

Notes:

- The *INITIALIZE* statement should be used only to create a new structure, and not to erase records contained within an existing structure.
- If the *INITIALIZE* statement is used to delete records within a structure, it does not update the *POPULATION* and *COUNT* control items that pertain to the structure or the *AA* words that point into the structure. To delete all records in a structure and ensure that *AGGREGATE* items, *POPULATION* items, *COUNT* items, and pointers are updated, you must delete each record of the data set by using either the *ERGO* program or a user program.
- If you are using the Open Distributed Transaction Processing product for the first time, use the *DMUTILITY INITIALIZE* statement to initialize the *RXGLOBALTR* and *RXSIBDESCS* data sets. When you update a database to include the *OPENOLTP* option and the *RMSUPPORT* library title, the Open Distributed Transaction Processing software automatically adds these two inquiry-only data sets to your database description file. If you try to update the data sets, a security error occurs.

REDISTRIBUTE Command

The REDISTRIBUTE command redistributes a database file of a disjoint data set across the family pack members of a multifamily pack while the database is in use. This command is useful when additional families are added to an existing multifamily pack. It can also improve I/O performance.

Syntax

```
— <options>— REDISTRIBUTE— <database file> —————|
```

<options>

```
OPTIONS ( ——— WORKERS = <integer> ——— )
```

<database file>

```
—————|  
| <structure name> —————|  
| <file name> —————|
```

<options>

You can specify the number of copy phase tasks to be run concurrently. Use this option for sectioned data sets or sectioned sets. The default is 1.

<database file>

The <structure name> must be a disjoint data set name. It redistributes the entire database file of the disjoint data set and it is set across the family pack member. The <file name> must be the file name of the data set or set.

Explanation

The REDISTRIBUTE command involves the following phases:

- INITIALIZE PHASE – initializes temporary files
- COPY PHASE – copies live database files to temporary files
- SWAP PHASE – swaps live database files with temporary files
- PURGE PHASE – purges old database files

There is no restart capability for this function. If DMUTILITY fails, DMUTILITY removes all temporary files. If a system fails during the REDISTRIBUTE command, you must manually remove temporary files. If a system fails during the SWAP phase, you must manually copy the old files as live database files.

Notes:

- *For disjoint index sequential sets, it is recommended to use online garbage collection to generate the new set where the areas will also be redistributed evenly.*
- *The pack must have sufficient space to accommodate the new and old database files.*
- *This command does not have the capability of redistributing a specific area of a file.*

Examples

The following example redistributes the disjoint data set and its sets:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB ON UITEST REDISTRIBUTE DS8");
```

The following example redistributes the one database file:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB ON UITEST REDISTRIBUTE  
TESTDB/DS8/DATA");
```

The following example redistributes the database file under the DS8 data set:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB ON UITEST  
REDISTRIBUTE TEST/DS8/=");
```

The following example redistributes one database file, which in this example is the set:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB ON UNITEST
REDISTRIBUTE TEST/DS8/SKEY");
```

Note: For disjoint index sequential sets, it is recommended to use the *OLGC*.

The following example redistributes the sectioned data set using the *WORKERS* option. In this example, the data set has 20 sections. The worker is set to 5 which means that 5 copy phase tasks will run concurrently until all the database files have been redistributed:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB ON UITEST OPTIONS(WORKERS=5)
REDISTRIBUTE TESTDB/DS9/DATA");
```

MIGRATEDB Command

The *MIGRATEDB* command allows you to add or delete disjoint data sets and their spanning sets and subsets without bringing down the database. This feature does not support adding sets or subsets to an existing data set; continue to use reorganization.

— *MIGRATEDB* —————|

You can automatically run the *MIGRATEDB* command through a *DASDL UPDATE* with the *MIGRATEDB*, *ZIP* and *DMCONTROL DASDL* options set. Refer to the *DASDL Programming Reference Manual* for the *MIGRATEDB DASDL* option. The command can also be run as a separate task if the *MIGRATEDB DASDL* option is set, and the *ZIP* and *DMCONTROL DASDL* options are reset.

Running *DMUTILITY* with the *MIGRATEDB* command performs the following tasks:

1. Initiates *USEREORGDB REORG* for new data set(s)
2. Waits for a quiet point, no transactions in progress
3. Stops all applications
4. Updates the control file
5. Replaces the *DMSUPPORT* library with a new *DMSUPPORT* library
6. Creates a new audit file for the update level change
7. Resumes all applications

Notes:

- The *MIGRATEDB* command only supports disjoint data sets.
- The *MIGRATEDB* command is not allowed for modeled databases.
- The data set cannot be a restart data set or a global data set.
- The data set cannot contain a link item or a structure of a link item.
- The data set cannot contain an embedded structure.
- The *MIGRATEDB* command is not supported for databases that have *Remote Database Backup* enabled.

Section 6

Backing Up a Database

In This Section

The following topics are covered in this section:

- Tools available for creating and managing database backups
- Understanding the database backup process
- Tasks related to creating and managing database backups
- Using the DMUTILITY *DUMP* and *APPEND* commands
- Using the DMUTILITY *VERIFYDUMP* command
- Copying database backups
 - Using the DMUTILITY *COPYDUMP* command
 - Using the DMUTILITY *DUPLICATEDUMP* command
- Listing a directory of a dump tape reel or disk file using the DMUTILITY *TAPEDIRECTORY* command
- Listing the contents of multidump tapes or recreating the fast access directory file for multidump tapes with the DMUTILITY *TAPESET DIRECTORY* command
- Cataloging the information in database backups
- Using the DMDUMPDIR program
- Using the DMUTILITY *BUILDDUMPDIRECTORY* command
- Recovering database backup catalog information
- Quick-reference information for all the syntax diagrams provided in this section

The information in this section applies to backing up a database to both tape and disk.

Terminology

Knowing the following terminology can help you understand the information in this section:

- A database backup is also referred to as a *database dump* or a *dump*.
- When a database is backed up to tape, the result of the operation is called a *tape dump*.
- Single dump tapes contain a single database backup from a single database.

Backing Up a Database

- Multidump tapes can contain many database backups from one or multiple databases.
- Database backups can be created on a multidump tape by using the DUMP and APPEND commands.
- When a database is backed up to disk, the result of the operation is called a *disk dump*.

Tools Available for Creating and Managing Database Backups

[Table 6-1](#) identifies the tasks you can perform in relation to creating and managing database backups and the headings in this section under which these tasks are described.

Note: *The tasks identified in this section can be initiated through Database Operations Center.*

Table 6-1. Tasks Related to Creating and Managing Database Backups

To perform this task . . .	Refer to . . .
Create a database backup.	DUMP and APPEND Commands (DMUTILITY)
Verify a database backup.	VERIFYDUMP Command (DMUTILITY)
Copy a database backup.	Copying Database Backups COPYDUMP Command (DMUTILITY) DUPLICATEDUMP Command (DMUTILITY)
Identify the information on a dump tape reel or in a dump disk file and report the contents of a multidump tape.	TAPEDIRECTORY Command (DMUTILITY) TAPESET DIRECTORY Command (DMUTILITY)
List the contents or recreate the fast access directory for a multidump tape or set of multidump tapes	TAPESET DIRECTORY Command (DMUTILITY)
Catalog the information stored in database backups.	Cataloging the Information in Database Backups DMDUMPDIR Program BUILDDUMPDIRECTORY Command (DMUTILITY)
Recover database backup catalog information.	Recovering Database Backup Catalog Information

Another method of creating backups is to create a quiesce database or a quiesce database copy. Refer to [Section 14, Using a Quiesce Database](#), for more information about this approach to creating a backup.

Understanding the Database Backup Process

Introduction

A database backup is a snapshot of an entire database or of parts of a database. A database backup is used for two primary purposes:

- To recover, with minimal loss of data, from a database or hardware failure
- To move or copy a database from one location to another

The DMUTILITY program through the DUMP command provides the facilities you need to create a database backup. Many options are provided with the DUMP command, including the ability to

- Create a backup copy of the database on tape or on disk.
- Generate the backup copy of the database when the database is offline or online.
- Select whether all or part of the database is to be backed up.
- Select whether the dump tape should be checked for errors as part of the dump process. By default, the DMUTILITY program automatically includes this check to ensure dump tapes have been written correctly and can be read.

Notes:

- *The DMSUPPORT title, including usercode and family, must be defined in the DASDL of a database for a backup to be accessed from a usercode that is different from the usercode of the initiator of the backup.*
- *While it is possible to back up a database by copying database and audit files with Library Maintenance, this method provides no database integrity checking and you are strongly discouraged from using it.*

The database backup process provided by the *DUMP* command does not back up the following files:

- Tailored software, such as the DMSUPPORT library and RMSUPPORT library
- Database description file
- Database application programs

While these files are not always needed for database recovery purposes, keeping a backup copy of the files enhances your ability to maintain, upgrade, and recover your database. Use the Library Maintenance COPY or ADD command with the COMPARE or VERIFY option to back up these files.

In addition, use the COPYAUDIT program to back up audit files. Refer to [Section 9, Copying Audit Files](#), for information on using the COPYAUDIT program.

Backing Up to Tape or Disk

You can choose to back up a database to tape or to disk. Choose the output medium that is most convenient for your site.

Multidump Tapes

Duplicate dump names are not permitted on the same multidump tape. In the event that an APPEND or COPYDUMP APPEND operation must be restarted, using the negative task value from the previous attempt does permit the dump name to be reused.

Dump names and multidump tape names can have only one node.

DMUTILITY creates a fast access directory for each multidump tape. The directory is created at the location specified by the DL LIBMAINTDIR system command. If there is no DL specification for LIBMAINTDIR, the directory is created on the default family of the user who created the multidump backup tape (that is, the family of the user who created the tape by using the DUMP command).

When multidump backup tapes are created, their ASSOCIATEDFILENAME attribute is set to the name of the fast access directory file. Each directory file is uniquely named and includes the date and time when the tape was created. This directory is used to provide the FASTLOCATE position for accessing the backup dumps stored on the tape.

Note: *Whenever you use an existing multidump tape on a system, the fast access directory for that tape must be present on the system. You can copy the directory from another system or create the directory for the tape by using the TAPESET DIRECTORY CREATE command.*

Unisys recommends that you do not move multidump tapes between systems to add dumps to them because this can result in inconsistent fast access directory files on the different systems. These inconsistent directory files can cause existing dumps to be overwritten.

Creating the Database Backup with the Database Offline or Online

You can control user access to a database while a database backup is in progress. During the backup process you can choose to

- Prevent all update users from accessing the database; that is, during the backup process, the database can be used only for inquiry purposes.
- Allow both update and inquiry users to access the database.

Selecting All or Part of the Database to Back Up

Each time you use the database backup process you can select which parts of the database to back up. The options available to you include selecting

- All database files
- Files according to the pack on which the files are located
- Files according to the directory
- Rows within a file
- Only those data blocks that have changed since the last dump

Database Control File

The database control file is automatically copied to the beginning of each tape reel or disk file of the backup dump.

For more information on the purpose and content of the control file, refer to [Section 2, Control File](#).

Timestamp Mismatch Errors

If a timestamp mismatch error occurs during the database backup process, the following message displays:

```
DISPLAY: <database name>: WRONG VERSION OF
         <file name>[<timestamp>]
ACCEPT: <database name>:
        CORRECT VERSION=<timestamp>.
        "AX RETRY OR OVERRIDE"
```

A timestamp mismatch error usually indicates an error that should be investigated and resolved before continuing with the backup process. If necessary, you can override the timestamp error—for example, when recovering the database control file. However, this action is not usually recommended.

To override the timestamp error, use the following command:

```
<mix number> AX OVERRIDE
```

If you choose to override the timestamp mismatch error, the backup process continues but the timestamp is not actually adjusted. The error repeats each time the structure with the timestamp mismatch is accessed during the backup process.

The timestamp of the structure is not adjusted to prevent any recovery operation that uses the backup dump from corrupting the database. Any timestamps that are overridden are not stored in the control file that is written to the backup media as part of the backup process.

Effect of DASDL *LOCKEDFILE* Option

If the DASDL *LOCKEDFILE* option is set in the database description file, then the *LOCKEDFILE* file attribute is set for the tape reels or disk files generated during the database backup process.

For more information on the LOCKEDFILE option, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*. For more information on the LOCKEDFILE file attribute, refer to the *File Attributes Programming Reference Manual*.

Tasks Related to Creating and Managing Database Backups

Introduction

Besides creating the actual database backup, you might want to perform several associated tasks as described in the following text.

Verifying the Database Backup

Before starting a database recovery process or before reloading a database from a backup tape, you can check that the database backup is free from the following types of errors:

- Block checksum errors
- Block sequencing errors
- I/O errors

You can use the DMUTILITY *VERIFYDUMP* command to perform this task.

Copying the Database Backup

Generating a copy of your database backup can provide you with extra security for your database since disk files and tapes can be lost or damaged, and the information can be accidentally erased. Two DMUTILITY commands are available for copying backup dumps:

- COPYDUMP
- DUPLICATEDUMP

Both commands generate a copy of the database backup, but the DUPLICATEDUMP command also enforces a reelforreel or fileforfile similarity between the original and the copy of the database backup.

Advantages of the DUPLICATEDUMP Command

- If a problem occurs with a particular tape reel or disk file during a reload or database recovery operation, you have a substitute reel or file and do not need to restart the complete reload or recovery operation.
- You can restart a DUPLICATEDUMP request, but you cannot restart a COPYDUMP request.

Refer to [Section 4, Using DMUTILITY](#), for information on restarting the DMUTILITY program.

Disadvantages of the DUPLICATEDUMP Command

- During the copy process if the destination tape reel or disk file cannot contain all the information stored on the source tape reel or disk file, the copy process terminates.
- You cannot copy from tape to disk or disk to tape using the DUPLICATEDUMP command.

Terminology Convention

To differentiate between the output generated by the COPYDUMP and the DUPLICATEDUMP commands, the following naming convention is used in this section:

- The terms *copying* and *copy* refer to the task performed by and the output generated by the COPYDUMP command.
- The terms *duplicating* and *duplicate* refer to the task performed by and the output generated by the DUPLICATEDUMP command.

Identifying the Information on a Dump Tape Reel or in a Dump Disk File

As the DMUTILITY program creates a database backup, it generates a directory of the information dumped.

Single Dump Tapes

A dump to a single dump tape can include several continuation reels. If continuation reels are necessary, the directory of each successive tape reel or disk file created in the database backup is cumulative.

You can choose to use one tape drive for the database dump. Or, you can use the TAPES option in the tape specification to split the database backup into two or more logical parts and dump the logical parts to different tape drives in parallel. In both instances, the last tape reel created contains the complete backup dump directory.

Be aware that the last tape reel created might not be the reel with the highest cycle and version number. For instance, the tape labeled cycle 1, version 2 might be created after the tape labeled cycle 3, version 2 and would therefore contain the more complete directory.

Multidump Tapes

The following are facts about multidump tapes:

- The WORKERS or TAPES options cannot be used when backing up a dump to a multidump tape. As a result, the CYCLE value is always 1.
- Multidump tapes do not have continuation reels. As a result, the value of VERSION is always 1. If the total dump does not fit on a reel, the dump contents that has been written on the reel is deleted and the operator is requested to initiate a new DUMP command rather than an APPEND command.
- Each multidump DUMP or APPEND statement creates a uniquely named dump file.

Notes:

- *When you are using the multidump format, be careful not to create a single point-of-failure scenario. For example, this situation could occur if a single tape were used to hold all the backups for a single database.*
- *Whenever you use an existing multidump tape on a system, the fast access directory for that tape must be present on the system. You can copy the directory from another system or create the directory for the tape by using the `TAPESET DIRECTORY CREATE` command.*
- *Do not append a dump to a multidump tape that was created on another system.*

Finding Information About Dumps

Knowing the dump information on a particular tape reel or disk file of a database backup can be useful when you are performing a database recovery operation.

Use the `TAPESET DIRECTORY` command to either list the contents or re-create the fast access directory file for a multidump tape. This command provides you with general information about the contents of the tape.

Use the `TAPEDIRECTORY` command to list specific information about one database backup only. If you want to track the information contained in *all* the database backups for a particular database, refer to the following discussion on cataloging the information in database backups.

Cataloging the Information in Database Backups

You can choose to back up all or part of a database. If you choose to use partial database backups and you must perform a database recovery process, ensure that you use the most current and the most accessible information.

Also, if you must perform a rollback or recovery process to undo incorrect transactions, you might not want to use the most current database backup. In such cases, you need a method of tracking the information in the database backups. Use the `DMDUMPDIR` program and the `BUILDDUMPDIRECTORY` command to track the information.

Use the `DMDUMPDIR` program to set up and maintain an automatic database backup catalog called a *dump tape directory*. The dump tape directory is a twolevel file directory that stores information about all the tape and disk dumps for a particular database.

The `BUILDDUMPDIRECTORY` command enables you to reconstruct catalog information for inclusion in a dump tape directory. You can use this command to reconstruct catalog information that has been lost or destroyed, or to generate catalog information for old database backups for which catalog information has not previously been created.

DUMP and APPEND Commands (DMUTILITY)

Introduction

Use the DUMP command to create a database backup. Use the APPEND command to create a database backup that is appended to an existing multidump tape. Using the DUMP or APPEND commands you can choose to

- Create the backup.
- Allow users or prevent users from accessing the database during the backup process.
- Identify the pieces of the database that you want to back up.
- Request the use of compressed tapes.
- Identify such items as tape serial numbers, tape density, and number of workers.

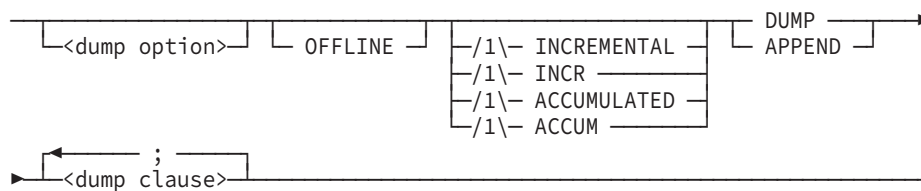
Notes:

- *APPEND enables you to add a database dump only to a multidump tape.*
- *When you use the <dump disk specification> clause for a permanent directory database, the dump is placed under the usercode from which the DMUTILITY task was initiated. The specification cannot be a permanent directory. Refer to [Section 16, Using Permanent Directory Databases](#), for examples.*

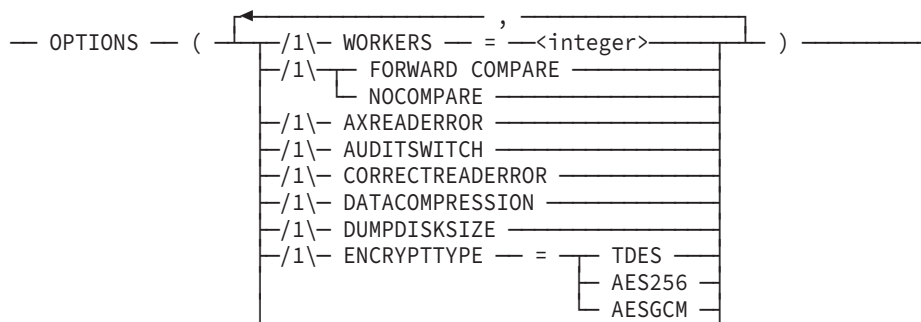
Syntax

The following diagrams illustrate the syntax for the DUMP command. Explanations of these syntax elements follow the diagrams. The syntax elements are explained in the order in which they appear in the syntax diagrams.

DUMP and APPEND Commands



<dump option>



Backing Up a Database

```

    /1\ NOENCRYPT
    /1\ OVERWRITEDISK
    /1\ WAITTIME
  
```

<dump clause>

```

    <dump list> BY FAMILYINDEX TO
    <dump tape specification>
    <multidump tape specification>
    <dump disk specification>
  
```

<dump list>

```

    <database file name> , <dump selector>
    = <dump selector>
    ( EXCLUDE <exclude list> ) <portion selector>
  
```

<dump selector>

```

    <portion selector>
    ( DUMPENCRYPT = FALSE <portion selector> )
    TRUE <portion selector>
  
```

<exclude list>

```

    <database name> / <data set name> / =
  
```



<portion selector>

```

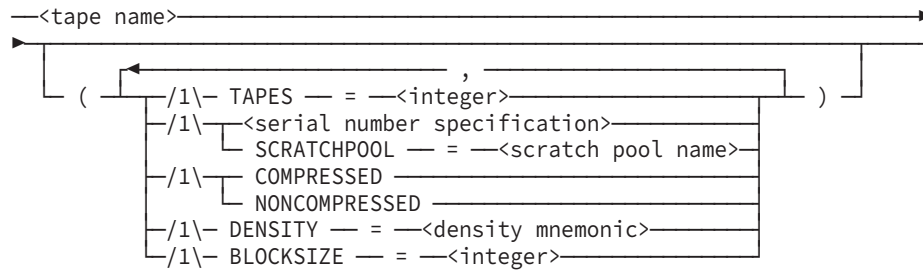
    AND
    &
    ( /1\ FAMILYINDEX = <range>
      /1\ ROW = <range>
      /1\ PACKNAME = <family name> )
  
```

<range>

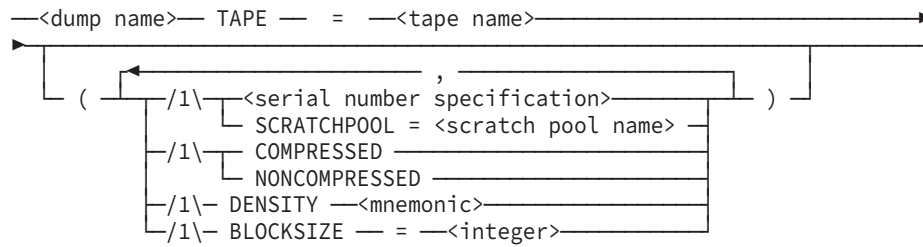
```

    <integer> , <integer>
    - <integer>
  
```

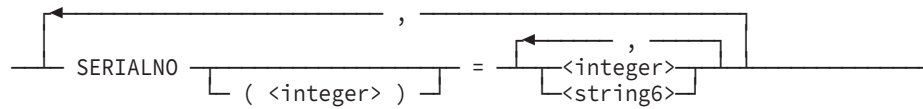
<dump tape specification>



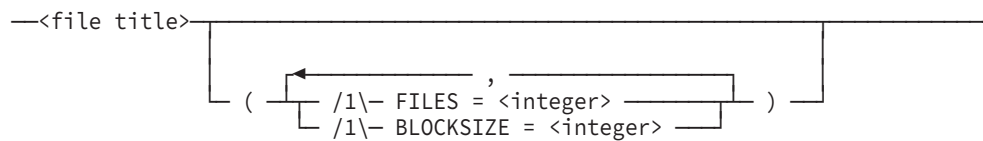
<multidump tape specification>



<serial number specification>



<dump disk specification>



Dump Option

Purpose

Use the dump option to designate the following options:

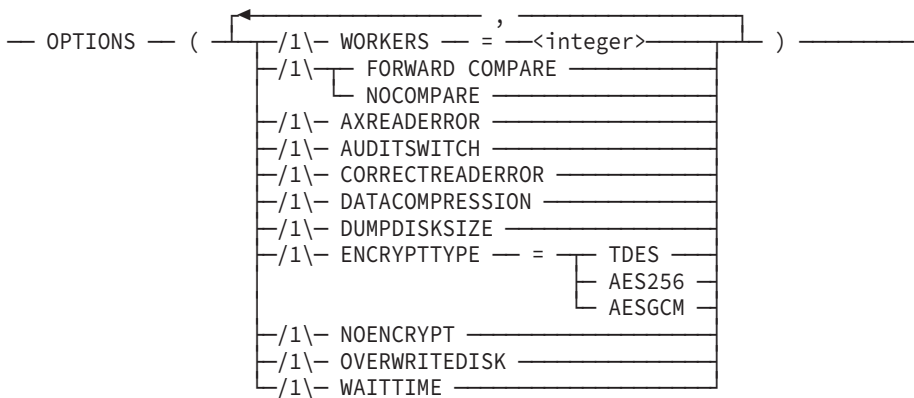
- WORKERS
- FORWARD COMPARE
- NOCOMPARE
- AXREADERROR
- AUDITSWITCH
- CORRECTREADERROR
- DATACOMPRESSION
- DUMPDISKSIZE

- NOENCRYPT
- OVERWRITEDISK
- WAITTIME

The WORKERS, FORWARD COMPARE, NOCOMPARE, and AXREADERROR options are valid only with tape output. If you use any of these options with disk output, a warning is displayed but no syntax error occurs.

Syntax

<dump option>



WORKERS Option

Single Dump Tapes

Use the WORKERS option to control the maximum number of tape drives to which you can dump files in parallel. You can designate a maximum of 50 tape drives or workers.

Increasing the number of workers can decrease the total time required to complete the database backup process. Limiting the number of workers can free up tape drives for other purposes.

The number you assign to the WORKERS option does not determine the number of tapes or disk files created. To designate the number of tapes, use the TAPES option in the dump tape specification. To designate the number of disk files, use the FILES option in the dump disk specification. You can designate a maximum of 50 tapes or files.

If you do not use the WORKERS option, then the number of parallel workers is the same as the number of tapes or files you designate in the TAPES or the FILES option.

Multidump Tapes

WORKERS and TAPES are not valid options when you are dumping a database to multidump tapes.

Disk Dumps

If you choose to do a tape or disk dump in the same DUMP command, then you are limited to processing one tape or disk dump list at a time. This limitation applies to any DUMP command that uses disk as the backup medium.

If you include more than one dump clause in your DUMP command but the output medium for all of the dump clauses is tape, the dump clause requests can process in parallel. The number of parallel workers in this instance is the smallest of the following values:

- The sum of all the values assigned to the TAPES option in each dump clause.
- The default value (20) for the WORKERS option.
- The number of workers you designate in the WORKERS option.

If you use one dump clause only, the number of parallel workers is set to the smallest of the following values:

- The number of tapes designated in the TAPES option
- The number of files designated in the FILES option
- The number of workers designated in the WORKERS option
- The default value (20) for the WORKERS option

For disk dumps, the WORKERS option is ignored and a warning message displayed. The number of parallel workers used is the number of disk files you designate in the FILES option in the dump disk specification.

FORWARD COMPARE Option

Use the FORWARD COMPARE option when the DMUTILITY program performs an implicit VERIFY operation to verify newly created dump tapes for tape drives that can only perform reads in the forward direction.

For tape drives that can only perform reads in the forward direction, the default comparison technique is forwardcompare. For tape drives that support the readreverse capability, the default comparison technique is readreverse.

FORWARDCOMPARE is the default option for multidump tape devices as they do not have read reverse capability. Multidump tapes are generally faster than single dump tapes when the tape is being repositioned for comparison, because multidump tapes use a fast access capability.

Setting the FORWARD COMPARE option has no effect on disk dumps.

NOCOMPARE Option

Use the NOCOMPARE option to skip automatic checking of a newly created tape. The primary reason for having this option is to delay verification of the tape or disk until a more convenient time or to allow verification of the tape on a different machine to make better use of resources. Using the NOCOMPARE option is not recommended without a subsequent DMUTILITY run to verify the newly created tape.

The NOCOMPARE and FORWARD COMPARE options are mutually exclusive. This means you cannot specify these two options at the same time. If you specify both options at the same time, a syntax error occurs. If you do not explicitly designate the NOCOMPARE option, the DMUTILITY program automatically starts the tape verification after each dump operation.

If you designate the NOCOMPARE option, the output listings from the DMUTILITY DUMP and TAPEDIRECTORY commands display the following:

```
NOCOMPARE OPTION HAS BEEN SET TO TRUE
```

Use of the NOCOMPARE option is recorded on the dump tape. This information is never changed on the tape, even if a successful VERIFYDUMP operation is subsequently completed, because data cannot be overwritten on tape devices. When performing a COPYDUMP operation of a dump tape created using the NOCOMPARE option, the newly created copy also indicates that the dump was created using the NOCOMPARE option.

The NOCOMPARE option enables you to make better use of available resources by postponing the verification operation until a later time or until you can perform the verification on another system. Because tapes created by using the NOCOMPARE option do not indicate if a VERIFYDUMP operation was subsequently performed, there is no guarantee of the integrity of the dump tapes created using the NOCOMPARE option. Therefore, it is recommended that you use the DMUTILITY program to explicitly verify all database dumps created using the NOCOMPARE option.

Note: *The NOCOMPARE option suppresses automatic verification of dump tapes and should only be used with a VERIFYDUMP operation.*

AUDITSWITCH Option

Use the AUDITSWITCH option to close the current audit before the dump is performed.

Note: *Only use this option for an OFFLINE dump. It is recommended to use this option when performing a dump for the first time after upgrading to a new release.*

AXREADERROR Option

Use the AXREADERROR option to enable you to take a specific action if a read operation error occurs during a DMUTILITY program dump. If the AXREADERROR option is specified and a read operation error occurs, the DMUTILITY program stops and displays a message asking you to enter one of the following commands:

- OK to continue
- SKIP ROW to skip the row
- TERMINATE to quit the DMUTILITY run

If the AXREADERROR option is not specified, the row on which an error occurred is automatically included in the dump and a warning message is included in the report.

CORRECTREADERERROR Option

Use the CORRECTREADERERROR option to enable DMUTILITY to reset the READERROR flag of a row if there is no checksum error and no I/O error. This option is mutually exclusive with the AXREADERROR option.

DATA_COMPRESSION Option

Use the DATA_COMPRESSION option to enable DMUTILITY to compress the data during the dump process.

Note: The DATA_COMPRESSION option is only supported for a full dump.

DUMPDISKSIZE Option

Use the DUMPDISKSIZE option to generate a report on the estimated disk size that the dump to disk will use. Use this option only for a full dump to disk. When the DUMPDISK option is specified, the dump to disk will not take place.

ENCRYPTTYPE Option

Use the ENCRYPTTYPE option to override the encryption type defined in the DASDL definition of the database.

If the ENCRYPTTYPE option is not defined in the DASDL definition and is not specified by the DMUTILITY DUMP and APPEND statements, TDES is the default encryption algorithm used for any encryption directives in the DASDL definition. The AESGCM authenticated encryption algorithm ensures the confidentiality, integrity, and the authenticity of the encrypted data.

Refer to [Section 15, Using Database Tape Encryption](#), for specific examples and additional information about the ENCRYPTTYPE option.

NOENCRYPT Option

Use the NOENCRYPT option to override any encryption directives in the DASDL definition of the database.

Refer to [Section 15, Using Database Tape Encryption](#), for specific examples and additional information about the NOENCRYPT option.

OVERWRITEDISK Option

Use the OVERWRITEDISK option if you want to overwrite an existing disk dump with the same file name. The existing disk dump will be removed and replaced by the new disk dump.

WAITTIME Option

Use the WAITTIME option to set the amount of time in seconds for a DMUTILITY DUMP/APPEND task to wait for all update users to leave the mix before the task is started. If all update users have not left the mix before the designated amount of seconds, the DMUTILITY task terminates and its task value is set to 4.

OFFLINE Option

Purpose

Use the OFFLINE option to prevent database users from updating the database during the database backup process.

A database backup that is performed with the database unavailable for update purposes is called an *offline* dump. A database backup that is performed with the database available for update purposes is called an *online* dump. Inquiry-only users can access a database during both offline and online dumps.

You can use offline dumps to back up and recover both audited and unaudited databases. For unaudited databases, you cannot perform online dumps and therefore do not need to include the OFFLINE option in the DUMP command. This limitation exists because the online dump can be used for recovery purposes only if the appropriate audit information is available.

If you want to perform an offline dump for an audited database, you must include the OFFLINE option in the DUMP command. By default, audited databases are available for both inquiry and update purposes during the backup process.

Syntax

To use the OFFLINE option, precede the keyword DUMP with the keyword OFFLINE in the DUMP command.

Offline Dump Process

Performing an offline dump ensures that the database is in a consistent state throughout the backup process.

Before initiating the offline dump process, the DMUTILITY program performs the following steps:

1. Waits for all programs that have the database open for update to complete processing
2. Locks the control file
Access to the database is limited to inquiry use only.

Following the successful completion of the offline dump, the DMUTILITY program unlocks the control file.

Resolving Locked Control File Situations

The DMUTILITY program locks the control file during the offline dump process. If the DMUTILITY program is discontinued during the offline dump, or if the DMUTILITY program fails to unlock the control file, use the *CANCEL* command to unlock the control file.

Storing the Last Audit File with an Offline Dump

Keep a copy of the last audit file with the offline dump in case the database needs to be copied back to disk at a later time using the *COPY* command. When using the *COPY* command, ensure that the exact copy of the audit file that was in use just before the dump process is also restored to disk. Without the audit file, a subsequent recovery process that uses the dump might fail.

Online Dump Process

When performing an online dump, the DMUTILITY program bypasses the database guard file and opens the database for inquiry only. This mechanism prevents security errors when dump worker processes try to open the database for online dumps.

At the beginning of a recovery process that uses an online dump, the audit file in use at the time that the dump was created, is opened and positioned two control points before the time the dump started. In order to position itself correctly, the recovery process may search backwards prior to these two control points.

Additional steps can be added to a backup script to ensure that use of an online dump for a database recovery does not require an audit file that was created prior to the audit file in use at the time of the dump. The following is an example of a pseudo-coded script:

```
<db> mix#> SM AUDIT CLOSE
RUN SYSTEM/DMUTILITY("DB=<db name> QUIESCE")
RUN SYSTEM/DMUTILITY("DB=<db name> RESUME")
RUN SYSTEM/DMUTILITY("DB=<db name> DUMP = TO FRIDAYDUMP")

ENDCOMMENT
```

Performing Reorganizations and Online Dumps Simultaneously Is Not Supported

An online dump cannot be performed while a reorganization is in progress. Any attempt to do so results in a fatal error and the following message is displayed:

ONLINE DUMP IS ILLEGAL WHEN REORGANIZATION IS IN PROGRESS

INCREMENTAL Option

Purpose



Use the INCREMENTAL option to back up all data sets, sets, and subsets that have been modified since the last full, incremental, or accumulated dump.

The data that is included in the dump depends on the physical option of the data set in the DASDL source for the structure.

- If the DUMPSTAMP option for a structure is not enabled, the whole structure is placed in the dump.
- If the DUMPSTAMP option is set for a structure, only the blocks with a dumpstamp that is equal to or greater than the last dumpstamp stored in the control file during the previous dump are placed in the dump.

For example, if the dumpstamp stored in the control file by the previous dump is 150, the incremental dump will include all blocks with a dumpstamp equal to or greater than 150.

For more information on the DUMPSTAMP option, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*. A full database dump is always required before any incremental dump can start. In addition, the first dump taken after any of the following operations must be a full dump:

- Any REORGANIZATION run
- Initialization of the control file
- RECOVER UPDATE of the control file
- Any online garbage collection
- Initialization of a database structure

When incremental or accumulated dumps are used to back up to a single dump tape, an additional tape is needed to store the updated tape directory information. You need an additional tape because the tape directory cannot be overwritten after it has been written out to tape, and information pertaining to modified blocks is not available until the end of the incremental and accumulated dumps.

When you create accumulated or incremental backups on a multidump tape, DMUTILITY stores the final tape directory information as an additional file appended to the backup on the same tape, so no extra tape is required.

Syntax

To use the INCREMENTAL option, precede the keyword DUMP with the keyword INCREMENTAL or INCR in the DUMP command. To dump all database structures, you must specify the "DUMP =" syntax.

Examples

The following examples illustrate the use of the INCREMENTAL option.

Example 1

The following command initiates an incremental tape dump, which includes all data blocks that have been modified since the last full, incremental, or accumulated dump. All incremental dumps must include the syntax "DUMP =" so that all database files are in the incremental dump.

```
INCREMENTAL DUMP = TO INCTAPE
```

Example 2

The following command initiates an incremental disk dump, which includes all data blocks that have been modified since the last full, incremental, or accumulated dump. All incremental dumps must include the syntax "DUMP =" so that all database files are in the incremental dump.

```
INCREMENTAL DUMP = TO INCDUMP ON DATAPACK
```

ACCUMULATED Option

Purpose



Use the ACCUMULATED option to back up all data sets, sets, and subsets that have been modified since the last full dump was performed.

The data that is included in the dump depends on the physical option of the dataset in the DASDL source for the structure.

- If the DUMPSTAMP option for a structure is not enabled, the whole structure is placed in the dump.
- If the DUMPSTAMP option is set for a structure, only the blocks with a dumpstamp that is equal to or greater than the last dumpstamp stored in the control file during the previous dump are placed in the dump.

For example, if the dumpstamp stored in the control file by the previous dump is 150, the incremental dump will include all blocks with a dumpstamp equal to or greater than 150.

Backing Up a Database

For more information on the DUMPSTAMP option, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

A full database dump is always required before any accumulated dump can start. In addition, the first dump taken after any of the following operations must be a full dump:

- Any REORGANIZATION run
- Initialization of the control file
- RECOVER UPDATE of the control file
- Any online garbage collection
- Initialization of a database structure

When incremental or accumulated dumps are used to back up to a single dump tape, an additional tape is needed to store the updated tape directory information. You need an additional tape because the tape directory cannot be overwritten after it has been written out to tape, and information pertaining to modified blocks is not available until the end of the incremental and accumulated dumps.

When you create accumulated or incremental backups on a multidump tape, DMUTILITY stores the final tape directory information as an additional file appended to the backup on the same tape, so no extra tape is required.

Syntax

To use the ACCUMULATED option, precede the keyword DUMP with the keyword ACCUMULATED or ACCUM in the DUMP command. To dump all database structures, you must specify the "DUMP =" syntax.

Example 1

The following command initiates an accumulated tape dump, which includes all data blocks that have been modified since the last full dump. All accumulated dumps must include the syntax "DUMP =" so that all database files are in the accumulated dump.

```
ACCUMULATED DUMP = TO ACCUMTAPE
```

Example 2

The following command initiates an accumulated disk dump, which includes all data blocks that have been modified since the last full dump. All accumulated dumps must include the syntax "DUMP =" so that all database files are in the accumulated dump.

```
ACCUMULATED DUMP = TO ACCUMDUMP ON DATAPACK
```

Dump Clause

Purpose

The dump clause enables you to define

- The parts of the database you want to back up
- The backup media you want to use

Single Dump Tapes

By supplying multiple dump clauses separated by semicolons (;), you can also use the dump clause to

- Dump different parts of the database to
 - Tapes with different names
 - Disk files with different names
 - A combination of tape and disk destinations
- Make multiple copies of the information you want to back up

Supplying multiple dump clauses also affects the number of dump workers assigned to the dump operation. For more information on the assignment of dump workers, refer to “WORKERS Option” under “DUMP Option” earlier in this section.

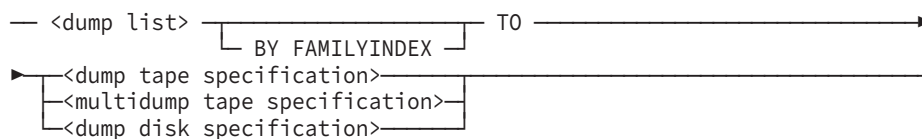
Multidump Tapes

The following usage restrictions apply to multidump tapes:

- Using more than one dump clause is not valid when you are dumping to a multidump tape.
- Duplicate dump names are not permitted when you use APPEND to add dumps to a multidump backup tape.
- If you must restart an APPEND action, use the negative task value from the previous attempt. To avoid leaving an incomplete dump on the tape, you must perform a restart action.
- Whenever you use an existing multidump tape on a system, the fast access directory for that tape must be present on the system. You can copy the directory from another system or create the directory for the tape by using the TAPESET DIRECTORY CREATE command. If you perform an APPEND action and the fast access directory is not present, DMUTILITY automatically creates the directory.
- Do not append a dump to a multidump tape that was created on another system.

Syntax

<dump clause>



Examples

The following examples illustrate the use of multiple dump clauses.

Example 1

The following DUMP command contains three dump clauses and produces three dumps:

- A dump containing database restart information only
- A dump containing client information only
- A dump containing vendor information only

```
DUMP      TESTDB/RDS/= TO RESTARTDUMP ;
          TESTDB/CLIENTS/= TO CLIENTDUMP ;
          TESTDB/VENDORS/= TO VENDORSDUMP
```

Example 2

The following DUMP command contains two dump clauses and backs up the complete database twice, once to a disk file called TESTDB/DUMP/1 and once to a disk file called TESTDB/DUMP/2. In both cases the backup disk files are created on a pack called TESTPK.

```
DUMP      = TO TESTDB/DUMP/1 ON TESTPK ;
          = TO TESTDB/DUMP/2 ON TESTPK
```

Dump List Clause

Purpose

Use the dump list clause to designate the parts of the database you want to back up. The dump list consists of one or more database file names and one or more dump selector clauses. The dump selector clauses refine the information provided by the database file names.

When you perform a partial database dump, dump all the related structures together. Otherwise, this dump is rendered useless for a rebuild recovery of these structures.

Related structures include all the following combinations:

- A data set and its sets and subsets
- A data set and its embedded structures
- Data sets and their linked data sets

Limitations

Be aware of the following limitations:

- When the DMUTILITY program dumps all, or a portion, of a preallocated direct data set, the DATAEOF value is recorded in the tape or file directory. Only the portion of the direct data set prior to the DATAEOF block is actually written to the dump media. If the DATAEOF value is unavailable (for example, because row 0 (zero) is locked out), all data set rows, including the preallocated ones, are dumped.

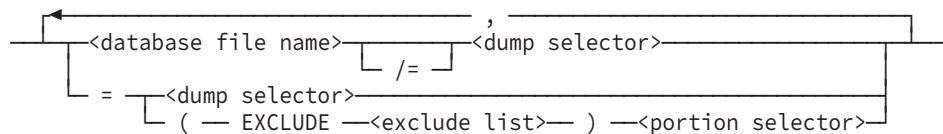
Refer to “DMUTILITY INITIALIZE Statement” in [Section 5, Initializing and Maintaining](#), for more information.

- You cannot make a backup of only the database control file.
- COPYDUMP and DUPLICATEDUMP do not support incremental and accumulated dumps.

Note: When you use the INCREMENTAL and ACCUMULATED dump options, you must specify “DUMP =” to perform a full dump.

Syntax

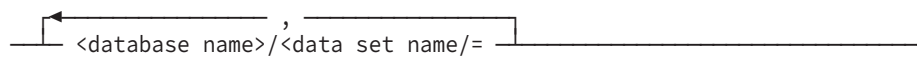
<dump list>



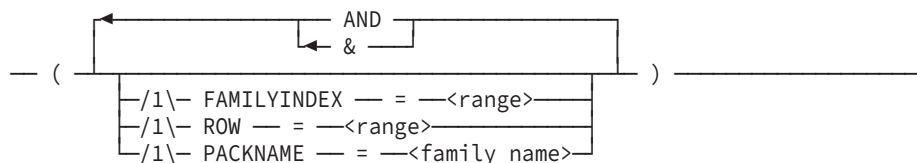
<dump selector>



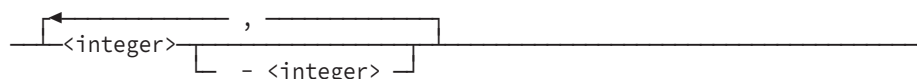
<exclude list>



<portion selector>



<range>



Database File Name Clause

Use the database file name clause to identify the title of a database file.

To dump a family of files, use /= (a slash followed by an equal sign). For example, to dump all files in the ORGDB database with the first nodes of ORGDB/PROJECT, supply the following file name:

```
ORGDB/PROJECT/=
```

To dump all the files in the database, use the = (an equal sign) branch of the <dump list> clause.

Dump Selector Clause

Use the dump selector clause to refine the information provided by the database file names in any of the following ways:

- Use the portion selector clause if the DUMPENCRYPT specification is not used for the specified file or files.
- Use the DUMPENCRYPT = FALSE clause if no encryption is necessary during this dump. As a result, the DUMPENCRYPT specification in DASDL for the specified file or files is ignored during the dump and the portion selector clause is used.
- Use the DUMPENCRYPT = TRUE clause if encryption must be done during this dump. As a result, the DUMPENCRYPT specification in DASDL for the specified file or files is ignored during the dump.

Refer to [Section 15, Using Database Tape Encryption](#), for specific examples and additional information on the DUMPENCRYPT option.

Exclude List Clause

Use the exclude list clause to select one or more database structures you want to leave out of a database dump. The exclude list can consist of one or more database files. This clause provides more flexibility to the DUMP command, and it is especially useful when you want to exclude a small percentage of structures from a DUMP operation.

When you want to perform a full database dump, but you want to exclude one or more database files, you must exclude all structures related to the file or files. Related structures include data sets, sets, subsets, and embedded structures.

Notes:

- *DMUTILITY tape encryption is not available when data sets are included in the exclude list clause of a dump list. For example, a full database dump performed without the data sets CLIENTS and VENDORS would use the following command:*

```
(EXCLUDE TESTDB/CLIENTS/= , TESTDB/VENDORS/=)
```
- *If the database is a permanent directory database, do not include the data path. Only include the database name and data set name.*

Example 1

The following command dumps all files that reside on family index 1 to 6 and pack family DBDATA, except all structures associated with the EMPLOYEE and CUSTOMER data sets:

```
DUMP = (EXCLUDE ALLDB/EMPLOYEE/=, ALLDB/CUSTOMER/=)
(PACKNAME=DBDATA & FAMILYINDEX = 1 - 6) TO TAPEX
```

Example 2

The following command dumps all rows that reside on pack family DBDATA, except the files that are part of the PAYROLL1, PAYROLL2, and PAYROLL3 disjoint data sets and all their related sets, subsets, and embedded structures.

```
DUMP = (EXCLUDE ALLDB/PAYROLL1/=, ALLDB/PAYROLL2/=,
ALLDB/PAYROLL3/=)
(PACKNAME=DBDATA) TO TAPEX
```

Note: If the database is a permanent directory database, do not include the data path. Only include the database name and data set name.

Portion Selector Clause

Use the portion selector clause to refine the information provided by the database file names in any of the following ways:

- Use the FAMILYINDEX construct to limit the backed up data to the data rows that reside on a particular family index.

For example, the following dump list backs up all the information on family index 1 through 4 for any files that have the first nodes ORGDB/PROJECT:

```
ORGDB/PROJECT/= (FAMILYINDEX = 1 - 4)
```

If none of the designated family indexes exist, or if no rows of the specified file exist on the designated family indexes, the following warning message is displayed:

```
NO ROWS MATCH THE REQUEST
```

- Use the ROW construct to limit the backed up data to the information in the designated rows.

For example, the following dump list backs up the information in rows 17 through 33 in the file ORGDB/PROJECT/DATA:

```
ORGDB/PROJECT/DATA (ROW = 17 - 33)
```

If none of the designated rows exist, or if no data from the specified file exists on the designated rows, the following warning message is displayed:

```
NO ROWS MATCH THE REQUEST
```

- Use the PACKNAME construct to limit the backed up data to the data that resides on a particular pack.

Backing Up a Database

For example, the following dump list backs up all the information that resides on the pack called DATAPACK:

```
= ( PACKNAME = DATAPACK
```

You can use the FAMILYINDEX, ROW, and PACKNAME constructs together in a dump selector clause. Frequently the constructs PACKNAME and FAMILYINDEX are used together as shown in the following example. In this example, only the data that resides on the pack MYDATAPACK on family index 1 through 5 is backed up.

```
ORGDB/PROJECT/DATA ( FAMILYINDEX = 1 - 5 AND  
PACKNAME = MYDATAPACK )
```

Creating Compound Dump Lists

To designate the same dump selector clause for several files, use a parenthetical statement to identify the database files and follow the parenthetical statement with a dump selector clause.

For example, the following statement includes in the database backup all data in rows 25 through 35 in the files ORGDB/PROJECT/DATA and ORGDB/INTERIM-MANAGER/DATA:

```
( ORGDB/PROJECT/DATA , ORGDB/INTERIM-MANAGER/DATA )  
( ROWS = 25 - 35 )
```

Database files in the dump list that already have a dump selector clause have the outer selection constraints related to the inner constraints through the Boolean construct OR.

For example, both of the following statements back up any data that resides on the pack called MYDATA or that resides on family index 25 through 35:

```
( ORGDB/= ( PACKNAME = MYDATA ) ) FAMILYINDEX = 25 - 35  
  
( ORGDB/= ( FAMILYINDEX = 25 - 35 ) ) ( PACKNAME = MYDATA )
```

BY FAMILYINDEX Option

Purpose

Use the BY FAMILYINDEX option to modify the order in which database information on disk is backed up. The BY FAMILYINDEX option causes information to be backed up by structure within a disk family index within a disk family. If the dump is likely to be used for reconstruction purposes, having all the structure-related information from a single disk family or family index together can make the reconstruction process faster.

A disk family can be made up of one or more disk family indexes. Each disk family index relates to a physical disk unit.

Syntax

To use the BY FAMILYINDEX option, precede the keyword TO with the keywords BY FAMILYINDEX in the DUMP command. For example,

```
DUMP = BY FAMILYINDEX TO TAPEX
```

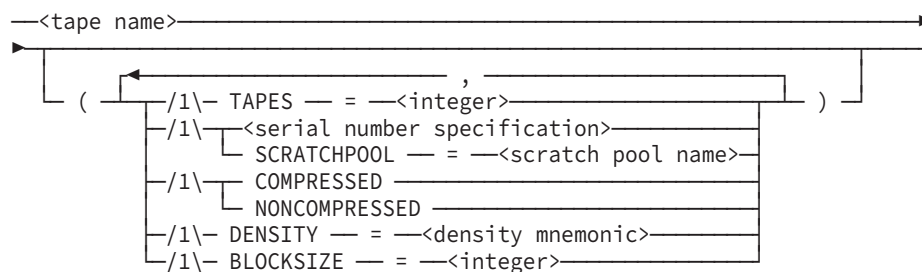
Dump Tape Specification

Purpose

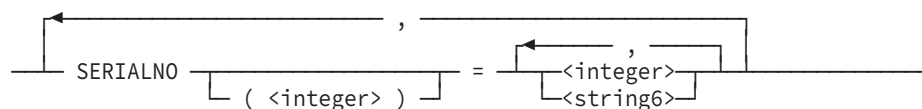
Use the dump tape specification to identify your output tape requirements for the backup process.

Syntax

<dump tape specification>



<serial number specification>



TAPES Clause

Single Dump Tapes

Use the TAPES clause to designate the maximum number of families (cycles) of tapes to which you want the database dumped. The database files in the dump list are divided into the maximum specified number of equal parts, and each part is dumped to a unique family of tapes. The minimum number of tapes you can assign is 1 and the maximum is 50. The default value is 1.

Specifying a large number of tapes for a small database might produce fewer tapes than you expect because there is not enough data to fill the designated number of tapes.

If you do not provide a value for the WORKERS option, then the TAPES clause also sets the number of parallel workers. If you request more than 50 tapes, the number of parallel workers is limited to 20 since this is the maximum value allowed for the WORKERS option.

After a successful tape dump, the printout provided by the DMUTILITY program displays the last cycle, version, and serial number of the tape containing the latest directory information.

Multidump Tapes

WORKERS and TAPES are not valid options when you are creating dumps to multidump tapes.

Tape Labels

Unique cycle and version numbers are assigned to the tape reels. The cycle number indicates the tape family; the version number indicates the position of the reel within the family. The version number of multidump tapes is always 1.

MTVERSION – revisit text and examples in this subsection when continuation reels for multidump tapes are reinstated.

For example, if you specify TAPES = 2, the tapes in the first family are labeled CYCLE=1 VERSION=1, CYCLE=1 VERSION=2, and so on. Similarly, the tape reels in the second family are labeled CYCLE=2 VERSION=1, CYCLE=2 VERSION=2, and so on. The result of a PER MT command on an ODT provides a display similar to the following:

```
----- MT STATUS -----
20*P[050543] 6250 #1 1:1 <05/25/2004> TESTDUMP
21*P[050544] 6250 #1 2:1 <05/25/2004> TESTDUMP
22*P[051754] 6250 #1 1:2 <05/25/2004> TESTDUMP
23*P[051755] 6250 #1 2:2 <05/25/2004> TESTDUMP
```

For multidump tapes, the result of a PER MT command on an ODT provides a display similar to the following:

```
----- MT STATUS -----
153*H[021404] 36TRK #1/1 1:1 <01/22/2004> DBDUMP01/TAPESET
250*H[021403] 36TRK C#1/1 1:1 <01/28/2004> TESTTAPE/TAPESET
723*H[021401] ST9840 C#1/1 1:1 <01/27/2004> MDRDBSTRC/TAPESET
724*H [021401] ST9840 C#1/1 1:1 <01/06/2004> ACU2TAPE/TAPESET
```

In the displays, the cycle and version numbers are provided in the following format:

```
<cycle number>:<version number>
```

Tape Security

By default, DMUTILITY dump tapes are created with the SECURITY=PRIVATE clause. This clause indicates that the dump tapes can be read only by the usercode under which the tapes were created or by a user under a privileged usercode.

COMPRESSED and NONCOMPRESSED Options

By default, if you mount

- An uncompressed tape, no data compression occurs
- A compressed tape, data compression occurs

To ensure that data compression either does or does not occur, include the COMPRESSED or the NONCOMPRESSED option in your DUMP command.

Density Specification

By including a tape density in the dump tape specification, you ensure that all dump tapes are written at that density. Refer to the explanation of the DENSITY file attribute in the *File Attributes Programming Reference Manual* for a complete list of available tape densities.

Data is written in different data block sizes, depending on the density that you designate. By default, 10922-word data blocks are written, and the wait time is 165 seconds. To improve tape use, include a density or block size specification in your DUMP command. The following table identifies tape densities and block sizes that you can designate.

Density (BPI)	Data Block Size (Words)	Wait Time
11000	10922	165 seconds

The wait time is the amount of time in which a tape drive must respond to a command. If the tape drive does not respond within the specified time, a timeout error occurs and is reported.

BLOCKSIZE Clause

Use the BLOCKSIZE clause to set the BLOCKSIZE value for the destination dump tape. The DMUTILITY program accepts a block size from 900 to 65535 words, however, the universal default block size 10922.

The assignment of the block size is based on the following criteria:

Is Block Size Specified?	Is Density Specified?	Block Size Calculated
Yes	Yes	Use the specified block size. If the block size specified is more than 10922 words, a warning message appears.
Yes	No	Use the specified block size. If the block size specified is more than 10922 words, a warning message appears.
No	Yes	Use the default block size and the specified density.
No	No	Use the default block size, 10922 words.

Serial Number Specification

Use the serial number specification to identify the specific tape or set of tapes you want to use. Serial numbers can be any 1 to 6digit integer or character string. The SERIALNO (<integer>) construct enables you to designate the serial numbers for a particular cycle of tapes. The integer designates the tape cycle number. This form of the serial number specification is valid only when the number of cycles is greater than one. Up to 1000 serial numbers can be recorded in a dump tape directory.

The following serial number specification associates the serial numbers 346783, 346784, and 346785 with two cycles of tapes using the SERIALNO(<integer>) construct:

```
SERIALNO (1) = 346783, 346784, 346785
SERIALNO (2) = 346783, 346784, 346785
```

The SERIALNO (<integer>) form of the serial number specification is not valid for multidump tapes.

You cannot designate a scratch pool name and serial numbers for the same tape.

If you are not using the SCRATCHPOOL option, and if you are using a cartridge tape library (CTL), tapes can have the same name but must have unique serial numbers. When retrieving tapes, the CTL needs this serial number to mount the correct tape on a cartridge library unit (CLU) device.

If you do not provide the serial numbers, the CTL might accept an incorrect tape with the same name as the correct one, or suspend the task until an operator response is received. After a successful tape dump, the printout provided by the DMUTILITY program displays the last cycle, version, and serial number of the tape containing the latest directory information.

When you use the UNSTACK function of the MCP TapeStack utility on a stacked dump tape, ensure that the serial number of the unstacked tape is the same as the serial number of the tape when it was first stacked. To ensure that the two numbers match, perform the following steps:

1. Use the DIR system command to determine the serial number of the stacked tape.
2. To assign the appropriate serial number to the unstacked tape, do one of the following:
 - a. Use the SN system command.
 - b. Use the UNSTACK AS syntax of the MCP TapeStack utility.

Refer to the *System Software Utilities Manual* for more information on tape stacking and unstacking.

SCRATCHPOOL Option

Use the SCRATCHPOOL option to designate the scratch pool from which you want to retrieve a tape. The scratch pool name construct is a 17-character identifier.

You cannot designate a scratch pool name and serial numbers for the same tape.

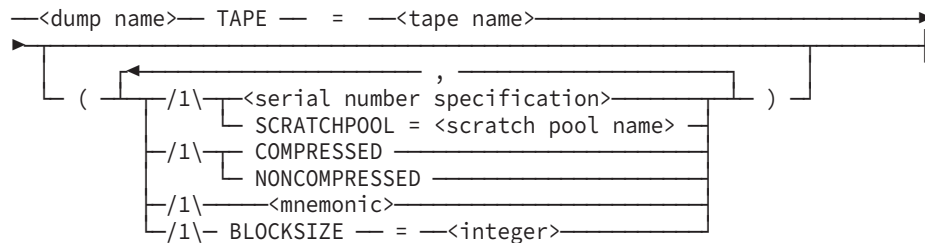
Multidump Tape Specification

Purpose

Use the multidump tape specification to identify your output tape requirements for the backup process when you want to store more than one backup dump on the same tape.

Syntax

<multidump tape specification>



Density Specification

When you use a density specification for a multidump tape, it must correspond to a tape drive that has fast access capability.

Multidump Serial Number Specification

The SERIALNO (<integer>) form of the serial number specification is not valid for multidump tapes.

Dump Disk Specification

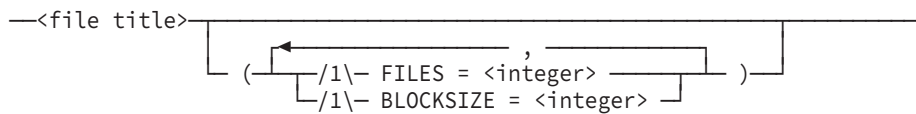
Purpose

If you are dumping to disk, use the dump disk specification to provide

- The base name for the output disk files
- The number of backup disk files you want to create
- The BLOCKSIZE attribute for dump files

Syntax

<dump disk specification>



FILES Clause

Use the FILES clause to identify the number of backup disk files that should be created for the dump. You can request that the dump be divided into at most 50 files. By default, only one file is created for a dump.

Examples

The naming convention for the output disk files is illustrated in the following examples.

Example 1

Assume you designate the following items:

- ORGDB/DUMP ON ISYS as the file title
- FILES = 1 as the number of files you want to create

In this instance, one disk file called ORGDB/DUMP on ISYS is created.

Example 2

Assume you designate the following items:

- ORGDB/DUMP ON BACKPACK as the file title
- FILES = 4 as the number of files you want to create

In this instance, the following five dump disk files are created. The file ORGDB/DUMP contains control information only.

- ORGDB/DUMP ON BACKPACK
- ORGDB/DUMP/01 ON BACKPACK
- ORGDB/DUMP/02 ON BACKPACK
- ORGDB/DUMP/03 ON BACKPACK
- ORGDB/DUMP/04 ON BACKPACK

BLOCKSIZE Clause

Use the BLOCKSIZE clause to set the BLOCKSIZE attribute for the dump files. This value must be a multiple of 60 words, and range from 900 to 65,520 words.

If the BLOCKSIZE option is not specified, the default disk dump block size is 19,200 words.

If you specify an invalid BLOCKSIZE value, the DMUTILITY program terminates with the following error message:

```
BLOCKSIZE MUST BE A MULTIPLE OF 60 WORDS AND RANGE FROM  
900 TO 65520 WORDS
```

Disk Usage

With a default disk dump block size of 19,200 words and the option to specify a BLOCKSIZE value up to 65,520 words, it is possible that increased disk space is required. The disk dump function reads each area from the source file and copies each area to one or more fixed-length data blocks. Consequently, the last block that corresponds to each area of the source file might contain some unfilled space. Depending on the area size of the source file, this unfilled space might increase as the BLOCKSIZE value increases.

When running disk dumps on databases where the average area size of each file is less than 900 words, it is recommended that you set the BLOCKSIZE value to 900 words to minimize the increased disk usage. For databases with files of varying sizes, expanding the BLOCKSIZE value improves the performance of the disk dump. However, some testing might be required to assess the impact of the increased disk usage.

Examples

The specification of the blocking factor for dump files is illustrated in the following examples.

Example 1

Assume you designate the following items:

- ORGDB/DUMP ON DATAPACK as the file title
- BLOCKSIZE = 3600 as the blocking factor you want to use

In this instance, one disk file called ORGDB/DUMP ON DATAPACK is created with the BLOCKSIZE value of 3600 words.

Example 2

Assume you designate the following items:

- ORGDB/DUMP ON BACKUP as the file title
- FILES = 3 as the number of files you want to create
- BLOCKSIZE = 6000 as the blocking factor you want to use

In this instance, the following dump files are created. Except for the first file, ORGDB/DUMP ON BACKPACK, all files are created with the BLOCKSIZE value of 6000 words.

- ORGDB/DUMP ON BACKPACK
- ORGDB/DUMP/01 ON BACKPACK
- ORGDB/DUMP/02 ON BACKPACK
- ORGDB/DUMP/03 ON BACKPACK

File Title Clause

Use the file title clause to provide the base name for the backup disk files. The backup disk files are always created under the same usercode as the usercode from which you initiate the DMUTILITY run.

Naming Requirements for Dump Disk Files

Adhere to the following rules for naming dump disk files:

- Do not supply a usercode.
If you supply a usercode, a syntax error occurs.
- Do supply a family name.
If you do not supply a family name, the DMUTILITY program assumes you want to back up the database to tape.

Naming Recommendations for Dump Disk Files

Avoid using any file name that might be the name of an existing database file. For example, avoid using any of the following file name formats:

- <database name>/< database name>/DATA
- <database name>/<data set name>/DATA
- <database name>/<data set name>/<set name>
- <database name>/CONTROL

If you do not supply a unique name, the following warning displays:

```
DISPLAY: **WARNING: **<pack name>** SPECIFIED DISKSTREAM
DUMP ALREADY EXISTS
          <dump file name>
ACCEPT: DISKSTREAM: AX '<file name>' OR 'QUIT DMUTILITY'
```

If, in response to this warning, you supply the name of a file that also exists, the following warning displays:

```
DISPLAY: DISKSTREAM: <dump file name> IS RESIDENT
```

DUMP Command Examples Where the Backup Medium Is Single Dump Tape

The following examples illustrate DUMP command syntax when the output medium is tape.

Example 1

The following command dumps all rows of the database to tape DBDUMP063094:

```
DUMP = TO DBDUMP063094
```

Example 2

The following command dumps all rows of the database that reside on family index 3 to tape TAPEX:

```
DUMP = (FAMILYINDEX = 3) TO TAPEX
```

Example 3

The following command divides the database into three equal parts and dumps each part to a separate tape. Three tapes are dumped in parallel if three tape drives are available. The first tape is labeled CYCLE=1, VERSION=1; the second tape is labeled CYCLE=2, VERSION=1; and the third tape is labeled CYCLE=3, VERSION=1. If an overflow tape is required, the version is incremented; that is, if the first tape overflows, it overflows to a tape labeled CYCLE=1, VERSION=2.

```
DUMP = TO TAPEX (TAPES = 3)
```

Example 4

The following command divides the database into three equal parts and dumps each part to a separate tape. Two tapes are dumped in parallel. The first tape is labeled CYCLE=1, VERSION=1; the second tape is labeled CYCLE=2, VERSION=1; and the third tape is labeled CYCLE=3, VERSION=1. If an overflow tape is required, the version is incremented; that is, if the first tape overflows, it overflows to a tape labeled CYCLE=1, VERSION=2.

```
OPTIONS (WORKERS = 2) DUMP = TO TAPEX (TAPES = 3)
```

Example 5

The following command dumps the entire database to tapes with the specified serial numbers. The first tape is labeled CYCLE=1, VERSION=1, SERIALNO=231; the second tape is labeled CYCLE=1, VERSION=2, SERIALNO=232; and the third tape is labeled CYCLE=1, VERSION=3, SERIALNO=233.

```
DUMP = TO TAPEX (SERIALNO = 231,232,233)
```

Example 6

The following command divides the database into two equal parts and dumps each part to a separate tape. Two tapes are dumped in parallel. The first tape is labeled CYCLE=1, VERSION=1, and SERIALNO=240. The second tape is labeled CYCLE=2, VERSION=1, and SERIALNO=250. If the first tape overflows, it overflows to a tape labeled CYCLE=1, VERSION=2, and SERIALNO=241. Similarly, if the second tape overflows, it overflows to a tape labeled CYCLE=2, VERSION=2, and SERIALNO=251.

```
DUMP = TO TAPEX (TAPES = 2, SERIALNO(1) = 240,241,  
SERIALNO(2) = 250,251)
```

Example 7

The following command explicitly divides the database by structure and dumps all rows of each file to the specified tape:

```
DUMP DB/A/DATA TO TAPE1; DB/B/DATA TO  
TAPE2; DB/C/DATA TO TAPE3
```

Example 8

The following command dumps each family index to a separate set of tapes. The rows on each family index are partitioned into three parts and dumped to different cycles of the same tape name. Nine tapes could be dumped simultaneously, but the WORKERS=3 clause limits the DMUTILITY program to three tapes at a time.

```
OPTIONS(WORKERS=3) DUMP  
= (FAMILYINDEX=1) TO T1(TAPES=3);  
= (FAMILYINDEX=2) TO T2(TAPES=3);  
= (FAMILYINDEX=3) TO T3(TAPES=3)
```

Example 9

The following example is identical to the previous example except that the tapes have been assigned serial numbers. SERIALNO(1) refers to the first cycle of that tape name. The first tape for cycle 1 of T1 is assigned serial number 100. If it overflows, the second tape, or version 2, is assigned serial number 101.

```
OPTIONS(WORKERS=3) DUMP  
= (FAMILYINDEX=1) TO T1(TAPES=3,  
SERIALNO(1)=100,101,  
SERIALNO(2)=200,201,  
SERIALNO(3)=300,301);  
= (FAMILYINDEX=2) TO T2(TAPES=3,  
SERIALNO(1)=110,111,  
SERIALNO(2)=210,211,  
SERIALNO(3)=310,311);  
= (FAMILYINDEX=3) TO T3(TAPES=3,  
SERIALNO(1)=120,121,  
SERIALNO(2)=220,221,  
SERIALNO(3)=320,321)
```

Example 10

The following command dumps all rows that reside on family index 1 and 4 of family DBDATA to tape TAPEX:

```
DUMP = (FAMILYINDEX=1,4 AND PACKNAME=DBDATA) TO TAPEX
```

Example 11

The following command dumps all rows in files DB/A/= and DB/B/= that reside on family index 1 of pack family DBDATA. In addition, all rows of DB/RDS/= are dumped if they satisfy either the inner condition (PACKNAME = DBDATA) or the outer condition (PACKNAME = DBDATA) and (FAMILYINDEX = 1). Because the first condition is less restrictive, all rows of DB/RDS/= on DBDATA meet this condition and are dumped.


```
DUMP (DB/A/=, DB/B/=, DB/RDS/= (PACKNAME=DBDATA))  
      (PACKNAME=DBDATA FAMILYINDEX=1) TO TAPEX
```

Example 12

The following command dumps all rows of the database to tape DUMPTAPE. After the dump has completed, the tape is rewound and read forward to check for correctness.

```
OPTIONS (FORWARD COMPARE) DUMP = TO DUMPTAPE
```

Example 13

The following command dumps all rows except files that are part of the EMPLOYEE1, EMPLOYEE2 disjoint data sets, subsets, and embedded structures. The NOCOMPARE option is set, and the structures are dumped according to the family index order.

```
OPTIONS (NOCOMPARE) DUMP = (EXCLUDE ALLDB/EMPLOYEE1/=,  
      ALLDB/EMPLOYEE2/=) BY FAMILY INDEX TO TAPEX
```

Example 14

The following command dumps all rows of the database, using the density FMTST9840 with data block size set to 25,000 words:

```
DUMP = TO TAPEX (DENSITY=FMTST9840, BLOCKSIZE=25000)
```

Example 15

The following command initiates an offline accumulated tape dump. All data blocks that have been modified since the last full dump are included in the file ACCUMTAPEX. The "DUMP =" syntax specifies that all database files must be included in the accumulated dump.

```
OPTIONS (FORWARD COMPARE, WORKERS=2) OFFLINE ACCUM DUMP =  
      TO ACCUMTAPEX
```

Example 16

The following command initiates an incremental dump. All data blocks that have been modified since the last full, incremental, or accumulated dump are included in the file INCRTAPE. The "DUMP =" syntax specifies that all database files must be included in the incremental dump.

```
INCR DUMP = TO INCRTAPE
```

DUMP and APPEND Examples Where the Backup Medium Is Multidump Tape

The following examples illustrate both dump and append functions for a multidump tape. APPEND allows backup to multidump tapes that already contain backups.

Example 1

The following commands dump all rows of the DB1 database and append all rows of the DB2 database to tape TESTTAPE. These dumps are labeled DB1DUMP and DB2DUMP.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = TO DB1DUMP TAPE  
= TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB2 APPEND = TO DB2DUMP TAPE  
= TESTTAPE")
```

Example 2

The following commands dump all rows of the DB1 database that reside on family index 3 to the multidump tape TESTTAPE and append all rows of the DB2 database to the same tape. These dumps are labeled DB1DUMP and DB2DUMP.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = (FAMILYINDEX = 3)  
TO DB1DUMP TAPE = TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB2 APPEND = (FAMILYINDEX = 3)  
TO DB2DUMP TAPE = TESTTAPE")
```

Example 3

The following commands dump and append each family index to the same tape TESTTAPE. The rows on each family index are partitioned into three parts on the same tape TESTTAPE.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = (FAMILYINDEX = 1)  
TO DB1DUMP TAPE = TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB1 APPEND = (FAMILYINDEX = 2)  
TO DB2DUMP TAPE = TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB1 APPEND = (FAMILIYINDEX = 3)  
TO DB3DUMP TAPE = TESTTAPE")
```

Example 4

The following commands dump to DB1DUMP and append to DB2DUMP on a multidump tape TESTAPE with a serial number specification 394239.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = TO DB1DUMP TAPE =  
TESTTTAPE(SERIALNO = 394239)")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB2 APPEND = TO DB2DUMP TAPE =  
TESTTAPE(SERIALNO = 394239)")
```

Example 5

The following commands dump to DB1DUMP and append to DB2DUMP on tape TESTTAPE all rows that reside on family index 1 of family DMTEST.

```
RUN * SYSTEM/DMUTILITY("DB = DB1 DUMP = (FAMILYINDEX = 1
AND PACKNAME = DMTEST) TO DB1DUMP TAPE = TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB2 APPEND = (FAMILYINDEX = 1
AND PACKNAME = DMTEST) TO DB2DUMP TAPE = TESTTAPE")
```

Example 6

The following commands dump to DB1DUMP and append to DB2DUMP on tape TESTTAPE all rows of the database. After each dump has completed, the tape is positioned to the beginning of the dump and read forward to check for correctness due to the FORWARD COMPARE option.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 OPTIONS(FORWARD COMPARE) DUMP =
TO DB1DUMP TAPE = TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB2 OPTIONS(FORWARD COMPARE)
APPEND = TO DB2DUMP TAPE = TESTTAPE")
```

Example 7

The following commands dump to DB1DUMP and append to DB2DUMP all rows of the databases DB1 and DB2, using the density FMTST9840 with data block size set to 25,000 words.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = TO DB1DUMP TAPE =
TESTTAPE(DENSITY = FMTST9840, BLOCKSIZE = 25000)")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB2 DUMP = TO DB1DUMP TAPE =
TESTTAPE(DENSITY = FMTST9840, BLOCKSIZE = 25000)")
```

Example 8

The following commands initiate a FULL OFFLINE dump, DB1DUMPFULL, and adds both an ACCUMULATED dump, DB1DUMPACCUM, and an INCREMENTAL dump, DB1DUMPINCR, to the same multidump tape. All data blocks that have been modified since the last full dump are included in the dumps DB1DUMPACCUM and DB1DUMPINCR.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 OFFLINE DUMP = TO DB1DUMPFULL
TAPE = TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB1 OFFLINE ACCUM APPEND =
TO DB1DUMPACCUM TAPE = TESTTAPE")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB1 OFFLINE INCR APPEND =
TO DB1DUMPINCR TAPE = TESTTAPE")
```

Example 9

The following commands place the first dump named DB1DUMP on the tape and appends a second dump named DB2DUMP with the AXREADERROR option, which enables you to take a specific action if a read operation error occurs during a DMUTILITY backup dump.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 OPTIONS(AXREADERROR) DUMP =  
TO DB1DUMP TAPE = TESTTAPE")  
  
RUN *SYSTEM/DMUTILITY("DB = DB2 OPTIONS(AXREADERROR) APPEND =  
TO DB2DUMP TAPE = TESTTAPE")
```

Example 10

The following command initiates a database dump, DB1DUMP, with the use of a COMPRESSED specification for a tape TESTTAPE, which will compress the data for compressed tapes.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = TO DB1DUMP  
TAPE = TESTTAPE(COMPRESSED) ")
```

Example 11

The following command initiates a database dump DB1DUMP with the use of a NONCOMPRESSED specification for a tape TESTTAPE, which will leave the data uncompressed for uncompressed tapes.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = TO DB1DUMP  
TAPE = TESTTAPE(NONCOMPRESSED) ")
```

Example 12

The following command initiates a database dump DB1DUMP with the use of a SCRATCHPOOL specification for a tape TESTTAPE, which designates the scratch pool from which you want to retrieve a tape.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 DUMP = TO DB1DUMP  
TAPE = TESTTAPE(SCTACHPOOL = <17-character identifier>")
```

Example 13

The following command dumps all rows of the database to a multidump tape. The dump name is TESTDB063094 and the multidump tape name is DBDUMP063094:

```
DUMP = TO TESTDB063094 TAPE = DBDUMP063094
```

Example 14

The following command uses APPEND to add a new dump of all database rows to an existing multidump tape. The dump name is ANOTHERDB063094 and the multidump tape name is DBDUMP063094:

```
APPEND = TO TESTDB063094 TAPE = DBDUMP063094
```

DUMP Command Examples Where the Backup Medium Is Disk

The following examples illustrate DUMP command syntax when the output medium is disk.

Example 1

Assume a user with the usercode DBA issues the following disk dump command against a database called MYDB:

```
DUMP = TO DBDUMP063094 ON SAVEPACK
```

The command causes the DMUTILITY program to attempt a dump of all rows of the MYDB database to a file called DBDUMP063094 on the disk pack called SAVEPACK under the DBA usercode.

If (DBA)DBDUMP063094 on SAVEPACK already exists; then the DMUTILITY program requires operator assistance. Refer to “Operator Interface to DMUTILITY for Disk Stream Dumps” in [Section 4, Using DMUTILITY](#), for more information.

Example 2

The following example is a multiple-file version of example 1:

```
DUMP = TO DBDUMP063094 ON SAVEPACK (FILES=4)
```

The results of this command are

- An empty file called DBDUMP063094 located on the pack SAVEPACK
- Four files each containing one quarter of the database: DBDUMP063094/01, DBDUMP063094/02, DBDUMP063094/03, and DBDUMP063094/04
- All four files are dumped in parallel.

Example 3

Assume a user with the usercode DBA issues the following disk dump command against a database called TESTDB:

```
DUMP = (FAMILYINDEX = 3) TO DISKX ON DMS
```

This command causes the DMUTILITY program to attempt a dump of all rows of the TESTDB database, residing on family index 3, to a newly created file called (DBA)DISKX on the pack DMS.

Example 4

The following command creates three files on the pack DMS: DISKX, DISKX/01, and DISKX/02. The files DISKX/01 and DISKX/02 are dumped in parallel.

```
DUMP = (FAMILYINDEX = 3) TO DISKX ON DMS (FILES=2)
```

Example 5

The following command dumps the data file for structure A for the DB database to a disk file, SAVE/A, on the pack PK1 and the data file for structure B to SAVE/B on pack PK2. Both are saved under the usercode that is running the DMUTILITY program.

```
DUMP DB/A/DATA TO SAVE/A ON PK1;  
DB/B/DATA TO SAVE/B ON PK2
```

Example 6

The following command creates two dump files: SAVE/A on pack PK1, and SAVE/B on pack PK2. The FILES=1 element of the command has no effect on the file naming convention.

```
DUMP DB/A/DATA TO SAVE/A ON PK1 (FILES=1);  
DB/B/DATA TO SAVE/B ON PK2
```

Example 7

The following command results in a warning message that the WORKERS option is ignored. The disk dump occurs, but only one worker is used. It dumps all the rows of the database to the file D1 on the pack DMS.

```
OPTIONS(WORKERS=3) DUMP = TO D1 ON DMS
```

Example 8

The following command results in a warning message that the FORWARD COMPARE option is ignored for a disk dump. However, the disk dump is allowed to continue. The block size of file D1 is 19,200 words because there is no BLOCKSIZE specification in the DUMP command.

```
OPTIONS(FORWARD COMPARE) DUMP = TO D1 ON DMS
```

Example 9

The following commands result in syntax errors:

- This command returns an error because you cannot use the TAPES option with disk files.

```
DUMP = TO TAPEX ON SAVEPACK (TAPES=3)
```

- This command returns an error because the keyword TO is missing from the statement.

```
DUMP = TAPEX ON DISK
```

- This command returns an error because the keyword FILES was expected and instead the keyword SERIALNO was found.

```
DUMP = TO TAPEX ON DISK (SERIALNO= 231,232,233)
```

Example 10

The following command creates four files on the pack DATAPACK: DBDUMP, DBDUMP/01, DBDUMP/02, and DDDUMP/03. These files are dumped in parallel and all have a BLOCKSIZE value of 3600 words, except the DBDUMP file.

```
DUMP = TO DBDUMP ON DATAPACK (FILES=3, BLOCKSIZE=3600)
```

Example 11

The following command dumps all rows of the database except files that are part of DATA1 and DATA2 disjoint data sets, subsets, and embedded structures. The destination dump file has a BLOCKSIZE value of 15,000 words.

```
DUMP = (EXCLUDE ALLDB/DATA1/=, ALLDB/DATA2/=)
      TO DISKDUMP(BLOCKSIZE=15000) ON SAVEPACK
```

Example 12

The following command initiates an offline accumulated disk dump. All data blocks that have been modified since the last full dump are included in the file ACCUMDUMP ON DATAPACK. The "DUMP =" syntax specifies that all database files must be included in the accumulated dump.

```
OPTIONS (FORWARD COMPARE, WORKERS=2) OFFLINE ACCUM
DUMP = TO ACCUMDUMP ON DATAPACK
```

Example 13

The following command initiates an accumulated disk dump. All data blocks that have been modified since the last full, incremental, or accumulated dump are included in the file INCRDUMP ON DATAPACK. The "DUMP =" syntax specifies that all database files must be included in the incremental dump.

```
INCR DUMP = TO INCRDUMP ON DATAPACK
```

DUMP Command Examples Where the Backup Medium Is Both Single Dump Tape and Disk

The following examples illustrate DUMP command syntax when the output medium is both tape and disk.

Example 1

The following command backs up all the information on

- Family index 3 to three files on the pack DMS
- Family index 1 and 4 to tape TAPEX

```
DUMP = (FAMILYINDEX = 3) TO DISKX ON DMS (FILES=3);
      = (FAMILYINDEX=1,4 ) TO TAPEX
```

Example 2

The following command backs up

- The data file for structure A to a disk file SAVE/A on the family PK1
- The data file for structure B to a disk file SAVE/B on the family PK2
- All of the database files on family index 1 to tape T1

In this instance, because both tape and disk media are being used, the WORKERS option setting is ignored.

```
OPTIONS(WORKERS=3) DUMP DB/A/DATA TO SAVE/A ON PK1;  
DB/B/DATA TO SAVE/B ON PK2;  
= (FAMILYINDEX=1) TO T1
```

VERIFYDUMP Command (DMUTILITY)

Purpose

Use the VERIFYDUMP command to ensure that a dump is usable. You can use the VERIFYDUMP command with complete and partial dumps of the database. The VERIFYDUMP command checks that a dump is free from the following types of errors:

- Block checksum errors
- Block sequencing errors
- I/O errors

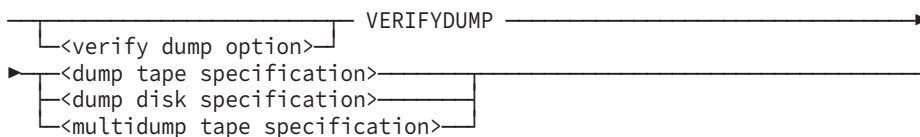
If the DMUTILITY program detects a problem with a dump, an error message is displayed. Depending on the type of error encountered and, if applicable, your response to the error message, the DMUTILITY program either continues or terminates.

Note: By default, the DMUTILITY program automatically performs the VERIFYDUMP operation when creating a dump. Unless the dump was created using the NOCOMPARE option, it is unnecessary to initiate a separate run to perform a VERIFYDUMP operation immediately following the creation of the dump tape. The NOCOMPARE option suppresses automatic verification of dump tapes and should only be used in conjunction with a VERIFYDUMP command at a later time or on another system.

Syntax

The following diagrams illustrate the syntax for the VERIFYDUMP command. Explanations of the syntax elements follow the diagrams. The syntax elements are explained in the order in which they appear in the syntax diagrams.

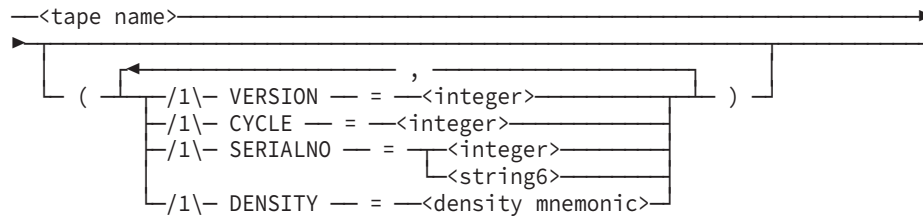
VERIFYDUMP Command



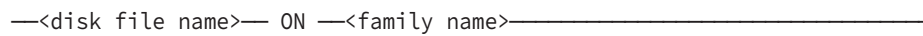
<verify dump option>

```
— OPTIONS — ( — WORKERS = <integer> — ) —
```

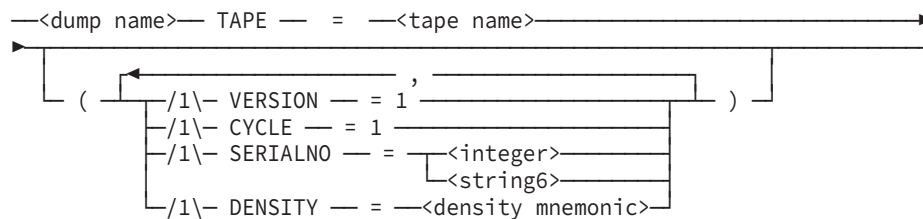

<dump tape specification>



<dump disk specification>



<multidump tape specification>



Verify Dump Option

If you are verifying a single tape dump, use the `WORKERS` option to identify the maximum number of tape reels that can be processed in parallel. The `WORKERS` option is not valid for use with multidump tapes. If you are verifying a disk dump, the `WORKERS` option is ignored.

Dump Tape Specifications and Dump Disk Specification

Use the dump tape specification or the dump disk specification to identify the name of the dump you want to verify.

For single tape dumps, provide the version, cycle, and serial number of the reel in the dump that was completed last. Otherwise, when automatic verification is done, the following conditions might occur:

- More reels of the dump tape than necessary might be processed in order to verify the desired rows.
- The list of the tapes needed to verify the desired rows might not contain every required cycle and version of the tape.

Copying Database Backups

Introduction

The `DMUTILITY` program provides the following two commands for copying existing database dumps from medium to medium: `COPYDUMP` and `DUPLICATEDUMP`.

COPYDUMP Command

Use the COPYDUMP command to copy a database dump from one device to another. The input and output devices do not have to be the same type of device. The copy of the dump is given the same timestamp as the original dump. While copying the dump, the DMUTILITY program verifies the dump for correctness. If a problem occurs, an appropriate error message displays.

DUPLICATEDUMP Command

Single Dump Tape

Use the DUPLICATEDUMP command to copy a database dump from one device to a device of the same type.

In the case of a tape dump, the DUPLICATEDUMP command enforces a reel-for-reel duplication. That is, if the original dump consists of five reels, the copy consists of five reels. The information on any reel of the original dump is copied to exactly one reel of the copy. As a result, if you need to, you can use reels 1, 3, and 5 from the original dump with reels 2 and 4 from the copy of the dump for a recovery of the database.

In the case of a disk dump, the DUPLICATEDUMP command enforces a file-for-file duplication. That is, if the original dump consists of three files, the copy consists of three files.

Multidump Tape

This option is not allowed for multidump tapes.

Diskstream Dumps

Use the DUPLICATEDUMP command to copy a database dump from one or more disk files to another set of one or more disk files.

COPYDUMP Command (DMUTILITY)

Purpose

Use the COPYDUMP command to create a copy of a database backup. The copy can be made on the same or a different type of media as the original database backup. If you want to enforce reelforreel or fileforfile similarity, use the DUPLICATEDUMP command.

You cannot use COPYDUMP to copy incremental or accumulated dumps. If you attempt to do so, DMUTILITY produces the following syntax error:

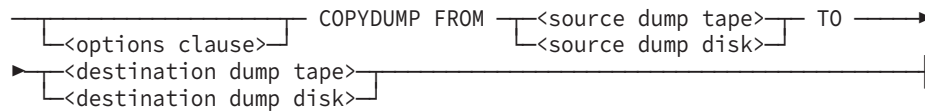
```
CAN'T COPY OR DUPLICATE DUMP FOR AN INCREMENTAL/ACCUMULATED DUMP
```

Note: *A dump cannot be copied to a permanent directory.*

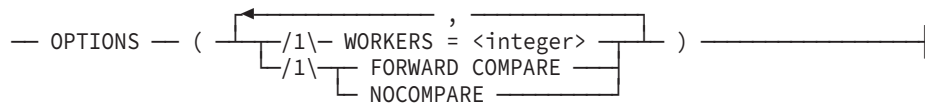
Syntax

The following diagrams illustrate the syntax for the COPYDUMP command. Explanations of the syntax elements follow the diagrams. The syntax elements are explained in the order in which they appear in the syntax diagrams.

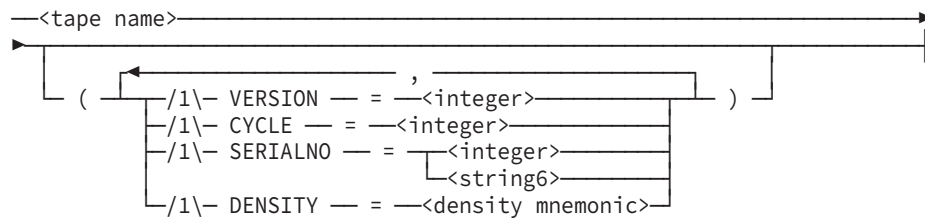
COPYDUMP Command



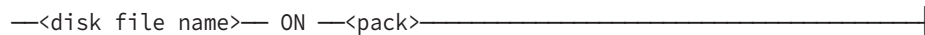
<options clause>



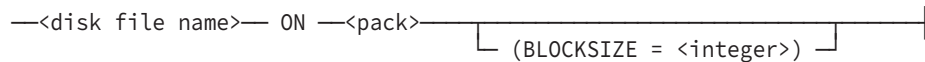
<source dump tape>



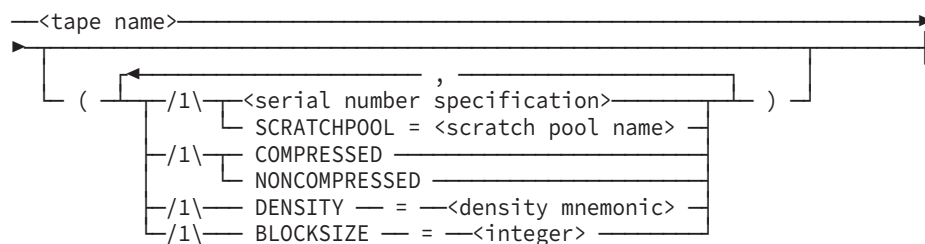
<source dump disk>



<destination dump disk>



<destination dump tape>



OPTIONS Clause

Use the OPTIONS clause to

- Identify the maximum number of tape reels that can be processed in parallel. The maximum value you can assign to the WORKERS option is 50.
- Request that a forwardcomparison technique be used to verify the copy of the dump.

Requesting a forward comparison is valid only when the copy destination is tape.

- Request that the automatic checking of a newly created tape be skipped by using the NOCOMPARE option. NOCOMPARE and FORWARD COMPARE are mutually exclusive options.

Note: *If you specify the NOCOMPARE option for a disk dump, a warning message is displayed and the option is ignored.*

If you specify the NOCOMPARE option for a tape dump, a message is displayed indicating that the output dump is not verified.

Source Dump Tape Clause

Use the source dump tape clause to identify the name of the source dump tape.

For tape dumps, provide the version, cycle, and serial number of the reel in the dump that was completed last. Otherwise, when automatic tape recovery is done, the following conditions might occur:

- More reels of the dump tape than necessary might be processed in order to load the desired rows.
- The list of the tapes needed to load the desired rows might not contain every required cycle and version of the tape.

Source Dump Disk Clause

Use the source dump disk clause to identify the name of the source dump if the dump is on disk. The dump is expected to be located under the same usercode as the usercode from which the DMUTILITY run is initiated.

Destination Dump Tape Clauses

Use the destination dump tape clause to identify the tape to which you want to copy the database dump. Even if you are copying a tape dump, the source and destination tape drives do not need to be the same type of device.

By default, if you mount

- An uncompressed tape, no data compression occurs
- A compressed tape, data compression occurs

To ensure that data compression either does or does not occur, include the COMPRESSED or the NONCOMPRESSED option in your COPYDUMP command.

Use the BLOCKSIZE clause to set the BLOCKSIZE value for the destination dump tape. The DMUTILITY program accepts a block size from 900 to 65535 words; however, the universal default block size is 10922 words.

Destination Dump Disk Clause

Use the destination dump disk clause to identify the disk file to which you want to copy the dump. The dump is copied to the same usercode from which the DMUTILITY run is initiated.

Use the BLOCKSIZE clause to set the BLOCKSIZE attribute for the destination dump file. This value must be a multiple of 60 words, and range from 900 words to 65,520 words. If you specify an invalid BLOCKSIZE value, the DMUTILITY program terminates with the following error message:

```
BLOCKSIZE MUST BE A MULTIPLE OF 60 WORDS AND RANGE
FROM 900 TO 65520 WORDS
```

Supply a unique name for the destination dump disk file. If you do not supply a unique name, the following warning appears:

```
DISPLAY: **WARNING: **<pack name>** SPECIFIED DISKSTREAM
DUMP ALREADY EXISTS
          <dump file name>
ACCEPT: DISKSTREAM: AX '<file name>' OR 'QUIT DMUTILITY'
```

Cataloging Tape and Disk Backup Information

Tape and disk backup information is cataloged if the DMDUMPDIR program is enabled, the control file is present, and the main directory is present.

COPYDUMP Command Examples

The following examples illustrate the COPYDUMP command syntax.

Example 1

The following command copies the tape dump TESTDBDUMP as TESTDBDUMP2. Both the original and the copy of the dump are on tape. In this example, four workers are requested and the copy of the dump is verified using a forwardcomparison technique.

```
OPTIONS (WORKERS=4, FORWARD COMPARE) COPYDUMP FROM
TESTDBDUMP TO TESTDBDUMP2
```

Example 2

The following command copies the disk dump TESTDISKDUMP to a tape, and the copy of the dump is verified using a forwardcomparison technique.

```
OPTIONS (FORWARD COMPARE) COPYDUMP FROM TESTDISKDUMP ON SYSPACK
TO TAPEDUMP
```

Example 3

The following command copies a disk dump TESTDISKDUMP to TESTDISKDUMP2 on another disk, and the destination dump file is created with a BLOCKSIZE value of 6000 words:

```
COPYDUMP FROM TESTDISKDUMP ON DATAPACK TO TESTDISKDUMP2
ON BACKUP (BLOCKSIZE = 6000)
```

DUPLICATEDUMP Command

Purpose

Use the DUPLICATEDUMP command to create a duplicate of a database backup or backups. The duplicate must be made on the same type of media as the original database backup. The DUPLICATEDUMP command enforces reelforreel or fileforfile similarity. Because of this similarity, you can mix and match tape reels or files from the original and the duplicate in processes such as database recovery.

If you are duplicating a tape dump or a set of tape dumps, ensure that the capacity of the destination tape reels is at least as great as the original tape reels. If a destination tape reel has a lesser capacity than the source tape reel and there is insufficient space on the destination reel for all the data on the source reel, an error occurs and the duplication process fails.

You cannot use DUPLICATEDUMP to duplicate incremental or accumulated dumps. If you attempt to do so, DMUTILITY produces the following syntax error:

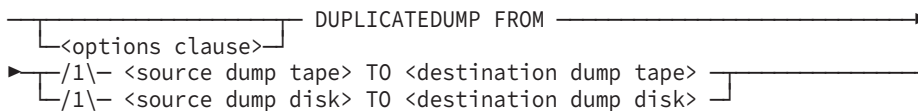
```
CAN'T COPY OR DUPLICATE DUMP FOR AN INCREMENTAL/ACCUMULATED DUMP
```

Note: A duplicate dump cannot be placed in a permanent directory.

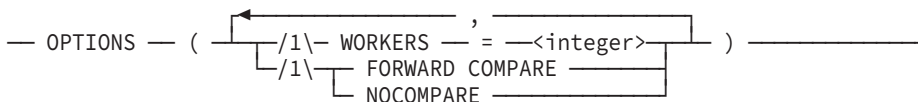
Syntax

The following diagrams illustrate the syntax for the DUPLICATEDUMP command. Explanations of the syntax elements follow the diagrams. The syntax elements are explained in the order in which they appear in the syntax diagrams.

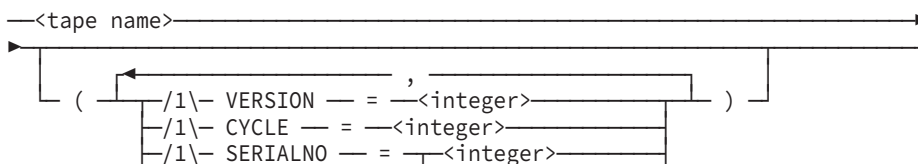
DUPLICATEDUMP Command



<options clause>



<source dump tape>



While not a recommended action, you can use a compressed source tape and a noncompressed destination tape, or a noncompressed source tape and a compressed destination tape. However, the COMPRESSED and NONCOMPRESSED options are provided only in the destination dump tape specification.

The same type of tape device must be used for the source and for the destination dump tape files.

You must also ensure that the capacity of any destination reel is at least as great as the capacity of any source reel. The DMUTILITY program terminates with an error if the destination tape reel cannot contain the data on the source tape reel. This error occurs because a reel-for-reel similarity is enforced by the DUPLICATEDUMP command.

Source Dump Disk and Destination Dump Disk Clauses

Use the source dump disk and the destination dump disk clauses to identify the names of the source and destination dump files when the dump is on disk. The source dump files must be located under the same usercode as the usercode from which the DMUTILITY run is initiated.

Supply a unique name for the destination dump disk files. If you do not supply a unique name, the following warning displays:

```
DISPLAY: **WARNING: **<pack name>** SPECIFIED DISKSTREAM
DUMP ALREADY EXISTS
                <dump file name>
ACCEPT: DISKSTREAM: AX '<file name>' OR 'QUIT DMUTILITY'
```

Cataloging Tape and Disk Backup Information

Tape and disk backup information is cataloged if DMDUMPDIR is enabled, the control file is present, and the main directory is present.

DUPLICATEDUMP Command Examples

The following examples illustrate the DUPLICATEDUMP command syntax.

Example 1

The following command duplicates the tape dump TESTDBDUMP as TESTDBDUMP2. In this example, four workers are requested and the duplicate of the dump is verified using a forwardcomparison technique.

```
OPTIONS (WORKERS=4, FORWARD COMPARE) DUPLICATEDUMP
FROM TESTDBDUMP TO TESTDBDUMP2
```

Example 2

The following command duplicates the disk dump DISKDUMP1:

```
DUPLICATEDUMP FROM DISKDUMP1 ON SYSPACK TO DISKDUMP2 ON BACKPACK
```


TAPEDIRECTORY Command (DMUTILITY)

Introduction

Use the TAPEDIRECTORY command to list the tape or disk file directory written by the DMUTILITY program at the beginning of each database backup tape reel or file. The directories of successive reels and files are cumulative. Thus, the directory of the last reel or last file written identifies all the database information that was backed up.

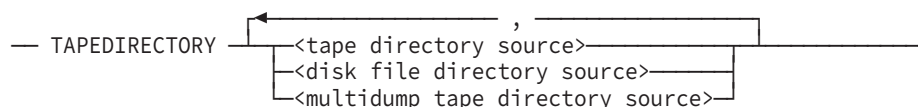
The directory generated by the TAPEDIRECTORY command is useful during recovery operations. From it, you can identify which reels or files contain the rows of interest, thus eliminating extra search time.

The directory provided by the TAPEDIRECTORY command describes the information backed up during one single database backup procedure. If you want to catalog all the database backup information for a particular database, use the DMDUMPDIR program and the BUILDDUMPDIRECTORY command. For more information on cataloging database backup information, refer to "Cataloging the Information in Database Backups" later in this section.

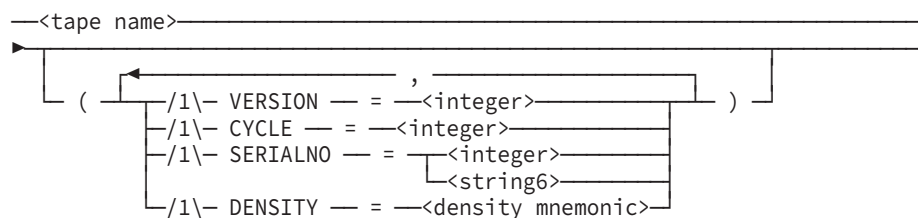
Syntax

The following diagrams illustrate the syntax for the TAPEDIRECTORY command. For explanations of the syntax elements, refer to "DMUTILITY DUMP Command" earlier in this section.

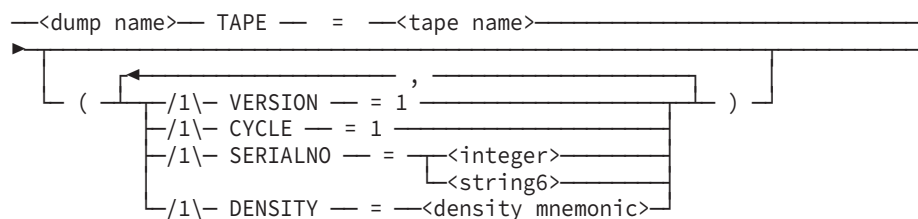
TAPEDIRECTORY Command



<tape directory source>



<multidump tape directory source>



Examples

The following examples illustrate the use of the TAPEDIRECTORY command.

Example 1

The following example produces a listing of the location of all rows dumped to the designated reel and dumped to all previous tape reels during the designated dump:

```
TAPEDIRECTORY DBTAPE020479 (CYCLE = 2, VERSION = 3)
```

If the designated reel is the last reel dumped by the DMUTILITY program, the tape directory contains information about all rows of the database that were dumped.

Example 2

The following example produces a listing of the locations of all rows that are dumped to a specific backup dump on the designated multidump reel.

```
TAPEDIRECTORY ANOTERHDB063094 TAPE = DBDUMPS063094 (CYCLE = 1,  
VERSION = 1)
```

TAPESET DIRECTORY Command (DMUTILITY)

Introduction

Use the TAPESET DIRECTORY command to either list the contents or recreate the fast access directory file for a multidump tape or set of multidump tapes.

Note: Do not use the database name specification "DB =" with this command.

The DMUTILITY program automatically creates or updates a directory file when the first reel of a set of tapes is created by the DUMP command with the multidump tape specification. Directory files are created on the pack identified by the DL LIBMAINTDIR command. If no location is specified, directory files are located on the system halt/load unit.

Directory file titles use the following format:

```
<tape name>/TAPESETDIRFILE/<yyyymmdd><hhmmss>  
ON <libmaint dir pack name>
```

The MCP automatically removes a fast access directory file (following a response to a waiting entry) when you purge a volume, or physical reel, from the set of tapes by using the PG or SN system command. Be sure to respond to the automatic MCP waiting entry that requests confirmation that you want to scratch the tape. If the fast access directory has been previously removed, the MCP does not ask for confirmation. Refer to the *System Commands Reference* for more information about the PG and SN commands. At times, you might want to copy the directory file manually so that you can transport it to another system.

Note: Unisys recommends that you do not move multidump tapes between systems to add dumps to them because this can result in inconsistent fast access directory files on the different systems. These inconsistent directory files can cause existing dumps to be overwritten.

Syntax

The following diagram illustrates the syntax for the TAPESET DIRECTORY command. Explanations of the syntax elements follow the diagram.



CREATE Option

Use CREATE to re-create the fast access disk directory that is used for positioning the tape.

If the multidump tape has been used to hold backups from more than one usercode, ensure that all of the usercodes have visibility to the fast access directory. One method to accomplish this task is to give the fast access directory PUBLIC security.

When the fast access directory is manually re-created by using the TAPESET DIRECTORY CREATE command, the usercode initiating the command must be able to create files under the usercode of the first dump on the tape. Otherwise, the operating system emits a security violation error.

LIST or PRINT Option

Use LIST to create reports of the multidump backup tape contents. This option sends the report to the terminal where you initiated the command.

PRINT creates reports of the multidump backup tape contents. This option sends the reports to a printer backup file.

DMUTILITY accesses the name of the directory file on disk by way of the ASSOCIATEDFILENAME attribute. You can obtain a directory listing in one of two ways:

- File-equate the file TSDIR to the appropriate directory file on the library maintenance directory (LIBMAINTDIR) pack. You can obtain the pack name by using the DL system command (see Example 2).
- Mount any of the tapes.

Tape contents can also be reported when a fast access directory file is not present. In the absence of a directory file, DMUTILITY reads the tape to create the report, and all members of the set must be available.

Examples

Example 1

The following example produces a printed listing of all dumps contained within the Fast access directory file for the multidump tape labeled QRESCFILETAPE1:

```
RUN SYSTEM/DMUTILITY ("TAPESET DIRECTORY PRINT
TAPE = QRESCFILETAPE1")
```

You can use any tape from the set to locate the directory.

The following is a sample of the output produced by the TAPESET DIRECTORY command with the LIST or PRINT option included:

```
Unisys Enterprise Database Server for ClearPath
MCP SYSTEM/DMUTILITY
                SSR 50.1 (50.140.0077)
                Thursday, January 8, 2004 15:56:08.1250
                SYSTEM: NX4800 SERIAL NUMBER: 9999
                FAST ACCESS DIRECTORY FILE REPORT
DIRECTORY FOR TAPE: QRESCFILETAPE1/TAPESET
MultiDump Directory File Name:
(PROD1)QRESCFILETAPE1/TAPESETDIRFILE/2004010813759 ON SYS009
REELNO: 1 (SERIALNO=HALLAJ)
DUMP#    1 = DBIDUMP          DB = (PROD1)QRESCFILE ON SYS009
                                DUMPED ON 01/08/2004 13:58:12
                                DUMP IS ONLINE
                                TYPE IS FULL
DUMP#    2 = DB2DUMP         DB = (PROD1)QRESCFILE ON SYS009
                                DUMPED ON 01/08/2004 13:58:23
                                DUMP IS ONLINE
                                TYPE IS ACCUMULATED
DUMP#    3 = DB3DUMP         DB = (PROD1)QRESCFILE ON SYS009
                                DUMPED ON 01/08/2004 13:58:55
                                DUMP IS ONLINE
                                TYPE IS INCREMENTAL
END OF TAPE.
```

Example 2

The following example uses the file-equate method to produce a printed listing of all dumps contained within the fast access directory file for the multidump tape labeled DBTAPE020479:

```
RUN SYSTEM/DMUTILITY ("TAPESET DIRECTORY PRINT
TAPE = DBTAPE020479");
FILE TSDIR = (TITLE = DBTAPE020479/TAPESETDIRFILE/20030903135109
ON DIRECTORYPACK)
```

Example 3

The following example sends the report of all dumps contained within the fast access directory file for the multidump tape labeled DBTAPE020479 back to your terminal when you manually run SYSTEM/DMUTILITY:

```
RUN SYSTEM/DMUTILITY
("TAPESET DIRECTORY LIST TAPE = DBTAPE020479")
```

You can use any tape from the set to locate the directory.

Example 4

The following example creates a new fast access directory for the dumps on the multidump tape labeled DBTAPE020479:

```
RUN SYSTEM/DMUTILITY
("TAPESET DIRECTORY CREATE TAPE = DBTAPE020479")
```

Cataloging the Information in Database Backups

Introduction

You can catalog information regarding tape and disk database backups in a two-level directory called the dump tape directory. The purpose of the catalog is to improve database recovery performance by making the information required for the database recovery operation easily located. [Figure 6-1](#) illustrates the dump tape directory concept. As shown in the illustration, the dump tape directory contains two types of files—the main directory file and a series of dump directory files.

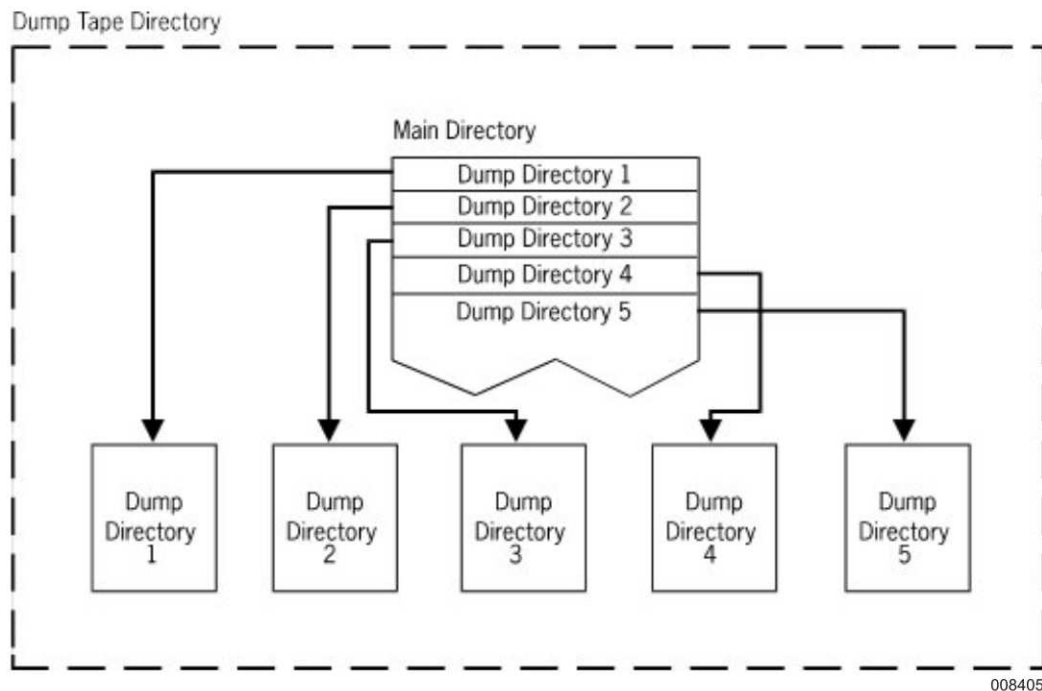


Figure 6-1. Dump Tape Directory

Main Directory File

Each database can have one main directory file. The main directory file contains one entry for each database backup for which you want to store information. [Figure 6-1](#) shows that each entry in the main directory file identifies a dump directory that contains information about a particular database backup.

The main directory file resides on the same pack under the same usercode as the database control file and has the name

```
<database name>/DMDUMPDIR
```

Dump Directory File

One dump directory file exists for each entry in the main directory. Each dump directory file contains information about one database backup—the name of the dump tape or disk file and a list of the database files backed up.

By default, dump directory files are stored on the same pack under the same usercode as the database control file. Using the DMDUMPDIR program you can choose to locate the dump directory files differently. Dump directory files have the following naming convention:

```
<database name>/DMDUMPDIR/<dump time>/<dump name>
```

The dump time identifies the date and time at which the dump was created. Starting with SSR 43.2, the dump time has the format YYYYMMDDHHMMSS. The following dump time example indicates that the dump was created on January 21, 1997, at 15:21:16:

```
19970121152116
```

The dump name is the name given to the tape or disk file in the DMUTILITY *DUMP*, *COPYDUMP*, or *DUPLICATEDUMP* command.

If you use the same dump name for the source dump tape and the destination dump tape in the *COPYDUMP* or *DUPLICATEDUMP* command, a new dump directory is created using the following naming convention:

```
<database>/DMDUMPDIR/<dump time>/<dump name>/<system time>
```

Tools for Developing and Maintaining Dump Tape Directories

Two tools are provided for developing and maintaining dump tape directories:

- DMDUMPDIR program
- BUILDDUMPDIRECTORY command of the DMUTILITY program

Use the DMDUMPDIR program to perform any of the following tasks for a particular database:

- Create a main directory file and initiate the automatic generation of dump directory files (ENABLE command).
- Discontinue the automatic generation of dump directory files and delete all files associated with the dump tape directory (DISABLE command).
- Insert a main directory file entry for an existing dump directory file (ADD command).
- Delete a main directory file entry and its associated dump directory file (DELETE command).
- List or print the information stored in the main directory file or in a dump directory file, or list or print information about particular items in the database (LIST and WRITE commands).

Use the DMUTILITY *BUILDDUMPDIRECTORY* command to construct a dump directory file for an existing database backup and generate an appropriate main directory file entry.

Process for Developing a Dump Tape Directory

To develop a dump tape directory for a database, perform one of the following procedures.

For a New Database

For a new database, run the DMDUMPDIR program and use the ENABLE command. Once the ENABLE command is processed, all new database backups are automatically logged in the dump tape directory.

For an Existing Database

For an existing database, perform the following steps:

1. Run the DMDUMPDIR program and use the ENABLE command.
All new database backups are automatically logged in the dump tape directory.
2. Identify all the previously generated database backups about which you want to store information.
3. Use the BUILDDUMPDIRECTORY command to build dump directory files for all the backups identified in step 2.

Appropriate entries are automatically made in the main directory file for each dump directory file you build.

Updating the Dump Tape Directory

Once you have issued the DMDUMPDIR *ENABLE* command for a database, all of the following three actions automatically update the dump tape directory:

- Creating a new database backup (DUMP command)
- Copying an existing database backup (COPYDUMP command)
- Duplicating an existing database backup (DUPLICATEDUMP command)

When you use the DMUTILITY *COPYDUMP* or *DUPLICATEDUMP* command and the database control file is present, the DMDUMPDIR program automatically generates a dump tape directory entry for the copy or duplicate of the dump. The new dump tape directory entry has the same timestamp as the entry for the original database dump. In a listing of the dump directory, duplicates and copies of a database dump are appropriately marked.

Running the DMDUMPDIR Program

The basic statement for running the DMDUMPDIR program is

```
RUN SYSTEM/DMDUMPDIR ("<DMDUMPDIR statement>");  
FILE DASDL = DESCRIPTION/<database name>;
```

To ensure that the DMDUMPDIR program can locate the database description file, use the appropriate file equation statement. The DMDUMPDIR statement provides the action you want performed by the DMDUMPDIR program. For details of the DMDUMPDIR statement syntax, refer to "DMDUMPDIR Program" later in this section.

DMDUMPDIR Program

Introduction

Dump directory information is retrieved and modified using the DMDUMPDIR program.

When you perform any of the following three actions and the DMDUMPDIR program is enabled, dump directory entries are created automatically:

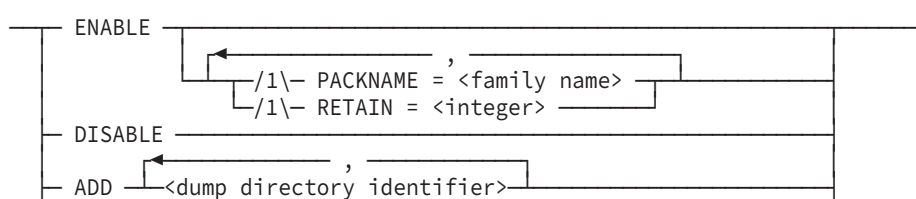
- Create a new database dump.
- Copy an existing database dump.
- Duplicate an existing database dump.

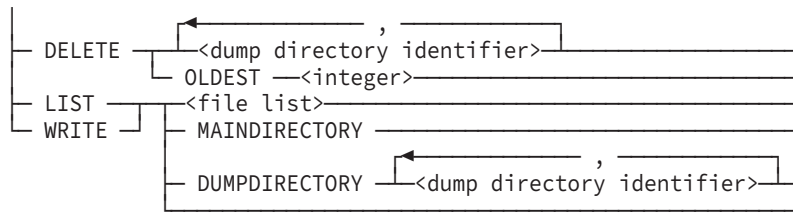
Note: When you use the DMDUMPDIR program with permanent directory databases, the dump directory files are placed under the usercode of the task initiator.

Syntax

The following diagrams illustrate the syntax for the DMDUMPDIR statement. The DMDUMPDIR statement commands are identified in functional order. Explanations of the syntax elements follow the diagrams.

<DMDUMPDIR statement>





<dump directory identifier>

— <dump time> / <dump name> _____

<file list>

— <file name> _____
 <row selector>

<row selector>

— (_____) _____
 /1\— FAMILYINDEX — = —<range>—
 /1\— ROW — = —<range>—
 /1\— PACKNAME — = —<family name>—
 /1\— ROWLOCK — = —<row selector>—
 /1\— LOCKEDROW —
 /1\— READEROR —

<range>

— <unsigned integer> _____
 - <unsigned integer>

ENABLE Command

Use the ENABLE command to establish a main directory for the database. The ENABLE command turns on a Boolean value in the database control file, creates a main directory file, and sets up the global section of the main directory.

Note: The database cannot be in use when the control file program updates the control file.

ENABLE Command Example

The following example causes a dump tape directory to be associated with the database. All dump directory files in this example reside on pack DBPACK. In addition, the directory retains information for a maximum of 10 dumps.

ENABLE PACKNAME = DBPACK, RETAIN = 10

PACKNAME Option

To identify the location for the dump directories, use the PACKNAME option. By default, dump directories are stored on the same pack as the main directory and the database control file.

Use the PACKNAME option primarily to change the location of an already enabled dump directory.

Note: *If the control file location changes, you must copy the main directory and dump directory to the new control file location.*

RETAIN Option

Use the RETAIN option to identify the number of dumps about which information is to be kept in the main directory. By default, information for 50 dumps is retained. When the RETAIN limit is exceeded, the oldest dumps are automatically deleted from the main directory.

Use the RETAIN option primarily to change the number of dumps about which information is maintained in an existing main directory.

DISABLE Command

Use the DISABLE command to disable the dump directory for a database. All available files associated with the dump tape directory, including the main directory and all dump directories, are deleted.

Note: *The database cannot be in use when the control file program updates the control file.*

ADD Command

Use the ADD command to manually add dump entries to the main directory if a dump directory file exists for the dump.

If the DMDUMPDIR program is enabled, then an entry for the dump is automatically generated in the main directory, and a dump directory file is made each time a dump is created.

DELETE Command

Use the DELETE command to remove dump entries from the main directory. If present, the corresponding dump directory is also removed from disk.

Use the dump identifier construct to designate the specific entries you want to remove. If you want to delete a number of entries because they are outofdate, use the `OLDEST` option. For example, use `OLDEST 3` to remove the three oldest entries in the main directory.

You can delete entries in the dump directory automatically or manually.

Entries are deleted automatically under either of the following circumstances:

- When the number of entries being added causes the maximum limit for entries to be exceeded

In this instance, the oldest entries are deleted to make room for the new entries. Refer to the `RETAIN` option for the `ENABLE` command earlier in this section for more details.

- When a database recovery process moves the database back in time

The recovery processes `ROLLBACK` and `REBUILD`, which can stop their processing before the end of the audit trail, perform these deletions. If a dump was made after the time at which the recovery process stops, the dump is no longer valid and is deleted automatically from the directory.

If you perform a manual recovery (for example, a copy or reprocessing operation), you must manually update the main directory to delete any invalidated dumps. Similarly, if invalid dumps become valid, you must manually add the information to the main directory.

DELETE Command Example

The following example removes from the main directory information about the tape dump `DMSDUMP` created on December 25, 1996 at 17:21:25. The corresponding dump directory file is automatically removed.

```
DELETE 19961225172125/DMSDUMP
```

LIST and WRITE Commands

Use the `LIST` and `WRITE` commands to display information on the contents of the main directory or dump directories, or to identify the dump tape or the dump file on which the most current copies of the specified rows reside. The `LIST` command directs output to a terminal. The `WRITE` command directs output to a printer.

File List Construct

Use the file list construct to identify the dump tape or the dump file on which the most current copies of the specified database rows reside.

File Name Construct

Use the file name construct to identify an Enterprise Database Server database file. The file name construct is an identifier or a series of identifiers that represents the Enterprise Database Server database file. Use the slash equal (/=) sign to represent a family of files. Use the equal (=) sign alone to designate all files in the database.

Row Selector Construct

Use the row selector construct to identify the dump tape or the dump file on which the specific rows in the designated database files reside. If you do not include a row selector construct, the report contains information about all rows in the designated database files.

FAMILYINDEX Option

Use the FAMILYINDEX option to limit the report to only those rows that currently reside on the specified family indexes.

ROW Option

Use the ROW option to limit the report to information on the data in the specified rows.

PACKNAME Option

Use the PACKNAME option to limit the report to information on the data on the specified pack. The PACKNAME option is normally used with the FAMILYINDEX option.

ROWLOCK = LOCKEDROW Option

Use the ROWLOCK = LOCKEDROW option to report on all locked rows.

ROWLOCK = READERROR Option

Use the ROWLOCK = READERROR option to report on all rows having read operation errors. The report does not include information about locked rows.

MAINDIRECTORY Option

Use the MAINDIRECTORY option to report on the contents of the main directory file.

DUMPDIRECTORY Option

Use the DUMPDIRECTORY option to report on the contents of the specified dump directory files.

Dump Directory Identifier Clause

Use the dump directory identifier clause to identify the dump entries you want to process.

The following examples illustrate the use of the dump identifier clause in the ADD command:

Example 1

This example adds information to the main directory for the tape dump DBDUMP created on July 1, 1996 at 12:52:05.

```
ADD 19960701125205/DBDUMP
```

Example 2

This example adds information to the main directory for the disk dump DBDUMP created on September 30, 1996 at 22:52:05.

```
ADD 19960930225205/DBDUMP ON BACKDUMP
```

DMDUMPDIR *WRITE* Command Examples

The following examples illustrate the use of the WRITE statement.

Example 1

This command lists information that identifies the dumps on which the most current copies of all rows in the database reside. The output is sent to a printer.

```
WRITE =
```

Example 2

This command lists information that identifies the dumps on which the most current copies of all rows dumped from family index 3 reside.

```
WRITE = (FAMILYINDEX = 3)
```

Example 3

This command lists information that identifies the dumps on which the most current copies of rows 1 through 5 of file DB/A/DATA reside.

```
WRITE DB/A/DATA (ROW = 1-5)
```

Example 4

This command lists information that identifies the dumps on which the most current copies of all rows of file DB/A/DATA and rows 1 through 8 of file DB/B/DATA reside.

```
WRITE DB/A/DATA, DB/B/DATA (ROW = 1-8)
```

Example 5

This command finds all rows in the database that are currently locked out or have read operation errors, and lists information that identifies the dump tapes or disk files on which the most current copies of these rows reside.

```
WRITE = (ROWLOCK = LOCKEDROW, READERROR)
```

Example 6

This command displays the contents of the main directory.

```
WRITE MAINDIRECTORY
```

Following is a sample listing of the contents of a main directory:

```
*** MAINDIRECTORY FOR DATABASE TESTDB ***

DESCRIPTION TIMESTAMP = 05/25/1995 14:20:44
BUILT BY DMDUMPDIR FORMAT LEVEL 4
  DUMPS TO RETAIN      = 50
  DUMP DIRECTORY PACKNAME = TESTPK
  NUMBER OF ENTRIES   = 4
  INDEX TO FIRST ENTRY = 13
  WORDS PER ENTRY     = 8
  WORDS IN MAIN DIRECTORY = 45
DUPLICATED DUMP TESTDB/DUMP/4
  DUMPED ON 06/09/1995 09:45:20
DUMP TESTDB/DUMP/1
  DUMPED ON 06/09/1995 09:45:20
DUMP TESTDB/DUMP/2
  DUMPED ON 06/09/1995 09:47:12
COPIED DUMP TESTDB/DUMP/3
  DUMPED ON 06/09/1995 09:47:126
```

Example 7

This command displays the dump directory for the dump called TESTDB/DUMP/1 created on June 9, 1995 at 09:45:20.

```
WRITE DUMPDIRECTORY 19950609094520/TESTDB/DUMP/1
```

Following is the output generated with this command:

```
*** DUMPDIRECTORY FOR DUMP TESTDB/DUMP/1 ON TESTPK ***

DUMPED ON 06/09/1995 09:45:20
DUMP IS OF DATABASE TESTDB
DESCRIPTION TIMESTAMP = 05/25/1995 14:20:44
UPDATE TIMESTAMP     = 05/25/1995 14:20:45
UPDATE LEVEL         = 1
BUILT BY DMDUMPDIR FORMAT LEVEL 4
  WORDS IN DUMP DIRECTORY = 83
  WORDS IN STRUCTURE ENTRY = 16
  INDEX TO FIRST STRUCTURE = 27
  TOTAL STRUCTURES DUMPED = 3
  INDEX TO TAPE INFORMATION = 81
```

```

AUDIT FILE NUMBER           = 1
DUMP BY FAMILYINDEX        = FALSE
FAMILYNAME TABLE SIZE     = 1
    
```

```

STRUCTURE # 2 ON TESTPK
  FORMAT TIMESTAMP          = 05/25/1995 14:20:44
  CREATION TIMESTAMP        = 05/25/1995 14:21:19
  VERSION TIMESTAMP         = 06/09/1995 09:45:22
  ROWS DUMPED               = 1
  ROWS IN STRUCTURE         = 1
  INDEX TO FIRST ROW        = 75
  FAMILY NUMBER             = 1
  ROW # 0 FAMILYINDEX       = 1 DMROWLOCK = 0
                           RESIDES ON TAPE NUMBER 0 TAPEBLOCK = 5
                           CYCLE = 1 VERSION = 1
    
```

```

STRUCTURE # 3 ON TESTPK
  FORMAT TIMESTAMP          = 05/25/1995 14:20:44
  CREATION TIMESTAMP        = 05/25/1995 14:21:20
  VERSION TIMESTAMP         = 05/25/1995 14:21:20
  ROWS DUMPED               = 1
  ROWS IN STRUCTURE         = 1
  INDEX TO FIRST ROW        = 77
  FAMILY NUMBER             = 1
  ROW # 0 FAMILYINDEX       = 1 DMROWLOCK = 0
                           RESIDES ON TAPE NUMBER 0 TAPEBLOCK = 6
                           CYCLE = 1 VERSION = 1
    
```

```

STRUCTURE # 4 ON TESTPK
  FORMAT TIMESTAMP          = 05/25/1995 14:20:44
  CREATION TIMESTAMP        = 05/25/1995 14:21:20
  VERSION TIMESTAMP         = 05/25/1995 14:21:20
  ROWS DUMPED               = 1
  ROWS IN STRUCTURE         = 1
  INDEX TO FIRST ROW        = 79
  FAMILY NUMBER             = 1
  ROW # 0 FAMILYINDEX       = 1 DMROWLOCK = 0
                           RESIDES ON TAPE NUMBER 0 TAPEBLOCK = 7
                           CYCLE = 1 VERSION = 1
    
```

Example 8

This command displays the dump information available for the database file TESTDB/DS1/DATA.

```
WRITE TESTDB/DS1/DATA
```

Following is a sample listing for the database file:

```

TESTDB/DS1/DATA ON TESTPK
-----
STRUCTURE # 3
  ROW # 0
  ON DUMP TESTDB/DUMP/3 ON TESTP
      CYCLE = 1 VERSION = 1
      DUMPED ON 06/09/1995 09:47:12
    
```

BUILDDUMPDIRECTORY Command (DMUTILITY)

Introduction

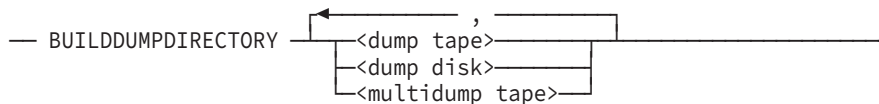
Use the BUILDDUMPDIRECTORY command for both of the following purposes:

- To build a dump directory file and insert a dump directory entry in the main directory file for each of the specified tape or disk dumps
- To recover dump directory files and to create dump directory files for dumps that were created prior to the DMDUMPDIR program being enabled

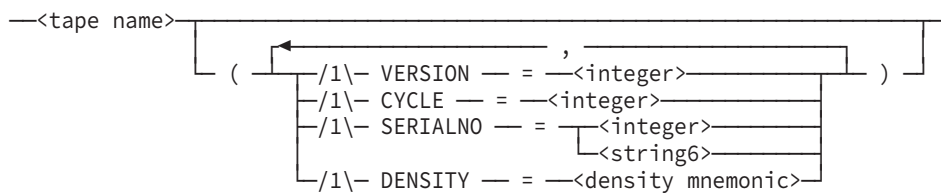
Syntax

The following diagrams illustrate the syntax for the BUILDDUMPDIRECTORY command. Explanations of these syntax elements follow the diagrams. The syntax elements are explained in the order in which they appear in the syntax diagrams.

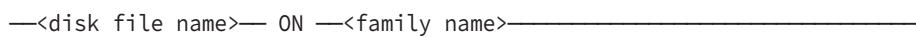
BUILDDUMPDIRECTORY Command



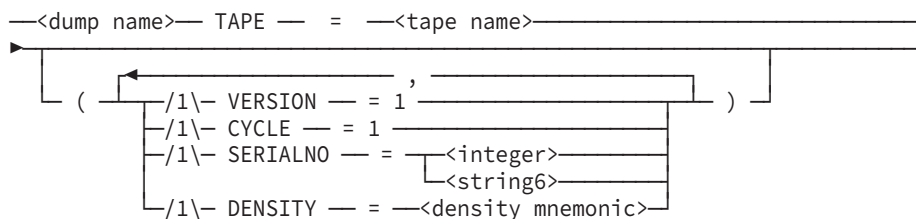
<dump tape>



<dump disk>



<multidump tape>



Explanation

For tape dumps, provide the version, cycle, and serial numbers for the reel in the dump that was completed last. Remember the following information when you are dumping to tape

For Multidump Tapes

- CYCLE and VERSION must be 1.
- Each DUMP or APPEND statement creates a uniquely named dump file. Thus, only one version of any given dump can exist.

For Single Dump Tapes

- More reels of the dump tape than necessary might be processed in order to load the desired rows.
- The list of the tapes needed to load the desired rows might not contain every required cycle and version of the tape.

For explanations of the syntax elements, refer to “DMUTILITY DUMP Command” earlier in this section.

Recovering Database Backup Catalog Information

Introduction

The information in the dump directory is a duplication, in a more convenient form, of information contained in database dumps. Consequently, the information is easily recovered if it is lost.

Recovering the Main Directory File

If the main directory file is lost, recover the information by using the `DMDUMPDIR ENABLE` command, followed by a series of `DMDUMPDIR ADD` commands. The `ENABLE` command creates an empty main directory file. And then the `ADD` commands populate the main directory file with information about the designated dumps.

Recovering a Dump Directory File

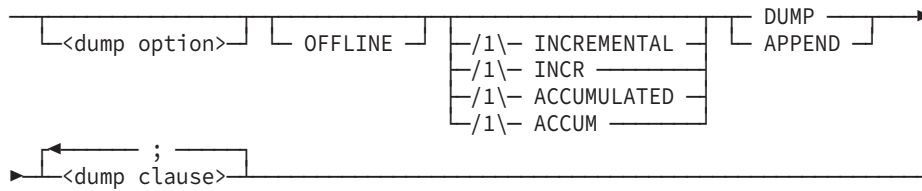
If a dump directory file is lost, recover the information using the `DMUTILITY BUILDDUMPDIRECTORY` command.

Quick-Reference Information

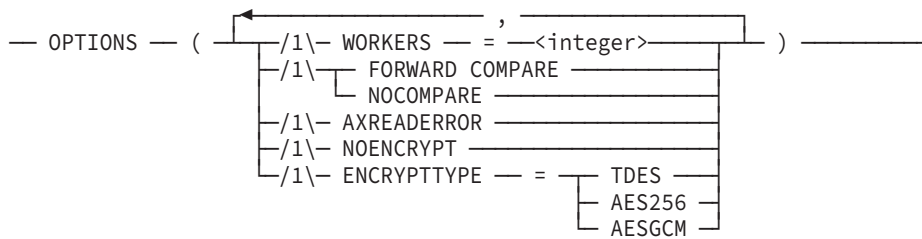
Introduction

The information presented here is for quick-reference purposes only. For an explanation of any element of a syntax diagram, refer to the appropriate information presented earlier in this section.

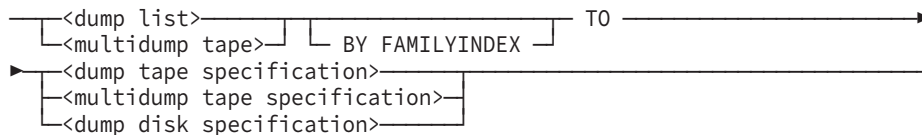
DUMP and Append Command



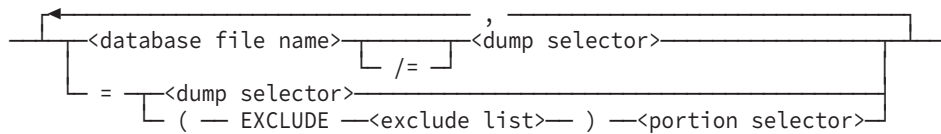
<dump option>



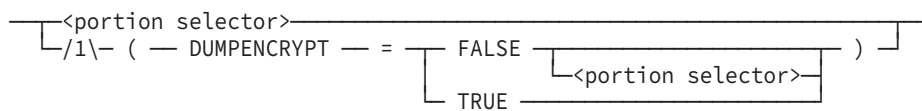
<dump clause>



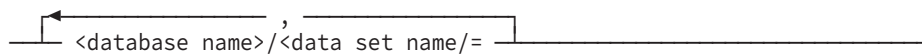
<dump list>



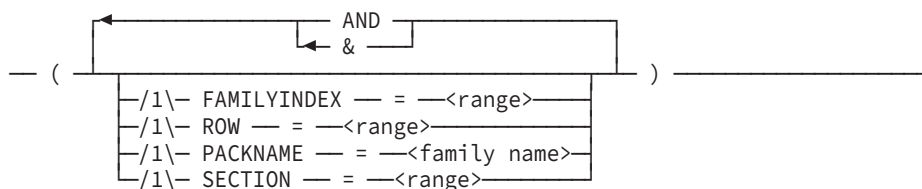
<dump selector>



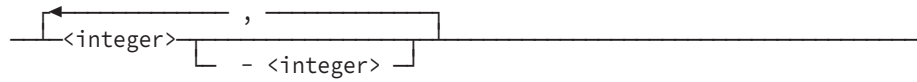
<exclude list>



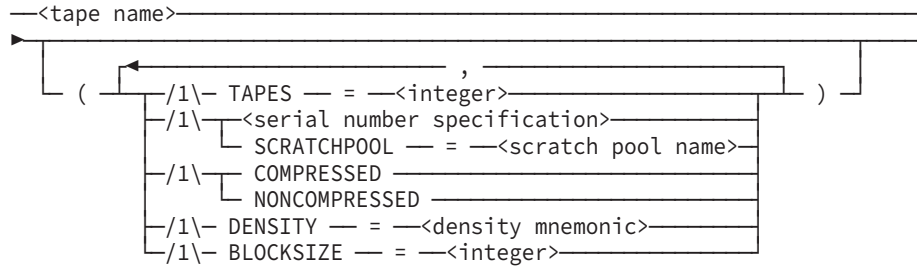
<portion selector>



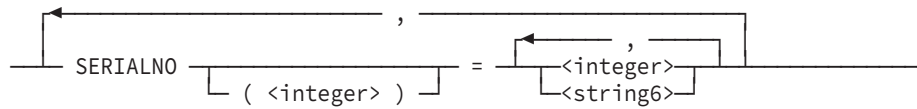
<range>



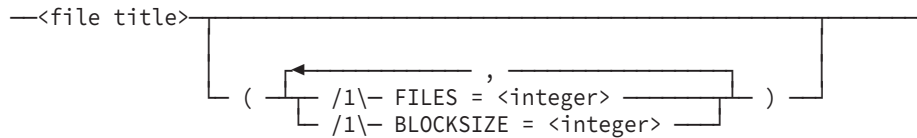
<dump tape specification>



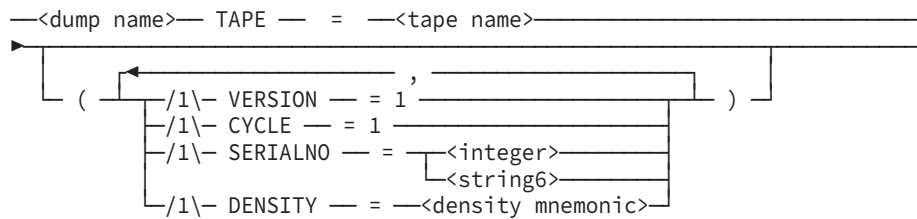
<serial number specification>



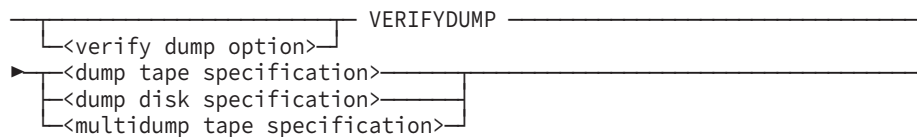
<dump disk specification>



<multidump tape specification>



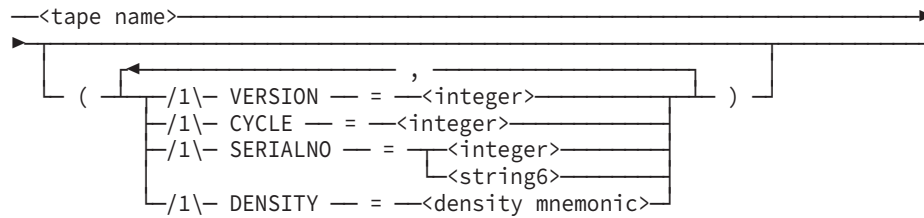
VERIFYDUMP Command



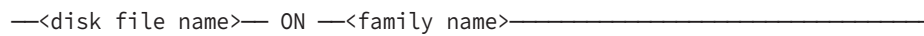
<verify dump option>



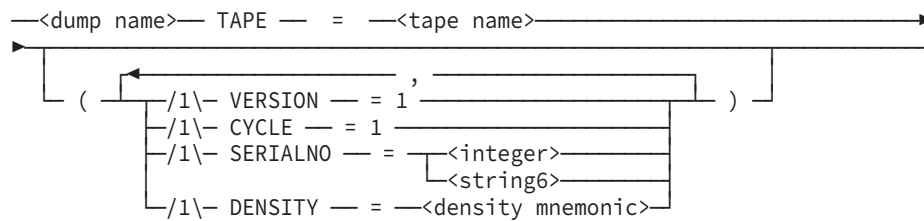
<dump tape specification>



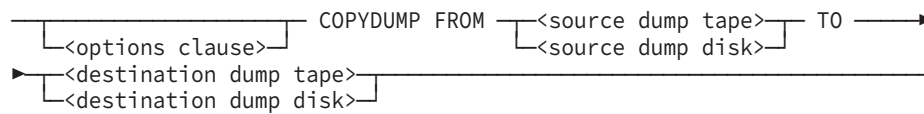
<dump disk specification>



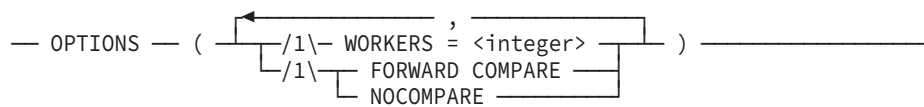
<multidump tape specification>



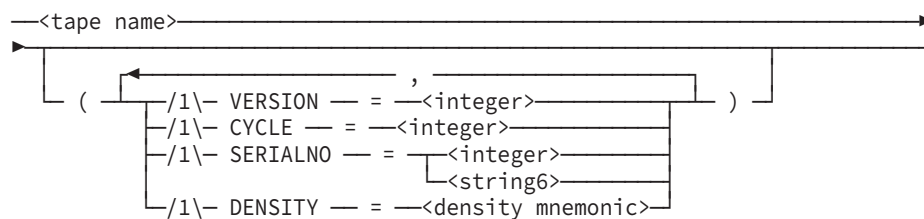
COPYDUMP Command



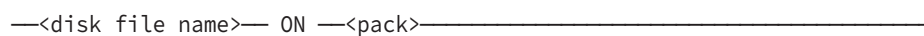
<options clause>



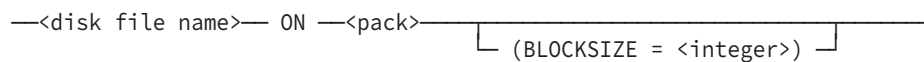
<source dump tape>



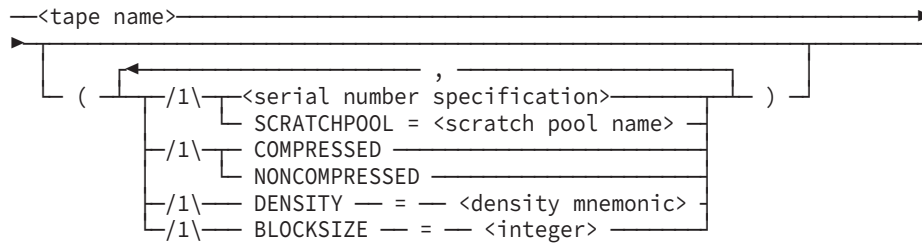
<source dump disk>



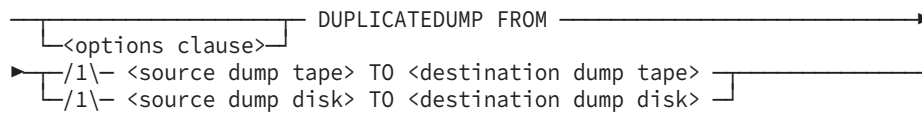
<destination dump disk>



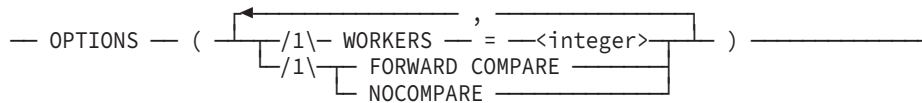
<destination dump tape>



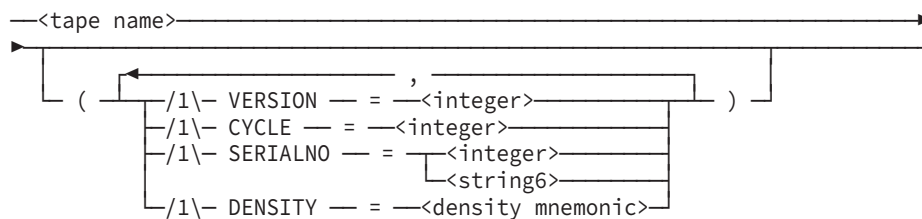
DUPLICATEDUMP Command



<options clause>

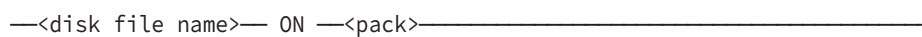


<source dump tape>

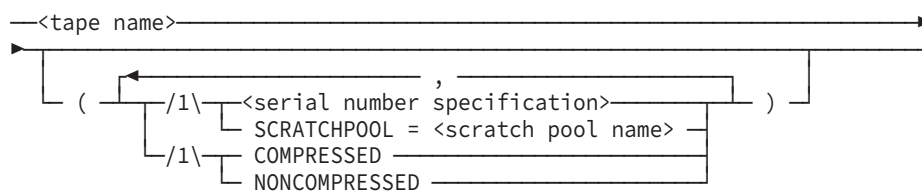


<source dump disk>

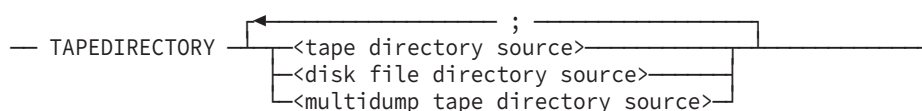
<destination dump disk>



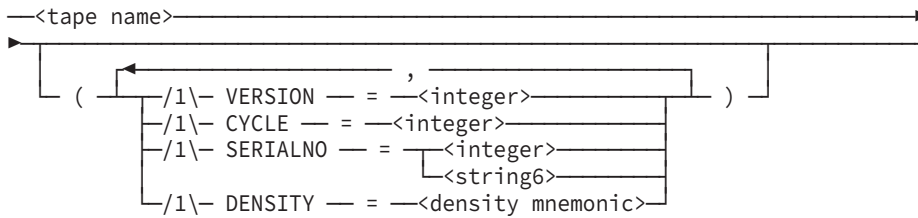
<destination dump tape>



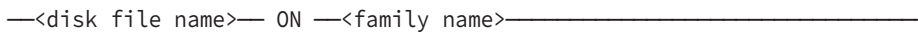
TAPEDIRECTORY Command



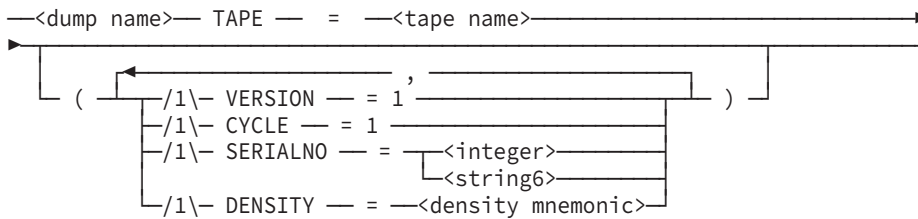
<tape directory source>



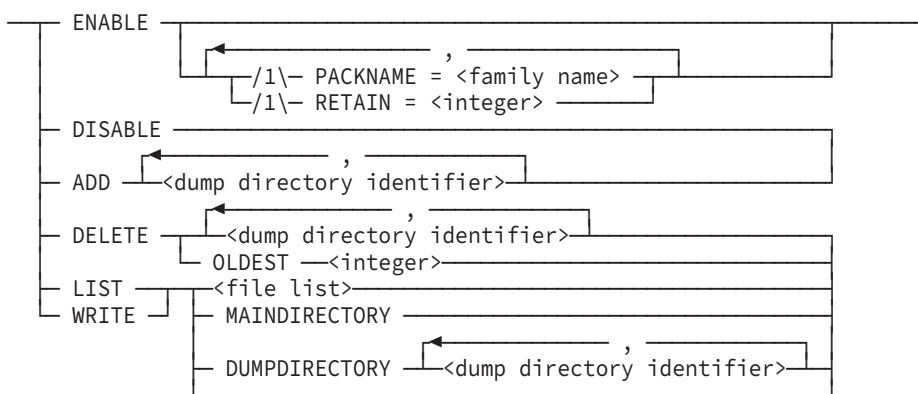
<disk file directory source>



<multidump tape directory source>



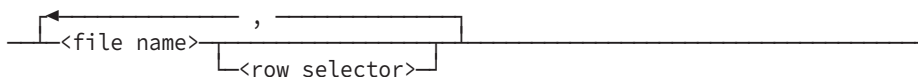
DMDUMPDIR Statement



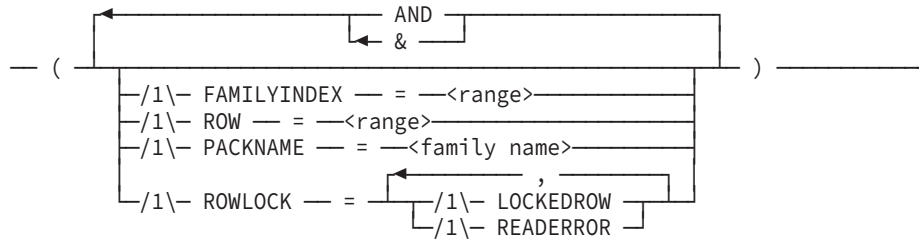
<dump directory identifier>



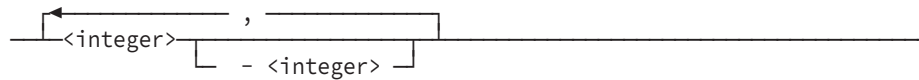
<file list>



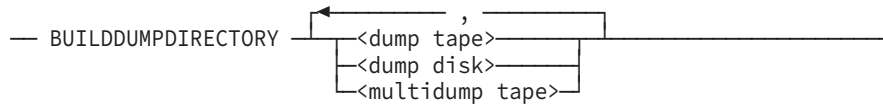
<row selector>



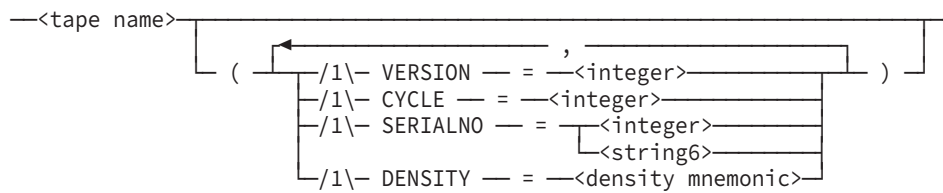
<range>



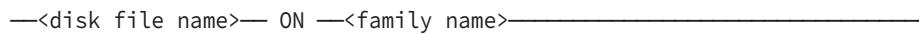
BUILDDUMPDIRECTORY Command



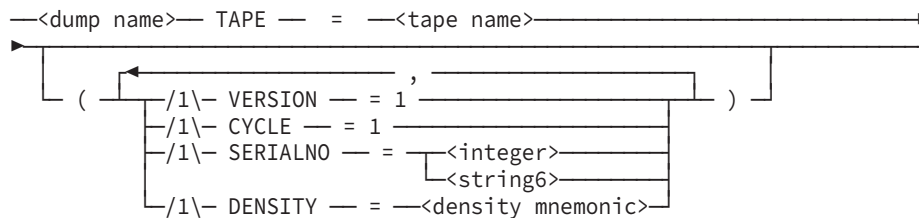
<dump tape>



<dump disk>



<multidump tape>



Section 7

Reorganizing the Database

During the life of the database, you can use the database reorganization process to make physical changes to database files and records.

You can reorganize the database by using the ONLINE, OFFLINE, or REORGDB option. The audited, restartable ONLINE or REORGDB option enables you to access the database during reorganization, while the unaudited, restartable OFFLINE option restricts your access to the database until the entire reorganization process is complete.

Reorganization involves

- The BUILDREORG utility. This utility designates the options used for reorganization and creates the REORGANIZATION program.
- The REORGANIZATION program. This program initiates the database reorganization and provides reorganization support functions.
- The Accessroutines. The Accessroutines does the actual changing of database files and records.

The Accessroutines reorganization tasks provide the following capabilities:

- Reordering data sets according to a specific index set (prime set), if sequential accessing through that set involves the greatest number of applications for the data set
- Consolidating deleted or unused space in data sets, sets, and subsets and returning this space to the system
- Generating sets or automatic subsets if a new method of access is desired for the data set
- Balancing index tables and achieving a uniform coarse/fine table distribution to optimize accesses through sets
- Changing the physical attributes of a structure, such as area size, population, modulus, and load factor
- Changing the record description of a data set or global data record by adding new items, or changing or deleting existing items, within the record
- Changing the KEY, KEY DATA, ASCENDING, DESCENDING, and DUPLICATES clauses for a set or automatic subset; changing the WHERE clause for an automatic subset
- Changing at most, 300 structures in one reorganization at a time

Note: The tasks identified in this section can be initiated through Database Operations Center.

Understanding Types of Reorganization

The reorganization tasks allow changes to the format of an existing database. These tasks can be used to reorder and consolidate data sets, sets, and subsets; to generate new sets or automatic subsets in order to allow more rapid access to data sets; and to change the record formats by adding, deleting, or changing fields. Multiple data sets and their spanning sets and subsets can be reorganized at one time. The exact number of structures allowed in a single reorganization is limited by the size of the code file that is generated. This number varies depending on the specific DASDL changes and the BUILDREORG options used, but generally is in the hundreds. Global data is treated like a data set.

The three types of reorganization available are as follows:

- Garbage collection
- File format conversion
- Record format conversion



You can also perform an online garbage collection of disjoint index sequential sets by using the Visible DBS GARBAGE COLLECT command while the database is running. Initiating this command results in a garbage collection of the specified sets while they are in use. You do not run the REORGANIZATION program. For the syntax for the Visible DBS GARBAGE COLLECT command, refer to [Section 12, Communicating with the Database](#).

Garbage Collection

Garbage collection is an important form of reorganization and should be performed regularly. Garbage collection consolidates deleted or unused space in data sets, sets, and subsets and returns this space to the system. In addition, records in a data set can be physically reordered and index structures can be rebalanced to achieve a uniform coarse/fine table distribution to minimize access time.

Garbage collection does not involve any changes to the database description; therefore, it does not require you to recompile the Data and Structure Definition Language (DASDL), the DMSUPPORT library, or your programs. To determine when to perform a garbage collection, use the dbaTOOLS product as described in the Software Product Catalog.

Note: During the migration phase of the garbage collection for databases that are not permanent directory databases, the title of the DMSUPPORT library temporarily changes to DMSUPPORT/<database name>/<update level>. If the garbage collection terminates while the DMSUPPORT library has its temporary name and you do not restart the garbage collection, you must manually rename the DMSUPPORT library to

DMSUPPORT/<database name>. When reorganizing permanent directory databases, the title is not temporarily changed and always includes the <update level> construct as the last node.

After a reorganization that includes file or record format conversions, the database description file contains both the new and the old reorganization information. In most cases, as long as you do not include the UPDATE option in the BUILDREORG specification the new description file can be used for most garbage collection reorganizations. However, under either of the following circumstances, you must recompile the DASDL source file to produce a new database description file prior to running a garbage collection reorganization:

The ORDER BY option of a GENERATE statement contains a new set or data set that was added during the last DASDL UPDATE compilation.

or



The GENERATE statement contains a previously existing data set that contains a set that was added during the last DASDL UPDATE compile with the USEREORGDB option.

To avoid any confusing errors when you perform a garbage collection reorganization, perform the following steps after a file or record format conversion reorganization. These steps remove the old reorganization information and change the status of newly added structures from *new* to *existing*.

1. Remove any reorganization clauses from the DASDL source file.
2. Perform a DASDL UPDATE compilation to create a new database description file.

File Format Conversion

File format conversion allows changes to database files and does not affect record formats. For example, file format conversion can be used to change the AREASIZE or BLOCKSIZE of a data set, the modulus of an access, or the TABLESIZE of a set or subset. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for DASDL requirements and allowable changes to the database during file format conversion.

File format conversion requires a DASDL update run. This update creates a new description file that defines the changes to be made during the reorganization run.

After error-free DASDL compilation, the DMSUPPORT library and other database software must be recompiled using the new description file. The DMSUPPORT compilation must be completed before the reorganization run is started.

Reorganizing the Database

For a permanent directory database, the DMSUPPORT library is in the same datapath as the database and the title always includes the <update level> construct as the last node. The description file for a permanent directory database is under the same usercode on the same family from which the BUILDREORG utility and the reorganization program are run.

During the reorganization run, the REORGANIZATION program and the description file are copied with the following titles for possible future use during a rebuild through the reorganization region of the audit:

```
REORGANIZATION/<database name>/<YYYYMMDD>/<HHMM>  
DESCRIPTION/<database name>/<YYYYMMDD>/<HHMM>
```

The <YYYYMMDD> construct is the date part of the update timestamp, and the <HHMM> construct is the time part of the update timestamp. To ensure that the correct files are saved, the file titles also include the usercode and the pack name of the actual files being used. Archive these files for possible later use in rebuild operations.

Because file format conversion does not affect record formats, user programs need not be recompiled. Existing user programs can continue to run without the need for reprogramming or recompilation.



When you use XE features, you must recompile applications if you change the value of the SECTIONS option from a nonsectioned data set to a sectioned data set, or from a sectioned data set to a nonsectioned data set.

Record Format Conversion

Record format conversion allows changes to the format of an existing data set, global data, set, or automatic subset record. Using record format conversion, items in the record can be changed, new items can be added, and items can be deleted. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information.

Record format conversion requires a DASDL update run. This update creates a new description file that defines the changes to be made during the reorganization run.

After an error-free DASDL compilation, the DMSUPPORT library and other database software must be recompiled using the new description file. The DMSUPPORT compilation must be completed before the reorganization run is started.

For a permanent directory database, the DMSUPPORT library is in the same datapath as the database and the title always includes the <update level> construct as the last node. The description file for a permanent directory database is under the same usercode on the same family from which SYSTEM/BUILDREORG and the reorganization program are run.

During the reorganization run, the description and reorganization files are copied with the following titles for possible future use during a rebuild through the reorganization region of the audit:

```
REORGANIZATION/<database name>/<YYYYMMDD>/<HHMM>  
DESCRIPTION/<database name>/<YYYYMMDD>/<HHMM>
```

The <YYYYMMDD> construct is the date part of the update timestamp, and the <HHMM> construct is the time part of the update timestamp. To ensure that the correct files are saved, the file titles also include the usercode and the pack name of the actual files being used. Archive these files for possible later use in rebuild operations.

Because record format conversion affects record formats, user programs that invoke the reorganized structure must be recompiled. User programs that do not invoke the reorganized structure continue to run without requiring reprogramming or recompilation. In some cases, the need for recompilation can be circumvented through the use of remaps.



When you use XE features, setting or resetting the EXTENDED attribute for a data set constitutes a record format conversion.

Understanding the Database Reorganization Process

A reorganization run can be performed when physical modifications to database files or records are necessary. The database administrator must decide which database structures require reorganization. You can perform as many reorganization runs as you need to achieve the desired database.

All structures can be reorganized at once in most cases. See “Understanding Types of Reorganization” earlier in this section for further information.

A reorganization run consists of the following steps:

1. For safety purposes, make backup copies of the following:
 - Control file and all of the database files through DMUTILITY DUMP
 - Audit trail
 - DASDL source
 - Database description file
 - DMSUPPORT library
 - RMSUPPORT library (if you are using the Open Distributed Transaction Processing product)

Reorganizing the Database

Depending on the open option specified in the BUILDREORG utility run, the database is available for inquiry only, update, or exclusively opened by the REORGANIZATION program. By default, the database is available for update, and all ONLINE reorganization operations are audited to the Enterprise Database Server audit trail.

Rebuild recovery is allowed through a reorganization. Rollback recovery is not allowed into a reorganization region of the audit. For this reason, a DMUTILITY dump of the database as close as possible to the start of a reorganization is desirable.

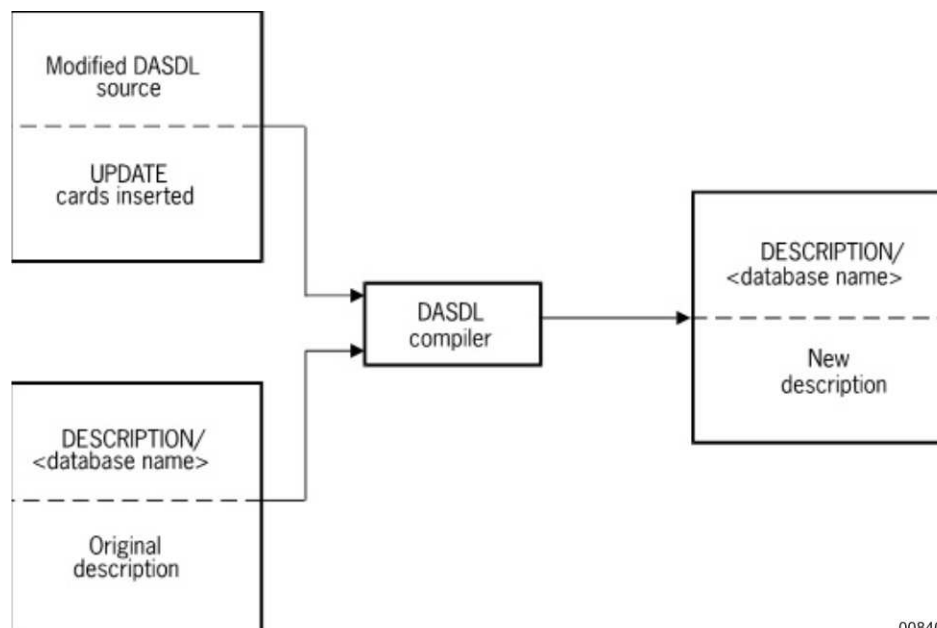
In a reorganization using either the OFFLINE or REORGDB options, the reorganization operations are not audited to the Enterprise Database Server audit trail. Following a reorganization using either the OFFLINE or REORGDB options, the ability to rebuild

- Reorganized structures depends on a post-reorganization dump of those structures
- Structures that were not reorganized depends on the dump taken prior to the reorganization that used either the OFFLINE or REORGDB options

For steps for performing a rebuild recovery through a reorganization, refer to “Rebuild Recoveries and Reorganizations” later in this section.

Note: Step 2 is not required if only garbage collection is to be performed.

2. If a file format or record format conversion is desired, you must perform a DASDL update run. This step must be performed when physical attributes such as AREASIZE, BLOCKSIZE, and TABLESIZE are changed or when record fields are added, deleted, or changed. When the modified DASDL source is compiled, a new description file is produced. This new file is specially marked for reorganization and contains both the old and new formats and an incremented update level. [Figure 7-1](#) illustrates this process.



008401

Figure 7-1. Creation of a New Database Description File

By default, the compiler control options ZIP and DMCONTROL are set. The ZIP compiler control option ensures that, when necessary, the DMSUPPORT library and the other tailored software are recompiled automatically. The DMCONTROL compiler control option ensures that, when necessary, the control file is updated automatically.

If a DMSUPPORT library migration is not needed, the DASDL compiler initiates the SYSTEM/DMCONTROL run. If a DMSUPPORT library migration is needed, the REORGANIZATION program initiates the SYSTEM/DMCONTROL run.

If the DASDL source is compiled for syntax only, DASDL does not compile the tailored software or run SYSTEM/DMCONTROL, even if the ZIP and DMCONTROL options are set.

When file format or record format conversion is desired, the update level of the database is incremented by the DASDL compiler. For permanent directory databases, the DMSUPPORT library has the same datapath as the database and the title always has the <update level> construct as the last node. As a result, the DMSUPPORT titles are not migrated during reorganization. For traditional databases, a new update level is appended to the DMSUPPORT title in preparation for the migration of the database from one update level to the next. For example, if the current DASDL update level is 3 and a DASDL update requiring reorganization has just been performed, the DMSUPPORT titles are as follows:

```
DMSUPPORT/<database name>/1
DMSUPPORT/<database name>/2
DMSUPPORT/<database name>
DMSUPPORT/<database name>/4
```

Next, the reorganization run changes the current DMSUPPORT title to an old title, and the next DMSUPPORT title to the current title. After the reorganization has been initiated, the DMSUPPORT titles are as follows:

```
DMSUPPORT/<database name>/1
DMSUPPORT/<database name>/2
DMSUPPORT/<database name>/3
DMSUPPORT/<database name>
```

Notes:

- *If a reorganization is terminated for any reason and is not restarted, you must*
 - *Manually restore the DMSUPPORT titles to their original titles.*
 - *Remove any DMSUPPORT libraries that cannot be used.*

In the scenario shown, you would perform the following steps:

 - a. *Remove DMSUPPORT/<database name>.*
 - b. *Change DMSUPPORT/<database name>/3 to DMSUPPORT/<database name>.*
- *If the DMSUPPORT title is not changed properly for some reason, such as a security violation, the reorganization waits for an AX (Accept) command and displays a message that the title must be changed manually.*
- *The DMSUPPORT code files for a permanent directory database reside in the permanent directory and their titles include the update level. As a result, the DMSUPPORT titles are not migrated during a reorganization.*

Reorganizing the Database

It is the responsibility of the database administrator to save old copies of the DMSUPPORT library and other tailored software in the event of rebuild recoveries through a reorganization region. Also, if the compiler control option ZIP is reset in the DASDL update, it is the responsibility of the database administrator to compile the new DMSUPPORT library with the appropriate title. Whenever a reorganization is required, the DASDL compiler displays the following information:

```
REORGANIZATION REQUIRED
NEW UPDATE LEVEL: <update level>
```

The database administrator must compile the DMSUPPORT library with the following title for traditional databases:

```
DMSUPPORT/<database name>/<update level>
```

For permanent directory databases, the DMSUPPORT library must be

```
<datapath>/DMSUPPORT/<database name>/<update level>
```

Alternately, if the DASDL compiler is initiated from a Work Flow Language (WFL) job, the TASKVALUE of the WFL job is set to the update level whenever a reorganization is required. The update level can be incremented even though a reorganization is not required, for example, when only a data set is added during a DASDL update. In this case, the TASKVALUE of the WFL job is zero. If the DASDL update does not require a reorganization, the TASKVALUE is zero and should not be appended to the DMSUPPORT title. For example:

```
COMPILE <database name> WITH DASDL;
IF MYJOB (VALUE) = 0 THEN
  COMPILE DMSUPPORT/<database name> WITH DMALGOL
ELSE
BEGIN
  S:= STRING (MYJOB (VALUE), *);
  COMPILE DMSUPPORT/<database name>/#S WITH DMALGOL;
END;
```

It should be noted that if the compiler control option ZIP is set in DASDL, all of these DMSUPPORT title assignments are handled automatically. The database administrator only need worry about backing up old DMSUPPORT code files and making them available in the event of future rebuild recoveries.

The DMSUPPORT title specification in the DASDL source is always without update level appended. That is, when a DASDL update is performed, the title should not be changed in the DASDL source to reflect the new update level.

Since the current DMSUPPORT title is always the default title (without update level appended), callers are unaffected by the migration; they should always call the DMSUPPORT library using the default title.

Both user programs and database software can be compiled against the new description file. However, user programs cannot run until the reorganization of the database has started. Old user programs can run with the old DMSUPPORT library until execution of the REORGANIZATION program. If the database is open at the time the REORGANIZATION program is executed, all current users must close the database before the reorganization can start. Once the reorganization has started, user programs can reopen the database. Those user programs unaffected by the new DASDL description can be rerun. A version of the program compiled with the new description file must be run if the program is affected by the DASDL update.

With the USEREORGDB option, there is no need to close the database before starting the reorganization. User programs compiled against the new description file should not be run until the entire reorganization process is completed.

You cannot use the BUILDREORG utility with the USEREORGB option when you are performing any form of reorganization or garbage collection for a modeled database. When you use the MODEL option in the DASDL specifications to create a modeled database, the DMSUPPORT library is shared between the production and the modeled databases. As a result, the control file for both the production and the modeled databases contains the same DMSUPPORT library title.

For the REORGDB reorganization process to function correctly, the DMSUPPORT library that belongs to the database being reorganized must be unique. However, this requirement is contrary to the modeling process.

Because there is no way to determine whether the DMSUPPORT library titles are unique to each database, BUILDREORG produces a syntax error whenever you use the USEREORGDB option for a modeled database.

However, you can perform a USEREORGDB reorganization on a modeled database by reusing the reorganization program compiled for the parent database in combination with the database equation.

In this instance, the internal database DB of the reorganization program is equated to the modeled database. In addition, if the USEREORGDB reorganization is an UPDATE reorganization, the file DESCRIPTION/<model db name>/<previous level> must be present. This requirement for an UPDATE reorganization ensures that the reorganization program cannot be run for the model unless the DASDL UPDATE compilation of the model has occurred. An example of the database equation statement is

```
RUN REORGANIZATION/<parent db>;  
    DATABASE DB (TITLE = <model database name>);
```

3. Run the BUILDREORG utility using the new description file and user-supplied reorganization specifications. [Figure 7-2](#) illustrates this process.

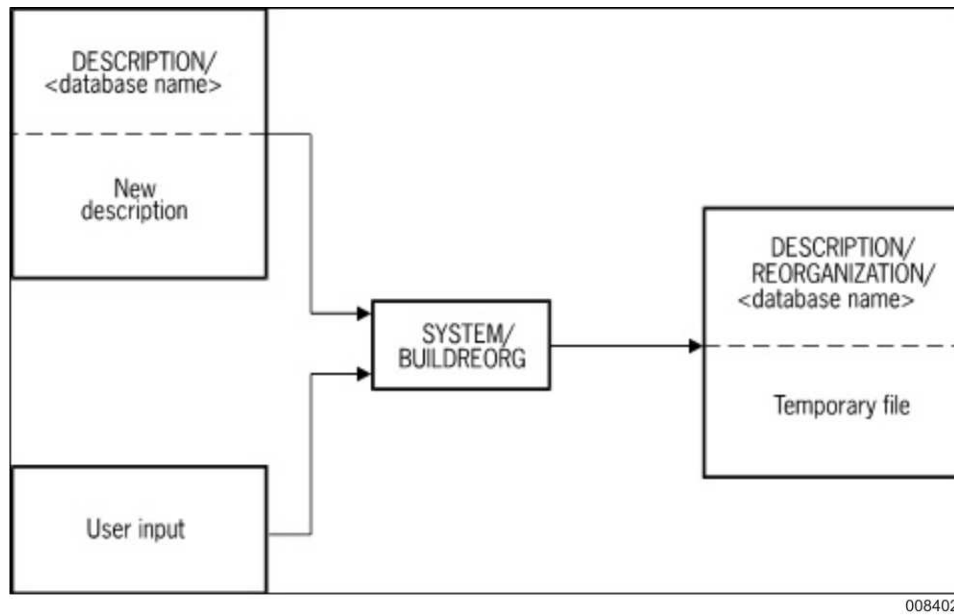


Figure 7-2. Creation of a Reorganization Description File (Scenario 1)

If file or record format conversion is required, then include the UPDATE option in the BUILDREORG specifications. If format changes are not required (garbage collection), the UPDATE option is not required in the BUILDREORG specifications. In addition, if format changes are not required and a DASDL update run was not performed, the original description file is used as input to the BUILDREORG utility. [Figure 7-3](#) illustrates this process.

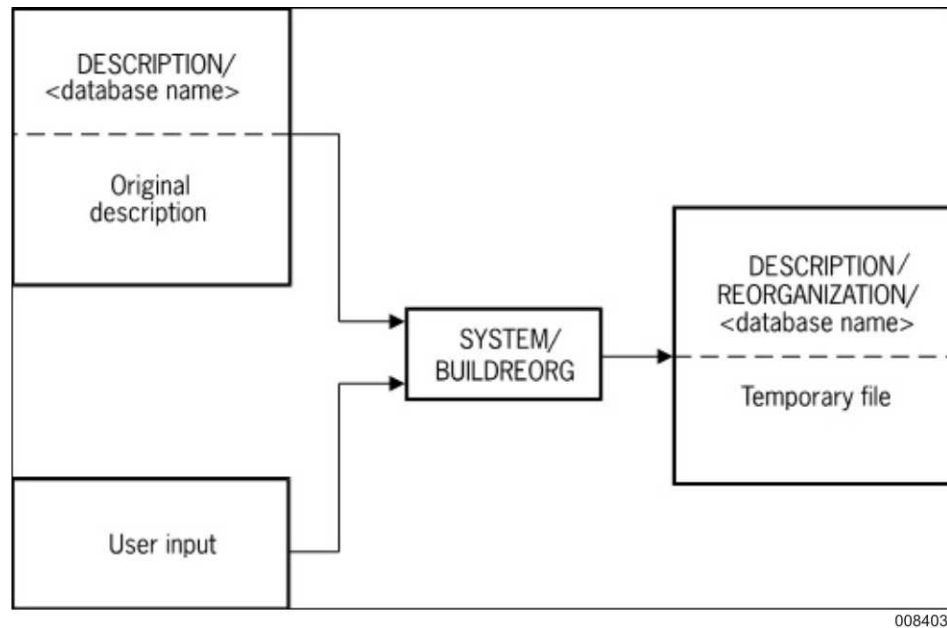


Figure 7-3. Creation of a Reorganization Description File (Scenario 2)

The specifications to the BUILDREORG utility identify structures requiring reorganization, describe how certain reorganizations are to be performed, and specify available system resources.

If no errors are produced, the BUILDREORG utility does the following:

- Creates the reorganization description file titled DESCRIPTION/REORGANIZATION/<database name> which contains the database description and the specifications for reorganization. The reorganization description file is created only for the purpose of compiling the REORGANIZATION program.
- Generates a report that shows both the user and default reorganization specifications. All structures affected by reorganization are noted on this report.
- Automatically initiates compilation of the REORGANIZATION program unless explicitly overridden.

The REORGANIZATION program, titled REORGANIZATION/<database name> unless a different name is supplied in the DASDL source file, is compiled with DMALGOL using DATABASE/REORGSYMBOLIC and the reorganization description file created by the BUILDREORG utility. [Figure 7-4](#) illustrates this process. Because the REORGANIZATION program invokes the database, the description file must be present for the compilation.

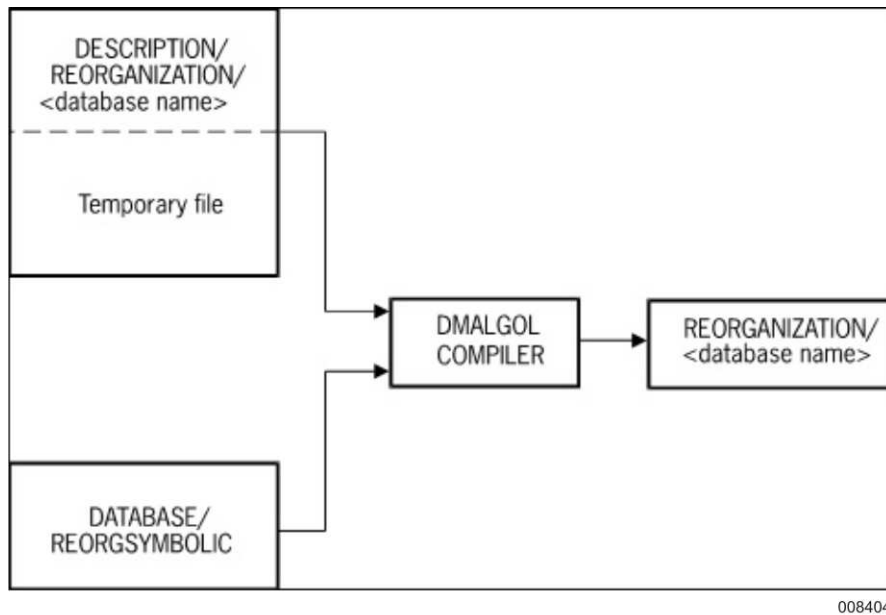


Figure 7-4. Creation of a REORGANIZATION Program

4. If file or record conversions were done, the DMSUPPORT library and other database software must be recompiled to run on the reorganized database. In addition, user programs that directly invoke converted structures must be recompiled.

If only file format conversion was done, user programs need not be recompiled. In addition, programs using other structures not affected by reorganization of the database need not be recompiled. The DMSUPPORT library and the RMSUPPORT library (if the Open Distributed Transaction Processing product is in use) must be recompiled before the REORGANIZATION program is run.

5. The REORGANIZATION program is run to perform the actual reorganization of the database. For detailed information on these phases, refer to "Understanding the Phases of Reorganization" later in this section.

Once the REORGANIZATION program, the new DMSUPPORT library, and the RMSUPPORT library (if the Open Distributed Transaction Processing product is in use) have been recompiled, the reorganization can be initiated. It is initiated by running the REORGANIZATION program. If the database is not open at the time, the REORGANIZATION program opens the database. If the database is open at the time, the database stack immediately marks itself as in a reorganization migration state. User programs are not allowed to open the database and the migration cannot proceed until current users of the database have closed the database. Also, all database files that are unaffected by the reorganization are kept open to minimize the migration time. Once the current users have closed the database, the migration continues. The migration should take a very short period of time.

The steps in the migration are as follows:

- a. The description file and the reorganization code file are copied using titles that include the date and time:

```
REORGANIZATION/<database name>/<YYYYMMDD>/<HHMM>  
DESCRIPTION/<database name>/<YYYYMMDD>/<HHMM>
```

These files should be saved along with the current DMSUPPORT library, which is discussed later in this section, in case they are needed in the future for rebuild recoveries.

- b. The database stack delinks from the DMSUPPORT library. For traditional databases, the stack migrates the current DMSUPPORT title to an old title and the next DMSUPPORT title to the current title.

This step is not required for the successful completion of the reorganization. However, if this step is not performed, problems can arise later. Therefore, if an error occurs that prevents the files from being properly handled by the reorganization, the reorganization waits and a message appears. When the waiting entry appears in the mix, you must respond with an AX (Accept) command. The reorganization then continues even if the problem with the files is not corrected. The titles must be manually corrected.

- c. The database stack links to the new DMSUPPORT library, and depending on the open option specified in the BUILDREORG specification, the user programs are allowed to reopen the database. The database becomes available to users on a structure-by-structure basis as described in "Running the REORGANIZATION Program" later in this section.

You can specify several open options in the BUILDREORG utility run. By default, all ONLINE reorganization operations are audited to the Enterprise Database Server audit trail and user programs are allowed to update the database while the reorganization is in progress. Using the OFFLINE option restricts your access to the database until the entire reorganization is complete. The REORGANIZATION program can be restarted if the system fails. The REORGANIZATION program is automatically reinitiated after a halt/load recovery.

The open options are described in detail under "Running the BUILDREORG Utility" later in this section. However, they are discussed briefly as follows:

- INQUIRYONLY
User programs are allowed to open the database during the reorganization, but only those that open the database INQUIRY.
- EXCLUSIVE
No user programs are allowed to open the database during the reorganization.
- PREVERIFY
All data sets to be generated have had their structures locked by a LOCK STRUCTURE operation, and the DASDL verify condition is applied to all records. If any records fail the verify condition, an error report is produced and the reorganization does not proceed.
- OFFLINE
If the OFFLINE option is used during a reorganization, you cannot access the structures being reorganized until the entire reorganization is complete. However,

structures that are not being reorganized can be accessed by user programs unless additional open options restricting access (for example, INQUIRYONLY or EXCLUSIVE) were included in the BUILDREORG specifications.

The audit overhead during a reorganization that uses the OFFLINE option is minimal. The audit includes control records showing state changes, but there is no record-level auditing during a reorganization using the OFFLINE option. In most cases, an unaudited reorganization using the OFFLINE option runs faster than an ONLINE, audited reorganization, and a reorganization using the OFFLINE option can be restarted at the structure level after a halt/load.

While a reorganization, in general, is faster using the OFFLINE option, the REORG SUPPORT library uses more SAVE memory, and the SAVE memory increases with each generate or fixup task. If you are reorganizing a large number of structures using the OFFLINE option, and your system is low on available memory, then your system might run out of memory or issue the SORT ERROR #24 error message or both. If the reorganization is abnormally terminated, the accumulated SAVE memory is freed. You can restart the reorganization and it should complete normally. For additional information, refer to "SORT Errors During Reorganization" later in this section.

Unless the USEREORGDB option is used in conjunction with the COPY TO option, the REORGANIZATION program always creates generated files on their final medium, that is, the pack specified in the latest DASDL update. The old file is either found on the same pack or the pack specified in the prior DASDL update, if the pack specification was changed.

In situations where two copies of the file are required by the reorganization and there is not enough space on the final pack for both copies of the file, you can use the BUILDREORG COPY option to copy the old file to a temporary pack location. For a permanent directory database, the appropriate datapath must exist at the temporary pack location so that the file can be copied. If you use the BUILDREORG COPY option, the REORGANIZATION program performs the following steps:

- a. Copies the old file to the specified pack
- b. Creates the new file on the final pack
- c. Proceeds with the required structure generation
- d. Once the structure generation is complete, removes the old file from the pack specified in the COPY option statement

The following three files must be saved after the reorganization run for future use in the event a rebuild recovery through the reorganization is needed:

```
DMSUPPORT/<database name>/<prior update level>  
DESCRIPTION/<database name>/<run date>/<run time>  
REORGANIZATION/<database name>/<run date>/<run time>
```

Use a DMUTILITY dump taken before the reorganization for rebuild recovery purposes. The rebuild recovery process repeats the reorganization. During the rebuild, the DMCONTROL utility is run again, which requires the description file used by the reorganization. In addition, the DMSUPPORT libraries and the REORGANIZATION program itself are required. To preserve unique versions of these files, they are copied by the REORGANIZATION program with the titles shown. The run date has the format YYYYMMDD, and the run time has the format HHMM.

6. Once the reorganization completes, make a backup copy of the database. The backup copy should include
 - A DMUTILITY dump of the database
 - Copies of the DMSUPPORT library and the description file from before and after the reorganization
 - A copy of the reorganization program
 - Copies of any tailored software

Creating this backup copy of the database avoids unnecessary overhead if a rebuild recovery is required at a later date.

Rebuild recoveries are allowed from a DMUTILITY dump created prior to the reorganization. If you use a dump from before a reorganization, the reorganization is essentially rerun as part of the rebuild recovery process. Refer to “Rebuild Recoveries and Reorganizations” later in this section for more information.

If you specify the UPDATE option when you use the BUILDREORG utility, the reorganization can only be rerun in the event a rebuild recovery is needed. Rebuild recovery uses the copy of the REORGANIZATION program with the reorganization timestamp incorporated in the program title. You should therefore remove the copy of the REORGANIZATION program titled REORGANIZATION/<database name>.

If you do not specify the UPDATE option when you use the BUILDREORG utility, you can save and rerun the REORGANIZATION program to perform the same reorganization. The REORGANIZATION program can be reused until the next DASDL update occurs. At that time, the REORGANIZATION program must be respecified and recompiled.

If the reorganization process fails and cannot be restarted successfully, the database files, control file, description file, and all tailored software, must be reloaded from backup dumps. Rebuild recovery through a reorganization is allowed, which might resolve pack problems, but if the problem is a reorganization logic problem, the restarted reorganization might also fail.

Note: *If the initiated reorganization task cannot open the database because another task has locked the control file, the initiated reorganization task aborts. This action is not a failure of the reorganization process since the REORGANIZATION program never actually starts. Wait until the task in progress completes and frees the control file before initiating the REORGANIZATION program.*

Understanding the Reorganization Algorithm

ONLINE Reorganization

The ONLINE reorganization algorithm employs a touch mechanism. That is, when a record that requires reorganization is touched by either an online user or by one of the background tasks, it is reorganized at that point and presented to the requester in the new format. The reorganization of an entire structure, therefore, appears to be instantaneous. All records, as far as user programs are concerned, are read in the new format whether they have been physically reformatted or not.

As records are physically reformatted, they are moved from the old file to the new file, and the fixup file is updated to reflect this move. This prevents a record from being reformatted twice. After all records have been reformatted, the fixup file is used to update index sets and records containing links to the reformatted records.

The background reorganization tasks touch records in sequential order while online users can touch records in random order. The sequential order of access by the background reorganization tasks might be the physical order in the old file, or it might be in the key order of a set, depending on whether or not an ORDER BY specification was included in the BUILDREORG specifications or a prime set was indicated in DASDL.

The BUILDREORG utility determines which reorganization tasks are required and the order in which they are processed. Each task is assigned a single structure to either generate or fixup. When the task is activated, it builds a database declaration including only the affected structure and opens the database. It then proceeds to access the structure in the BUILDREORG designated order. In the case of an ORDER BY generation of a structure, this can be as simple as doing a FIND NEXT as any user program would.

When the structure being reorganized is a standard fixed-format data set with REBLOCK set, deleted records might exist in the structure after the structure is reorganized. These deleted records are unused space in the last big block of the structure and cause the creation of a DKTABLE. Under certain circumstances, these deleted records can result in the data set being larger after a reorganization.

OFFLINE Reorganization

Database structures reorganized through the OFFLINE option cannot be accessed by the user programs until the entire reorganization is complete.

The phases and operational procedures for the OFFLINE option are similar to those of the ONLINE reorganization. Although the reorganization does not generate audit images for every record that is reorganized through the OFFLINE option, it does generate certain audit control records indicating state changes.

A reorganization that uses the OFFLINE option can be restarted. If the reorganization is restarted, it restarts at the structure level.

While the reorganization, in general, is faster using the OFFLINE option, the REORGSUPPORT library uses more SAVE memory, and the SAVE memory increases with each generate or fixup task. If you are reorganizing a large number of structures using the OFFLINE option, and your system is low on available memory, then your system might run out of memory or issue the SORT ERROR #24 error message or both. If the reorganization is abnormally terminated, the accumulated SAVE memory is freed. You can restart the reorganization and it should complete normally. For additional information, refer to "SORT Errors During Reorganization" later in this section.

REORGDB Reorganization



The REORGDB mode of reorganization provides essentially uninterrupted user access to production database structures while a background reorganization is in progress. This process is initiated through the USEREORGDB option, which is only available for XE structures or those structures that are migrating to the XE features.

The background reorganization takes place on a copy of the database. For a permanent directory database, the copy of the database resides in the same permanent directory as the production database. Updates to the production database are captured, and after a successful reorganization, the captured updates are applied to the reorganized database copy. When the reorganization and synchronization completes, the database administrator can swap the reorganized database copy with the production database. The AUTOSWAP option setting determines whether the swap occurs automatically or under manual control.

Caution

Reorganizing a whole database with the USEREORGDB option might require a significant amount of system resources and could adversely affect overall system performance.

Unless otherwise specified, the system automatically balances the performance ratio between application programs and update tasks by starting with a minimum of five update tasks plus one driver task for each structure that is either explicitly or implicitly reorganized. If the number of update users is greater than five, the number of update tasks is higher—up to a maximum of 50. For cases involving embedding and links, the number of update tasks is one rather than five or more.

To manually control the number of update tasks, use the MAXUPDATERS phrase of the <reorgdb control> specification.

An overview of the REORGDB process follows:

1. Affected data structures are copied to the database copy. At the end of the copy process, the QUIESCE procedure is used to briefly pause the production database while all of the copied files are synchronized. This operation is transparent to the user programs.

Note: The database copy is allowed to reside on a different set of disk packs than the production database. This is accomplished with a COPY TO option.

Reorganizing the Database

You can specify whether the reorganized data structures remain at the new location or are copied back to the original packs during a swap. If a corresponding COPY BACK option does not accompany a COPY TO option, the result is similar to performing a DMCONTROL family change.

2. The database copy is reorganized using the OFFLINE option.

Updates to the production database are captured to a flat file known as the capture file. These updates are then applied to the database copy.

3. When all updates have been applied, the production database structures involved in the reorganization are available to be swapped with the corresponding database copy structures.

The swap availability point occurs when the update application process reaches the first EOF point on the capture files.

4. Either automatically or on command, the reorganized structures in the database copy are swapped to the production database.

Updates to the production database continue to be captured and are applied to the database copy until the swap occurs. When the swap starts, Accessroutines first suspends all user programs in their code, using the production database stack, and there is a brief pause while synchronization takes place.

After the swap completes, the user programs are allowed to reenter the Accessroutines code and all paths are restored. No program coding changes are necessary to use the REORGDB feature.

Unless previously recompiled against the new description file, programs will receive version errors when accessing a structure whose format timestamp has changed due to reorganization. If the reorganization specifying the USEREORGDB option is a garbage collection, programs do not receive version errors. User programs cannot access the database copy.

Programs using any new structures can be initiated as soon as the swap process completes. Programs initiated during the swap receive a DMOPENERROR 72 error. The database administrator initiates the swap through either a graphical or command-line interface.

The Database Operations Center provides the graphical user interface for a REORGDB reorganization so that the database administrator can begin the USEREORGDB process and swap in the newly reorganized database copy when it is ready and synchronized.

Notes:

- *After a REORGDB reorganization of a database running the Remote Database Backup feature, you must perform a structure clone operation on the affected data structures.*
- *When the USEREORGDB option is specified for a reorganization, both the updated and previous description files must be available. Both the newly updated description file, DESCRIPTION/<database name>, and the backup copy created by DASDL, DESCRIPTION/<database name>/<previous level>, must be present at the beginning of the reorganization process.*

REORGDB Memory Usage

In addition to the ALLOWEDCORE and REORGALLOWEDCORE settings for the two databases, the following numbers can be used as a rough guide for estimating memory requirements for each data set.

The actual values for your reorganization will be different.

Catch-up buffers for apply updates (CAUDIT) CAUDIT buffers (two for read operations and two for write operations) are owned by the production database stack and are not counted toward the database ALLOWEDCORE limit. Additional REORGDB items that use memory on the database stack are also not counted toward the ALLOWEDCORE total.	36K * number of data sets
REORGANIZATION program	x * number of data sets
REORGSUPPORT library	x * number of data sets
Apply updates control program (UPDATEP)	8K * number of data sets
Apply updates worker program (UPDATER)	(7K * number of data sets) * maximum number of user programs or 5

Note: In the preceding table, the variable x is the memory used by the REORGSUPPORT library and the REORGANIZATION programs, depending upon the BUILDREORG ALLOWEDCORE setting and the populations of the structures being reorganized. For additional information, refer to "Using the Central Data Set Control Options," "Using the ALLOWEDCORE Phrase," and "Using the REORGDBALLOWEDCORE Option" later in this section.

For standard OFFLINE option reorganizations, task limits determine how many central data set sequence tasks can run at once and how many tasks, such as fixups, can run at one time within a central data set sequence. A standard offline reorganization is serial, that is, it sequentially runs central data set sequence operations based on TASKLIMIT values (each getting its own share of the ALLOWEDCORE total). Within a central data set sequence, the TASKLIMIT values control the number of fixup tasks that can run together. Each sequential phase deallocates its memory when each central data set task completes.

For the REORGDB reorganization, the task limits only apply to the OFFLINE reorganization phase of the REORGDB process. As a result, during the apply updates phase of the REORGDB reorganization, the ALLOWEDCORE value applies to each data set and is a minimum of 2 million words each. Unlike standard OFFLINE reorganizations, memory cannot be released until the swap even though the offline reorganization phase has completed. In particular, the fixup information used to map the record addresses as updates are applied to the newly reorganized, or garbage collected, data structures from the transactions in the CAUDIT files. Therefore, whatever memory is brought into the reorganization is accumulated.

The fixup file buffers comprise most of the ALLOWEDCORE memory and none of the memory can be deallocated at the end of the offline reorganization phase. The memory remains allocated until the swap, even after all data set offline reorganization phases have completed. This is because all apply updates are still active until that point. Standard offline reorganizations can start deallocating fixup file buffers and ALLOWEDCORE memory as soon as each data set and set fixup operation finishes. The REORGDB reorganization cannot deallocate anything until all of the apply updates tasks have completely finished and the swap starts.

The REORGDB reorganizations can be memory intensive. The alternatives are to either reduce very large ALLOWEDCORE values or specify a smaller quantity of structures to be garbage collected or reorganized. Garbage collection is well suited for subdivision because there is no concurrent requirement to recompile applications. It is likely that a number of garbage collections run successively might run faster than one large garbage collection of many structures because each garbage collection could specify more ALLOWEDCORE memory without concern for memory buildup.

This is a reorganization example that cannot be restarted due to the NORESTART specification:

```
GLOBAL
TASKLIMIT = 3;
INTERNAL FILES (FAMILYNAME=MYWORKPACK);
ALLOWEDCORE; = 1500000
USEREORGDB;
    REORGDBTITLE           = REORGPACEDB;
    AUTOSWAP               = FALSE;
    NORESTART;
    TOTALCOPYCORE         = 2000000;
    REORGDALLOWEDCORE     = 1000000;
```

Running the BUILDREORG Utility

The purpose of the BUILDREORG utility is to specify and control the reorganization process. This specification is used to designate the databases to be reorganized, the intermediate media to use, and various other options.

The BUILDREORG utility uses the database description file and user card input to create the reorganization description file titled

```
DESCRIPTION/REORGANIZATION/<databasename>
```

The reorganization description file is used by the DMALGOL compiler and DATABASE/REORGSYMBOLIC to generate a tailored REORGANIZATION program. In addition, the files DATABASE/PROPERTIES and DATABASE/DMCONTROL must be present on disk during the BUILDREORG run. This compilation produces the REORGANIZATION program, which is tailored to the new database description. If file or record format conversion is performed, reorganization is suitable for only one run and should be removed after a successful run. However, in case a rebuild recovery is required through the reorganization period, keep the copy of the REORGANIZATION program that has the reorganization timestamp incorporated in the program title.

The actual reorganization is accomplished through procedures in the Access routines in conjunction with the tailored REORGANIZATION program. For this reason, it is important that the REORGANIZATION program is generated and run on the same level of Enterprise Database Server software.

The BUILDREORG utility reads your input in free format, using the first 72 columns of each card image.

If the BUILDREORG utility detects an error, the utility assigns the value TERMINATED to the STATUS task attribute and the value 1 to the TASKVALUE task attribute. If there are no errors, the STATUS task attribute has the default value of COMPLETEDOK, and the TASKVALUE task attribute has the default value of 0 (zero).

If the BUILDREORG utility input is error free and the ZIP option was not reset, the REORGANIZATION program is compiled automatically. The BUILDREORG utility has a structure limit of 300 structures which includes generated structures and structures requiring fix-ups. A warning message appears when BUILDREORG detects a reorganization of 200 structures. A syntax error occurs when BUILDREORG detects a reorganization of 300 or more structures. When a very large number of structures are being reorganized at the same time, it is possible for the program to get compilation errors because of its size.

The following errors can sometimes be circumvented:

```
TOO MANY STACK CELLS AT THIS LEVEL
USE THE SEGDESCABOVE OPTION TO AVOID THIS PROBLEM

THE CODE FILE HAS BECOME TOO LARGE
```

To circumvent the first error, compile the program with the SEGDESCABOVE option set to a value between 4 and 4095. It is recommended you have the SEGDESCABOVE value greater than 3000. The second error can be avoided by setting the NOBINDINFO option. If you have set both options, the resulting reorganization program might cause a stack overflow when it is run.

If you have set both options and an error still occurs, or a stack overflow occurs at run time, then you must reduce the number of structures being reorganized. This process might require two separate reorganizations to make all the desired changes. Refer to [Running Through a Batch Job](#) for information on compiling the REORGANIZATION program.

For a permanent directory database, the description file, reorganization description file, and the reorganization program must be under the usercode of the initiator and not in the datapath of the database being modified.

Running Through a Batch Job

You can run the BUILDREORG utility through a WFL batch job. The job must file-equate the database description file. In addition, the files DATABASE/PROPERTIES and DATABASE/DMCONTROL must be on disk.

Reorganizing the Database

The following compiler control options are initially assigned the value SET in the BUILDREORG utility:

- LIST option

When the LIST option is set, the BUILDREORG utility produces a report on the user and default specifications in the REORGANIZATION program. The report also lists all structures to be affected by the REORGANIZATION program.

- ZIP option

When the ZIP option is set, compilation of the REORGANIZATION program—REORGANIZATION/<database name>—begins automatically.

When the ZIP option is reset, you must manually compile the REORGANIZATION program.

- RESTARTSORT option

When using the SORT specification during the generation of sets or data sets, you can encounter SORT ERROR #24. This error occurs when the number of words specified for memory exceeds the memory available on the system. When this error occurs, you need to manually adjust the ALLOWEDCORE value. To do this, rerun the BUILDREORG task with the compiler control option RESTARTSORT. When this option is set, you can then enter a new ALLOWEDCORE value and the SORT is restarted with the new ALLOWEDCORE value. This option only applies to generate tasks, and has no effect on fixup tasks. For additional information, refer to "SORT Errors During Reorganization," later in this section.

You can use the following WFL job statements to run the BUILDREORG utility.

The syntax for the reorganization specification is provided under "Syntax for the BUILDREORG Utility" later in this section.

```
?BEGIN JOB COMPILEBUILDREORG;
?TASK T;
?RUN SYSTEM/BUILDREORG[T];
?FILE DASDL =DESCRIPTION/<database name>
?DATA CARD
$LIST ZIP RESTARTSORT
.
.<reorganization specification>
.
?IF T ISNT COMPILEDOK THEN
  BEGIN
    DISPLAY ("BUILDREORG detected syntax errors");
  END;
?IF T ISNT COMPLETEDOK THEN
  BEGIN
    DISPLAY ("BUILDREORG aborted");
  END;
?END
```

If the ZIP option is set, the following WFL job is automatically initiated:

```
? BEGIN JOB COMPILEREORG;
  COMPILER REORGANIZATION/<database name>
  WITH DMALGOL LIBRARY
  COMPILER FILE TAPE = DATABASE/REORGSYMBOLIC;
  COMPILER FILE DASDL =
  DESCRIPTION/REORGANIZATION/<database name>
  COMPILER FILE PROPERTIES = DATABASE/PROPERTIES;
  COMPILER DATA CARD
$ MERGE
? END JOB
```

When a large number of structures are reorganized at the same time, the compilation of the reorganization might fail with the following error:

```
TOO MANY STACK CELLS AT THIS LEVEL
USE THE SEGDESCABOVE OPTION TO AVOID THIS PROBLEM
```

To correct this error, compile the REORGANIZATION program by adding the ALGOL compiler control option \$SET SEGDESCABOVE to a value between 4 and 4095 after the \$MERGE command in the previously listed WFL job. The WFL job with this added option appears as follows:

```
? BEGIN JOB COMPILEREORG;
  COMPILER REORGANIZATION/<database name>
  WITH DMALGOL LIBRARY
  COMPILER FILE TAPE = DATABASE/REORGSYMBOLIC;
  COMPILER FILE DASDL =
  DESCRIPTION/REORGANIZATION/<database name>
  COMPILER FILE PROPERTIES = DATABASE/PROPERTIES;
  COMPILER DATA CARD
$ MERGE
$ SET SEGDESCABOVE 4095
? END JOB
```

If the following error occurs, it might be possible to correct the problem by adding \$SET NOBINDINFO in place of or along with the SEGDESCABOVE option, depending on the syntax errors that have occurred:

```
THE CODE FILE HAS BECOME TOO LARGE
```

The resulting WFL job would appear as follows:

```
? BEGIN JOB COMPILEREORG;
  COMPILER REORGANIZATION/<database name>
  WITH DMALGOL LIBRARY
  COMPILER FILE TAPE = DATABASE/REORGSYMBOLIC;
  COMPILER FILE DASDL =
  DESCRIPTION/REORGANIZATION/<database name>
  COMPILER FILE PROPERTIES = DATABASE/PROPERTIES;
  COMPILER DATA CARD
$ MERGE
$ SET SEGDESCABOVE 4095
$ SET NOBINDINFO
? END JOB
```

Use the NOBINDINFO and SEGDESCABOVE compiler control options only if they are required because of the aforementioned syntax errors. If syntax errors still occur after setting both, or if a stack overflow occurs when the reorganization program is run, then you must reduce the number of structures being reorganized.

For a permanent directory database, the description file, reorganization description file, and reorganization program must be under the usercode of the initiator and not in the datapath of the database being modified.

Syntax for the BUILDREORG Utility

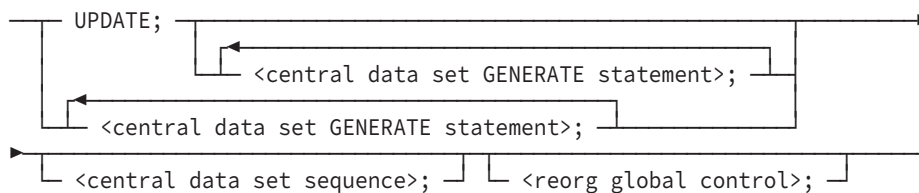
You can use the BUILDREORG utility to perform the following tasks:

- Generate database structures with the UPDATE option.
- Generate database structures implicitly.
- Generate database structures explicitly.
- Exercise global control to designate where the REORGANIZATION program maintains its internal files and to designate the availability of database structures during a reorganization.
- Sequence the reorganization to optimize the way in which tasks are performed.
- Optionally include SORT specifications on index set generations or data set generations that use the ORDER BY and OFFLINE options.

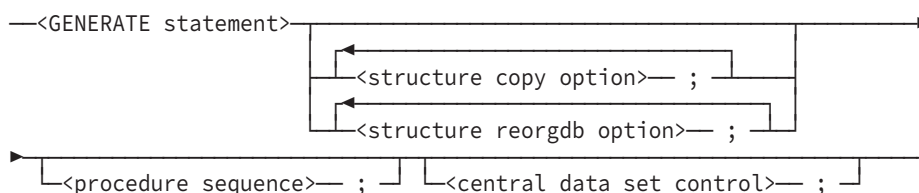
For easier access of information, the explanation of the BUILDREORG utility syntax is presented in the following discussions of these tasks.

The syntax for the BUILDREORG utility is illustrated and explained on the following pages.

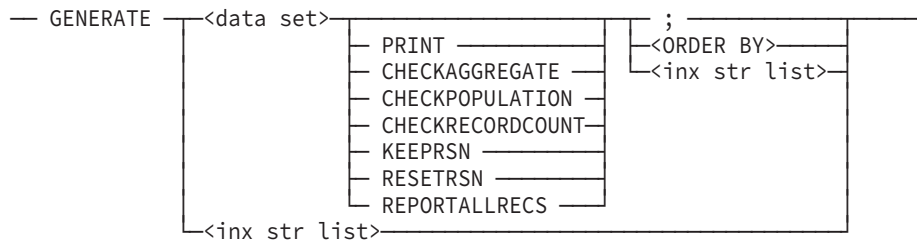
<reorganization specification>



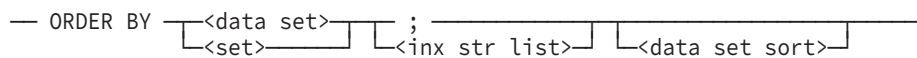
<central data set GENERATE statement>



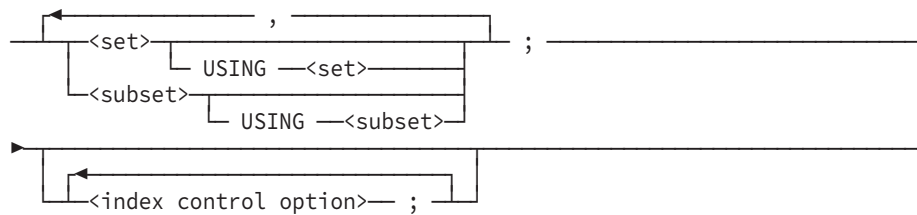
<GENERATE statement>



<ORDER BY option>



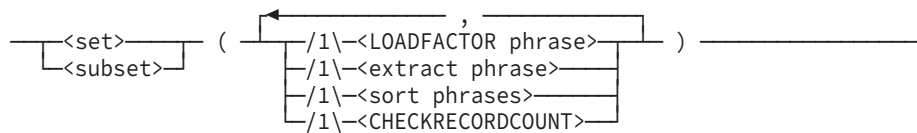
<inx str list>



<data set sort>



<index control option>



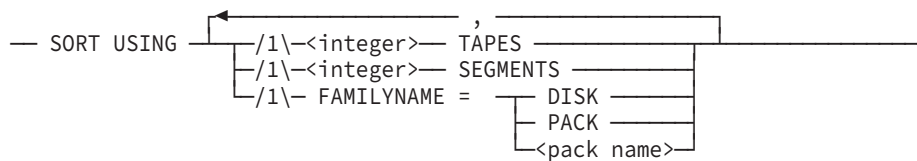
<LOADFACTOR phrase>



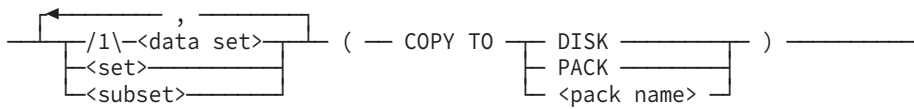
<extract phrase>



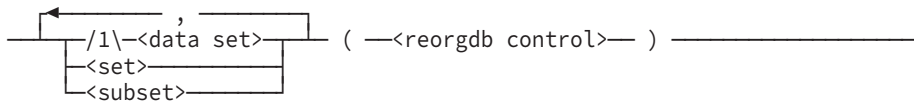
<sort phrases>



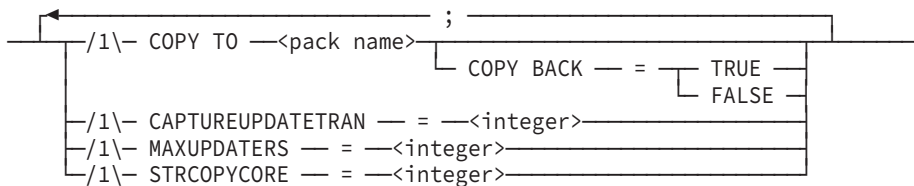
<structure COPY option>



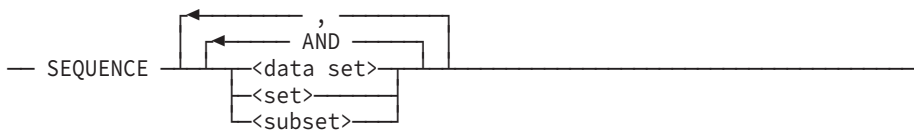
<structure reorgdb option>



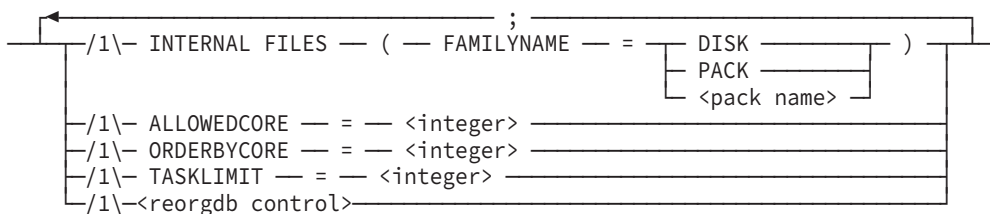
<reorgdb control>



<procedure sequence>

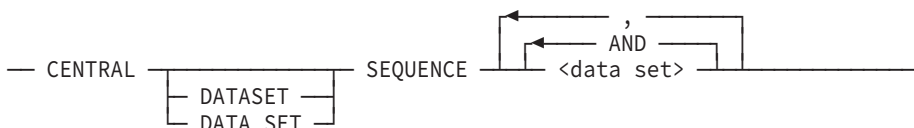


<central data set control>

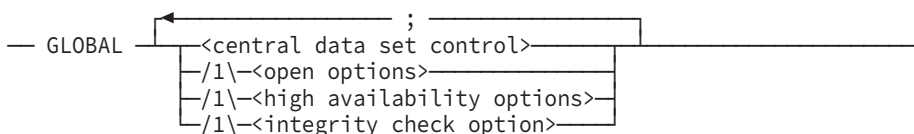


Note: ORDERBYCORE is the memory limit that can be used by the SORT routine when generating the dataset record using a set.

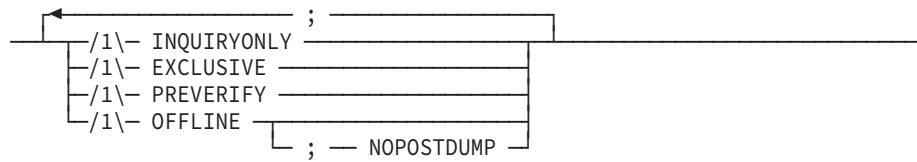
<central data set sequence>



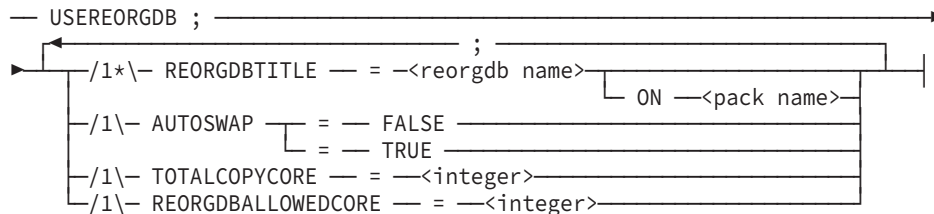
<reorg global control>



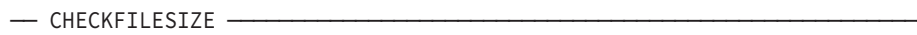
<open options>



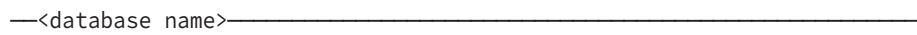
<high availability options>



<integrity check option>



<reorgdb name>



Using the BUILDREORG UPDATE Option

The UPDATE option must be used when the DASDL UPDATE compilation specifies that a reorganization is required. The UPDATE option cannot be used otherwise. When the DASDL UPDATE compilation specifies that a reorganization is required, the description file is marked as requiring reorganization and contains information on both old and new formats.

If the DASDL UPDATE compilation specifies that a generation by the REORGANIZATION program is required for a structure, then a garbage collection reorganization must be run using an explicit GENERATE statement for the structure. This reorganization must be performed when new sets or subsets are added to existing data sets. Adding new disjoint structures does not require an UPDATE reorganization. However, if the UPDATE reorganization is done in conjunction with other changes, the UPDATE option performs the necessary implicit generations for the new structures.

When the UPDATE option is used, it must precede all other reorganization specifications. The UPDATE option causes the REORGANIZATION program that is generated to make all the changes that were specified in the previous DASDL update. Additional specifications used with the UPDATE option override the defaults supplied by the UPDATE option to allow more control over the reorganization processes.

You should review the BUILDREORG report to verify the changes that are to be done by the REORGANIZATION program.

Reorganizing the Database

When the REORGANIZATION program runs, it first calls a version of the DMSUPPORT library, which should match the current database (an update level that is 1 less than the description file used to compile the REORGANIZATION program). If the update level of the DMSUPPORT library is greater than the update level of the database, the run is aborted with a DMS OPEN ERROR number 29.

The DMCONTROL utility is run to migrate the control file to the next update level. This utility checks that the update level of the control file is one less than the update level on the description file. If the levels are incorrect and you run the REORGANIZATION program, the following error message terminates the run:

```
CONTROL FILE UPDATE LEVEL SHOULD BE 1 LESS THAN DESCRIPTION
```

If the DMCONTROL utility code file and the description file cannot be found, they can be file-equated using a statement with the following syntax:

```
RUN REORGANIZATION/<database name>;  
FILE DMCONTROL (TITLE = SYSTEM/DMCONTROL ON SYS39);  
FILE DASDL (TITLE = DESCRIPTION/<database name> ON PACK);
```

If you accidentally reorganize the database with the UPDATE option a second time, the reorganization is aborted with the following error message:

```
DATA BASE ALREADY AT REORG PROGRAM UPDATE LEVEL
```

If you do not designate the UPDATE option, the REORGANIZATION program checks that the update level of the REORGANIZATION program matches the current update level of the database. If the update levels do not match, the following error message terminates the run:

```
DATA BASE DOES NOT MATCH REORG PROGRAM UPDATE LEVEL
```

Using an Alias Name

You can use an alias name to refer to a data set or set name using 16-bit (double-byte) character structure names. You can use an alias name in the following statement or options:

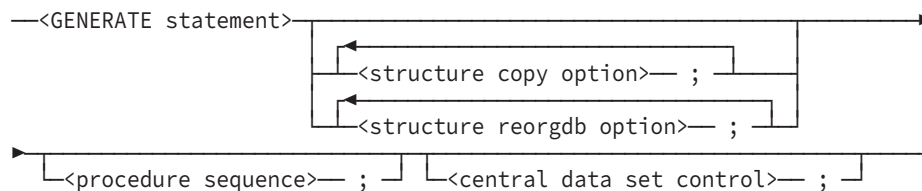
- GENERATE statement
- Index control option
- Structure COPY option
- Procedure sequence option
- Central data set sequence statement

For additional information about alias names, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

Using the Central Data Set GENERATE Statement

The central data set GENERATE statement provides the specifications for a reorganized data set. The GENERATE statement specifies the data set, sets, and subsets that are to be reorganized.

<central data set GENERATE statement>



Using the GENERATE Statement

If you are using the UPDATE option, designating the GENERATE statement is optional. If the data set was reorganized in the previous DASDL compilation, then the BUILDREORG utility uses the following statement by default:

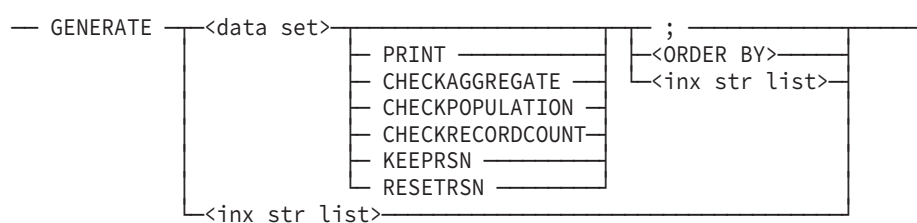
```
GENERATE <data set>
```

If the GENERATE statement designates a data set, the system reads the original data set, writes a new data set, and performs a complete garbage collection.

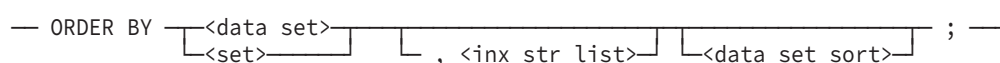
If a new set or subset is added, it is implicitly generated from the data set.

If you want to generate a structure and specify particular options explicitly, use the following statement.

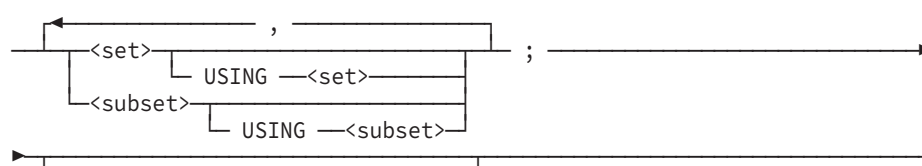
<GENERATE statement>



<ORDER BY option>



<inx str list>



```
┌──────────────────────────────────────────┐  
└──────────────────────────────────────────┘  
<index control option>— ; ─┘
```

If you generate both a data set and its set, and the set is generated before the data set, the BUILDREORG utility only creates and shows the generate tasks. This action improves availability of the data set and causes the REORGANIZATION program to run a separate fixup task to fix the set after data set generation is complete. This separate fixup task is created while the reorganization is running and is not shown on the BUILDREORG report.

Generating a Data Set

Only one data set can be specified for each GENERATE statement. All sets and subsets in the GENERATE statement must refer to a common data set. All sets, subsets, and data sets belonging to the same data set family must be generated in the same GENERATE statement.

Global data is treated like a data set. No other data set can be included in the GENERATE statement along with global data.

Structures can be generated only once for each reorganization run.

Whenever a data set and its embedded data set are both generated, the master data set is generated before the embedded data set. Any subsequent rebuild through a reorganization fails when both the master data set and its embedded data set are generated. Therefore, a full database dump should be taken after a reorganization when both the master data set and its embedded data set were generated during the reorganization.

Using the PRINT Option

You can add the PRINT option to the GENERATE statement to print a “beforeimage” and an “afterimage” of the records in a reorganized data set when running the REORGANIZATION program. Images are printed in the DMUTILITY record format. This format uses hexadecimal notation, record by record, with the segment address and record offset.

The following statement prints the records of the EMPLOYEES data set:

```
GENERATE EMPLOYEES PRINT;
```

Using the CHECKAGGREGATE and CHECKPOPULATION Options

You can calculate the value of an aggregate item or a population item during reorganization by using the following options:

1. CHECKAGGREGATE. The CHECKAGGREGATE option calculates the value of an aggregate item. If the value has changed, the system updates the aggregate item with the new value. Use this option only during initialization, when adding a new aggregate item, or when an existing aggregate item is corrupted. This option is processor intensive because it makes these calculations for every structure.

The following example calculates the aggregate item for the EMPLOYEESk data set:

```
GENERATE EMPLOYEES CHECKAGGREGATE;
```

2. CHECKPOPULATION. The CHECKPOPULATION option calculates the value of a population item. If the value has changed, the system displays the following prompt:

```
Skip to ignore or override to correct
```

3. Use this option only during initialization or when a population item is corrupted. This option is processor intensive because it makes these calculations for every structure. The following example calculates the population item for the EMPLOYEES data set:

```
GENERATE EMPLOYEES CHECKPOPULATION;
```

Using the CHECKRECORDCOUNT Option

Use the CHECKRECORDCOUNT option to recalculate the number of records in a structure when the structure record count is corrupted. This requires that the RECORDCOUNT option be set through the DASDL syntax. The first example generates the EMPLOYEES data set and uses the CHECKRECORDCOUNT option to recount the number of records in the EMPLOYEES data set. The second example generates the set S1 using itself and using the CHECKRECORDCOUNT option to recount the number of records in the S1 set.

```
GENERATE EMPLOYEES CHECKRECORDCOUNT;  
GENERATE S1; S1(CHECKRECORDCOUNT);
```

Using the KEEPRSN Option

Use the KEEPRSN option to preserve the RSN value of new records created during a USERORGDB reorganization. This option is reset by default.

Using the RESETRSN Option

Use the RESETRSN option to request the reorganization to reassign the RSN values starting from 1. This can be used to correct any RSN corruption problem caused by a hardware or software error.

You can only use this option in conjunction with the OFFLINE option. When using the RESETRSN option, all sets must be generated from the data set.

If all of the following conditions occur, you cannot use the RESETRSN option in the GENERATE statement:

- The database structures are generated with the UPDATE option.
- The generated structure contains a DATAMASK item.

- The OBFUSCATELEVEL is equal to 3.

Using the REPORTALLRECS Option

Use the REPORTALLRECS option to request the reorganization task to print all records which cause DUPLICATE ERROR or DATA ERROR (VERIFY STORE ERROR) in a DATASET/SET generation. One printer file is generated per structure. The failed records are printed in the DMUTILITY data set record format. Note that when SORT is involved during SET generation, the failed records are reported in the SET format.

Using the ORDER BY Option

Use the ORDER BY option to specify the order in which records are inserted into a reorganized data set.



If you are reorganizing an XE sectioned data set, you must use the OFFLINE option in order to use the ORDER BY option. You cannot perform an ONLINE reorganization and use the ORDER BY option. The reorganization process does not use the round-robin method to store records in sectioned data sets. The reorganization process accomplishes the ORDER BY task by dividing the total number of records by the number of sections and storing the appropriate number of records in the first section before starting the second section, and so forth. However, any new records created after the reorganization are added to the sections using the standard round-robin method.

You can perform an ONLINE reorganization using the ORDER BY option on a nonsectioned XE structure.

If you designated a prime set in DASDL, (and did not designate the ORDER BY option in the GENERATE statement) the generated data set is ordered in the key order of the prime set. An index that orders a data set is always generated after the data set is generated. The ORDER BY option in the GENERATE statement supersedes any prime set designated in DASDL.

If the OFFLINE option is used for the data set being generated with the ORDER BY option, then the SORT intrinsic can be used. By default, the data set is generated using the same mechanism as online reorganization without auditing, which might be slower than using the SORT intrinsic. You can override this behavior by explicitly specifying the SORT option for the data set in the syntax. This forces the use of the SORT intrinsic for the data set generation and can improve performance. An exception to this behavior is the generation of an XE data set. Since it can only be generated using the OFFLINE option, it uses the SORT intrinsic by default.

If the SORT intrinsic is used for a compact data set and a SORT phrase is not used to specify the number of segments, the number is calculated based on an estimated population. If the average record size is specified in the DASDL source, it is used to estimate the population. Otherwise, the average of the minimum and the maximum record size for the structure is used.

By default, the memory used for this ORDER BY sort is 2,000,000 words. Use the ORDERBYCORE phrase to specify the amount of memory to be used if you want to override the default.

The ORDER BY option can designate one of the following structures for ordering the data set:

- Set. The ORDER BY option can only use index sequential, unordered list, or ordered list sets that span the generated data set. The set must already exist in the database. Thus, you cannot use a set recently declared in the DASDL source. To use such a set, compile the DASDL source twice with the UPDATE option. The first compilation marks the set as a new structure and allows it to be generated. The second compilation marks the set as an old structure that can be used in the ORDER BY option.

For example, the following GENERATE statement orders the EMPLOYEES data set using the BYLASTNAME set:

```
GENERATE EMPLOYEES ORDER BY BYLASTNAME;
```

- Data set. This data set must be the generated data set. The generated data set maintains the same physical order. The following GENERATE statement orders the EMPLOYEES data set using the physical order of the EMPLOYEES data set:

```
GENERATE EMPLOYEES ORDER BY EMPLOYEES;
```

Note: Using the ORDER BY option for an extremely large compact or variable format structure that involves fixup of its sets might use large amounts of memory for the fixup file. This could result in the run-time error "RESIZE ABORTED – INSUFFICIENT MEMORY." If this error occurs or if memory usage becomes a problem, do not use the ORDER BY option.

Using the Index Structure List Option

The <inx str list> option (sets and subsets) can be generated from the data set, from another index structure, or from itself. The method of generation is determined by the presence of a USING clause. If a USING clause is not specified, the index structure is generated from the data set.

Generating from a Data Set

The system uses the data set as input and generates the index structure, performing complete table balancing. Only sets and automatic subsets spanning a common data set can be generated in this manner.

When generating an index structure from a data set, the system cannot determine the order in which duplicates were originally entered. Therefore, if DUPLICATES, DUPLICATES FIRST, or DUPLICATES LAST is specified in DASDL, the duplicate entries are entered in the new structure in a random sequence. Also, if there is a NO DUPLICATES condition on the key of the index structure and duplicates are encountered, the reorganization task terminates with an error. Other reorganization tasks continue.



When generating an XE set, the order of the duplicate entries is preserved by the presence of the record serial numbers (RSNs).

You also have a choice affecting the availability of sets and data sets. If you include SORT specifications in the ONLINE BUILDREORG run, a sort operation occurs that restricts the availability of sets and data sets as follows:

- A set is unavailable until its generation is complete.
- A data set is available if it has other sets that are available, and no changes are made that would affect the set that is being generated.

By default, no sorting occurs, which maximizes the availability of sets and data sets.

When data sets and sets are generated using the OFFLINE option, and a set is generated from the data set, the SORT intrinsic is used by default.

Generating from Index Structures

Generating an index structure from an index structure has two advantages over generating from a data set:

- The order of duplicates is preserved.
- Keys need not be extracted or sorted.

If the index structure being generated is a manual subset and the data set is generated, all key entries referring to deleted records in the data set are deleted. This prevents these keys from pointing past the end of file or to valid data.



If your intent is to perform a garbage collection only on disjoint index sequential sets, you can use the Visible DBS GARBAGE COLLECT command while the database is running. Initiating this command results in a garbage collection of the specified sets while they are in use. You do not run the REORGANIZATION program. For the syntax for the Visible DBS GARBAGE COLLECT command, refer to [Section 12, Communicating with the Database](#).

You can generate index structures from two types of index structures:

- Another index structure. Both index structures must have the same key, DUPLICATES condition, WHERE condition, and data in key against the data set. The system reads the index structure specified in the USING clause and generates the new index structure, balancing the tables completely. This method can generate sets as well as automatic and manual subsets.

The index structure used as the source of generation is designated after the USING clause. The following example generates the set BYNUMBER from the set OLDNUMS, and the subset TEMPS from the set OLDEMPS:

```
GENERATE BYNUMBER USING OLDNUMS, TEMPS USING OLDEMPS;
```

Do not use circular USING clauses because they do not generate index structures. The following are examples of circular USING clauses:

```
GENERATE S1 USING S2, S2 USING S1;  
GENERATE S1 USING S2, S2 USING S3, S3 USING S1;
```

- Same index structure. This is the most common structure used with the USING clause. Such generations follow the same rules as generating from another index structure. You must use the same index structure when generating embedded index structures.

The following example generates the set OLDNUMS from itself and the subset OLDEMPS from itself:

```
GENERATE OLDNUMS USING OLDNUMS, OLDEMPS USING OLDEMPS;
```

Sorts do not occur when generating from index structures.

Implicitly Generating Structures

Implicit generations for certain structures are provided by BUILDREORG. When the UPDATE option is specified for BUILDREORG, all operations are automatically provided to implement the changes specified in the previous DASDL update run; as a result, no GENERATE statement is required.

Whenever a data set and its embedded data set are both generated, the master data set is generated before the embedded data set. Any subsequent rebuild through a reorganization fails when both the master data set and its embedded data set are generated. Therefore, a full database dump should be taken after a reorganization when both the master data set and its embedded data set were generated during the reorganization.

Certain implicit generations can be modified by explicit specification. However, those modifications must adhere to current GENERATE statement restrictions.

- When the UPDATE option is specified to the BUILDREORG utility, the following implicit generations are provided for those structures that have been redescribed in the new DASDL source:
 - The data set having no prime set specified in DASDL has the following implicit generation:

```
GENERATE <data set>
```

- The data set having a prime set specified in DASDL has the following implicit generation:

```
GENERATE <data set> ORDER BY <index str>
```

The <index str> parameter references the prime set.

- All indexing structures, where DASDL indicates REORGANIZE <index str>, have the following implicit generation:

```
GENERATE <index str>
```

This implicit generation cannot be modified.

- All indexing structures specified in DASDL whose key is not changed have the following implicit generation:

```
GENERATE <index str> USING <index str>
```

- If the data set is generated, the following implicit generations are provided for spanning index structures that are not generated as described previously:

- All bit vectors and manual subsets have the following implicit generation:

```
GENERATE <index str> USING <index str>
```

- Except for sectioned structures with XE features, all automatic sets or subsets that are unordered lists—or that are indexsequential structures with DUPLICATES allowed, but FIRST or LAST unspecified—have the following implicit generation:

```
GENERATE <index str> USING <index str>
```

No partitioned structure or structure with a partitioned master can be reorganized. In addition, a data set being linked to by a partitioned structure cannot be reorganized.

Optimizing Set Generation Performance When Migrating to XE Features



Sets are implicitly generated from a data set when one of the following situations occurs:

- An Enterprise Database Server data set is converted to XE data set with sections.
- Sections are added to an existing XE data set that was not previously sectioned.
- A set that contains duplicate entries without DUPLICATES FIRST or DUPLICATES LAST specified in DASDL is converted to an XE set.

During the conversion to an XE set with sections, all the sets are generated. The reason for this is that section numbers must be added to the set keys when a data set is converted to an XE data set with sections. Sets with DUPLICATES specified, but FIRST or

LAST unspecified, must have record serial numbers (RSNs) extracted from the data set. RSNs are used to resolve duplicates. This extraction from the data set causes DASDL to specify REORGANIZE <index str>, which defaults to GENERATE <set> USING <data set>.

To speed up performance for the three types of conversions previously described, it is recommended that you include SORT specifications for the sets in the BUILDREORG specifications, as described earlier in this section under “Generating from a Data Set.”

Once the conversion is complete, the SORT specification does not affect the order of duplicates for future set generations. It is also recommended that you use the SORT specification in the BUILDREORG specifications for the XE to the XE set generation.

Another option for improving performance is to include the USING clause in the BUILDREORG specification to force the set to be generated from itself. Refer to “Generating from Index Structures” earlier in this section for additional information. There are cases, such as when a key item has been changed, in which generating a set from itself is not allowed.

Either of the two alternatives enables speedy generation of sets during the conversion process from data sets to XE sectioned data sets or from the XE data sets to XE sectioned data sets.

Explicitly Generating Structures

Although the UPDATE option can implicitly generate the reorganized database structures, this option cannot generate all structures. In such a case, and during garbage collection, you must explicitly generate structures with the GENERATE statement as follows:

```
GENERATE <structure>; <options>;
```

The designated structure is either one data set, global data, or a list of index structures such as sets and subsets. These sets and subsets must refer to a common data set and must be processed in the same GENERATE statement. Structures can be generated only once for each REORGANIZATION program run.

The designated options, which can be left out, control how the structure is generated.

Whenever a data set and its embedded data set are both generated, the master data set is generated before the embedded data set. Any subsequent rebuild through a reorganization fails when both the master data set and its embedded data set are generated. Therefore, a full database dump should be taken after a reorganization when both the master data set and its embedded data set were generated during the reorganization.

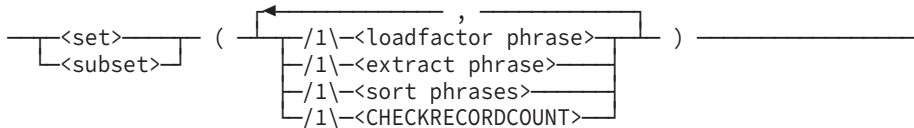
Using the Index Control Option

The index control option controls the resources used when generating index structures. An index control option can be specified for index structures that are generated implicitly, as well as those generated explicitly. You can control these resources with the following options:

Reorganizing the Database

- LOADFACTOR phrase
- EXTRACT phrase
- SORT phrases
- CHECKRECORDCOUNT

<index control option>



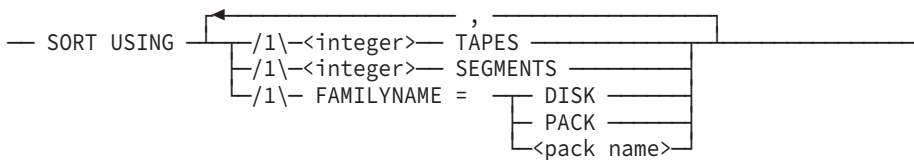
<loadfactor phrase>

— LOADFACTOR = —<integer>—

<extract phrase>

— EXTRACT KEYS TO —<medium>—

<sort phrases>



Using the LOADFACTOR Phrase

The LOADFACTOR phrase overrides, for reorganization only, the load factor specified in the DASDL source. The LOADFACTOR phrase can be specified only for generated index structures of type index random, index sequential, and ordered list. The LOADFACTOR phrase represents the percentage of each table that contains entries following reorganization. Depending on the degree of table compaction desired, the integer can be any integer value from 1 to 99.

The following example designates a load factor of 58 percent for the BYNAME set:

```
GENERATE EMPLOYEES, BYNAME; BYNAME(LOADFACTOR = 58);
```

By default, the LOADFACTOR phrase is the value of LOADFACTOR specified in the DASDL source. If no load factor is specified in DASDL, the default value for INDEX SEQUENTIAL and ORDERED LIST is 66 percent, and for INDEX RANDOM the default value is 50 percent.

Following reorganization, the load factor specified in the DASDL source is then used.

Using the EXTRACT Phrase

If an index structure is generated from a data set or an INDEX RANDOM structure, REORGANIZATION unconditionally creates a file consisting of the keys for the index structure. This key file is an unordered version of the index structure and can be quite large. The EXTRACT phrase controls where the key file is written.

By default, the key file is written to the medium on which the index structure is to reside.

When the database is a permanent directory database, the key file is created under the same permanent directory as the data files. If the EXTRACT phrase is used for a permanent directory database, ensure that the necessary permanent directories exist for the key file to be created.

Using SORT Phrases

The REORGANIZATION program uses the SORT intrinsic to order the extract key file. The SORT phrases enables you to specify the resources to be used by the SORT intrinsic. Disk (or pack), tape, or a combination of disk (or pack) and tape sorting is permitted. Refer to the *System Software Utilities Operations Reference Manual* for a complete description of the SORT intrinsic.

Choosing SORT phrases enables you to specify to the SORT intrinsic the number of tapes, the number of segments, and the pack family where the internal files are to be maintained.

Note: A SORT phrase is used only when a sort operation is to be performed. A SORT phrase must not be used to control the sequence of set and data set generation.

If SORT phrases are specified, default values are assigned for those options not explicitly stated by the user. Default values for the SORT phrases declarations not explicitly specified are as follows:

- If <integer> TAPES is not specified, SORT USING 0 TAPES is assumed by default.
- If <integer> SEGMENTS is not specified, SORT USING 2 TIMES PHYSICAL FILE SIZE SEGMENTS is assumed by default.

If the ORDER BY option causes the sort operation, the default value is POPULATION TIMES (NEW RECORD SIZE +1) TIMES 2 AND A HALF.

- If FAMILYNAME is not specified, the internal files family name declared in the reorganization global control specifications is assumed by default. If an internal files specification is not declared, DISK is assumed by default.

The maximum value that can be specified for SORT SEGMENTS is 268,435,454.

There are two cases when the SORT intrinsic is not done for generating a set even though the SORT option was specified for that set in the BUILDREORG specifications:

- The first case is when a set is generated using itself.
- The second case is when the set is generated from the data set and is also used in the ORDER BY clause to order the data set. In both of these cases, the set is actually

generated from itself and the SORT intrinsic specifications are ignored because they are meant to be used when generating a set from the data set.

When a set is generated from itself, and the data set was also generated and contains more than 2 million records, it is possible that the fixup of the set can use the SORT intrinsic and then it can use the given specifications.

Using the Data Set SORT Option

The data set SORT option forces the use of the SORT intrinsic when a data set is generated using the ORDER BY and OFFLINE options. The resources used for the sorting of the data set can be controlled using the SORT phrases.

The syntax is as follows:

```
——<data set>—— ( ——<sort phrases>—— ) —— ; ——|
```

Using the Structure COPY TO Option

The structure COPY TO option enables you to copy structures to a temporary pack location before the reorganization begins.

If the COPY TO option is used with a permanent directory database, ensure that the necessary permanent directories exist for the file to be successfully copied.

The syntax is as follows:

```
—— /1\——<data set>—— ( —— COPY TO —— DISK —— ) ——|
      |——<set>——
      |——<subset>——
      |——<pack name>——
```

The REORGANIZATION program always creates generated structures on their final medium, that is, the pack specified in the latest DASDL update. For data sets and garbage collect on index sets, the REORGANIZATION program first renames the old file, creates an empty new file and then moves records from the old file to the new file. For data sets, rows of the old file are purged as the new file grows.

In general, enough disk space is required for the new file, plus one row of the old file, plus the fixup file. Some situations that require more disk space are as follows:

- If COPY TO is specified for multiple structures of a data set family, all copies of the structures for that family are performed prior to the start of the generation of the data set. Enough space must be made available for copies of all of the structures being copied.
- If an ORDER BY clause is specified in BUILDREORG for a data set, no rows of the old file are purged until the generate process is complete. Therefore, disk requirements are the size of the entire old file, plus the entire new file, plus the fixup file.
- No rows of COMPACT, ORDERED or UNORDERED data sets are purged until the generate process is complete.
- When the new file of a RANDOM data set is initialized, the primary scramble area is

always allocated. Therefore, at the start of the generation, the entire old file exists, along with the primary scramble area of the new file and the fixup file.

- If user programs are allowed during the ONLINE reorganization (either inquiry or update users), any access to a DIRECT data set record causes all records prior to that record to be allocated in the new file. Therefore, if a user program accesses the last record in a DIRECT data set, the entire new file is immediately allocated.
- If a set is generated from a set, there must be enough disk space for both the old file and the new file.

The size of fixup files depends on the file type. COMPACT and variable format data sets require two words per record in the old file, and all other data sets require one word per record in the old file.

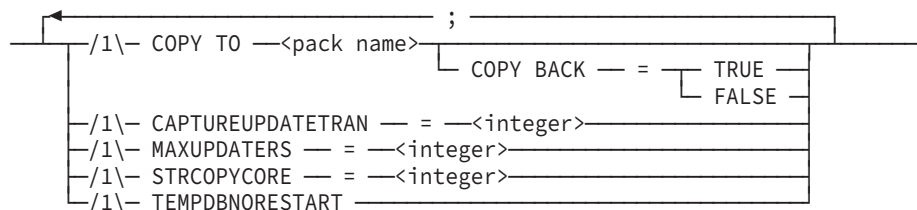
In situations where two copies of the file are required and there is not enough space on the final pack for both copies, the old file can be copied first to a temporary pack using the structure COPY option. The REORGANIZATION program first copies the old file to the specified pack. It then creates a new file on the final pack and proceeds with the generation. When the generation is complete, the old file is removed from the pack specified in the structure COPY option.

Using the Structure REORGDB Option

<structure reorgdb option>



<reorgdb control>



Using the REORGDB Control COPY TO Option

The REORGDB control COPY TO option specifies the database copy location where the production database structures are captured and applied during a REORGDB reorganization. This option is similar to the structure COPY TO option, with the exception that it can be specified as a global, central data set, or structure control, and with the COPY BACK option. If the COPY TO option is used with a permanent directory database, ensure that the necessary permanent directories exist for the file to be successfully copied. The default pack name for the REORGDB control COPY TO option is the same as the existing database structure.

While database structures are being copied from the production database to the copy database, if a halt/load occurs while the production database is quiesced, or suspended, and waiting to be resumed, you must restart the reorganization. This applies to all copy process activities because destination files are not in a state that would permit recovery.

Notes:

- *If the COPY BACK option is not used with the COPY TO option, rebuilding from a time prior to the reorganization will not be possible. A full dump of the entire database must be performed after the reorganization for future rebuild recoveries.*
- *When reorganizing a permanent directory database, the database administrator must create the necessary permanent directories on the pack specified in the COPY TO option.*

The mechanism for changing data structure pack families during a reorganization is slightly different when combined with the USEREORGDB option. When using the BUILDREORG USEREORGDB option, include the COPY TO syntax without the COPY BACK syntax if you want to change pack families. Using this syntax also has the effect of setting the family change bit in the control file. The effect of this procedure is identical to having manually made the change using DMCONTROL.

Using the REORGDB Control COPY BACK Option

The REORGDB control COPY BACK option specifies that a file be returned to the original location when the swap occurs at the end of the REORGDB reorganization process. This option does not specify pack locations. The COPY BACK option can only be used in combination with the USEREORGDB and COPY TO options.

The COPY BACK option default is FALSE. If the COPY BACK option is not used with the COPY TO option, or if the COPY BACK option is FALSE, then the final destination for the reorganized structures is the same as the COPY TO location.

Note: *Be sure that there is sufficient space on the disk pack to accommodate both the production database structures named in the reorganization specification and the corresponding reorganized structures from the database copy when it is copied back to the production database. When the COPY BACK option is specified, both the production database structures and the reorganized structures reside on the same disk pack while waiting for the swap to occur.*

COPY BACK operations begin once the reorganization of all structures is complete. The COPY BACK operations must complete before updates can be applied to the production database. Once the reorganized structure files are moved to their COPY BACK location, the files are removed from the COPY TO location.

If the swap has not begun, a halt/load restarts the COPY BACK process at the beginning of any partially copied file and the process continues until all files are copied to the COPY BACK location.

Using the CAPTUREUPDATETRAN Phrase

Specialized apply-updates tasks enable you to update newly reorganized structures with the application program updates that are captured during the REORGDB reorganization process. These tasks use standard data management statements such as BEGINTRANSACTION and ENDTRANSACTION to perform the work.

The CAPTUREUPDATETRAN phrase controls the number of captured data set changes for each transaction of the apply-updates tasks as the captured updates are applied to the files belonging to the copy database.

If the CAPTUREUPDATETRAN phrase is not specified, the default of 500 updates per transaction is used. The minimum value for the CAPTUREUPDATETRAN phrase is 500 and the maximum is 10,000. You can achieve better performance by specifying larger values. However, these larger values can potentially cause longer recovery times if a recovery becomes necessary.

The CAPTUREUPDATETRAN phrase is a component of the <reorgdb control> construct, which can be used with both the <central data set control> and <structure reorgdb option> syntax. When the <structure reorgdb option> form is used, the CAPTUREUPDATETRAN phrase is available only for use with the <central data set control> option.

Using the MAXUPDATERS Phrase

Specialized apply-updates tasks enable you to update newly reorganized structures with the application program updates that are captured during the REORGDB reorganization process.

The MAXUPDATERS phrase controls the number of apply-updates tasks that apply the captured updates to the file belonging to the copy database.

If the MAXUPDATERS phrase is not specified, the system uses the number of update applications to determine the number of apply-updates tasks. A default of five applyupdates tasks plus one driver task for each structure that is either explicitly or implicitly reorganized is used. If the number of update applications is greater than five, the number of tasks is higher—up to a maximum of 50.

The minimum value for the MAXUPDATERS phrase is one and the maximum is 50. For cases involving embedding and links, the number of apply-updates tasks is one.

The MAXUPDATERS phrase is a component of the <reorgdb control> construct, and can be used with both the <central data set control> and <structure reorgdb option> syntax. The MAXUPDATERS phrase is available only for use with <data set> when the <structure reorgdb option> form is used.

Using the TOTALCOPYCORE and STRCOPYCORE Phrases

The TOTALCOPYCORE and STRCOPYCORE phrases enable you to specify values within the BUILDREORG process that determine the number of copy tasks that can be running at the same time.

The overall TOTALCOPYCORE default value is 5 million words, with a maximum of 100 million words and a minimum of 1 million words. Specifying larger values within that range might be useful for databases with structures having a large number of sections.

The default STRCOPYCORE value for each structure is 250,000 words, with a maximum of 20 million words and a minimum of 150,000 words. This value is applied at the structure level. For sectioned data sets, the value is divided among the sections.

The specified STRCOPYCORE value cannot exceed the value of the TOTALCOPYCORE specification. If the STRCOPYCORE value does exceed the TOTALCOPYCORE specification, the BUILDREORG process will increase the TOTALCOPYCORE specification to equal the highest single STRCOPYCORE value so that the copy tasks can run.

If the specified STRCOPYCORE value is allowed to default to 250,000, then, during the COPY phase of a REORGDB reorganization, only one section of a structure is copied at a time, even if TOTALCOPYCORE specification is set to a large value. This applies to sectioned data sets and physically-sectioned sets.

If the STRCOPYCORE value is set to $N \times 250,000$, then N sections (up to a limit of 40) is copied in parallel as long as the TOTALCOPYCORE specification is not exceeded.

Each copy task, by structure or by data set section, is allocated two 65K buffers. These buffers can be less than 65K if the AREASIZE value equals less than 65,000 words.

As the copy task progresses, if there is an update to the production database the update is duplicated to the database copy file. The update duplication process maintains a pool of 10 small buffers for duplicate updates and rotates serially through the pool. If excessive wait times result during the process, the number of buffers in the pool is increased to the structure limit of the STRCOPYCORE values. When this occurs, the following message is displayed:

```
Pool buffer in use from a previous duplicated write
```

Note: Copy memory for the small buffers is held only until all copy tasks finish.

Before all copy tasks are completed, all modified buffers are written. At that point, all copied files are identical to the production database files. The capture process takes over. No more production database updates are duplicated to the copy files, and all copy memory is deallocated. The two 65K copy buffers are de-allocated as each file completes its copy task (is duplicated to the corresponding file on the copy database), but the small buffer pool continues to be used to copy the production database updates until all of the remaining copy tasks finish.

This mechanism allows the parameters to be tuned for activity-based processing rather than task-based processing. The advantage is that heavily updated structures can be given more memory and more buffers to be used for quickly processing updates during the copy.

USING the TEMPDBNORESTART Phrase

The TEMPDBNORESTART phrase controls the restart capability of the offline reorganization performed on the name of the database specified in REORGDBTITLE phrase.

If specified, the database specified in REORGDBTITLE phrase will not be restartable.

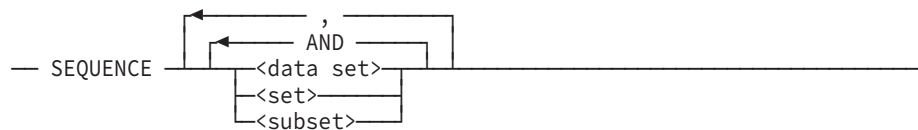
This option can be used to speed-up the REORGDB performance for datasets with a large number of sections.

Using the Procedure Sequence Option

The <procedure sequence> option controls the order in which reorganization processes occur within a data set family. This sequence enables you to optimize both resources and throughput. For most applications, default sequencing along with the possible specification of the TASKLIMIT phrase and index control options should provide ample optimization.

It is not necessary to specify the data set, but all sets and subsets in a single <procedure sequence> option must refer to the same data set.

You can override the default sequence by using a statement with the following syntax:



The same criteria used to determine the default sequence of processes must be followed when explicitly specifying a procedure sequence. Some flexibility is allowed in determining which independent processes should be done in parallel. Depending on the TASKLIMIT value, all structures in the procedure sequence that are joined by AND are processed simultaneously. If the number of these structures is greater than the TASKLIMIT, the number of processes executing simultaneously is limited by the TASKLIMIT value. Each set of structures separated by a comma (,) in the procedure sequence is processed only after the prior set of structures have been processed.

The <procedure sequence> option sequences processes within a data set family. For example, the following option sequences the data set EMPLOYEES, the set BYNAME, and the subset EXEMPT in the designated order within a data set family.

```
SEQUENCE EMPLOYEES, BYNAME, EXEMPT
```

The following default sequencing criteria are used within a data set family:

- The generation of an embedded index set cannot occur in parallel with the generation of its master.
- If an index structure is generated from another index structure that is generated or requires fixup, then the index structure in the USING clause must appear before the generated index structure, and cannot be joined by AND.
- Structures not explicitly specified in the <procedure sequence> option are automatically placed in sequence according to the preceding criteria. These structures are joined by AND as much as possible.
- When a data set and its spanning set are both generated, the set is generated first and then fixed up at a later time. This maximizes structure availability, since some updates to data sets are not allowed when a set is being generated. However, the set is generated after the data set if any of the following conditions are true:
 - The set is a bit vector, a manual subset, or an unordered list.
 - The set is an index sequential or an ordered list that allows duplicates, but DUPLICATES FIRST or DUPLICATES LAST is not specified.
 - The data set is ordered by the set.
 - The set is generated from the data set using the SORT option.
 - The structure is being reorganized using the OFFLINE option.

or



The EXTENDED attribute is being added to or removed from a structure.

Example

The following example processes the BYNAME set, then the BYSALARY and BYZIP sets at the same time, and finally, the BYYEAR set:

```
GENERATE EMPLOYEES, BYNAME, BYYEAR, BYSALARY, BYZIP;  
SEQUENCE BYNAME, BYSALARY AND BYZIP, BYYEAR;
```

Using the Central Data Set Control Options

The <central data set control> statement applies to the global level using the <reorg global control> statement or applies to the data set level using the central data set GENERATE statement. The values you assign at the global level apply to the overall reorganization as well as to any data set family that does not have a specified value in the central data set GENERATE statement. For example, TASKLIMIT = 2 indicates that at most, two generate tasks could be active, and each active generate task that does not

have a TASKLIMIT assigned would have at most two active subtasks. When you assign a value in the central data set GENERATE statement, it applies only to that data set family and overrides any global values that might be specified. The default values for the <central data set control> options are

- FAMILYNAME = DISK
- TASKLIMIT = 1
- ALLOWEDCORE = 2,000,000 words for each sort task, or
ALLOWEDCORE = 2,000,000 words for each fixup task
- ORDERBYCORE = 2,000,000 words

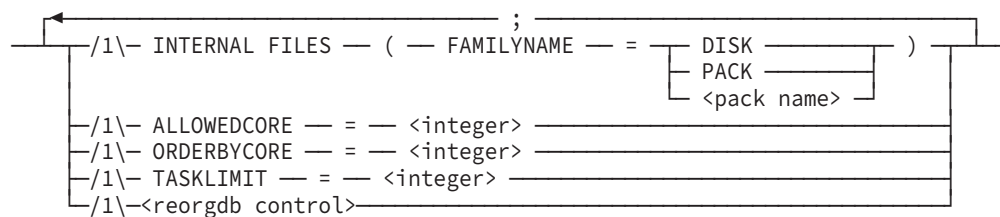
The following table shows the minimum and the maximum values that reorganization uses if values outside these ranges are included in the BUILDREORG specifications:

Function	Minimum Words	Maximum Words
Fixup	2 million	549,755,813,887 (2**39-1)
Orderby Sort	2 million	268,435,455 (2**28-1)
All other Sorts	2 million	268,435,455 (2**28-1)

The default sort task memory size is calculated based upon the information in the *System Software Utilities Operations Reference Manual*.

The following diagram illustrates the syntax for the <central data set control> statement:

<central data set control>



Note: ORDERBYCORE is the memory limit that can be used by the SORT routine when generating the dataset record using a set.

Using the INTERNAL FILES Phrase

The INTERNAL FILES phrase designates the location of the internal files generated by the REORGANIZATION program. These files are used in such functions as restarting the program and retrieving records. Before designating the location of the internal files, you must understand how the REORGANIZATION program generates and names these files.

Reorganizing the Database

Notes:

- Though not required, Unisys recommends that you use the *INTERNAL FILES* phrase. If you do not use the *INTERNAL FILES* phrase, the reorganization program could display the *REQUIRES *PK DISK* error message and hang. If you do not use the *INTERNAL FILES* phrase, the default family name is *DISK*.
- For a permanent directory database, the *INTERNAL FILES* phrase is ignored and the internal files are created on the same pack family as the related structure, using the same datapath.
- The names of some internal files are different for permanent directory databases to conform to the file naming rules for permanent directories.

The following example designates the internal files family to be *DBDATA*:

```
GENERATE EMPLOYEES ORDER BY BYNUMBER ;
INTERNAL FILES (FAMILYNAME = DBDATA) ;
```

Given the previous pack specification, *DBDATA*, the following files for the *EMPLOYEE* data set and *BYNUMBER* set are stored on the *DBDATA* pack:

File Name	Explanation	Contents
TEST/EMPLOYEES/DATA/FIXUP	Fix up file	Cross-reference addresses between the old and new files
TEST/EMPLOYEES/BYNUMBER/FIXEDUP	Fixed up file	Blocks of the file that have been fixed up
TEST/EMPLOYEES/BYNUMBER/RESTART This file is specific to the OFFLINE option.	Internal file	Data for restart purposes
TEST/EMPLOYEES/DATASPACE This file is specific to the OFFLINE option.	Compact data set file	Available space information
TEST/EMPLOYEES/BYNUMBER/COARSE1 This file is specific to the OFFLINE option.	Temporary file	Data created and purged during the creation of available space tables
TEST/EMPLOYEES/BYNUMBER/COARSE2 This option is specific to the OFFLINE option.	Temporary file	Data created and purged during the creation of available space tables

When a data set is generated, or an index set is generated with a *USING* clause, the *REORGANIZATION* program immediately changes the title of the existing file, appending */OLD* to the end of the file name. The *REORGANIZATION* program then initializes a new, empty file. For example, the *EMPLOYEES* data set with the *BYNUMBER* set would be changed as follows:

Original Structure File Name	Renamed Structure File Name
TEST/EMPLOYEES/DATA	TEST/EMPLOYEES/DATA/OLD
TEST/EMPLOYEES/BYNUMBER	TEST/EMPLOYEES/BYNUMBER/OLD

The following files would then be initialized as new, empty files:

```
TEST/EMPLOYEES/DATA
TEST/EMPLOYEES/BYNUMBER
```

Note: You can copy the old files to a temporary pack using the structure COPY option. For more information, refer to "Using the Structure COPY Option" earlier in this section.

Once the data set has been generated, the fixup process begins with a fixup file being created. Every time a record is moved from the old file to the new file, an entry is made in the fixup file indicating that any references to the old address (such as in index sets or in links in other data set records) must be fixed up. The length of the fixup file depends on the type of data set being generated.

Data Set Type	Fixup File Contents
Compact data sets and variable format data sets	Two words for each record in the old file
All other data set types	One word for each record in the old file

The fixup file has FIXUP appended to the file name. For example, the EMPLOYEES data set could have the following fixup file associate with it:

```
TEST/EMPLOYEES/DATA/FIXUP
```

For each structure that requires a fixup, a fixed up file is created. The fixed up file is used to identify blocks of the file that have been fixed up. For all fixed up structures, the fixed up file contains an on bit for each block in the new file.

The fixed up file has FIXEDUP appended to the file name. For example, if BYNUMBER were an index set spanning EMPLOYEES, all disk file addresses in BYNUMBER would need to be fixed up. The fixed up file identifies blocks in the BYNUMBER file whose addresses have been fixed up. If the BYNUMBER file contained 100 blocks, the fixed up file would contain 100 bits.

```
TEST/EMPLOYEES/BYNUMBER/FIXEDUP
```

Using the ALLOWEDCORE Phrase

The ALLOWEDCORE phrase controls the amount of memory that a reorganization uses for both sort and fixup tasks. Without the ALLOWEDCORE phrase, a default ALLOWEDCORE value of 100,000 words per task applies for sort tasks. For fixup tasks, a default ALLOWEDCORE value of 2,000,000 words per task applies. The default ALLOWEDCORE value for fixup tasks overrides the user-specified ALLOWEDCORE value

per task if less than 2,000,000 words per task is specified. The actual memory used is dependent upon the data set population. Small data set populations do not require the full amount, while large populations use the entire amount.

If you specify an ALLOWEDCORE value in the <central data set control> statement, that value overrides the default value. If you are generating many data sets and some involve sort tasks and some involve fixup tasks, specifying an ALLOWEDCORE value in the <central data set control> statement within the central data set GENERATE statement gives you the flexibility to assign a high value for fixup tasks and a low value for sort tasks.

Each generated data set consists of allocated ALLOWEDCORE divided by TASKLIMIT words of memory. For each set of the data set, the allocation is data set ALLOWEDCORE divided by TASKLIMIT. For example, a database has 3 data sets and each data set has 4 sets; TASKLIMIT = 3; ALLOWEDCORE = 30,000,000; and the <central data set sequence> specification uses commas (DS1, DS2, DS3). The work for each dataset and its sets is completed before the next data set and set combination can be started. The allocation for each data set is 10,000,000 (30,000,000/3); and the allocation for each set is $10,000,000/3 = 3,333,333$.

Each DS1 is generated and 10 million words are used. When it finishes, sets 1, 2, and 3 start and use 3.333 million words each for a total of 9.999 million. As soon as one of the sets finishes, set 4 is started.

Using the same example, if the goal is to utilize 30 million words, then the ALLOWEDCORE specification must change to 90,000,000. Thus, 30 million can be used by the data set being generated, and when it completes, 3 set generates start and utilize 10 million words each for a total of 30 million.

Because the ALLOWEDCORE value in the <central data set control> statement is a total value that applies to the number of tasks allowed, you must multiply the per-task value you desire by the number of simultaneous tasks allowed to obtain a total value to assign to the ALLOWEDCORE phrase. (In other words, the total ALLOWEDCORE value is divided by the TASKLIMIT value to obtain a per-task value.)

The maximum value that can be specified to the BUILDREORG utility is 549,755,813,887 ($2^{**}39-1$).

Using the ORDERBYCORE Phrase

Use the ORDERBYCORE phrase to control the amount of memory that the SORT routine can use when generating a data set using the ORDER BY option.



If the ORDERBYCORE phrase is not specified, 2,000,000 words is the default. This SORT routine occurs only when the OFFLINE option is used with the ORDER BY option and the data set SORT option, or when the data set being ordered is an XE sectioned data set.

The maximum value that can be specified to the BUILDREORG utility is 549,755,813,887 ($2^{39}-1$). The maximum value that the REORGANIZATION program uses for ORDERBYCORE is 268,435,455 ($2^{28}-1$). If large values are used for ORDERBYCORE, be cautious when using AND within the <central data set sequence> statement because ORDERBYCORE is the sum of all parallel data set order by memory.

Using the TASKLIMIT Phrase

Use the TASKLIMIT phrase to limit the total number of reorganization tasks that can run simultaneously. You can specify a TASKLIMIT value at the global level, the data set level, or both. If you do not specify a value for TASKLIMIT, a default of 1 applies. If you specify a TASKLIMIT value greater than 1 (allowing multiple reorganization tasks to run simultaneously), refer to "DASDL ALLOWEDCORE" under "Enhancing Reorganization Performance" later in this section.

If you specify a TASKLIMIT value at the data set level (in the central data set GENERATE statement), it applies to the number of simultaneous tasks for that data set family and overrides any value specified at the global level.

If you specify a TASKLIMIT value at the global level, it applies to the number of active generate tasks and to the number of active subtasks for any data set families that do not have a specified local task limit.

Using the TASKLIMIT phrase places an upper limit on the number of reorganization processes that can occur at any one time. Depending on the value of TASKLIMIT, all structures in the procedure sequence that have been either implicitly or explicitly joined by AND are processed simultaneously. The effective value of TASKLIMIT depends on the machine configuration on which the REORGANIZATION program is running.

The following example designates a global task limit of 50:

```
GENERATE EMPLOYEES ORDER BY BYNAME;
GLOBAL TASKLIMIT = 50;
```

The following example designates a local task limit for the data set CUSTOMERS:

```
GENERATE EMPLOYEES;
GENERATE CUSTOMERS;
    TASKLIMIT = 2;
GENERATE PRODUCTS;
GLOBAL TASKLIMIT = 20;
```

Using the Central Data Set Sequence Statement

The <central data set sequence> statement enables you to control the order in which reorganization processes occur between data set families. This sequence enables you to optimize both resources and throughput. Some flexibility is allowed in determining which central data set processes should be done in parallel. Depending on the TASKLIMIT value and the central data set sequence specification, the central data sets in the central data set sequence joined by AND are processed simultaneously. If the number of these structures

Reorganizing the Database

is greater than the specified TASKLIMIT, the number of processes executing simultaneously is limited by the TASKLIMIT value. Each group of structures delimited by commas in the central data set sequence is processed serially. By default, the TASKLIMIT value is one.

Certain rules must be followed when specifying a central data set sequence. These rules are as follows:

- The generation of an embedded data set must occur after the generation of its master. Any rebuild through a reorganization fails when both the master data set and its embedded data set are generated. Therefore, a full database dump should be taken after a reorganization when both the master data set and its embedded data set were generated during the reorganization.
- The generation of a disjoint data set cannot occur in parallel with the generation of another data set that has a link reference to the disjoint data set.
- When both a master and an embedded data set are being generated, the embedded data set can be in the <central data set sequence> statement without the master, but by default, the master is still generated first.
- The data set that owns the manual subset must be generated before the data set that is spanned by the manual subset. Otherwise, a sequence error occurs.

The following diagram illustrates the syntax for the <central data set sequence> statement:

<central data set sequence>



The <central data set sequence> statement controls the order in which processes between data sets occur. The master structure must be reorganized before its embedded structure. The following example designates that the data set EMPLOYEES is to be processed before the data set PRODUCTS:

```
CENTRAL DATA SET SEQUENCE EMPLOYEES, PRODUCTS;
```

The following example processes the data sets EMPLOYEES and PRODUCTS at the same time:

```
CENTRAL DATA SET SEQUENCE EMPLOYEES AND PRODUCTS;
```

Example

The following example processes the EMPLOYEES data set, then the PRODUCTS and ORDERS data sets, and then the CUSTOMERS data set:

```

GENERATE EMPLOYEES;
GENERATE CUSTOMERS;
GENERATE PRODUCTS;
GENERATE ORDERS;
CENTRAL DATA SET SEQUENCE EMPLOYEES, PRODUCTS AND ORDERS,
CUSTOMERS;

```

Using the Reorg Global Control Statement

The <reorg global control> statement enables you to

- Identify how the database can be opened during the reorganization.

If you do not specify how the database can be accessed during an ONLINE reorganization, user programs can open the database for update and inquiry purposes. In addition, the ONLINE reorganization process is audited and is therefore restartable.

If you use the OFFLINE option for the reorganization, only state control records are put in the audit to indicate the beginning and end of the reorganization. If you use the EXCLUSIVE option, you cannot access the database until the entire reorganization process is complete. The reorganization process can be restarted at the structure level.

You can update database structures that are not being OFFLINE reorganized. To rebuild the database, you must perform a database dump of the OFFLINE reorganized structures when the reorganization process is complete. A successfully completed reorganization using the OFFLINE option prompts you to perform the database dump.

Use the open options to restrict user access to a database and to request that the structures being reorganized are online or offline during the reorganization.

- Identify the location of internal files generated by the reorganization.
- Limit the number of tasks that can be processed simultaneously.
- Control the amount of sort core and fixup core to be used during the reorganization.

You can include at most one <reorg global control> statement in your reorganization specification. If included, the <reorg global control> statement must be the last statement in the reorganization specifications for the BUILDREORG program. Any statements included after the <reorg global control> statement are ignored by the BUILDREORG program.

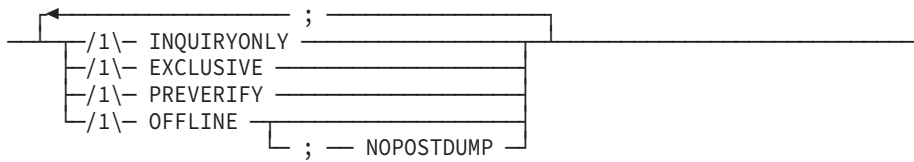
The syntax is as follows:

```

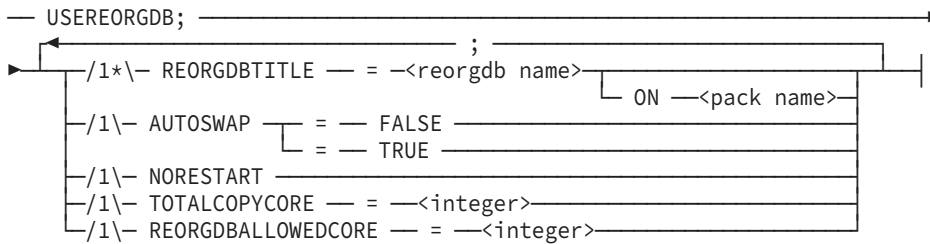
— GLOBAL — |
               |
               | <central data set control> ;
               | /1\ <open options>
               | /1\ <high availability options>
               | /1\ <integrity check option>
               |
               |

```

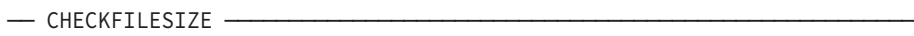
<open options>



<high availability options>



<integrity check option>



<reorgdb name>



Refer to “Using the Central Data Set Control Options” earlier in this section for information about the <central data set control> statement.

In a reorganization using the OFFLINE option, you can override the <reorg global control> statement at the central data set level. For example, if you are reorganizing 200 structures and want to reorganize all but one of them with the OFFLINE option, you can override the GLOBAL OFFLINE setting by specifying OFFLINE = FALSE at the central data set level. This approach is more efficient than specifying the OFFLINE option for 199 structures.

Using the INQUIRYONLY Option

The INQUIRYONLY option enables the database to be opened for inquiry only during reorganization. Since updates are not allowed, no loss of updates can occur if the reorganization run does not complete. This option does not affect the level of auditing that is carried out during the reorganization. If the INQUIRYONLY option is used with the OFFLINE option, any structure that is not being reorganized is available for inquiry only.

The following example designates the INQUIRYONLY option:

```
GENERATE EMPLOYEES ORDER BY BYNAME;  
GLOBAL INQUIRYONLY;
```

Using the EXCLUSIVE Option

The EXCLUSIVE option prevents user programs from opening the database during reorganization. However, because reorganization updates are still audited, the REORGANIZATION program can be restarted.

The following example designates the EXCLUSIVE option:

```
GENERATE EMPLOYEES ORDER BY BYNAME; GLOBAL EXCLUSIVE;
```

Caution

Although the syntax enables you to specify both the INQUIRYONLY and the EXCLUSIVE options in the same statement, it is recommended that you specify only one of these options in a statement. If both options are specified, the BUILDREORG program ignores the INQUIRYONLY option since this option has fewer restrictions than the EXCLUSIVE option.

Using the PREVERIFY Option

The PREVERIFY option initiates a preverification task for each generated data set before reorganization actually begins. This task locks the structure, preventing changes until reorganization starts. It then reads all records in the data set to ensure the verify condition for records (for example, that required items are not null).

If a program in transaction state attempts to lock a record in a structure while a preverification task is occurring, a DEADLOCK number 2 exception occurs.

If the program attempting the lock is not in transaction state, then the program waits until either of the following occurs:

- The time period designated by the MAXWAIT task attribute expires.
For more information on the MAXWAIT task attribute, refer to the *Task Attributes Programming Reference Manual*.
- The preverification of the structure completes.

For more information on structure locking, refer to the *Application Programming Guide*.

If all data sets pass preverification, the database structures are unlocked and the REORGANIZATION program runs. If any data set fails the preverification, reorganization does not begin. The errors should be corrected and the reorganization program restarted. If the database administrator decides the reorganization should not be run, it can be purged by restoring the description file and DASDL source that existed before the DASDL update that caused the reorganization, and removing the new DMSUPPORT file that was created. Since no reorganization tasks actually started and no audit records were written for the preverify task, nothing else should be changed. The PREVERIFY option cannot be used to test for duplicate conditions on spanning sets.

The following example designates the PREVERIFY option:

```
GENERATE EMPLOYEES ORDER BY BYNAME;  
GLOBAL PREVERIFY;
```

If the reorganization is abnormally terminated during preverification, it must be restarted by running the REORGANIZATION program. Since the database is not being updated, DMRECOVERY does not encounter audit records that indicate a reorganization is in progress. Consequently, if you run DMRECOVERY, the system most likely displays the following message:

```
RECOVERY NOT NECESSARY
```

Using the OFFLINE Option

The OFFLINE option prevents user programs from accessing the database structures being reorganized until the reorganization process is complete. No reorganization updates are audited. However, the reorganization is restartable at the row level of the structure.

During the reorganization process using the OFFLINE option, database structures that are not being reorganized can be accessed by user programs unless additional open options restricting access (for example, INQUIRYONLY or EXCLUSIVE) were included in the BUILDREORG specifications. To rebuild the database after a reorganization using the OFFLINE option, you must perform a dump of any reorganized structures. When the reorganization has successfully completed, you are prompted to perform a database dump unless you have specified the NOPOSTDUMP option. Refer to "Using the NOPOSTDUMP Option" in this section for more information.

While the reorganization, in general, is faster using the OFFLINE option, the REORGSUPPORT library uses more SAVE memory, and the SAVE memory increases with each generate or fixup task. If you are reorganizing a large number of structures using the OFFLINE option, and your system is low on available memory, then your system might run out of memory and/or issue the SORT ERROR #24 error message. If the reorganization is abnormally terminated, the accumulated SAVE memory is freed. You can restart the reorganization and it should complete normally. For additional information, refer to "SORT Errors During Reorganization" later in this section.

If the dataset being reorganized is sectioned, the reorganization task can take much longer than if it is not sectioned. The records are evenly distributed throughout the sections. At restart points, the buffers for all the sections are flushed. The time needed to create these restart points becomes significant as the number of sections and the number of areas increase. To lessen the impact of the increased reorganization time, it might be desirable to use the USEREORGDB option, which allows the database to remain available throughout the reorganization.

Caution

The rebuild or reconstruct recovery operations cannot run through an OFFLINE reorganization. To rebuild or reconstruct the database, you must dump reorganized structures after the reorganization.

The following example designates the OFFLINE option:

```
GENERATE EMPLOYEES ORDER BY BYNAME; GLOBAL OFFLINE;
```

Combinations for the Open Options

The following tables summarize the possible combinations of the open options.

INQUIRYONLY

Reorganization Audited?	Can Reorganization be Restarted?	Can a Rebuild be Done Through the Reorganization?	Which Structures Can be Accessed?
Yes, reorganization updates are audited.	Yes, restartable at the record level depending on the type of structure.	Yes, but cannot be done through a reorganization that adds or deletes structures or changes the number of sections in an extended data set. Cannot rebuild through an update level increase that did not involve an update reorganization.	User programs can open the database for inquiry only.

Reorganizing the Database

EXCLUSIVE

Reorganization Audited?	Can Reorganization be Restarted?	Can a Rebuild be Done Through the Reorganization?	Which Structures Can be Accessed?
Yes, reorganization updates are audited.	Yes, restartable at the record level depending on the type of structure.	<p>Yes, but cannot be done through a reorganization that adds or deletes structures or changes the number of sections in an extended data set.</p> <p>Cannot rebuild through an update level increase that did not involve an update reorganization.</p>	User programs cannot open the database.

OFFLINE

Reorganization Audited?	Can Reorganization be Restarted?	Can a Rebuild be Done Through the Reorganization?	Which Structures Can be Accessed?
Reorganization updates are not audited, but there are audit records for items such as database stack initiations and reorganization state changes.	Yes, restartable at the row level depending on the type of structure.	<p>Yes, but cannot be done through a reorganization that adds or deletes structures or changes the number of sections in an extended data set.</p> <p>Cannot rebuild through an update level increase that did not involve an update reorganization.</p> <p>To rebuild through a reorganization that uses the offline option, a dump is required of the reorganized structure or structures after the reorganization.</p>	<p>User programs can only access structures that are not being reorganized.</p> <p>The reorganization generally runs faster but uses more save memory.</p>

INQUIRYONLY and EXCLUSIVE

Reorganization Audited?	Can Reorganization be Restarted?	Can a Rebuild be Done Through the Reorganization?	Which Structures Can be Accessed?
Yes, reorganization updates are audited.	Yes, restartable at the record level depending on the type of structure.	Yes, but cannot be done through a reorganization that adds or deletes structures or changes the number of sections in an extended data set. Cannot rebuild through an update level increase that did not involve an update reorganization.	User programs cannot open the database.

INQUIRYONLY and OFFLINE

Reorganization Audited?	Can Reorganization be Restarted?	Can a Rebuild be Done Through the Reorganization?	Which Structures Can be Accessed?
Reorganization updates are not audited, but there are audit records for items such as database stack initiations and reorganization state changes.	Yes, restartable at the row level depending on the type of structure.	Yes, but cannot be done through a reorganization that adds or deletes structures or changes the number of sections in an extended data set. Cannot rebuild through an update level increase that did not involve an update reorganization. To rebuild through a reorganization that uses the offline option, a dump is required of the reorganized structure or structures after the reorganization.	User programs can open the database for inquiry only. User programs can only access structures that are not being reorganized. The reorganization generally runs faster but uses more save memory.

EXCLUSIVE and OFFLINE or INQUIRYONLY, EXCLUSIVE and OFFLINE

Reorganization Audited?	Can Reorganization be Restarted?	Can a Rebuild be Done Through the Reorganization?	Which Structures Can be Accessed?
Reorganization updates are not audited, but there are audit records for items such as database stack initiations and reorg state changes.	Yes, restartable at the row level depending on the type of structure.	<p>Yes, but cannot be done through a reorganization that adds or deletes structures or changes the number of sections in an extended data set.</p> <p>Cannot rebuild through an update level increase that did not involve an update reorganization.</p> <p>To rebuild through a reorganization that uses the offline option, a dump is required of the reorganized structure or structures after the reorganization.</p>	<p>User programs cannot open the database.</p> <p>Reorganization generally runs faster but uses more save memory.</p>

Default

Reorganization Audited?	Can Reorganization be Restarted?	Can a Rebuild be Done Through the Reorganization?	Which Structures Can be Accessed?
Yes, reorganization updates are audited.	Yes, restartable at the record level depending on the type of structure.	<p>Yes, but cannot be done through a reorganization that adds or deletes structures or changes the number of sections in an extended data set.</p> <p>Cannot rebuild through an update level increase that did not involve an update reorganization.</p>	<p>User programs can open the database for inquiry and update during the reorganization.</p> <p>Refer to "Availability of Structures During Reorganization" later in this section for additional information.</p>

Using the NOPOSTDUMP Option

Note: The *NOPOSTDUMP* option can only be used with the *OFFLINE* option.

If you specify the *OFFLINE* option, the *REORGANIZATION* program generates a message reminding you to perform a database dump and then waits for an *AX* (Accept) command at the end of the reorganization process. The reorganized structures are not available to user programs until you provide a response in the form of an *AX OK* command or an *AX SKIP* command.

If you also specify the *NOPOSTDUMP* option, the *REORGANIZATION* program only prompts you to perform a dump; it does not wait for you to perform the dump. The reorganized structures are made available to the user programs right after the message is displayed. You are responsible for performing the database dump as soon as possible.

The *NOPOSTDUMP* option is reset by default.

The *NOPOSTDUMP* option is set implicitly in any of the following situations:

- When using the *OFFLINE* option and *EXCLUSIVE* option
- When using the *OFFLINE* option and reorganizing a structure that is a member of the *RESTART* data set family

Using the USEREORGDB Option



Before initializing the *REORGDB* reorganization through the *USEREORGDB* option, the database must have the *INDEPENDENTTRANS* option set through a separate *DASDL UPDATE*. The *INDEPENDENTTRANS* option cannot be added during a *REORGDB* reorganization.

When the *USEREORGDB* option is specified for a reorganization, both the updated and previous description files must be available. Both the newly updated description file, *DESCRIPTION/<database name>*, and the backup copy created by *DASDL*, *DESCRIPTION/<database name>/<previous level>*, must be present at the beginning of the reorganization process.

The procedures handling the Swap process that occurs when the *USEREORGDB* option is used cannot safely handle global transactions. If the *USEREORGDB* option is used in an Open Distributed Transaction Processing environment, the *AUTOSWAP* option must be set to *FALSE* and any Open Distributed Transaction Processing activity processing must be completed before you allow the Swap process to start.

Note: If you are reorganizing a permanent directory database, the database administrator must make space available for the work files in the permanent directory in which the database resides.

The *USEREORGDB* option can be used for garbage collection or with the *DASDL UPDATE* option to reorganize structures for

Reorganizing the Database

- File and record format conversions on structures that have the XE attribute set
- Conversions from a structure without the XE attribute set to a structure that does have this attribute set
- XE structures, during garbage collection

The USEREORGDB option cannot be specified for use

- With databases that are not audited
- With databases that have been created by setting the MODEL option in DASDL
- With data structure pack family changes made using DASDL
- For embedded structures or manual subsets
- During a reorganization of the restart data set
- When specified in combination with the INQUIRYONLY, PREVERIFY, EXCLUSIVE, OFFLINE, or NOPOSTDUMP options
- With data sets containing aggregate items
- If sets are reorganized, unless the data set is also generated
- To delete sets or data sets
- To add new sets or datasets with no other changes
- To reorganize the global data set
- With data sets that are accessed by a manual subset
- With a structure that is the object of an unprotected link, when the structure owning the link meets one of the following conditions:
 - Contains a nonqualifying link type
 - Is referred to by an aggregate item
 - Is an embedded data set
 - Is accessed by a manual subset

The following supported structure types can be reorganized when the USEREORGDB option is specified:

- Standard fixed-format
- Standard variable-format
- Compact
- Direct
- Index sequential

The preceding disjoint structure types can also include, or be the target of, unprotected links. When using a REORGDB reorganization to generate a data set that is an object data set of a link item, both the data set and the data set containing the link item need to be involved in the reorganization. To accomplish this, REBUILDREORG implicitly includes a fixup for the owning data set. This action enables both the owner and object data set sides of the link updates to be captured.

If only the owning data set is reorganized using REORGDB, it is also necessary to capture information for both the owner and object data sets because the actual reorganization process takes place offline in another database. To accommodate this information, an explicit generate statement for the object data set must be included in the BUILDREORG specifications.

The unprotected link is the only link type supported by the REORGDB reorganization. No other link types are allowed to be used with structures, owners, or targets that use this form of reorganization.

Allowed disjoint data set types are permitted to contain embedded data sets. However, the embedded structures cannot be reorganized using the USEREORGDB option. This also means structures without the XE attribute set that contain embedded data sets cannot migrate to XE structures using this mechanism.

Caution

The rebuild or reconstruct recovery operations cannot run through an OFFLINE or REORGDB reorganization. To rebuild or reconstruct the database, you must dump reorganized structures after the reorganization.

Using the REORGDBTITLE Phrase



The REORGDBTITLE phrase specifies the name of the database copy used during a REORGDB reorganization. The specified name must not be the same as that of the production database. However, the usercode of the REORGDBTITLE phrase must be the same as that of the production database. This option is required in combination with the USEREORGDB option.

Notes:

- *When reorganizing multiple databases under the same usercode, be sure to specify different values for the REORGDBTITLE phrase.*
- *For a permanent directory database,*

- *The specified title must not include a usercode. The title is used with the datapath of the live database to determine the name of the database copy.*
- *The administrator must also create a node for the temporary database prior to running the reorganization.*

Using the AUTOSWAP Option



The AUTOSWAP option indicates how the swap phase is to be initiated during a REORGDB reorganization. When set to TRUE, the option signifies that manual intervention is not used and that the reorganized structures are to be put into service as soon as possible following completion of the reorganization. Once all of the reorganized structures have reached their first synchronization point, the swap phase is initiated.

When database activity resumes, any active application that does not match the level of the opened structures receives a version error.

When this option is set to the default value of FALSE, the REORGANIZATION program issues the following message, indicating that the reorganized structures have reached a synchronization point and that the structures are ready to be put into service:

```
AUTOSWAP IS RESET FOR USEREORGDB REORGANIZATION,  
WAITING FOR OPERATOR SWAP REQUEST.
```

The REORGANIZATION program waits for you to respond with the following system command, indicating readiness to begin the swap phase:

```
SWAP NOW
```

Once commanded, the following swap actions occur:

1. The active applications pause briefly
2. The final synchronization takes place
3. The reorganized structures are enabled for service
4. The applications are allowed to continue.

When database activity resumes, any active application that does not match the level of the structures that it has open will receive the customary Version Error.

Using the NORESTART Option

The NORESTART option indicates that reorganization cannot be restarted. Selecting this option enables the reorganization of the database copy designated by the REORGDBTITLE to proceed without performing excessive I/O operations that can slow the process. The process is especially slowed when reorganized structures are sectioned.

Using the TOTALCOPYCORE and STRCOPYCORE Phrases

The TOTALCOPYCORE and STRCOPYCORE phrases enable you to specify values within the BUILDREORG process that determine the number of copy tasks that can be running at the same time.

The overall TOTALCOPYCORE default value is 5 million words, with a maximum of 100 million words and a minimum of 1 million words. Specifying larger values within that range might be useful for databases with structures having a large number of sections.

The default STRCOPYCORE value for each structure is 250,000 words, with a maximum of 20 million words and a minimum of 150,000 words. This value is applied at the structure level. For sectioned data sets, the value is divided among the sections.

The specified STRCOPYCORE value cannot exceed the value of the TOTALCOPYCORE specification. If the STRCOPYCORE value does exceed the TOTALCOPYCORE specification, the BUILDREORG process will increase the TOTALCOPYCORE specification to equal the highest single STRCOPYCORE value so that the copy tasks can run.

Each copy task, by structure or by data set section, is allocated two 65K buffers. These buffers can be less than 65K if the AREASIZE value equals less than 65,000 words.

As the copy task progresses, if there is an update to the production database the update is duplicated to the database copy file. The update duplication process maintains a pool of 10 small buffers for duplicate updates and rotates serially through the pool. If excessive wait times result during the process, the number of buffers in the pool is increased to the structure limit of the STRCOPYCORE values. When this occurs, the following message is displayed:

```
Pool buffer in use from a previous duplicated write
```

Note: Copy memory for the small buffers is held only until all copy tasks finish.

Before all copy tasks are completed, all modified buffers are written. At that point, all copied files are identical to the production database files. The capture process takes over. No more production database updates are duplicated to the copy files, and all copy memory is deallocated. The two 65K copy buffers are de-allocated as each file completes its copy task (is duplicated to the corresponding file on the copy database), but the small buffer pool continues to be used to copy the production database updates until all of the remaining copy tasks finish.

This mechanism allows the parameters to be tuned for activity-based processing rather than task-based processing. The advantage is that heavily updated structures can be given more memory and more buffers to be used for quickly processing updates during the copy.

Using the REORGDBALLOWEDCORE Option

The REORGDBALLOWEDCORE option indicates the amount of core memory that can be used for the reorganized database copy. This option for the REORGDB mode of reorganization serves the same purpose as the database ALLOWEDCORE parameter that is specified in DASDL. The maximum number of words allowed for the REORGDBALLOWEDCORE option is $2^{28}-1$, and the minimum allowed is 1,000,000. The default is 1,000,000 words.

The REORGDBALLOWEDCORE option serves the role of database ALLOWEDCORE parameter only during the processing of updates. It does not have a role during the background OFFLINE reorganization.

Using the CHECKFILESIZE Option

If you specify the CHECKFILESIZE option, the REORGANIZATION program will check the LASTRECORD file attribute at the beginning of the reorganization process. If the file size is larger than $2^{28}-1$ sectors, the REORGANIZATION program will exit gracefully and require users to perform a DASDL update to change or add more physical sections to the structure. This prevents the REORGANIZATION program from failing with a fatal error when the database file exceeds the file size limit. By default, this option is reset. However, setting this option is recommended.

The REORGANIZATION will perform normally if the number of physical sections are increased even though the option is set and file size exceeds the limit.

Running the REORGANIZATION Program

To perform database reorganization, you run the REORGANIZATION program. The actual reorganization is accomplished through procedures in the Accessroutines in conjunction with the tailored REORGANIZATION program. For this reason, it is important that the REORGANIZATION program is generated and run on the same level of Enterprise Database Server software.

For a permanent directory database, the reorganization program must be run with a datapath specification.

```
RUN REORGANIZATION/<database name>; DATAPATH=<datapath>  
ON <pack name>
```



You can also perform an online garbage collection of disjoint index sequential sets by using the Visible DBS GARBAGE COLLECT command while the database is running. Initiating this command results in a garbage collection of the specified sets while they are in use. You do not run the REORGANIZATION program. For the syntax for the Visible DBS GARBAGE COLLECT command, refer to [Section 12, Communicating with the Database](#).

Preparing to Reorganize

If the reorganization allows multiple reorganization tasks to run simultaneously (that is, TASKLIMIT option is greater than 1), refer to “DASDL ALLOWEDCORE” under “Enhancing Reorganization Performance” later in this section.

Before you reorganize the database, back up the following from the original database:

- Control file and all the database files

The reorganization creates a discontinuity in these files. This discontinuity prevents ROLLBACK recovery into the reorganization region of the audit file. The rebuild process can recover these files from a dump performed before reorganization. The rebuild process essentially reorganizes the database again.

Use the DMUTILITY *DUMP* command as described in [Section 6, Backing Up a Database](#). If you cannot dump the entire database, dump only the database files that need reorganization.

- Audit trail
- DASDL source
- Description file

If the reorganization process fails and cannot be restarted successfully, the database files, control file, description file, and all tailored software must be reloaded from backup dumps.

You can restrict user access to a database to inquiry-only by using the BUILDREORG open option INQUIRYONLY. Then, you can use the backup copy of the database to restore the database after a fatal error or to essentially cancel a reorganization.

If you use the LOCKEDFILE attribute for database files, you must also use the LOCKEDFILE option in the DASDL. If you did not set the LOCKEDFILE option in the DASDL and the LOCKEDFILE attribute is set on any of the structures involved in the REORGANIZATION, you must reset the LOCKEDFILE attribute for all pertinent structures before you start the reorganization. Otherwise, problems can occur.

Understanding the Phases of Reorganization

If any of the reorganization tasks fail and leave a structure marked as dead, the REORGANIZATION program is assigned a status of ABORTED and the TASKVALUE attribute is assigned a task value of 2. If such a condition occurs, identify the REORGANIZATION program task that failed and take an appropriate action.

The program REORGANIZATION/<database name> initiates the reorganization and acts as a driver and monitor for the reorganization tasks. The actual reorganization tasks (generates and fixups) are processes of the database stack. Each generated structure goes through a series of phases during the reorganization. These phases are independent of other structures. The phases are as follows:

Reorganizing the Database

1. The rename phase. The old file title has /OLD appended to the end.
2. The COPY phase. If a STRUCTURE COPY specification is included in the BUILDREORG specifications, the old file is copied to the designated pack.
3. The FIXUP file initialize. The fixup file is preinitialized to all zeros.
4. The new file initialize. An empty new file is created and its timestamp entered into the control file.
5. The GENERATE phase. Records are moved and reformatted from the old file to the new file.
6. The FIXUP phase. Affected structures are fixed up.
7. The REMOVE phase. Temporary internal files are removed.

Starting the REORGANIZATION Program

To perform or to restart the REORGANIZATION program, enter the following from CANDE:

```
RUN $REORGANIZATION/<database name>
```

For permanent directory databases, the reorganization program must be run with a datapath specified, as follows:

```
RUN REORGANIZATION/<database name>; DATAPATH=<datapath>  
ON <packname>
```

Refer to [Section 16, Using Permanent Directory Databases](#), for more information.

If the control file and the description file are located on different packs, file-equate the description file title using the internal file name DASDL. For example, if the reorganization program for the database MYDB resides on the ISYS pack and the description file resides on the DMS pack, use the following statement to run the reorganization program:

```
RUN $REORGANIZATION/MYDB; FILE DASDL=DESCRIPTION/MYDB ON DMS
```

If the SYSTEM/DMCONTROL code file does not have the same usercode as the REORGANIZATION program or is not located on the same pack family, the SYSTEM/DMCONTROL code file can be file-equated as follows:

```
RUN REORGANIZATION/<database name>;  
FILE DMCONTROL (TITLE=*SYSTEM/DMCONTROL ON <pack name>)
```

In addition to the optional equation statements, you must satisfy the following requirements when reorganizing a modeled database with the USEREORGDB option:

1. The internal database DB must be equated.
2. The file DESCRIPTION/<model db name>/<previous level> must be present if the reorganization is an UPDATE reorganization. This requirement ensures that the reorganization program cannot be run for the model unless the DASDL UPDATE compilation has occurred. An example of the database equation statement is

```
RUN REORGANIZATION/<parent db>;  
DATABASE DB (TITLE = <model database name>);
```

Note: If a guard file is used to guard a database, the usercode under which the REORGANIZATION program is to be run should be allowed to access the database files.

A reorganization run can also be executed as in the following Work Flow Language (WFL) job deck:

```
? BEGIN JOB REORGANIZATIONRUN;
  TASK REORG;

  RUN REORGANIZATION/<database name> [REORG];
  IF REORG ISNT COMPLETEDOK THEN ABORT
  "*** BAD REORG RUN ***";
```

Note: If a failure occurs that requires the entire reorganization to be rerun, then the COMPLETEDOK status is false.

However, if one of the reorganization tasks fails and leaves a structure marked as dead, the REORGANIZATION program is assigned a status of COMPLETEDOK and the TASKVALUE attribute is assigned the value 2. Identify the reorganization task that failed and take appropriate action.

Reorganizing a Nonusercoded Database

To reorganize a nonusercoded database, run the REORGANIZATION program with the asterisk (*) designation. If a guard file is used to guard the database, the REORGANIZATION program running under the nonusercode (*) should be allowed to access database files during the reorganization run. If the ZIP option is reset during a BUILDREORG run, then the security type of the REORGANIZATION program should be made public.

Using the Transaction Processing System (TPS) During Reorganization

During a reorganization that allows updates, TPS update programs can run. The following rules apply when the DASDL TPSDUALUPDATE option is not set:

- If a regular data management program is the first to open the database for updates after the reorganization starts, then TPS update programs cannot access the database during the reorganization.
- If a TPS program is the first to open the database for updates after the reorganization starts, then regular data management update programs cannot access the database during the reorganization.
- During the reorganization, both TPS and regular data management inquiry programs can run.

For more information on the TPSDUALUPDATE option, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

Availability of Structures During Reorganization

The entire reorganization is complete when the REORGANIZATION program finishes. Depending on the open options designated in the BUILDREORG specification, structures in the database become available on a structure-by-structure basis. Structures that are reorganized using the OFFLINE option cannot be accessed until the entire reorganization is complete. Given that user programs can access the database during an ONLINE reorganization, the following rules apply to the availability of structures:

- Structures unaffected by the reorganization are available immediately.
- A deadlock occurs if the user program is in transaction state and waiting on a structure to become available, and the REORGANIZATION program needs to take a syncpoint. If the INDEPENDENTTRANS option is set for the database, the user program receives a DMVERSIONERROR #3 message because the reorganized structure is unavailable, and the REORGANIZATION program continues. If the INDEPENDENTTRANS option is not set, the database terminates with a fatal error. If update programs are to be run during an ONLINE reorganization, it is recommended that the INDEPENDENTTRANS option be set for the database.
- Application programs accessing structures that are being reorganized cannot be run at a higher priority than the reorganization tasks because records must be reorganized before being presented to the application program. If a compact structure is being reorganized, an application program that performs a FIND NEXT operation through the data set and is running at a higher priority can receive an erroneous NOTFOUND condition. This condition occurs because no records have been reorganized to the new structure yet, and the application program tries to read an empty file.
- Structures affected by the reorganization are not available until at least the associated reorganization task has started.
- Disjoint ordered or unordered data sets are not available until their reorganization is complete.
- Direct and random data sets are available for inquiries that do not involve a linear search, links, or access through the data set.
- Reorganized data sets other than direct, random and disjoint ordered or unordered are available as soon as their associated reorganization task starts if no spanning set is generated in the same run and if no spanning set is an ordered list that allows DUPLICATES and must be fixed up. The generation of the spanning set might occur automatically, or it can be explicitly requested in the BUILDREORG specifications.

If the spanning set is generated or an ordered list set allowing DUPLICATES is fixed up, the data set is available for inquiry but not for any of the following updates that affect the key or the key data of the set:

- CREATE/STORE
- DELETE
- MODIFY/STORE
- The operation *FIND LAST <data set>* waits until the generation of the data set is complete.
- Sets that are generated are not available (even for inquiry) until their generation is complete, as opposed to started.

- If a SORT specification is requested when generating a set from a data set, both the set and the data set are available only for inquiries through a set that is not being generated by this reorganization run.
- If the restart data set is to be reorganized and you are using the Open Distributed Transaction Processing product, the RMSUPPORT library waits until the reorganization of the restart data set is complete before opening the database.

If a user program attempts to access an unavailable structure, one of the following situations occurs:

- The program might wait in or out of transaction state. If the waiting program has locked records and another program attempts to lock those records, the program already waiting for the structure to become available receives a VERSIONERROR number 3 message. (VERSIONERROR number 3 is a reference to an unavailable reorganized structure.)
- If the MAXWAIT task attribute is nonzero, the program waits MAXWAIT seconds or until the structure becomes available, whichever is first. If MAXWAIT seconds expires, a VERSIONERROR number 3 message is returned.
- If the MAXWAIT task attribute is zero, the program waits until the structure becomes available, or until the program is discontinued.
- The operation *FIND <data set>* for RANDOM data sets or their accesses waits until the random data set generation is complete. However, if the program is in transaction state, it immediately receives a VERSIONERROR number 3 message.

A user program can interrogate the availability of a structure by setting the MAXWAIT task attribute to a small value, and performing an innocuous inquiry (such as FIND FIRST). If no error is returned, the structure is available, otherwise it is unavailable.

For more information on the MAXWAIT task attribute, refer to the *Task Attributes Programming Reference Manual*.

Availability of Structures During REORGDB Reorganization

All structures, including those affected by the REORGDB reorganization, are available during REORGDB reorganization. However, updates to LOB items in the reorganized structures are not allowed during a REORGDB Reorganization. In this case, a READONLY 4 is returned to the application.

To provide proper recovery synchronization, the Accessroutines must wait at two points for a real end transaction (ETR) request from all programs in transaction state; this includes OLTP global transactions. All programs updating the production database are briefly suspended during these two phases of the reorganization:

- After structures are copied to the database copy
- During the swap phase of the reorganization

Swap Phase

The frequency of control points and the number of user application updates per transaction greatly affects the time between the swap phase and the completion of the updates application phase. Typically, at the end of the apply-updates tasks, the amount of time that the applications are actually suspended is not very long. However, lengthy transactions and high transaction volumes in conjunction with infrequent control points can extend this period.

The following steps explain the process that occurs during the swap phase of the reorganization:

1. The apply-updates actions that place the production database updates into the copy database start once the reorganization of the copy database completes. Many updates, control points, and so on can occur on the production database while the reorganization of the copy database is running, thus accumulating a large number of changes to apply to the newly reorganized data structures.
2. Once the apply-updates period starts, the copy database is at least two control points behind the production database. Production database (capture audit file) updates are not released to the apply-updates tasks until they have been committed for two control points.
3. When the apply-updates tasks have reached the first EOF on all files containing the structure updates (the capture audit files), the first potential swap point has been reached. Either the automatic swap mechanism starts, or the AX TO SWAP NOW RSVP message is issued.

When the swap begins, two control points are forced in order to cut off the activity of the current two control points on the live database. This action enables the apply-updates tasks to work on those database updates while the production database is starting a new two-control-point cycle. But, if the new two-control-point cycle is going to be long too, only a comparatively small amount of progress might be made. This is when having a very large number of updates per database transaction can be a disadvantage. The apply-updates tasks are always two control points behind the production database.

The biggest issues are the control point frequency and the number of production database updates per transaction. As a general rule, long transactions are not desirable for any database performing online transactions. Specifying the DASDL option SYNCWAIT does help the situation, but there can still be long recovery times in order to back out the long transactions.

Note that all applications remain fully active during this period.

4. After step 3 completes, there is another EOF check on all capture audit files, and the actual swapping code is entered. Once the swapping code is entered, all updates to the production database are stopped. However, inquiries continue. Two final control points are forced on the production database to release all remaining updates. These control points are forced so that there are fewer transactions to apply.

However, if these remaining transactions are mostly composed of lengthy transactions and each contains a very large number of database updates, then the apply-updates period is extended. This extension is especially noticeable if the forced control points occur near the time when a control point would normally occur. In this

case, it is not so much the long transactions themselves but rather the quantity of updates that they contain. For a different case, such as an online application with low transaction volumes, the apply-updates actions could be much faster than the actual time needed to create the transactions.

Once the remaining updates have been applied, the inquiries are also stopped and the actual swap takes place.

For nearly all of this time, the production database remains available. The time between the “application stopped” message and the end of the swap when the live database becomes available is short. This process is designed to minimize the length of time that application programs are unavailable during the swap activity.

Updating During Generation: The Fixup Process

Database structures that are not reorganized but have links that point to reorganized structures must be updated during reorganization. This process is referred to as a fixup, or address update, and is automatically performed in place by the REORGANIZATION program. There are three cases where fixup is performed:

- If the reorganized structure is embedded, and root word links in the master are updated.
- If the data set is generated, all index structures in the data set family not explicitly or implicitly generated have their links to the data set updated.
- DASDL link items pointing to it have their links updated.

All structures requiring fixup must be present during reorganization. The report produced by SYSTEM/BUILDREORG lists all structures that require fixup.

Fixup performance is generally influenced by the following factors:

- Population of the master data set structure as determined by the REORGANIZATION program
- The specified BUILDREORG ALLOWEDCORE value
- The memory available for the fixup task
- The bias of the set with respect to the data set records

When the set structures are identified by the REORGANIZATION program as large sets or when available memory is deficient, the fixup algorithms provide optimization for better performance. One of these fixup algorithms involves sorting the fixup file. If the sort operation fails, a message is displayed and the fixup task continues as if the sort operation had not been attempted. This action allows the reorganization to complete successfully, but the reorganization takes longer than it would have if the sort operation had been successful.

The fixup file is used in most cases, but the REORGANIZATION program does not need, nor does it create, a fixup file when all of the following conditions are met:

- The data set has no links.
- The data set is not embedded.

- All sets are being generated using the data set.
- The data set is generated first.
- The reorganization process is run using the EXCLUSIVE option or the OFFLINE option.

When a fixup file is not created, because of the conditions previously mentioned, the REORGANIZATION program saves system resources.

Sequencing Fixups: Three Cases

A data set might require fixup for several different structures during the same reorganization run. For example, a data set can contain multiple embedded structures and can have multiple links referencing other structures. Any number of these embedded structures or linked structures can be generated during the same reorganization.

Three rules control the order and the combination of generations and fixups:

- No fixups start until all of the affected structure generates are complete.
- All fixups for a given data set are combined in the same fixup run.
- A data set fixup is never combined with its generation. If a data set and one of its index sets are both generated in the same reorganization run, the generates can be ordered data set first or index set first. If the index set is generated first, availability is maximized because the data set is not completely available until the index set generation is complete. However, a second pass of the index set is required for the index set fixup. If the data set is generated first, availability is restricted until the index set generation is completed. In this case, the index set fixup can be combined with the index set generation, which minimizes the time it takes to complete a reorganization.

Examples of Reorganization Sequencing

In this example, the database administrator performs garbage collection on the following database:

```
D1 DATA SET
(
  A1 ALPHA (6);
  A2 ALPHA (8);
  A3 ALPHA (10);
  G GROUP
  (
    G1 NUMBER (3);
    G2 NUMBER (6);
    G3 ALPHA (6);
  );
  N1 NUMBER (4);
  N2 NUMBER (6);
  R1 REAL;
);

DS1 SET OF D1 KEY IS A1;
DS2 SET OF D1 KEY IS A2;
```

```

DS3 SET OF D1 KEY IS (G1, G2);
DS4 SET OF D1 KEY IS N1;

E1 DATA SET
(
  A1 ALPHA (6);
  A2 ALPHA (8);
  A3 ALPHA (10);
  G GROUP
  (
    G1 NUMBER (3);
    G2 NUMBER (6);
    G3 ALPHA (6);
  );
  N1 NUMBER (4);
  N2 NUMBER (6);
  R1 REAL;
);

ES1 SET OF E1 KEY IS A1;
ES2 SET OF E1 KEY IS A2;
ES3 SET OF E3 KEY IS (G1, G2);

```

The database administrator can specify the following reorganization sequence and resource allocation to the BUILDREORG utility:

```

GENERATE D1, DS1, DS2, DS3 USING DS3, DS4;

DS1(LOADFACTOR = 90);
DS3(LOADFACTOR = 80);

D1(COPY TO TEMPPACK);
DS1,DS2(COPY TO DISK);

SEQUENCE DS2, D1, DS1 AND DS4, DS3;

GENERATE E1 ORDER BY ES1, ES1, ES2, ES3 USING ES3;

ES1(LOADFACTOR = 90);
ES3(LOADFACTOR = 80);

E1(COPY TO TEMPPACK);
ES1,ES2(COPY TO DISK);

SEQUENCE ES2, E1, ES1 AND ES3;

CENTRAL DATASET SEQUENCE D1 AND E1;
GLOBAL TASKLIMIT = 2;
  ALLOWEDCORE = 300000;
  INTERNAL FILES(FAMILYNAME = SYSPACK);

```

Finishing the Reorganization Process

During the reorganization process, the REORGANIZATION program and the description file are copied with the following titles:

Reorganizing the Database

```
REORGANIZATION/<database name>/<YYYYMMDD>/<HHMM>  
DESCRIPTION/<database name>/<YYYYMMDD>/<HHMM>
```

If the reorganization was required because of a DASDL update, the title of the DMSUPPORT library is also changed to include the old update level:

```
DMSUPPORT/<database name>/<update level>
```

Save these files in case a rebuild recovery through the reorganization region of the audit is necessary.

During the life of a database, the database reorganization process is used to make physical changes to database files and records. The reorganization tasks allow changes to the format of an existing database. These tasks can be used to

- Reorder and consolidate data sets, sets, and subsets.
- Generate new sets or automatic subsets in order to allow more rapid access to data sets.

Note: *Multiple data sets and their spanning sets and subsets can be reorganized at one time.*

- Change the record formats by adding, deleting, or changing fields.

The REORGANIZATION program accepts properties described through DASDL and BUILDREORG and uses these properties to perform file and record format conversions. The REORGANIZATION program also increments the database update level.

During the REORGDB mode of reorganization, record and file format changes take place on a production database copy that is then synchronized at a designated point called the swap point. While the swap takes place, the active application programs are suspended to preserve the synchronization. Upon completion of the swap, the applications are allowed to resume.

Note: *Once the applications resume, version errors are given to those applications that have not been compiled to match the level of the reorganized structures that they affect.*

The synchronization process between the production database and the reorganized database copy is accomplished by initiating a Catch-up task for each structure that is being reorganized. The task reads a capture file (unique to each task) and applies the updated information to the reorganized data sets.

Once all of the tasks are completed, either the REORGANIZATION program emits a message to inform the operator that the reorganized structures can be exchanged between the production database and the reorganized database copy, or it performs the exchange automatically. The choice of methods (operator control or automatic) is a BUILDREORG option.

If the AUTOSWAP option is set to FALSE, either explicitly or by default, then the REORGANIZATION program emits the following message, indicating that the reorganized structures have reached a synchronization point:

```
AUTOSWAP IS RESET FOR USEREORGDB REORGANIZATION,  
WAITING FOR OPERATOR SWAP REQUEST.
```

You must respond with the AX SWAP NOW system command to begin the swap phase, which means the newly reorganized data structures can be placed into service at the next synchronization point.

If the automatic swap mechanism is enabled by setting the AUTOSWAP option to TRUE in the BUILDREORG syntax, the following message is emitted at the first opportunity to begin the swap phase:

```
AUTOSWAP IS SET FOR USEREORGDB REORGANIZATION
```

As soon as the message is delivered, the automatic process begins.

Note: The reorganization synchronization points are not Enterprise Database Server syncpoints.

Reorganization Status Report

A reorganization status report is generated and sent to the printer at the completion of the reorganization.

If the STATISTICSLOC command is specified, the report is stored at the designated pack with the title

```
<dbusercode>REORGSTATS/<dbname>/YYYYMMDD/HHMMSS on <pack name>
```

For a permanent directory database, the path name is assumed and the report is stored as

```
<path name>/REORGSTATS/<dbname>/YYYYMMDD/HHMMSS on <pack name>
```

The date and time when the report is generated is represented by YYYYMMDD and HHMMSS.

The report includes the mix number, name, number of entries processed, start time, and end time for each task processed during the reorganization.

DATABASE NAME:T5DB

MIX	PROCESS	STATUS	START TIME	END TIME
---	-----	-----	-----	-----
1065	GEN/T4/2	COMPLETED (650RECORDS)	6/10/2009 12:43:01	06/10/2009 12:43:02
1066	FIX/T4-S0/3	COMPLETED (8BLOCKS)	6/10/2009 12:43:02	06/10/2009 12:43:03
1067	GEN/T4-S1/7	COMPLETED (650RECORDS)	6/10/2009 12:43:03	06/10/2009 12:43:08
1073	GEN/T5/4	COMPLETED (800RECORDS)	6/10/2009 12:43:08	06/10/2009 12:43:09
1075	FIX/T5-S0/5	COMPLETED (10BLOCKS)	6/10/2009 12:43:09	06/10/2009 12:43:10
1076	GEN/T5-S1/8	COMPLETED (800BLOCKS)	6/10/2009 12:43:10	06/10/2009 12:43:15
1083	GEN/T5-S2/9	COMPLETED (800BLOCKS)	6/10/2009 12:43:15	06/10/2009 12:43:20

Reorganizing the Database

The following is an example of the additional information provided for a USREORGDB type of reorganization. SWAP PHASE (FILE) indicates that FILES SWAP has started, and SWAP PHASE (FINAL) indicates that FILES SWAP is completed.

DATABASE NAME:T5DB

USERORGDB PHASES	STATUS	START TIME	END TIME
COPY TO DATABASE COPY PHASE	COMPLETED	6/10/2009 13:08:10	6/10/2009 13:08:12
CAPTURE UPDATES PHASE	COMPLETED	6/10/2009 13:08:18	6/10/2009 13:08:42
OFFLINE REORGANIZATION PHASE**	COMPLETED	6/10/2009 13:08:18	6/10/2009 13:08:42
COPYBACK PHASE	NOT APPLICABLE		
APPLYUPDATESPHASE	COMPLETED	6/10/2009 13:08:42	6/10/2009 13:09:14
SWAP PHASE (FILE)	COMPLETED	6/10/2009 13:09:16	6/10/2009 13:09:23
SWAP PHASE (FINAL)	COMPLETED	6/10/2009 13:09:23	6/10/2009 13:09:30

**OFFLINE REORGANIZATION STATISTICS:

MIX	PROCESS	STATUS	START TIME	END TIME
2989	GEN/T4/2	COMPLETED(650RECORDS)	6/10/2009 13:08:18	6/10/2009 13:08:19
2990	FIX/T4-S0/3	COMPLETED(8RECORDS)	6/10/2009 13:08:19	6/10/2009 13:08:20
2991	GEN/T4-S1/7	COMPLETED(650RECORDS)	6/10/2009 13:08:20	6/10/2009 13:08:25
3001	GEN/T5/4	COMPLETED(800RECORDS)	6/10/2009 13:08:25	6/10/2009 13:08:26
3002	FIX/T5-S0/5	COMPLETED(10RECORDS)	6/10/2009 13:08:26	6/10/2009 13:08:27
3003	GEN/T5-S1/8	COMPLETED(800RECORDS)	6/10/2009 13:08:27	6/10/2009 13:08:32
3008	GEN/T5-S2/9	COMPLETED(800RECORDS)	6/10/2009 13:08:32	6/10/2009 13:08:37

Terminating and Reccessing the REORGDB Reorganization Process

The REORGDB mode of reorganization can be paused or discarded without affecting the production database. Either of these actions could prove useful for last-minute requirement changes or if you need to make additional system resources, such as memory or processors, available to the production environment.

You can recess the reorganization process before the reorganized data structures are placed into production. To recess the reorganization, give the REORGANIZATION program the RECESS directive by using the AX (Accept) system command. The REORGANIZATION

program removes all structure-specific reorganization tasks and proceeds to the end-of-job state. These actions do not affect the update captures from the production database. You can resume the reorganization process by restarting the REORGANIZATION program.

After the reorganization process is recessed, you can discard the entire reorganization by initiating the USEREORGDB TERMINATE command using the Visible DBS interface of the production database. After initiating this command, the database stops capturing updates for the reorganization process, and it is no longer possible to resume the reorganization.

Caution

Updates to the database continue to be captured following a RECESS command. If the reorganization is not terminated or restarted, the capture audit files continue to grow in size.

Perform the following procedure if you want to terminate and discard the entire reorganization with a single command:

1. Prior to placing the reorganized structures into service, give the REORGANIZATION program the TERMINATE directive by using the AX (Accept) system command.
The REORGANIZATION program signals all of the reorganization tasks to leave the mix and removes all reorganization-related files.
2. If the update level of the description file was changed during the reorganization process and you do not intend to restart the REORGANIZATION program, restore the original description file.
3. Proceed to the end-of-job state.
The production database will remain fully active.

Restarting a Reorganization

The REORGANIZATION program can be restarted after a halt/load. ONLINE reorganization tasks enter and leave transaction states just like any user of the database, and audit their changes to the Enterprise Database Server audit. After a halt/load, the DMRECOVERY utility can run explicitly, or the first user program to open the database runs the utility implicitly. After SYSTEM/DMRECOVERY runs, the REORGANIZATION program is automatically restarted and each reorganization task continues from where it left off.

When SYSTEM/DMRECOVERY restarts the REORGANIZATION program, it expects the REORGANIZATION program to have either the default title REORGANIZATION/<database name> or the title specified in the DASDL source file. In addition, it expects to locate the description file under the usercode and pack family of the user who initiated the recover. If the description file cannot be located, the reorganization program pauses with a NO FILE prompt. Use the MCP FA command to provide the usercode and pack name from where the reorganization was originally started, for example, <mix number> FA TITLE = (ORIGINALUC) DESCRIPTION/DBNAME/<level> ON ORIGINALPK.

The options are to either manually restart the reorganization program from the original usercode and pack family, or to restart the reorganization program using file equation as described under the subsection “Starting the REORGANIZATION Program” earlier in this section.

Reorganization audit records are not actually complete “beforeimages” and “afterimages” of the reorganized records. The beforeimage remains in the old file. The afterimage can be recreated by applying the reorganization move text command to the beforeimage. Therefore, reorganization audit records contain simply an address in the old file where the record came from, and an address in the new file where the new record was put.

If the PREVERIFY option is specified and the reorganization abnormally terminates during the preverification phase, the reorganization can be restarted only by rerunning the REORGANIZATION program. This restriction is because the database is not being updated and there are no audit records to indicate a reorganization is in progress.

Reorganizations using the OFFLINE option, although unaudited, can be restarted after a halt/load. In some cases, the reorganization can be restarted at the structure level.

Rebuild Recoveries and Reorganizations

Rebuild recoveries are allowed

- Completely through or into the middle of a reorganization region of the Enterprise Database Server audit
- Through multiple reorganizations

Whenever possible, avoid performing rebuild recoveries through reorganization because the rebuild recovery process requires all audit records associated with each ONLINE reorganization to be reprocessed, and because all the DMSUPPORT libraries and REORGANIZATION programs must be made available.

When rebuilding a database through one or more reorganization runs using a dump tape prior to reorganization and one or more dump tapes after reorganization, DMUTILITY requires that the control files on all dump tapes contain the exact same structures. That is, the reorganization must not have added or deleted any structures. The reorganization must also not have changed the number of sections in an extended data set.

When using the REORGDB option, if the COPY BACK option is not used with the COPY TO option, then you cannot rebuild from a time prior to the reorganization. If the COPY BACK option is not used with the COPY TO option, then you must perform a full dump of the entire database after the reorganization for future rebuild recoveries.

If a reorganization adds or deletes structures, or changes the number of sections in an extended data set, the following actions can be taken to help ensure a successful rebuild of the database:

- For rebuilding through an ONLINE reorganization, use only the database dump created prior to the reorganization.
- For OFFLINE or ONLINE reorganizations, a full database dump must be performed after the reorganization to avoid having to rebuild through the reorganization.

When rebuilding through one or more reorganizations, the REORGANIZATION program is responsible for migrating between DMSUPPORT libraries each time the database update level increases. Update level increases outside of the reorganization process cannot be processed with a rebuild recovery through a reorganization.

If possible, complete one of the following actions when performing an update level increase:

- Create a database backup after the update level increase.
- Combine the update level increase with a reorganization.

If a rebuild recovery is to recover through a reorganization region, you must copy the control file from the dump tape before you initiate the rebuild recovery.

When you use rebuild recovery to recover through a reorganization region, do not use the FROM MOST CURRENT option. You must also specify a recover source in the RECOVER statement.

The required naming convention for the DMSUPPORT and REORGANIZATION program code files and the description file are as follows. The run date has the format YYYYMMDD, and the run time has the format HHMM.

```
DMSUPPORT/<database name>/<prior update level>  
REORGANIZATION/<database name>/<run date>/<run time>  
DESCRIPTION/<database name>/<run date>/<run time>
```

For permanent directory databases, the DMSUPPORT code file should be in the permanent directory.

```
<datapath>/DMSUPPORT/<database name>/<prior update level>
```

A reorganization using the OFFLINE or REORGDB option does not audit the reorganization of structures at the record level. Therefore, you can perform the rebuild recovery function only after you have performed a database dump of the reorganized structures.

For a successful rebuild, you must perform a database dump before the reorganization and perform a dump of the reorganized structures after the reorganization has completed.

Note: *If a reorganization using the OFFLINE option adds a new set, adds a new data set, or increases the number of sections in a data set, then a full database dump must be taken after the reorganization in order to perform a successful rebuild recovery.*

If the dump is not available or is not included in the input specification to DMUTILITY, then the recovery program displays the following message:

```
STR # = <n> IS OFFLINE REORGANIZED; A POST REORGANIZATION DUMP  
IS REQUIRED TO RECOVER THRU OFFLINE REORGANIZED STRUCTURES
```

After this message appears, you can use one of the following three commands:

- Use the AX:CONTINUE command if you want to proceed with the rebuild recovery that recovers user updates except for structures reorganized using the OFFLINE option. Note the following points when using this command:
 - After initiating this command, perform a complete offline database dump before using the database. Otherwise you cannot successfully complete future rebuild recoveries.
 - If you performed an update reorganization, copy the DMSUPPORT and description code files of the database that correspond to the level of the database before the reorganization was performed. This action enables you to run future reorganizations and DASDL updates.
 - If the reorganization adds or deletes any structures, or if the reorganization changes the number of sections in an extended data set, this command cannot be used.
 - The rebuild recovery terminates normally when it encounters
 - A stopping point requested by the user
 - The first user audit record for an offline structure
 - The beginning of another reorganization in the audit
- Use the AX:TERMINATE command to terminate the rebuild recovery normally and restore access to the database. You can run the reorganization again if needed.
- Use the AX:ABORT command to terminate the rebuild recovery abnormally. You must perform the rebuild recovery again before you can access the database.

The workload for rebuilding through a reorganization region of the audit is significant and can normally be avoided by taking DMUTILITY dumps immediately after every reorganization. Because of the high workload associated with this type of rebuild operation, use it only as a last resort. For instance, perform this type of rebuild operation if the last usable dump was taken prior to the last reorganization.

If the reorganization involved in the rebuild recovery is a garbage collection reorganization, no special handling of the control file is needed. Simple updates, such as adding a new set and then generating the new set by using reorganization, are considered to be garbage collections.

However, if a file or a record format conversion is involved in the rebuild recovery, then you must perform the following steps before initiating the rebuild recovery:

1. Copy the control file that is to be used for the rebuild recovery from the DMUTILITY dump(s). If multiple dumps are specified, the control file must be copied from the oldest dump specified. If the correct control file is not specified for the rebuild recovery, it might result in the following message:

```
POSSIBLE REORGANIZATION BETWEEN DUMPS; STRUCTURE INVOLVED: <n>
```

The variable <n> is a structure number.

Note: This message might be displayed for multiple structures, depending on the reorganization performed.

2. Ensure that the correct DMSUPPORT tailored software files are available.

The recovery operation identifies the required DMSUPPORT files, even if the DMSUPPORT file contains the update level.

[Table 7-1](#) summarizes the handling of the control file for a rebuild recovery.

Table 7-1. Control File Handling for Rebuild Recoveries

Type of Reorganization	Copy Control File?
Garbage collection	No
File format conversion	Yes
Record format conversion	Yes

Rollback Recoveries and Reorganizations

You cannot perform a rollback recovery into or through a reorganization region of the audit. This limitation exists because new images written in the audit cannot be transformed into the old-style images. The data files for the two types of images are not compatible. This limitation is one reason that you should perform a full database dump prior to performing a reorganization.

Row Recoveries and Reorganizations

You cannot perform a row recovery through a reorganization region of the audit. This limitation exists because the beforeimages and afterimages of the records being reorganized do not exist in the audit file. You can perform only a rebuild recovery through a reorganization region.

Reorganization I/O Errors

During the reorganization of a data set, an I/O error can occur because insufficient space was allocated to store all the records currently in the data set.

Examples of when this problem can occur include the following:

- When the value of the MAXRECORDS option is specified and the number of records in the data set exceeds this value. A DASDL UPDATE compilation causes the AREASIZE and AREAS values to be recalculated based on the value given for the MAXRECORDS option. As a result, during the reorganization the allocated areas can be used up before all the existing records have been reorganized.
- When a population is assigned to a variable-format data set, the number of areas the Enterprise Database Server assigns to the data set is calculated by multiplying the average record length by the population. If the data set contains many long records, the areas can be filled before the population is exceeded.

If the OFFLINE option is being used and the current population cannot be contained in 1000 areas, the following message is displayed and the reorganization is terminated with a FAULT IN ACR:

```
<data set> EXCEEDED 1000 AREAS. INCREASE AREASIZE
```

As a result of this error, the following steps must be taken:

1. Restore the database to a point before the reorganization.
2. Correct DASDL to handle the population.
3. Run the REORGANIZATION program created using the corrected description file.

If the OFFLINE option is not used and the current population cannot be contained in 1000 areas, a reorganization I/O error message category 10 (limit error), subcategory 1 (an attempt to store too many records in a structure), is displayed for the structure being generated. A message appears stating that the reorganization can be restarted and the file is to be expanded during recovery. Recovery expands the AREASIZE value of the file by 10 percent and then initiates reorganization.

If the 10 percent expansion is not sufficient to handle the current population, the reorganization continues to fail. If this type of error occurs and the restarted reorganization continues to fail, the database must be restored to a point before the reorganization. Then DASDL must be corrected to handle the population, and the corrected REORGANIZATION program run. If this type of error occurs, and the reorganization is restarted and completes satisfactorily, DASDL must still be updated. This DASDL update requires another reorganization to be run.

When doing a DASDL UPDATE compilation, ensure that the values assigned to the AREAS, AREASIZE, and MAXRECORDS options take into consideration the current population of the data set and its expected growth rate.

Reorganization Data Errors

The following types of data errors can occur during a reorganization:

- A data set verify store error
Normally, this error occurs because an item has been made required by the DASDL UPDATE compilation, and the item is found to be null in some of the data set records. The item might be null because the user-specified verify condition has changed or because an item has been designated as a key for a new index set.
- A duplicate error
Duplicates are found in the data set spanned by the index set, and either of the following is also true:
 - A new index set, with duplicates not allowed, has been added.
 - The key of an existing set, with duplicates not allowed, has been changed.

Data set verify store errors can be completely avoided by specifying PREVERIFY in the BUILDREORG specifications. If any verify store errors are encountered during the PREVERIFY run, the reorganization cannot proceed.

Duplicate errors cannot be detected in a PREVERIFY run.

If either type of error is encountered during a reorganization, the reorganization completes, but the affected structure is marked as *Disabled by Reorg* and is unavailable. Any user program trying to access the structure receives a VERSION error 3 (reference to a structure that has not yet been made available by the REORGANIZATION program). To make the structure available again, you must first initiate a reorganization that reverses the error condition. For data set verify store errors, you must change the verify condition or delete the index set whose keys were found to be null. For duplicate errors, correct the records in the data set containing the offending duplicate key items by either delete the record or modify the key items so that no duplicates exist.

You must then perform a garbage collect reorganization to generate the index structure from the corrected data set. Alternatively, you can correct the duplicate errors by deleting the disabled index set with a DASDL update, updating the key items or changing the index set appropriately, and performing another DASDL update and reorganization to add the index set back again.

Reorganization data errors are written to the printer in the DMUTILITY print record format. In the case of a PREVERIFY error report, up to 100 records in error per structure are written before the PREVERIFY task terminates.

SORT Errors During Reorganization

When using SORT during the generation of sets or data sets, you can encounter SORT errors. For SORT ERROR #24 (the number of words specified for memory exceeds the memory available on the system), if the compiler control option RESTARTSORT is set for the BUILDREORG task, the following messages are displayed on generate tasks:

```
*** SORT ERROR 24 SPECIFIED CORESIZE <value> TOO LARGE ***.  
ENTER <MIX #> AX <NEW ALLOWEDCORE> TO RESTART SORT OR  
<MIX #> AX QUIT TO QUIT.
```

If a new ALLOWEDCORE value is entered, the sort operation is restarted using the new ALLOWEDCORE value. If the RESTARTSORT option is reset or if other SORT errors occur, the following applies:

Set generation results in SORT error:

When a set generation fails, the set is marked "Disabled by Reorg," and the reorganization completes. Any user program that attempts to access the disabled structure receives DMVERSIONERROR #3. To re-enable this structure, REORGANIZATION must generate the set successfully. To proceed with the set generation, wait for the reorganization program to complete all other structures that did not encounter the SORT ERROR, then compile and initiate a reorganization for the disabled set.

Data Set generation results in SORT error:

When a data set generation fails, reorganization terminates abnormally. To proceed with the reorganization:

- If the failure was due to a lack of available memory (SORT ERROR #24) , re-run the reorganization when the necessary memory is available.
- If the failure was a result of sort errors other than SORT ERROR #24, correct the errors that caused the failure by recompiling REORGANIZATION (for example, lowering the ALLOWEDCORE or changing the BUILDREORG SORT) and re-running the Reorganization.

Note: Generally, you might want to set the *RESTARTSORT* option to aid in correcting SORT ERROR #24, before compiling the REORGANIZATION program.

OFFLINE FIXUP EXCHANGE Errors During Reorganization

During an OFFLINE reorganization, an OFFLINE FIXUP EXCHANGE error might occur if a file is accessed while it is being fixed up. Using the FILEDATA system software utility or a PD command to access file header information could cause this error. A pack squash or library maintenance copy might cause this error as well.

If an OFFLINE FIXUP EXCHANGE error occurs, an error message displays the name of the structure and the fixed up block that could not be exchanged. The FIXUP task waits on the ACCEPT (AX) command. Wait until the file is no longer being accessed and enter <mix #> AX OK. Once you enter this information, the reorganization continues as normal.

If you want to terminate the reorganization, enter <mix #> AX DS. The reorganization is terminated with FAULT IN ACR @20635600. If the reorganization is terminated at this point, the reorganization can be restarted.

Displaying Reorganization Status

A Visible DBS command is available to display the progress of a reorganization. The command displays a 1-line message for each reorganization task (generate or fixup task) associated with the reorganization. In the case of a generate task, the number of records reorganized is displayed and, where possible, the percent of the total records to be reorganized is displayed. (The total number of records to be reorganized is not known for variable format or compact data sets.) For a fixup task, the number of blocks fixed up and a percent of the total blocks to be fixed up is displayed.

The status appears only for the tasks that are created by BUILDREORG and shown on the BUILDREORG report. In some cases, separate generate and fixup tasks are created for a set. This separate fixup task is created while the reorganization is running, and when the Accessroutines determines a separate fixup task is necessary. An example of this occurrence is when the set is generated before the data set. This additional fixup task does not appear on the BUILDREORG report. The progress of fixup tasks added at run time is reported while the tasks are running when the Visible DBS command SM STATUS REORG is entered. The Visible DBS command SM STATUS REORG does not report the completed generate task. However, once the added fixup task completes, the SM STATUS REORG command shows the completed generate task, and no longer shows anything for the fixup task.

The syntax of the Visible DBS command is as follows:

```
<mix number> SM STATUS REORG
```

The <mix number> is the job number of the database stack.

An example of the displayed and printed output is included here:

PROCESS	STATUS
GEN/D1/3	COMPLETED
FIX/S1/4	50 BLOCKS FIXED UP (25 %)
GEN/D2/5	20000 RECORDS REORGED (10 %)
FIX/S2/6	NOT STARTED YET

Enhancing Reorganization Performance

Several parameters can be specified to enhance reorganization performance. Some of these are DASDL specifications, and some are BUILDREORG specifications. In the case of DASDL specifications, it might be that parameters should be changed for the reorganization, but returned to their original value after the reorganization is done. Although specific values for parameters cannot be given, some general guidelines are given in the following paragraphs.

DASDL ALLOWEDCORE

Because the REORGANIZATION program uses the Accessroutines procedures to perform reorganizations, the reorganization is affected by the database ALLOWEDCORE value. A large ALLOWEDCORE value generally improves performance for a reorganization, as it would for typical day-to-day database activity. This would be especially true for generating index sets in random key order. More coarse tables could be made resident with a large ALLOWEDCORE value.

Allowing multiple reorganization tasks to run simultaneously (that is, setting the TASKLIMIT option to greater than 1) increases the amount of memory needed for buffers. To avoid excessive overlays and improve performance, make sure the database ALLOWEDCORE value is sufficient to handle that increased database usage. In extreme cases, if the ALLOWEDCORE value is insufficient, tasks terminate with a 'SEQUENCE ERROR' or 'REORG IO ERROR ON <structure>, CATEGORY = 9, SUBCATEGORY = 256' error message. If one of these error messages occurs, the database must be reloaded with the dump that was taken before the start of the reorganization, the database ALLOWEDCORE value must be increased, and the reorganization must be restarted.

AUDIT BLOCKSIZE

As a general rule, a large AUDIT BLOCKSIZE value is desirable when a reorganization is running, or during typical daytoday database activity. Since audit I/O is serial, and writes are variable length, performance is never degraded by long I/O transfer time.

AREASIZE for Audits and Database Files

Each time an area boundary is crossed in the audit, the audit is closed and reopened. A larger AREASIZE value for the audit file decreases this overhead.

During a reorganization, the AREASIZE value for data sets is a consideration. For most data set generates, areas of the old file are purged and the new file grows. To purge the old areas, the old file must be closed and reopened. Again, a larger AREASIZE value reduces this overhead.

The disadvantage of large area sizes is that larger requests for pack space are required and space might be wasted for small files.

REBLOCKFACTOR

The REORGANIZATION program makes use of normal Accessroutines buffer management routines. Therefore, for serial accessing of standard and direct data sets, reblocking can be enabled. In the case of a data set generate, the old file is read using the REORGANIZATION program read routines, but the new file is created using Accessroutines procedures. Thus, reblocking can be enabled for the new file. In the case of an index set generate using the data set, the FIND NEXT command is performed on the data set to extract the keys. Again, reblocking can be enabled.

Because reblocking is dynamically activated and deactivated, depending on whether access is serial, there would rarely be a case in which reblocking would be a disadvantage. It is recommended that this feature always be enabled. If the feature is not enabled, then consider turning on or increasing reblocking during a reorganization.

When using an OFFLINE reorganization for a sectioned direct data set, reorganization does not automatically use larger data blocks when reading and writing to disk. It is recommended that you use reblocking for an OFFLINE reorganization of sectioned direct data sets.

TASKLIMIT

In general, specifying the value of TASKLIMIT as a number greater than 1 increases the speed at which a reorganization runs. The optimum value is dependent on system resources available at the time the reorganization is run.

CENTRAL DATASET SEQUENCE

Setting the CENTRAL DATASET SEQUENCE option enables you to

- Control the order of structures to be reorganized.
- Specify groups of structures to be processed simultaneously.

OFFLINE

The OFFLINE option provides the users who are interested in performance, rather than availability, the means to perform a restartable reorganization that produces a minimal amount of audits, while giving generally good performance. If the OFFLINE option reorganizes structures, the structures are not available until the entire reorganization is complete. The OFFLINE option generates sets after the data set is generated. If sets are generated from the data set, the default used is the SORT option.

If the dataset being reorganized is sectioned, the reorganization task can take much longer than if it is not sectioned. The records are evenly distributed throughout the sections. At restart points, the buffers for all the sections are flushed. The time needed to create these restart points becomes significant as the number of sections and the number of areas increase. To lessen the impact of the increased reorganization time, it might be desirable to use the USEREORGDB option, which allows the database to remain available throughout the reorganization.

Set Generation

Generating a set from itself is generally faster than generating the set from the data set if the task has a sufficient ALLOWEDCORE value. A sufficient ALLOWEDCORE value means that at least one word for each record is in the corresponding data set. However, when the set is new, corrupted, or its key changed, the set must be generated from the data set. If a set is generated from the data set and a large number of records is involved, use the SORT option to decrease the generation time.

When multiple sets from the same data set are generated and you are using the SORT option, it is best to use the SORT option for all the sets from that data set. Using the SORT option is best because the ALLOWEDCORE value specified for the data set is used for both FIXUP and SORT options. Generally you need a large ALLOWEDCORE value to ensure a fast fixup. However, a large ALLOWEDCORE value when using the SORT option can cause SORT ERROR 24 if there is insufficient memory available on the system at run time.

Disk Storage Requirements

The REORGANIZATION program always creates generated structures on their final medium, which is the pack specified in the latest DASDL update.

Note: *If you are reorganizing a permanent directory database, the database administrator must make space available for the work files in the permanent directory in which the database resides.*

If you are reorganizing data sets, or performing garbage collection on index sets, the REORGANIZATION program does the following:

- Renames the old file
- Creates an empty new file
- Moves records from the old file to the new file

Reorganizing the Database

For more detailed information on the naming conventions used by the REORGANIZATION program for old files, new files, and fixup files, refer to “Using the INTERNAL FILES Phrase” in this section.

For data sets, rows of the old file are purged as the new file grows. Therefore, the old file shrinks as the new file grows. However, data sets that have an ORDER BY clause, or index sets in the process of garbage collection, cause the old file (or portions of the old file) to be purged only after the new file is built. Because of this, data sets with ORDER BY clauses and index sets in the process of garbage collection require at least enough pack space for two copies of the file. If there is not enough space on the final medium for both copies, the old file can first be copied to a temporary pack. The old file is then purged on the temporary pack after the new file is created.

For OFFLINE option generation tasks that are restarted at the structure level, adequate space is required to accommodate the old file, the new file, and the fixup file.

During a reorganization, be careful when using system software utilities—such as PD and FILEDATA—that access file header information. If these utilities access a data file at the same time as the file is being reorganized, old rows might not be purged as they are reorganized. Instead, the old rows are purged at the end of the reorganization and the total space requirement of the reorganization process is greater than it need be.

Caution

Use of the SQUASH or Reserve Disk (RES) utility during a reorganization is not recommended. This utility could cause the reorganization to experience either or both of the following:

- Require operator intervention.
- Terminate abnormally.

If the reorganization is abnormally terminated, it can be restarted after the files are no longer in use.

Use the structure COPY option to make a copy of the old file on a temporary pack.

For ONLINE reorganizations, enough disk space is needed for the new file, one row of the old file, and the fixup file.

Sometimes, however, you need more disk space. Some of these situations are as follows:

- If you specified data sets with an ORDER BY clause in BUILDREORG.
No rows of the old file are purged until the generate process has completed. In this case, disk requirements are the size of the combination of the following:
 - The entire old file
 - The entire new file

- The fixup file
- If you specified compact, ordered, or unordered data sets.

No rows of compact, ordered, or unordered data sets are purged until the generate process is complete. You need additional disk space for two copies (old and new) of the rows in these data sets.
- If you specified random data sets.

When the new file of a random data set is initialized, the primary scramble area is always allocated. In this case, disk requirements have to accommodate all the files that exist at the start of the generation. Therefore, disk requirements are the size of the combination of the following:

 - The entire old file
 - The primary scramble area of the new file
 - The fixup file
- If inquiry or update user programs access direct data sets during reorganization.

When a user program accesses a direct data set record during reorganization, all records prior to that record are allocated in the new file. Therefore, if a user program accesses the last record in a direct data set, the entire new file is immediately allocated.

The size of fixup files depend on the file type. Compact and variable format data sets require two words per record in the old file. All other data set types require one word per record in the old file.

If you do not have enough disk space on the final medium to handle the previous situations, you should use the structure COPY option.

Limitations of Database Reorganization

Certain types of database reorganizations are not permitted. When the BUILDREORG utility detects these types of reorganization, it terminates with the fatal error REORGANIZATION LIMITATION after printing its report.

The limitations of database reorganization are described in the following list. Where possible, a description is given of how the limitation might be overcome by changing the specification given to SYSTEM/BUILDREORG.

- Generation of an index sequential structure from an index random or ordered list structure where the index sequential had DUPLICATES allowed, FIRST or LAST not specified, is not permitted. Generation of the index sequential structure from itself is a possible alternative that can be taken to avoid this limitation. If the index sequential structure is a disjoint set or automatic subset, generation of the index structure from the data set is another alternative that can be taken.
- When the data set is generated, generation of an unordered list from another index structure is not permitted. In addition, fixup of an unordered list does not occur if the data set is ordered by a set or prime set that is not the generated unordered list, or if

the data set is of type random or unordered. If the unordered list is a disjoint set or automatic subset, you can try either of the following to avoid this limitation:

- Generate the unordered list from the generated data set.
- Use the unordered list to order the data set.
- When the data set is generated, generation of an ordered list from another index structure where the ordered list had DUPLICATES allowed, but FIRST or LAST not specified, is not permitted if the data set is ordered by a set or prime set that is not the generated ordered list, or if the data set is of type RANDOM or UNORDERED. You can try either of the following to avoid this limitation:
 - Generate the ordered list from the generated data set.
 - Use the ordered list to order the data set.
- When the data set is generated, generation of an embedded index sequential structure where the index sequential structure has DUPLICATES allowed, but FIRST or LAST not specified, is not permitted if the index sequential structure is not used to order the data set. Using the index sequential structure to order the generated data set is a possible alternative that can be taken to avoid this limitation.
- When the data set is generated so that the physical order of records change, then generation of an embedded index structure is not permitted in the following two situations:
 - The index is an unordered list or an index sequential or ordered list structure allowing DUPLICATES with FIRST and LAST not specified.
 - The index is not used to order the data set. (The physical order of records change in a generated data set when it is a random or unordered data set or when it is ordered by an index.)
- When using an OFFLINE reorganization for a sectioned direct data set, reorganization does not automatically use larger data blocks when reading and writing to disk. Specifying the REBLOCKFACTOR option can enhance reorganization performance.
- When using a REORGDB reorganization to generate a data set—which is an object data set of a link item—both the data set and the data set containing the link item need to be generated.
- When using a REORGDB reorganization to generate a data set—which is the owner of a link item—both the owning data set and the data set that is the objective of the link item need to be explicitly generated.
- When the compression of LOB items is changed between schemas, the following restrictions apply:
 - Only the OFFLINE option of the Reorganization is allowed.
 - A database dump, containing at a minimum all tank structures, is performed prior to running the Reorganization. If you need to restart the Reorganization, reload the database dump, and then restart the Reorganization.

Section 8

Recovering the Database

Enterprise Database Server software generally recovers itself automatically after a system crash. However, there are times when the database requires manual recovery. The DMUTILITY program, along with the DMRECOVERY, DMDATARECOVERY, and RECONSTRUCT programs, provides the following capabilities:

- Partial database recovery, to
 - Recover one or more damaged rows of database files.
 - Recover one or more damaged structures.
 - Recover write errors using the Quickfix process. This process recovers locked-out rows using only the audit trail; the rows need not be loaded from a backup dump.
- Whole database recovery, to
 - Rebuild the entire database, including recovery from data corruption.
 - Roll back the database to a point in time.
 - Recover unaudited databases using the COPY statement.

Visible Recovery commands enable you to tune and monitor whole database recovery operations, display recovery status information, and print recovery statistics. For more information on the Visible Recovery commands, refer to “Visible Recovery Commands” later in this section.

It is important that you are thoroughly familiar with all of the information contained in this guide, particularly “Database Recovery” in [Section 2, Control File](#).

Note: *The tasks identified in this section can be initiated through Database Operations Center.*

RECOVER Statement (DMUTILITY)

The RECOVER statement is used to initiate all manual forms of recovery for audited databases; that is, all forms of recovery with the exception of halt/load and recovery from abnormal termination, which are initiated automatically. Manual forms of recovery fall into two classes:

- Partial database recovery
- Whole database recovery

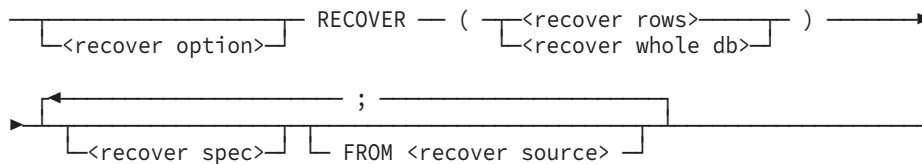
Recovering the Database

DMSII does not support recovering a database using audit files and a dump created under a previous DMSII release version.

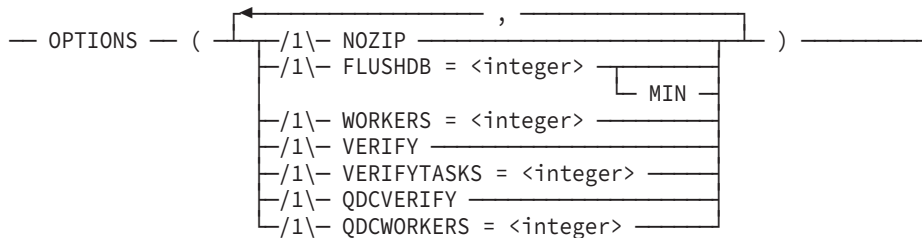
The syntax for this statement is illustrated and explained on the following pages.

The explanation is divided into two parts. The first part describes partial database recovery—that is, RECOVER ROWS and its related constructs. The second part describes whole database recovery together with its constructs.

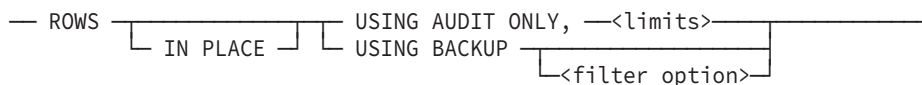
Syntax



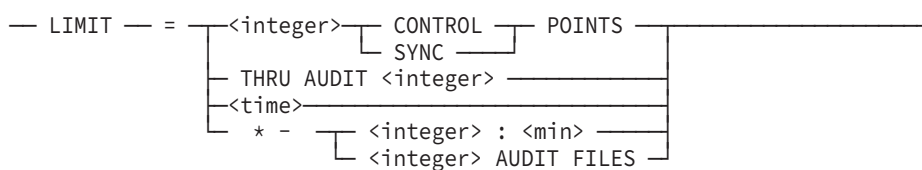
<recover option>



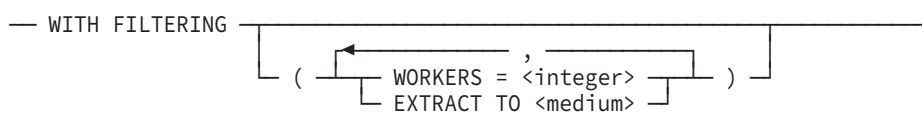
<recover rows>



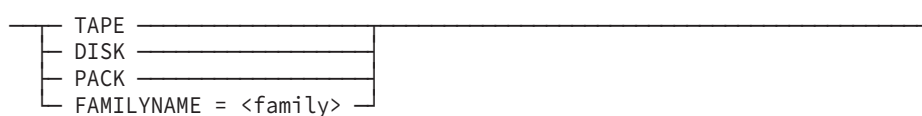
<limits>



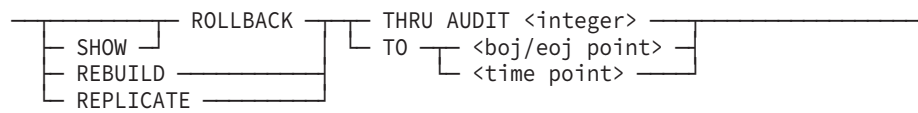
<filter option>



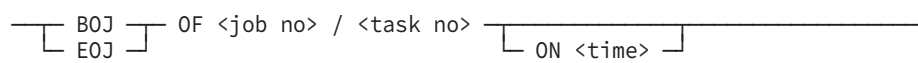
<medium>



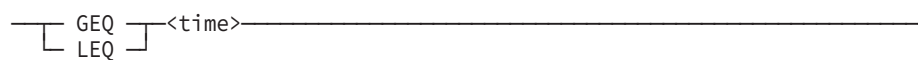
<recover whole db>



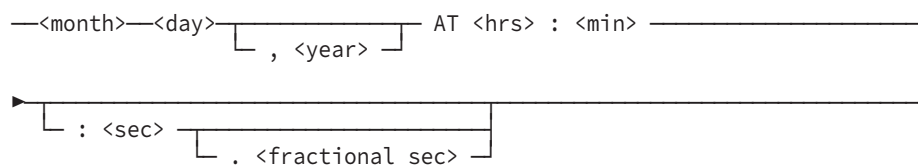
<boj/eoj point>



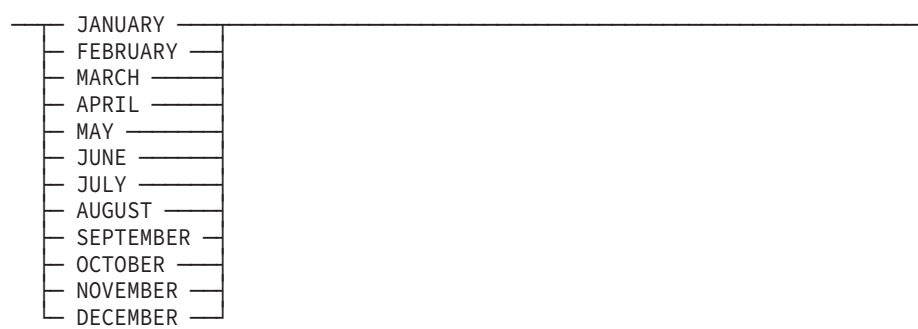
<time point>



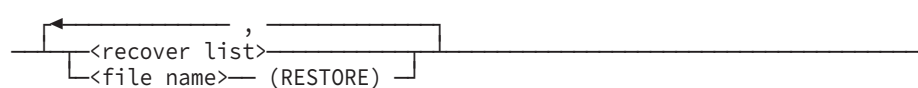
<time>



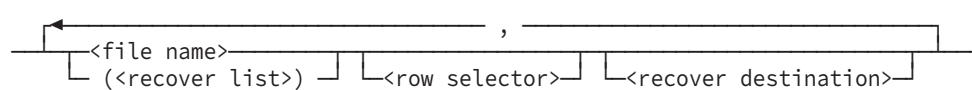
<month>



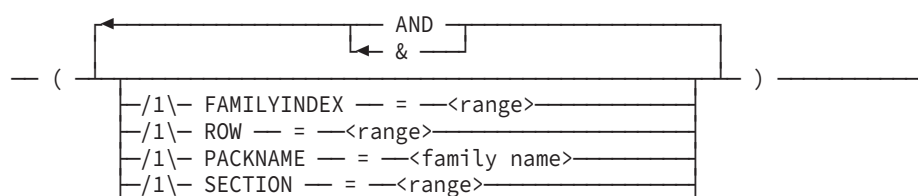
<recover spec>



<recover list>



<row selector>



Recovering the Database

```

/1\ - ROWLOCK = ( /1\ - LOCKEDROW , /1\ - READERROR )

```

<recover destination>

```

<recover as> _____
|
|   <recover to> _____
|   <recover onto> _____
|   <recover to> _____
|
|_____

```

<recover as>

```

AS <file name> _____

```

<recover onto>

```

ONTO <file name> _____

```

<recover to>

```

TO ( ( FAMILYINDEX = <integer> )
    RETAIN ) _____

```

<recover source>

```

MOST CURRENT _____
MOST CURRENT FULL _____
MOST CURRENT ACCUM _____
|
|   <tape specification> _____
|   <disk specification> _____
|   <multidump tape specification> _____
|
|   QUIESCE DB _____
|   QDC ( <QDC title clause> ) _____
|
|_____

```

<tape specification>

```

<tape name> _____
|
|   ( ( /1\ - VERSION = <integer> _____
|       /1\ - CYCLE = <integer> _____
|       /1\ - SERIALNO = <integer> _____
|                       <string6> _____
|       /1\ - DENSITY = <density mnemonic> _____
|   ) ) _____
|
|_____

```

<disk specification>

```

<disk dump file name> ON <family name> _____

```

<multidump tape specification>

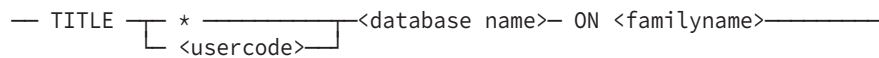
```

<dump name> TAPE = <tape name> _____
|
|   ( ( /1\ - VERSION = 1 _____
|       /1\ - CYCLE = 1 _____
|       /1\ - SERIALNO = <integer> _____
|                       <string6> _____
|       /1\ - DENSITY = <density mnemonic> _____
|   ) ) _____
|
|_____

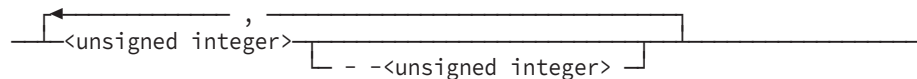
```




<QDC title clause>



<range>



Explanation

The explanation for the syntax diagrams is divided into three parts, as follows:

- The first part contains a table describing the simpler elements of the syntax diagrams.
- The second part describes partial database recovery, RECOVER ROWS, and its related constructs.
- The third part describes whole database recovery and its constructs.

The following information explains the simpler elements of the syntax diagrams.

Option	Explanation
<job no>	Mix number for the job. Unsigned integer between 100 and 65535. If the MCP is enabled to use the 5-digit mix and job numbers, a compatible version of DMUTILITY should be run that can handle the larger numbers.
<task no>	Mix number for the task. Unsigned integer between 100 and 65535. If the MCP is enabled to use the 5-digit mix and job numbers, a compatible version of DMUTILITY should be run that can handle the larger numbers.
<day>	Day of the month. Unsigned integer of not more than two digits.
<year>	Year. Unsigned integer of either two or four digits that identify the year. When specified in four digits, the year must be between 1970 and 2035. When specified in two digits with a value of 00 to 35, the year is treated as 20xx. When specified in two digits with a value of 70 to 99, the year is treated as 19xx. Two-digit years between 35 and 70 are invalid.
<hrs>	Hours. Unsigned integer of not more than two digits.
<min>	Minutes. Unsigned integer of not more than two digits.
<sec>	Seconds. Unsigned integer of not more than two digits. The default is 0.
<fractional sec>	Fractional seconds. Unsigned integer of not more than three digits. The default is 0.

Partial Database Recovery

RECOVER(<recover rows>) is used to recover damaged rows of database files. The database can be in use during row recovery, except when a fatal error has occurred. In this case, the database must be brought down first; otherwise, the recovery fails when the database is opened. DMUTILITY initiates row recovery; RECONSTRUCT completes the process. RECONSTRUCT processes an external coroutine called DMDATARECOVERY, which reads the audit trail and applies the audit images. RECONSTRUCT is automatically initiated by DMUTILITY unless NOZIP is specified in the RECOVER option or if DMUTILITY cannot find all the specified rows from the dump tapes. If NOZIP is specified, the entire row recovery can be controlled by a single Work Flow Language (WFL) job.

The following terms are all used to refer to partial database recovery and can be treated as synonyms: RECOVER ROWS, row recovery, row reconstruction, reconstruct, reconstruction.

For all forms of row recovery, except RESTORE, database files must be resident before DMUTILITY can load database files from backup dumps. The rows to be recovered are selected from the files as they exist at the time DMUTILITY is run for the row recovery. It is recommended that, in the case of multiple pack family structures, duplicate directories exist on one of the continuation packs. You must use the RESTORE option to recover any structures that have been initialized since the dump was performed.

If the row recovery is from a dump ordered by family index (that is, if BY FAMILYINDEX was specified at dump time), some extra tape processing might occur.

For tape dumps, the extra processing appears as numerous tape rewinds or reel switches.

Extra processing occurs only when the dump tape specified in the <recover source> construct does not reflect the current physical placement of the rows on the pack family indexes. The situation can arise if, sometime after the family index dump is taken, a COPY or RECOVER ROWS is performed without retaining the family index. (Refer to the <recover to> explanation in this section for more information).

You can recover all, some, or parts of the database files. To recover all of the files, use an equal sign (=) in the recover list syntax. You cannot perform a row recovery through a reorganization region. You must use REBUILD recovery to recover through a reorganization region.

Because rewinding a tape increases the chance of a bad tape or a tape drive causing parity errors, follow one of these steps to avoid these situations:

- Copy or reconstruct using a FAMILYINDEX = RETAIN statement.
- Dump each family index to a different tape or tapes as shown in the following examples:
 - Single dump tape example:

```
DUMP = (FAMILYINDEX = 1) TO T1;  
      = (FAMILYINDEX = 2) TO T2;  
      = (FAMILYINDEX = 3) TO T3
```
 - Multidump tape example:

```
RUN SYSTEM/DMUTILITY("DB = ANOTHERDB DUMP = (FAMILYINDEX = 1)
  TO ANOTHERDB063094X1 TAPE = T1")
RUN SYSTEM/DMUTILITY("DB = ANOTHERDB APPEND = (FAMILYINDEX = 2)
  TO ANOTHERDB063094X2 TAPE = T1")
RUN SYSTEM/DMUTILITY("DB = ANOTHERDB APPEND = (FAMILYINDEX = 3)
  TO ANOTHERDB063094X3 TAPE = T1")
```

Note: If any of these dumps were the first written to that particular multidump tape, the command would have been *DUMP* rather than *APPEND*.

- If FAMILYINDEX = RETAIN is not used, take a dump immediately after the COPY or reconstruction is performed. Use only the newest dump.
- Rebuild the entire database.

If recovery fails to reconstruct some rows, a report is generated to indicate the rows that failed. To reconstruct these rows, you must initiate a new task to process the partial data recovery.

In most cases, the following statements are valid:

- Partial database recovery can be run while the database is online.
- Halt/load recovery runs before partial database recovery if the recovery is started after a halt/load.

However, if the halt/load recovery might reapply transactions or back out long transactions, the partial database recovery is run before the halt/load recovery. The database is locked until both recovery processes are complete. This situation occurs if either or both of the following DASDL options are set:

- REAPPLYCOMPLETED
- SYNCWAIT

In a Remote Database Backup system, if your RECOVER statement includes the NOZIP option, use the following syntax for the reconstruct operation. The local pack name is the location of the control file for the database that you want to reconstruct.

```
RUN RECONSTRUCT/<database name>; DATABASE DB(TITLE = <database name>
ON <local pack name>)
```

For example, when the NOZIP recovery option is specified, use the following statement to recover the database PAYDB located on the MYPACK pack on the secondary host:

```
RUN RECONSTRUCT/PAYDB;DATABASE DB(TITLE = PAYDB ON MYPACK)
```

In contrast, if the NOZIP recovery option is **not** specified, running recovery through DMUTILITY automatically initiates the RECONSTRUCT program. And the reconstruct operation automatically uses the structures available on the local host.

For easier access of information, the explanation of syntax for database recovery is divided into the following three subjects:

- Designating how to recover
- Designating what to recover and where
- Designating a backup dump

Information on recovering partition files is also included in the table that follows.

Designating How to Recover

The following options are available for recovering a database.

<recover option>

This option permits control of various aspects of the recovery process. By default, DMUTILITY automatically initiates DMRECOVERY or RECONSTRUCT to perform the actual recovery of the files. The only role played by DMUTILITY in the process is to load any necessary rows or files, pass the parameters to the recovery program, and, if desired, initiate that program.

To control the entire recovery process, specify OPTIONS(NOZIP) and add either one of the following statements to the job deck that runs DMUTILITY:

```
RUN SYSTEM/DMRECOVERY(" <db statement>")
RUN SYSTEM/DMRECOVERY(" <db statement>");
DATAPATH = *DIR/TEST ON TESTPACK;

RUN RECONSTRUCT/<database name>
RUN <path name>/RECONSTRUCT/<database name>;
DATAPATH = *DIR/TEST ON TESTPACK;
```

The first set of run statements are for REBUILD or ROLLBACK. The second set are for RECOVER ROWS.

The recovery code file name is either the name specified in the Data and Structure Definition Language (DASDL) or the default SYSTEM/DMRECOVERY. Thus, the recovery programs do not run unseen, and all of the printer output for a single recovery operation is collected automatically with a single job summary.

You can specify the following options for recovery:

- WORKERS

This option controls the number of tapes that are processed in parallel for those forms of recovery requiring tape input. If the WORKERS option is not specified, each tape is processed serially.

Do not use the WORKERS option with either disk dumps or multidump tapes. If you do, a warning occurs.

- FLUSHDB

This option helps determine when DMRECOVERY or DMDATARECOVERY needs to take restart points. The default value for the FLUSHDB option is 20 minutes.

The elapsed time designated by the FLUSHDB option does not in itself cause a restart point. Instead, restart points are taken after specific tasks are completed or are determined by a combination of the designated control records in the audit trail and the time specified by the FLUSHDB option.

In the following scenario, restart points are taken after 2, 7, and 14 minutes:

- The FLUSHDB option is set to 5 minutes.
- Control records that could trigger a restart point appear in the audit trail after 1, 4, 6, 7, 11, and 14 minutes.
- An action that requires a restart point occurs after 2 minutes.

- BYCYCLE

When reloading the database, DMUTILITY processes one READVOLUME task for each physical task for each physical tape required. The READVOLUME task is processed in the following order if DMUTILITY has access to all the cycles and versions from which the dump is created: (CYCLE 1, VERSION 1), (CYCLE 1, VERSION 2), . . . (CYCLE 1, VERSION n), (CYCLE 2, VERSION 1), (CYCLE 2, VERSION 2), . . . (CYCLE 2, VERSION m), . . . (CYCLE x, VERSION y). Each READVOLUME task reads its own tape and usually the beginning of the next tape, since the last row normally splits across two tapes.

If you did not include a cycle and version specification in the syntax, DMUTILITY reads the tape directory from the first tape. This tape has information about the number of cycles only, not the number of versions for each cycle. In this case, DMUTILITY processes one READVOLUME task for each cycle because the directory shows that only one version exists for each cycle.

If you include a cycle and version number specification in the syntax, DMUTILITY reads the tape directory from that tape, which probably has more information about all the cycles and the actual number of versions for each cycle. In this case, one READVOLUME task is processed for each version in the order previously described.

This process is designed to locate the desired rows as soon as possible. However, it can appear that the same tape has unnecessarily been requested multiple times by different READVOLUME tasks if all the rows are being reloaded.

When specified, the BYCYCLE option forces DMUTILITY to process one READVOLUME task for each cycle instead of for each version even if DMUTILITY detects multiple versions for some cycles. This action is most efficient when the entire database is being reloaded, since all versions have to be read. One READVOLUME task per cycle means an independent task for each cycle so that tapes do not need to be requested multiple times by different READVOLUME tasks.

When DMUTILITY is reconstructing rows, the BYCYCLE option can require extra searching time for the desired rows that do not reside on the first version of the required cycle.

Note: *Because CYCLE is always 1 for multidump tapes, tape processing becomes single threaded when DMUTILITY is loading from a multidump tape.*

- VERIFY

This option controls preverification when QUIESCE DB is designated. The verification detects CHECKSUM and ADDRESSCHECK errors of the selected data before the recovery starts.

- VERIFYTASKS

This option can be used only when QUIESCE DB is designated as a recovery source and indicates the number of tasks allowed to perform a VERIFY process in parallel.

The maximum value of VERIFYTASKS is 50. If the VERIFYTASKS option is not specified, the value 1 is assumed. The VERIFY process cannot be restarted.

- QDCVERIFY

This option controls verification during the recovery when QDC (<QDC title clause>) is designated in the <recover source> construct. This verification detects CHECKSUM and ADDRESSCHECK errors of the selected data.

- QDCWORKERS

This option controls the number of workers to be processed in parallel during a recovery that had QDC (<QDC title clause>) designated in the <recover source> construct. From 1 to 50 workers can be specified. If the QDCWORKERS option is not specified, the value 1 is assumed.

IN PLACE

You can specify this option for all forms of row recovery. If specified, the audit images are applied to the database files directly. This requires that the rows be locked out by DMUTILITY if they are not already locked out.

If you do not specify IN PLACE, the audit images are applied to temporary files, and the appropriate rows are exchanged into the database files upon completion of image application.

If you used the RC command to reconfigure a family pack with the KEEP option, an I/O error occurs after any attempt to read or write on the rows on that pack. Therefore, the recovery of a file with the IN PLACE option also fails with an I/O error. As a result, it is recommended that the IN PLACE option not be used in this situation.

USING BACKUP

This option causes the rows to be reconstructed from backup dump tapes and the entire audit since the time of the dump. DMUTILITY copies old versions of the rows to be recovered from backup dump tapes to temporary files if IN PLACE is not specified, and onto the actual database files if IN PLACE is specified. DMDATARECOVERY then applies audit images to the rows. The advantage of this form of row recovery is that it always works, unless irrecoverable I/O errors have occurred. The disadvantage is the time consumed locating the proper dump tapes, loading the rows, and processing all the audit images from the time of the dump to the end of the audit trail.

If USING BACKUP is specified and IN PLACE is not specified, then any recovered rows initially having a ROWLOCK value of 0 (normal state) are changed to a value of 2 (READERROR) by DMUTILITY. This is done so that DMDATARECOVERY can determine which rows have been exchanged and which have not in the event of a halt/load.

<filter option>

This option specifies that the DMRECONFILTER utility program is to extract the applicable audit file information necessary for the particular reconstruction request. After analysis of the RECOVER parameters, DMUTILITY initiates SYSTEM/DMRECONFILTER and passes

to it the needed information by way of a file called <dbname>/RECONSTRUCTFILTERINFO. DMRECONFILTER examines all of the audit files created since the dump used by DMUTILITY and extracts the necessary audit information. DMRECONFILTER runs in parallel with the reloading of the database after each worker has completed its updating of the RECONSTRUCTFILTERINFO file.

Note: *If you use the filter option, the last row of the requested structure is always reconstructed.*

You can specify the following two options for filtering:

- WORKERS

This option determines the number of worker stacks used by DMRECONFILTER for filtering audit files. Each worker is assigned one of the audits to extract and analyze. As each worker finishes the analysis of an audit file, it locates the next audit file for processing.

The WORKERS value should typically be set to the number of tape drives that are available for the audit files. If the WORKERS value is larger than the number of audits to filter, only as many workers as there are audits to filter are initiated. The maximum number of workers that can be specified is 50.

Do not use the WORKERS option with disk dumps. If you do, a warning occurs.

- EXTRACT TO

This option determines where to place the filtered audit information. By default, this information is placed on the same medium as the primary audit files, unless the primary audit files are on tape. In this case, the filtered audit information is placed on the same medium as the control file.

USING AUDIT ONLY

This option causes the rows to be reconstructed using only the audit.

<limits>

This option informs DMDATARECOVERY how far to scan the audit in the reverse direction during the Quickfix process. LIMIT can be specified in terms of CONTROL POINTS or SYNC POINTS, audit files, or a date and time. If the audit trail contains entries that do not appear to be in chronological order, do not use the date, time, or * - <integer>: <min> specification of this option.

If * - <integer>: <min> is specified, DMDATARECOVERY limits its reverse scan to audit records created no more than <integer>: <min> before the time in the last audit record. For example, if * -2:30 is specified, DMDATARECOVERY limits its reverse scan to audit records created no more than two and one-half hours prior to the time in the last audit record.

If * -<integer> AUDIT FILES is specified, DMDATARECOVERY limits its reverse scan to the last integer audit files created. For example, if * -2 AUDIT FILES is specified, DMDATARECOVERY limits its reverse scan to the last two audit files.

If IN PLACE is specified for the Quickfix process, rows damaged by irrecoverable errors in the magnetic recording media cannot be recovered. This makes this form of Quickfix with IN PLACE specified slightly less powerful than a normal Quickfix.

Reconstructing Rows Using the Quickfix Process

If rows are reconstructed using only the audit, the rows need not be loaded from a backup dump. This process is referred to as *Quickfix*. Quickfix recovers only locked rows (rows having write errors) and reduces the time required for row recovery. A fatal error occurs if no rows are locked out, so be sure to check for locked rows before you initiate the Quickfix process. Be aware that Quickfix might not always be able to reconstruct all locked rows.

Quickfix begins by scanning the audit in the reverse direction starting at the current end of the audit. The user specifies a limit to this reverse scan by using the <limits> construct. At the end of the reverse scan, DMDATARECOVERY determines if any rows can be reconstructed. If so, it reverses direction in the audit and performs a normal row recovery. The backward scan of the audit stops short of the specified limit if the recoverability status of all locked rows is determined prior to reaching the limit.

Example of Quickfix

```
BEGIN JOB RECOVERDB;
  TASK UTILTASK;
  RUN SYSTEM/DMUTILITY
    ("DB = MYDB OPTIONS(NOZIP) RECOVER (ROWS USING AUDIT ONLY, "
      "LIMIT = * - 1:30)") [UTILTASK];
  IF UTILTASK(TASKVALUE) = 0 THEN ABORT "UTILITY ERROR"
  ELSE IF UTILTASK(TASKVALUE) = 2 THEN DISPLAY "UTILITY WARNING"

  RUN RECONSTRUCT/MYDB;
END JOB.
```

This job performs a Quickfix row recovery with DMUTILITY and RECONSTRUCT running together in the same job.

Designating What to Recover and Where

The following elements of the RECOVER statement enable you to

- Designate the information you want to recover.
- Identify the location where the information should be placed.

<recover spec>

This option designates the files and rows to be recovered, reconstructed, or restored. For partial database recovery, if the file exists in the disk file directory on pack, the <recover list> syntax should be used. This initiates a reconstruction of the damaged rows of the file.

If the file does not exist, use the RESTORE option to recover the entire file from the backup dump and the audit files. You must use the RESTORE option to recover any structures that have been initialized since the dump was performed.

If a RESTORE is specified, the file is restored through the end of the current audit. DMUTILITY copies the file to be restored from the specified recover source and creates a new file on disk. The title of the new file is the name of the file on the dump tape. The audit images are applied directly to the new file. This requires that all of the rows of the file be locked out by DMUTILITY. If the file was present on disk, it is removed.

<recover list>

This option designates the files and rows to be recovered. The slash equal sign (/=) can be used to recover a family of files. The equal sign (=) alone designates that all files and rows in the database are to be recovered.

<row selector>

This option specifies which rows of the file are to be recovered. If a recover list construct is enclosed in parentheses and a row selector is specified, all database files in that recover list are restricted by that row selector. Database files in the recover list that already have a row selector specification have the outer selection constraints related to the inner selection constraints through the Boolean construct OR.

If FAMILYINDEX is specified in the row selector, only those rows that currently reside on the specified family indexes are reconstructed. If ROW is specified, only those particular rows are reconstructed. PACKNAME allows the user to limit DMUTILITY to a particular pack family without enumerating the files that exist on that pack family. PACKNAME is normally used in conjunction with FAMILYINDEX. If ROWLOCK = READERROR is specified, all rows having read errors (but not those locked out) are reconstructed. If ROWLOCK = LOCKEDROW is specified, all locked rows are reconstructed.



If SECTION is specified in the row selector, only those rows that belong to the specified sections are reconstructed.

<recover destination>

This option designates where the rows are to be placed prior to recovery. If a <recover destination> is not specified, the rows are allocated on arbitrary family indexes but on the same pack family as the file to which they correspond.

<recover as>

This option causes DMUTILITY to copy the rows of a file from the dump tape and create a new file on disk. The title of the newly created file is the file name specified in the <recover as> specification.

You cannot use this option if you use either the IN PLACE or the RESTORE option, or if you are rebuilding a database.

<recover onto>

This option causes DMUTILITY to copy the rows of a file from the dump onto the file specified in the <recover onto> specification. The title of the copied file is the file name specified in the <recover onto> specification. If the file being copied onto is not present, a No File condition results.

If <recover as> or <recover onto> is not specified, special files are created to which the rows to be reconstructed are copied. If the structure is not partitioned, the form of the title is

```
<database name>/RECONSTRUCT/<structure number>
```

If the structure is partitioned, the title is in the form

```
<database name>/RECONSTRUCT/  
  <structure number>/<partition number>
```

You cannot use this option if you use either the IN PLACE or the RESTORE option, or if you are rebuilding a database.

<recover to>

This option identifies the family index where the rows are to be copied for recovery. If RETAIN is specified, the rows are recovered on the same family index they occupied when they were dumped.

If the FAMILYINDEX specified in the <recover to> specification does not exist, DMUTILITY waits on a SECTORS REQUIRED condition for that FAMILYINDEX to be present.

Designating a Backup Dump

The following information explains the element of the RECOVER statement syntax diagrams that identifies the tape dumps or disk dumps you want to use during the recovery process.

<recover source>**Tape Dumps**

The following information explains how to use the RECOVER statement syntax if you are using tape dumps.

When using single dump tapes, the tape directories of successive single dump tape reels are cumulative. The tape directory of the last tape dumped contains information about all rows of the database that were dumped. For example, assume a database was dumped to TAPEX, cycle 1 and cycle 2, and that each has three versions. Also assume that cycle 1, version 3, was the last tape written. Under usual processing, you would specify cycle 1, version 3, in the recovery source.

Notes:

- *Multidump tape functionality is not available for the REPLICATE function.*
- *Because CYCLE is always 1 for multidump tapes, tape processing becomes single threaded when DMUTILITY is loading from a multidump tape.*
- *Whenever you use an existing multidump tape on a system, the fast access directory for that tape must be present on the system. You can copy the directory from another system or create the directory for the tape by using the TAPESET DIRECTORY CREATE command.*
- *Do not append a dump to a multidump tape that was created on another system.*

With the exception of incremental and accumulated dumps, if all the rows to be recovered are on cycle 1, version 3, then specify cycle 1, version 3 as the single dump tape recovery source. When you use incremental and accumulated dumps to recover the database, each tape name must include "CYCLE = last cycle" and "VERSION = last version number" in the tape specification clause.

A dump tape should be named only once in the recover source list, and the specified cycle and version should contain the latest tape directory. DMUTILITY then processes each physical tape in parallel, dependent on the number of workers and the number of tapes specified. The maximum number of workers allowed is 50. If the WORKERS recover option is not specified, the value of the TAPES option at the time the dump is created determines the number of workers.

DMUTILITY expects all files named in a recover list to reside on the corresponding recover source.

DMRecovery expects the current audit to be present for correct operation.

A DMUTILITY-initiated REBUILD, RECONSTRUCT, or COPY using the direct data set rows that were in the preallocated region at the time of the dump causes DMUTILITY to simulate the loading of these rows by preallocating them. The effects of the row preallocation cause these rows to appear as though they had actually been written to the dump tape. Refer to "DMUTILITY INITIALIZE Statement" in [Section 5, Initializing and Maintaining](#).

When more than one dump tape (or dump file in the case of multidump tapes) is used by SYSTEM/DMUTILITY, it is possible to have the same database file on different tapes or contained within dumps stored on multidump tapes. When the same file is encountered more than once, the dump timestamp of the current tape or dump is compared against the dump timestamp of the dump from which the database file on disk was loaded. If these timestamps are different, the file with the latest dump timestamp is selected and the older file is ignored.

Because the selection process might require removing a file that was previously loaded, it is advantageous to first specify an input list that gives the name of the latest dump tape or dump, except when an incremental or accumulated dump is involved. For details, refer to “Database Recovery Using Incremental and Accumulated Dumps” later in this section. If you provide the name of the latest dump tape or dump, DMUTILITY ignores the duplicate file with the earlier timestamp when it encounters the file in additional locations. For more information, refer to “DMUTILITY INITIALIZE Statement” in [Section 5, Initializing and Maintaining](#).

When the database has a dump tape directory, the recovery process can be automated significantly. DMUTILITY can, under certain circumstances, use the dump tape directory to determine which tapes are to be read to load the database files. This determination is possible for

- All forms of reconstruction (RECOVER ROWS)
- REBUILD THRU AUDIT <integer> when the most current dumps are to be used

Note: Do not rebuild the database using an audit file number as a stopping point if the audit file number has rolled back to 1 since you performed the last database dump. For example, if the dump was performed at audit file number 9998, and the rebuild process has to go through audit file number 2, then perform the rebuild recovery using a time as the stopping point.

The automatic selection of dump tapes is invoked when the <recover source> construct is omitted from the RECOVER statement, the dump tape directory is enabled for the database, and the preceding restrictions on the form of recovery are met. RECOVER FROM MOST CURRENT construct must be used when the REBUILD is THRU AUDIT and automatic tape selection is desired.

The DMUTILITY and DMDUMPDIR programs allow recovery using the RECOVER FROM MOST CURRENT FULL construct to automatically access the most current full backup with the database DUMPSTAMP option enabled. This recovery occurs even when corresponding accumulated backups, incremental backups, or both backups exist.

The DMDUMPDIR program must be enabled if you want to use the FROM MOST CURRENT construct. The following fatal error occurs if the DMDUMPDIR program is not enabled:

```
DMDUMPDIR MUST BE ENABLED TO USE - MOST CURRENT
```

Disk Dumps

The following information explains how to use the RECOVER statement syntax if you are using disk dumps.

A disk dump should be named only once in the recover source list.

The DMUTILITY program expects all files named in a recover list to reside on the corresponding recover disk dump source.

DMRECOVERY requires the current audit to be present for correct operation.

A DMUTILITY-initiated REBUILD, RECONSTRUCT, or COPY operation using the direct data set rows that were in the preallocated region at the time of the dump causes the DMUTILITY program to simulate the loading of these rows by preallocating them. The effects of the row preallocation cause these rows to appear as though they had actually been written to the dump.

When the database has a dump directory, the process of performing a RECOVER operation can be automated significantly. The DMUTILITY program can, under certain circumstances, use the dump directory to determine which dumps are to be read to load the database files. This determination is possible for

- All forms of reconstruction (RECOVER ROWS)
- REBUILD THRU AUDIT <integer> when the most current dumps are to be used

Note: Do not rebuild the database using an audit file number as a stopping point if the audit file number has rolled back to 1 since you performed the last database dump. For example, if the dump was performed at audit file number 9998, and the rebuild process has to go through audit file number 2, then perform the rebuild recovery using a time as the stopping point.

The automatic selection of dumps is invoked when the <recover disk dump source> construct is omitted from the RECOVER statement, the dump directory is enabled for the database, and the preceding restrictions on the form of recovery are met. RECOVER FROM MOST CURRENT construct must be used when the REBUILD is THRU AUDIT and automatic selection is desired.

QUIESCE DB

The QUIESCE DB recovery source can be used only for REBUILD recovery.

QDC (<QDC title clause>)



Refer to “Using a Quiesce Database Copy as a Recovery or a Copy Source” in [Section 14, Using a Quiesce Database](#), for information.

Partial Database Recovery of Partition Files

If a database has partitioned structures, it is possible to have partition files in the database that were created after a dump is taken. If a recover source is specified, DMUTILITY identifies the partitions that were created after the specified dump and builds empty reconstruct files for them. However, if no recover source is specified, no reconstruction of the new partitions takes place. DMUTILITY reconstructs any new partition that meets the requirements specified in the recover list and the row selection. If any row meets the row selection criteria, all of the rows are reconstructed. The rows that are not already locked are given a read error.

When more than one dump needs to be specified in the recover source, it is advantageous to specify the latest dump first. This allows DMUTILITY to detect if a partition, which could appear as new on the dump being processed, was already found on another dump. Depending on the order in which the dumps are processed, DMUTILITY loads the partition file from tape or creates an empty file for it. When the latest dump is processed first, the partition file is loaded from tape and only selected rows are reconstructed. If an earlier tape is then processed, DMUTILITY ignores the new partition and does not build an empty file for it.

Note: *You can designate only one disk dump in the recover source statement.*

However, if the earlier dump is processed first, DMUTILITY identifies a new partition, creates an empty reconstruct file, and reconstructs all of the rows. Then, when the latest dump is processed, DMUTILITY loads the partition file from the dump, overlaying the empty file that was created. If the second form of processing occurs, you notice additional processing time, but neither process loses information.

Whole Database Recovery

RECOVER (<recover whole db>) is used to recover the entire database. The database must not be in use during this form of database recovery. Recovery is initiated by DMUTILITY and completed by DMRECOVERY. SYSTEM/DMRECOVERY is automatically initiated by DMUTILITY, unless NOZIP is specified in the RECOVER option.

Recovery Methods

Two methods are available for recovering the entire database: REBUILD and ROLLBACK. The current audit must be present.

REBUILD

This method moves the entire database forward in time. DMUTILITY begins the REBUILD process by loading the entire database from one or more sets of dump tapes, by reading a complete database copy that is in a state of QUIESCE, or by using a quiesce database copy. DMRECOVERY then applies the audit trail afterimages to bring the database forward.

Note: You can perform a rebuild recovery through a reorganization region of the audit. For restrictions and capabilities, refer to “Rebuild Recoveries and Reorganizations” in [Section 7, Reorganizing the Database](#).

If the REBUILD process is unsuccessful, the only alternative is to reload a backup copy of the database and reprocess all updates to the database.

The last audit file that is available to the recovery process must either contain the specified EOJ/BOJ record or contain at least one record having a timestamp that is subsequent to the completion of the online dump. Records having a timestamp include control records and the following types of audit records: RDSO, RDSC, BLKIMG, RDERR, and CDI. In the case of REBUILD THRU AUDIT <integer>, if the specified audit file does not contain at least one record with a timestamp subsequent to the completion of the online dump, the REBUILD process cannot be used.

If a file or a record format conversion is involved in the rebuild recovery, you must perform the following steps before initiating the rebuild recovery:

1. Copy the control file that is to be used for the rebuild recovery from the DMUTILITY dump(s). If multiple dumps are specified, the control file must be copied from the oldest dump specified. If the correct control file is not specified for the rebuild recovery, it might result in the following message:

```
POSSIBLE REORGANIZATION BETWEEN DUMPS; STRUCTURE INVOLVED: <n>
```

The variable <n> is a structure number.

Note: This message might be displayed for multiple structures, depending on the reorganization performed.

2. Ensure that the correct DMSUPPORT tailored software files are available.

The recovery operation identifies the required DMSUPPORT files, even if the DMSUPPORT files contain the update level. See [Table 7-1](#) for further information.

If the REBUILD process uses a quiesce database copy as the recovery source and the control file of the live database is missing, restore the missing control file by using the CFRESTORE command and initiate the REBUILD process again. Refer to “CFRESTORE Command (DMUTILITY)” in [Section 14, Using a Quiesce Database](#), for more information.

Note: Although the automatic population increase information of a control file from a dump or an initialized control file might not be accurate immediately after a rebuild recovery, the information is corrected at the next population increase.

ROLLBACK

This method moves the entire database backward in time. ROLLBACK begins with the current database files. DMRECOVERY then applies audit trail beforeimages to move the database back to a specified point in time. ROLLBACK produces a report of all jobs that are completely or partially backed out. Because of the possible interaction and interdependency of programs, ROLLBACK must use the entire database.

Unlike REBUILD, ROLLBACK cannot be used to recover from data corruption. ROLLBACK can potentially save time over REBUILD, however, to correct logical errors such as an update program that was run with incorrect input.

You cannot perform a rollback recovery into or through a reorganization region of the audit. This limitation exists because new images written in the audit cannot be transformed into the old-style images. The data files for the two types of images are not compatible.

To simulate a rollback and generate a report similar to the one that would be generated for the actual rollback, use the SHOW ROLLBACK command.

Starting Points

If a dump is taken at the time a database is quiesced, the starting point of a REBUILD recovery using that dump will be the time when the quiesce was performed. If the database was not quiesced at the time the dump was taken, the starting point will not be the time of the creation of the dump.

Stopping Points

You can specify a stopping point for REBUILD and ROLLBACK in terms of

- An audit file
- A starting (BOJ) or an ending (EOJ) point of processing for a specific job
- A specific date and time

The recovery process always ensures that the database is left in a state of integrity. To leave the database in a state of integrity, the recovery process might need to rebuild the database to a more recent point or rollback the database to an earlier point than the one you designate.

Do not rebuild the database using an audit file number as a stopping point if the audit file number has rolled back to 1 since you performed the last database dump. For example, if the dump was performed at audit file number 9998, and the rebuild process has to go through audit file number 2, then perform the rebuild recovery using a time as the stopping point.

When setting time as a stopping point, if the system clock of the machine has been moved backward (for example, to switch from daylight saving time to standard time), do not recover to a point of time during the time change. For example, if the system clock has been set backward from 2 a.m. to 1 a.m., do not use a stopping point between 1 a.m. and 2 a.m. Instead, choose a time after 2 a.m. or before 1 a.m.

Both rollback and rebuild recoveries perform halt/load recovery wrap-up procedures before the recoveries finish. The halt/load recovery phase begins at the most recent point in time when no programs were in transaction state.

If the DASDL SYNCWAIT option is set to a nonzero value, transactions might prevent natural quiet points from occurring. These long transactions are temporarily taken out of transaction state to allow pseudo syncpoints to occur. These pseudo syncpoints are valid stopping points for the first phase of rollback and rebuild recoveries. During this halt/load recovery phase, long transactions that were in progress at the time of the last syncpoint are individually backed out. The actual stopping point is determined by the REAPPLYCOMPLETED option and either the ROLLBACK or the REBUILD option. Audit records after the stopping point become invalid and should be discarded.

For disk file auditing, the current audit file is extended from the point where the ROLLBACK/REBUILD finishes. Thus, the latter portion of the audit file is overwritten. Any audit files that were created after the current audit file are invalid and should be discarded.

For tape auditing, the current audit file is closed at the ROLLBACK/REBUILD termination point, and a new audit file is opened when processing is resumed. Discard any audit files that were created after the current audit file.

If an audit medium switch took place during normal operations and the ROLLBACK/REBUILD terminated at a point on a tape audit file, the tape audit file is closed, and a new audit file is created on the primary audit medium when processing resumes. The tape file becomes the last valid audit file. Discard all audit files that are created after the current audit file.

For example, consider a database for which the last audit is 6000. If ROLLBACK terminates in audit file 5888, then audit files 5889 through 6000 are invalid. If audit file 5888 is a disk audit file, then auditing begins at the end of this file when normal auditing resumes. If audit file 5888 is a tape audit file, then it is closed and auditing begins in audit file 5889 on the primary audit medium when normal updating resumes. As the user, it is your responsibility to discard any invalid audit files.

Note: *You must wait for the DMSUPPORT library and database stack to terminate. The two stacks must terminate completely before any user application can access the database.*

If desired, it is possible to recover through a change in the number of audit sections. If you perform rollback recovery through an audit section change, that change is still in effect.

If a rebuild recovery must recover through a reorganization region, you must copy the control file from the dump tape before you run DMUTILITY. Do not use the FROM MOST CURRENT option. You must also specify a recover source in the RECOVER statement. For more information on rebuilding through a reorganization region, refer to “Rebuild Recoveries and Reorganizations” in [Section 7, Reorganizing the Database](#).

For partitioned databases, not all of the partition files have to be loaded from the dump tape for the rebuild recovery. The partition files that have not been updated since the dump was performed do not need to be loaded.

Note: If the DMDUMPDIR program is enabled, dump directory entries can be deleted automatically under some conditions. If a dump was made after the time at which the recover process stops, the dump is considered invalid and is deleted automatically from the directory. This deletion can happen after a rebuild or rollback recovery is performed.

If you want to add this dump back to the directory, use the DMUTILITY BUILDDUMPDIRECTORY function. For more information on using BUILDDUMPDIRECTORY, see “DMUTILITY BUILDDUMPDIRECTORY Command” in [Section 6, Backing Up a Database](#), of this guide.

Both rollback and rebuild recoveries are supported in the Open Distributed Transaction Processing environment. Because all of the databases in the Open Distributed Transaction Processing environment must be synchronized, ensure that you

- Rebuild to the end of the current audit file. However, if some audit information is lost, the databases might still be out of synchronization.
- Roll back all databases to the same point in time if the databases are all on the same machine or if all of the machines are synchronized.

Because the Enterprise Database Server recovery processes affect only one database, integrity is ensured only for the database on which the recovery is performed. In the Open Distributed Transaction Processing environment, a transaction can affect more than one database. When you perform a recovery on a database in an Open Distributed Transaction Processing environment, you must ensure that you resynchronize the recovered database with all other databases in that environment.

The following information explains the REBUILD, ROLLBACK, and REPLICATE elements of the RECOVER statement syntax diagrams.

REBUILD THRU AUDIT <integer>

This option applies the afterimages from all audit files created since the time of the dump or the time of the QUIESCE. This process continues until all afterimages in the audit file specified by <integer> have been applied. REBUILD then performs a halt/load type of recovery. Following this halt/load recovery phase, if the REAPPLYCOMPLETED option is set, all completed transactions through the end of the specified audit file are reapplied.

Note: Do not rebuild the database using an audit file number as the stopping point if the audit file number has rolled back to 1 since you performed the last database dump. For example, if the dump was performed at audit file number 9998, and the rebuild recovery has to go through audit file number 2, then perform the rebuild recovery using a time as the stopping point.

ROLLBACK THRU AUDIT <integer>

This option begins with the current database files and applies beforeimages from the audit. This process continues until all beforeimages in the audit file specified by <integer> have been applied. ROLLBACK then backs up into the end of audit <integer> – 1 to begin the halt/load recovery phase. Upon completion of the halt/load recovery, if the REAPPLYCOMPLETED option is set, all completed transactions through the end of audit <integer> – 1 are reapplied.

REBUILD/ROLLBACK TO <boj/eoj point>

REBUILD TO <boj/eoj point> applies all afterimages created since the dump or QUIESCE and stops at the BOJ or EOJ point of the specified job. ROLLBACK TO <boj/eoj point> applies all beforeimages in the audit trail until it encounters the BOJ or EOJ point of the task. Following the halt/load recovery phase, if the REAPPLYCOMPLETED option is set, all completed transactions up to the specified BOJ or EOJ point are reapplied. The existence of BOJ and EOJ points implies that the task opened and closed the database. If the task opens and closes a database several times, REBUILD TO <ej point> stops at the first database CLOSE statement, and ROLLBACK TO <boj point> stops at the last database OPEN statement of the task.

The REBUILD/ROLLBACK TO <boj/eoj point> option does not enable the MOST CURRENT option in the <recover source> construct.

Caution

When using a BOJ or EOJ point, ensure that you identify the job and task that actually opened or closed the database. If you are unsure that the job and task actually opened or closed the database, use the REBUILD/ROLLBACK TO <boj/eoj point> ON <time> option.

SHOW ROLLBACK

Using this option you can verify the results of a rollback before performing the rollback. You can designate the stopping point for the verification by using any of the following:

- Audit file number
- Starting point of a job (BOJ)
- Ending point of a job (EOJ)
- Date and time

This option simulates a database rollback and generates a report as a printer backup file. The information in the report is similar to the report that would be generated for the actual rollback.

The SHOW ROLLBACK report contains the following:

- A list of the jobs that would be completely or partially backed out.
- A list of the database open and close occurrences that would be backed out.
- A description of the rollback stopping point in the audit trail, including the time stamp in the audit file at which the rollback would finish.

REBUILD TO <time point>

This option applies all afterimages created since the dump or QUIESCE, and stops at the first control record in the audit with a timestamp greater than or equal to the specified time. The LEQ (less than or equal to) operator is not valid for the REBUILD process.

When setting time as a stopping point, if the system clock of the machine has been moved backward (for example, to switch from daylight saving time to standard time), do not recover to a point of time during the time change. For example, if the system clock has been set backward from 2 a.m. to 1 a.m., do not use a stopping point between 1 a.m. and 2 a.m. Instead, choose a time after 2 a.m. or before 1 a.m.

Note: *If you want to specify a future month as a recovery time, also specify the desired year. If you do not specify a year, DMUTILITY subtracts 1 from the current year.*

The REBUILD TO <time point> option does not enable the MOST CURRENT option in the <recover source> construct.

ROLLBACK TO <time point>

This option applies all beforeimages from the audit until it encounters a control record with a timestamp less than or equal to the specified time. Since ENDTRANSACTION audit images do not have timestamps, the point in time at which a given transaction completed is not recorded in the audit. Therefore, no audit image reapplication occurs following the halt/load recovery phase, regardless of the setting of the REAPPLYCOMPLETED option. The GEQ (greater than or equal to) operator is not valid for ROLLBACK.

When setting time as a stopping point, if the system clock of the machine has been moved backward (for example, to switch from daylight saving time to standard time), do not recover to a point of time during the time change. For example, if the system clock has been set backward from 2 a.m. to 1 a.m., do not use a stopping point between 1 a.m. and 2 a.m. Instead, choose a time after 2 a.m. or before 1 a.m.

REBUILD TO <boj/eoj point> ON <time>

This option applies all afterimages created since the dump or QUIESCE and stops at either <boj/eoj point> or <time point>, depending on which condition is satisfied first.

The REBUILD TO <boj/eoj point> ON <time> option does not enable the MOST CURRENT option in the <recover source> construct.

ROLLBACK TO <boj/eoj point> ON <time>

This option applies all beforeimages in the audit and stops at either <boj/eoj point> or <time point>, depending on which condition is satisfied first.

REPLICATE



This option uses a current copy of an online dump to move a database to a new location, and then performs a rebuild recovery to make the database operational. This option can be particularly useful for creating test or temporary reporting environments for production databases that must remain active on a daily basis.

When you use the REPLICATE option, note the following:

- In general, the REPLICATE option rebuilds the database using new description and control files. The database built by using the REPLICATE option must have the same database name as the original database, but can have a different usercode and be on different packs than the original database. The process expects to find the necessary audit files at the new location.

Note: *This process excludes the audit file that is currently in use by the parent database. The audit file is excluded because an active audit file of any database is always opened exclusively so that the Accessroutines can periodically update its attributes in the disk file header.*

- Use extra caution when the replication is either to or from a nonusercoded (*) database. Nonusercoded databases are visible to all users on the system, which can make it easy to mix structures and audit files. In addition, it is possible to create a confusing situation when an application program has visibility to both usercode and nonusercoded databases.
- Avoid replicating to different disk packs on the same system while attempting to retain the same usercode.

The anomalies just described can also be created by indiscriminate database compilations. However, errors are more likely to occur when you move databases rather than create them.

While a true rebuild operation is designed to function across a reorganization boundary, there are limitations for replicating through a reorganization. These limitations exist because the information pertaining to work files used by the reorganization, including pack locations, is captured in the audit file.

Note: *This limitation occurs because pack locations specified in DASDL can be changed when replicating. However, reorganization pack locations cannot be changed because they are described through BUILDREORG, which is not part of the replicate process.*

The REPLICATE option does not enable the MOST CURRENT option in the <recover source> construct. This removes a possible dependence to have a dump tape directory present and available at the new location. The REPLICATE THRU AUDIT <integer>, REPLICATE TO <boj/eoj point>, and REPLICATE TO <time point> options are valid.

Performing Replication with a Traditional Database

1. Create a utility dump of the source database, either online or offline.

Match the Enterprise Database Server software level at the destination with the Enterprise Database Server software level used at the source location.

Note: *The source database must not be reorganized between the time the dump is captured and the time indicated by the stopping point in the audit files to be used for the replicate process.*

2. Transfer the dump, DASDL source, description file, and audit files from their original location to the new location and usercode.

Note: *The DASDL source and description file must match the dump.*

3. Modify all of the appropriate places in the DASDL source to reflect the new pack location, and change the usercode to match the new location. Make Enterprise Database Server software title changes at this time.

4. Ensure that the following DASDL options are also included:

- UPDATE
- \$SET ZIP
- \$RESET DMCONTROL

5. Generate an updated description file and compile a new DMSUPPORT library by compiling the modified DASDL source. For example,

```
COMPILE MYNEW/DASDL AS $MYDBNAME
```

In the preceding example, the database name must remain the same as that of the original database name, MYDBNAME.

6. Create a compatible control file by using the DMCONTROL RECOVER INITIALIZE option. For example,

```
RUN $SYSTEM/DMCONTROL("DB=MYDBNAME RECOVER INITIALIZE")
```

The DMCONTROL option requires the most current audit file number. The existing rules for matching utility dumps and audit files apply to the REPLICATE feature.

Note: *For online dumps, the audit must be closed after the dump is completed. The audit file must also be available for use by the replicate process whether the dump is online or offline.*

7. Use DMUTILITY to replicate the database using the REPLICATE variant of <recover whole db>. If you use the THRU AUDIT syntax, ensure that the audit file number is the same as that specified in step 6. For example,

```
RUN $SYSTEM/DMUTILITY("DB=MYDBNAME OPTIONS (NOZIP)  
RECOVER(REPLICATE THRU AUDIT x) = AS (NEWUC)=  
ON NEWPACK FROM MYDBNAMEDUMP ")
```

8. Run the RECOVERY program. For example,

```
RUN$SYSTEM/DMRECOVERY("DB=MYDBNAME")
```

The replicated database is now ready for use.

Either compilation or database equation can be used to prepare applications for use with the replicated copy of the database. The following example uses the WFL MODIFY command to perform a permanent database equation:

```
WFL MODIFY <codefile title> DATABASE
MYDBNAME (TITLE=(NEWUC)MYDBNAME ON NEWPACK)
```

Performing Replication with a Permanent Directory Database

Caution

To avoid overwriting your description file, be sure to run your DASDL update from a different usercode. In addition, the pack name of the control file must be different from the pack name of the existing control file.

1. Create a utility dump of the source database, either online or offline.
Match the Enterprise Database Server software level at the destination with the Enterprise Database Server software level used at the source location.
Note: *The source database must not be reorganized between the time the dump is captured and the time indicated by the stopping point in the audit files to be used for the replicate process.*
2. Transfer the dump, DASDL source, and description file from their original location to the new pack location and usercode.
The DASDL source and description file must match the dump.
3. Transfer the audit files to the permanent directory to which the database is to be replicated.
4. Modify all of the appropriate places in the DASDL source to reflect the new pack location, and change the DBPATH specification of the control file to match the new location. Make Enterprise Database Server software title changes at this time.
5. Ensure that the following DASDL options are also included:
 - UPDATE
 - \$SET ZIP
 - \$RESET DMCONTROL
6. Generate an updated description file and compile a new DMSUPPORT library by compiling the modified DASDL source. For example,

```
COMPILE MYNEW/DASDL AS $MYDBNAME
```

In the preceding example, the database name must remain the same as that of the original database name, MYDBNAME.
7. Create a compatible control file by using the DMCONTROL RECOVER INITIALIZE option. For example,

```
RUN $SYSTEM/DMCONTROL("DB=MYDBNAME RECOVER INITIALIZE");
```

Recovering the Database

```
FILE CF =  
*DIR/<new nodes>/MYDBNAME/CONTROL ON <pack name>;
```

The DMCONTROL option requires the most current audit file number. The existing rules for matching utility dumps and audit files apply to the replicate process.

Note: For online dumps, the audit must be closed after the dump is completed. The audit file must also be available for use by the replicate process whether the dump is online or offline.

8. Use DMUTILITY to replicate the database using the REPLICATE variant of <recover whole db>. If you use the THRU AUDIT syntax, ensure that the audit file number is the same as that specified in step 7. For example,

```
RUN $SYSTEM/DMUTILITY("DB=MYDBNAME OPTIONS (NOZIP)  
RECOVER(REPLICATE THRU AUDIT x) = AS = ON NEWPACK  
FROM MYDBNAMEDUMP ");  
DATAPATH = *DIR/<new nodes> ON <pack name>;
```

9. Run the RECOVERY program. For example,

```
RUN$SYSTEM/DMRECOVERY("DB=MYDBNAME"); DATAPATH =  
*DIR/<new nodes> ON <pack name>;
```

The replicated database is now ready for use.

Application programs must be run using a DATAPATH specification. The following example uses a permanent database equation:

```
RUN <codefile title>; DATAPATH = *DIR/<new nodes> ON <pack name>;
```

Restrictions

Some forms of the RECOVER statement require both a recover specification construct and a <recover source> construct; others do not. [Table 8-1](#) summarizes these requirements when you are using tape dumps.

Table 8-1. Recover Specification and Source for Tape Dumps

RECOVER Statement	Recover Specification
RECOVER (ROWS USING BACKUP)	Required. If ONTO <file name> or AS <file name> is specified, the file name must not be the same as any database file name in this or any other database.
RECOVER (ROWS IN PLACE USING BACKUP)	Required. The <recover destination> statement is not permitted.
RECOVER (ROWS...USING AUDIT ONLY...)	Not required.
RECOVER (ROLLBACK...)	Not permitted.

Table 8–1. Recover Specification and Source for Tape Dumps (cont.)

RECOVER Statement	Recover Specification
RECOVER (REBUILD...)	<p>Not required.</p> <p>The following constructs are not permitted:</p> <ul style="list-style-type: none"> • RESTORE • ONTO <file name> • AS <file name> <p>The recover list for a rebuild recovery must be specified so that all database files are loaded from one or more sets of dump tapes. If all database files are not specified, the recovery is not successful.</p>
RECOVER (ROWS USING BACKUP)	<p>Required.</p> <p>The file name must not be the same as any database file name in this or any other database.</p>
RECOVER (REPLICATE...)	<p>Required.</p> <p>Must include the AS <file name> construct.</p> <p>The recover list for the REPLICATE option must be specified so that all database files are loaded from one or more sets of dumps. If all database files are not specified, the recovery is not successful.</p>

[Table 8–2](#) summarizes the requirements when you are using disk dumps.

Table 8–2. Recover Specification and Source for Disk Dumps

RECOVER Statement	Recover Specification
RECOVER (ROWS IN PLACE USING BACKUP)	<p>Required.</p> <p>The <recover destination> statement is not permitted.</p>
RECOVER (ROWS...USING AUDIT ONLY...)	<p>Not required.</p>
RECOVER (ROLLBACK...)	<p>Not permitted.</p>
RECOVER (REBUILD...)	<p>Not required.</p> <p>The following constructs are not permitted:</p> <ul style="list-style-type: none"> • RESTORE • ONTO <file name> • AS <file name> <p>The recover list for a rebuild recovery must be specified so that all database files are loaded from one or more sets of dumps.</p>

Table 8–2. Recover Specification and Source for Disk Dumps (cont.)

RECOVER Statement	Recover Specification
RECOVER (REPLICATE...)	Required. Must include the AS <file name> construct. The recover list for the REPLICATE option must be specified so that all database files are loaded from one or more sets of dumps. If all database files are not specified, the recovery is not successful.

Examples of the RECOVER Statement

Example 1

```
RECOVER(ROWS USING BACKUP)
      = (ROWLOCK = LOCKEDROW, READERROR)
      FROM T1, T2, T3, T4, T5
```

This command recovers all rows having write operation errors (ROWLOCK = LOCKEDROW) or read operation errors (ROWLOCK = READERROR), using the data in the specified dump tapes plus the changes recorded in the audit since the time of the dumps. Because the WORKERS clause is not specified, DMUTILITY processes each dump serially.

Example 2

```
OPTIONS(WORKERS=3) RECOVER(ROWS USING BACKUP)
      = (ROWLOCK = LOCKEDROW, READERROR)
      FROM T1, T2, T3, T4, T5
```

This example is identical to Example 1 except for the WORKERS option. The WORKERS option controls the number of dump tapes that can be processed in parallel. DMUTILITY initiates a worker for each cycle in the first dump tape. If there are more cycles than workers, DMUTILITY processes the next available cycle when one of the current workers finishes. If there are more workers than cycles, DMUTILITY initiates a worker to begin processing the next dump tape. When a worker finishes, it begins processing the next available cycle.

Example 3

```
RECOVER(ROWS USING BACKUP)
      = (ROWLOCK = LOCKEDROW, READERROR
        PACKNAME = DBDATA FAMILYINDEX = 1-3,5)
      FROM TAPEX
```

This command recovers all rows having write operation errors or read operation errors that reside on family index 1, 2, 3, or 5 of pack family DBDATA using the data in the specified dump tape plus the changes recorded in the audit since the time of the dump. No rows on family index 4 are selected for row recovery.

Example 4

```
RECOVER(ROWS USING BACKUP) DB/D/DATA FROM TAPEX
```

This command recovers only rows in the file DB/D/DATA that have copies on the tape TAPEX. Resident rows allocated since the time of the dump are also recovered.

Example 5

```
RECOVER(ROWS USING BACKUP)  
      = (FAMILYINDEX = 3) FROM TAPEX
```

This command recovers all rows of the database that were present on family index 3 at the time of the dump. In addition, any rows resident on family index 3 that were allocated since the time of the dump are also recovered.

Example 6

```
RECOVER(ROWS USING BACKUP)  
      (DB/D/= AS DB/E/=,  
       DB/F/= ONTO DB/G/=)  
      (ROWLOCK = LOCKEDROW) FROM TAPEX
```

This command recovers all rows of files DB/D/= and DB/F/= that are currently locked out. All rows copied to the tape TAPEX file and all resident rows allocated since the time of the dump are recovered. The rows of file DB/D/= are copied as file DB/E/= before reconstruction. Following reconstruction, the old locked-out rows resides in file DB/E/=, which is not removed. The rows of file DB/F/= are copied as file DB/G/= before reconstruction. Following reconstruction, all locked-out rows of file DB/F/= resides in file DB/G/=, which is not removed.

Example 7

```
RECOVER(ROWS USING AUDIT ONLY, LIMIT = * - 1:30)
```

This command recovers all locked-out rows, using the audit only. This form of row recovery scans the audit in the reverse direction, starting from the current end of the audit. The example limits the reverse scan to audit records created not more than 1.5 hours before the time in the last audit record.

Example 8

```
RECOVER(REBUILD THRU AUDIT 4567) = FROM TAPEX
```

This command loads the entire database from the specified dump tape and moves the entire database forward by processing all the audit since the time of the dump. This process terminates after audit file 4567 is processed.

Example 9

```
RECOVER (REBUILD THRU AUDIT 999)  
      = FROM MOST CURRENT
```

This command is similar to Example 9, except that it causes the database to be rebuilt using the most current dump tapes that were created. The DMUTILITY program automatically selects the dump tapes. In addition to the most current full dump, the dump tapes include the most current accumulated dump if one is present that is more current than the full dump plus any set of incremental dumps that are more current than the full or accumulated dumps.

If the DMDUMPDIR program is disabled, the use of the FROM MOST CURRENT construct results in the following fatal error:

```
DMDUMPDIR MUST BE ENABLED TO USE - MOST CURRENT
```

Example 10

```
RECOVER(ROLLBACK TO BOJ OF 1234/1235)
```

This command moves the entire database backward to the beginning of task 1234/1235.

Example 11

```
RECOVER (ROWS USING  
        BACKUP WITH FILTERING(WORKERS=3)) =  
        (ROWLOCK=READERROR) FROM TAPEX
```

This command recovers all rows marked as having read operation errors and invokes the DMRECONFILTER utility to filter the audit information necessary for the reconstruction. Three filter workers are used in parallel to filter the audits generated since the dump tape TAPEX was produced.

Example 12

```
RECOVER (ROWS USING BACKUP) DB/D/DATA (RESTORE) FROM TAPEX
```

This command restores all rows of the file DB/D/DATA, using the data on the dump tape TAPEX, and any changes recorded in the audit through the most current audit file.

Example 13

```
RECOVER (ROWS USING BACKUP) DB/A/DATA (ROWLOCK=LOCKEDROW),  
        DB/D/DATA (RESTORE) FROM TAPEX
```

This command recovers all rows of the file DB/A/DATA that have write operation errors, using the data on the dump tape TAPEX and any changes recorded in the audit. In addition, this command restores all rows of the file DB/D/DATA, using the data on the dump tape TAPEX and the audit. The file DB/A/DATA must be resident.

Example 14

```
("DB=TESTDB00 RECOVER (ROWS USING BACKUP) = (ROWLOCK =  
LOCKEDROW, READERROR)  
  
FROM DUMPDISK4 ON UITEST")  
  
#RUNNING 3210  
#3210 DISPLAY:>>>INPUT WAS:.
```

```
#3210 DISPLAY:DB=TESTDB00 RECOVER (ROWS USING BACKUP) =
(ROWLOCK = LOCKEDROW,  READERERROR)      FR.

#3210 DISPLAY:OM DUMPDISK4 ON UITEST.

#BOT 3213  (ALMA)TAPEWORKER01/DUMPDISK4 ON UITEST

#3213 PK17096 (ALMA)TESTDB00/CONTROLOLD/01 REMOVED
ON UITEST

#EOT 3213  (ALMA) (ALMA)TAPEWORKER01/DUMPDISK4 ON
UITEST

#3210 DISPLAY:** WARNING: DID NOT FIND ANY FILES
WHICH MATCH REQUEST.

#3210 PK17096 (ALMA)TESTDB00/HLDUMPINFO/56812 REMOVED
ON UITEST

#ET=5.7 PT=0.1 IO=0.5
```

Note: When no files are found to match a RECOVER (ROWS USING BACKUP) request, DMUTILITY issues a warning message, removes the HLDUMPINFO and terminates DMUTILITY as normal.

Example 15

```
DB=ORIGINALDB OPTIONS (NOZIP) RECOVER
(REPLICATE THRU AUDIT 2) = FROM TESTDUMP

#RUNNING 0499
#0499 DISPLAY:>>>INPUT WAS:

#0499 DISPLAY:DB=ORIGINALDB OPTIONS (NOZIP) RECOVER
(REPLICATE THRU AUDIT 2) = FROM TESTDUMP

#BOT 6522  (SOMEUC)TAPEWORKER01/TESTDUMP ON THISPACK
#6519 PK131 (SOMEUC)ORIGINALDB/HLDUMPINFO/38367
REPLACED ON THISPACK

#6522 PK131 (SOMEUC)ORIGINALDB/CONTROLOLD/01
REPLACED ON THISPACK

#6522 DISPLAY:** SYNTAX ERROR: ** FROM **
REPLICATE REQUIRES AN "AS" CLAUSE.

#P-DS 6522 (SOMEUC) (SOMEUC)TAPEWORKER01/TESTDUMP
ON THISPACK

#6519 DISPLAY:** FATAL ERROR: PROCESS TERMINATED: .
#P-DS 23424000, 37031000, 49617000, 99612000.
```

This statement attempts to copy all structures to the same pack name or pack names that were in effect when the DMUTILITY dump (TESTDUMP) was created in preparation for a rebuild recovery. However, it terminates with an error because no destination was specified.

Example 16

```
DB=ORIGINALDB OPTIONS (NOZIP) RECOVER
(REPLICATE THRU AUDIT 2) =
AS (NEWUC)= FROM TESTDUMP

#
#RUNNING 0590
#0590 DISPLAY:>>>INPUT WAS:

#0590 DISPLAY:DB=ORIGINALDB OPTIONS
(NOZIP) RECOVER (REPLICATE THRU AUDIT 2) = AS
(NEWUC)= FROM TESTDUMP

#BOT 0593 (NEWUC)TAPEWORKER01/TESTDUMP
#0590 PK131 (NEWUC)ORIGINALDB/HLDUMPINFO/36648
REPLACED ON THISPACK
```

This statement copies all structures to the same pack name or pack names that were in effect when the DMUTILITY dump (TESTDUMP) was created. In addition, this statement changes the database usercode to NEWUC.

Example 17

```
DB=ORIGINALDB OPTIONS (NOZIP) RECOVER
(REPLICATE THRU AUDIT 2) = AS (NEWUC)
= ON NEWPACK FROM TESTDUMP

#RUNNING 0626
#0626 DISPLAY:>>>INPUT WAS:

#0626 DISPLAY:DB=ORIGINALDB OPTIONS
(NOZIP) RECOVER (REPLICATE THRU AUDIT 2)
= AS (NEWUC)= ON NEWPACK FROM TESTDUMP

#BOT 0629 (NEWUC)TAPEWORKER01/TESTDUMP
#0629 PK131 (NEWUC)ORIGINALDB/HLDUMPINFO/36648
REPLACED ON THISPACK
```

This statement copies all structures to a new location (NEWPACK) in preparation for a rebuild recovery. In addition, this statement changes the database usercode to NEWUC. The ON option is included in this example.

Example 18

```
DB=ORIGINALDB OPTIONS (NOZIP) RECOVER (REPLICATE THRU AUDIT 2)
= AS (NEWUC)=ON NEWPACK FROM MOST CURRENT

#RUNNING 9123
#9123 DISPLAY:>>>INPUT WAS:
```

```
#9123 DISPLAY:DB=ORIGINALDB OPTIONS (NOZIP)
RECOVER (REPLICATE THRU AUDIT 2) = AS (NEWUC)=
ON NEWPACK FROM MOST CURRENT
```

```
#9123 DISPLAY:**SYTNAX ERROR:**NEWDB** 'FROM'
<RECOVERY SOURCE> EXPECTED.
#P-DS 23424000, 45107000, 49558000, 99612000.
#ET=1.4 PT=0.5 IO=0.4
```

This statement attempts to copy all structures to a new location (NEWPACK) and also changes the database usercode to NEWUC. This syntax is not valid because a source database specification is required.

Example 19

```
DB=ORIGINALDB OPTIONS (NOZIP) RECOVER (REPLICATE THRU AUDIT
2)= AS (NEWUC)= ON NEWPACK FROM MOST CURRENT
```

```
#RUNNING 9996
#9996 DISPLAY:>>>INPUT WAS:
```

```
#9996 DISPLAY:DB=ORIGINALDB OPTIONS (NOZIP) RECOVER
(REPLICATE THRU AUDIT 2) = AS (NEWUC)= ON NEWPACK FROM
MOST CURRENT
```

```
#9996 DISPLAY:**FATAL ERROR:DMDUMPDIR MUST BE ENABLED
TO USE - MOST CURRENT.
#P-DS 23424000, 45120460, 49558000, 99612000.
#ET=1.4 PT=0.4 IO=0.4
```

This statement attempts to copy all structures to a new location (NEWPACK) and also changes the database usercode to NEWUC. Because a dump tape directory has not been enabled, the process terminates with an error.

Example 20

```
RECOVER (REBUILD THRU AUDIT 4567) FROM QUIESCE DB
```

This command moves the entire database forward by processing all audit since the time of the database QUIESCE. This process terminates after audit file 4567 is processed.

Example 21

```
RECOVER(REBUILD THRU AUDIT 999)
=FROM MOST CURRENT FULL
```

This statement selects only the most current full dump.

Example 22

```
RECOVER(REBUILD THRU AUDIT 999)=FROM MOST CURRENT ACCUM
```

This statement selects the most current full dump and those in the most current accumulated dump (if one is present and more current than the full dump).

Example 23

```
RECOVER(REBUILD THRU AUDIT 4567)= FROM ANOTHERDB063094 TAPE =
DBDUMPS063094
```

This command loads the entire database from the specified dump on a multidump tape and moves the entire database forward by processing all the audit since the time of the dump. This process terminates after audit file 4567 is processed.

Examples of the RECOVER Statement for Partial Database Recovery

The following examples illustrate the RECONSTRUCT (RECOVER ROWS USING BACKUP) function, which is used to recover damaged rows of database files.

Example 1

The following example causes the rows to be reconstructed from the backup dump DB1DUMP and tape DB1TAPE, and the entire audit since the time of the dump.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= FROM DB1DUMP TAPE = DB1TAPE")
```

Example 2

The following example causes the rows to be reconstructed from the backup dump DB1DUMP and tape DB1TAPE, and the entire audit since the time of the dump. The example uses the RESTORE option to recover the entire file from the backup dump and the audit files.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= (RESTORE) FROM DB1DUMP TAPE = DB1TAPE")
```

Example 3

The following examples recover damaged rows of the database DB1 since the most current dump, most current full, and most current accumulated dump respectively. When you are using the most current construct, you must enable DMDUMPDIR.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= FROM MOST CURRENT")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= FROM MUST CURRENT FULL")
```

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= FROM MUST CURRENT ACCUM")
```

Example 4

The following example causes the rows to be reconstructed from the backup dump DB1DUMP and tape DB1TAPE, and the entire audit since the time of the dump. The example enables the NOZIP option, which allows entire row recovery to be controlled by a single WFL.


```
RUN *SYSTEM/DMUTILITY("DB = DB1 OPTIONS(NOZIP)
RECOVER(ROWS USING BACKUP) DB1/= FROM DB1DUMP TAPE = DB1TAPE")
```

Example 5

The following example causes the rows to be reconstructed from the backup dump DB1DUMP and tape DB1TAPE, and the entire audit since the time of the dump. The example enables the FLUSHDB option, which helps determine when DMRECOVERY needs to take restart points.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 OPTIONS(FLUSHDB = 1)
RECOVER(ROWS USING BACKUP) DB1/= FROM DB1DUMP TAPE = DB1TAPE")
```

Example 6

The following example recovers rows from the specified family index 0 of the database DB1 from the dump DB1DUMP, which is stored on tape DB1TAPE.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= (FAMILYINDEX = 0) FROM DB1DUMP TAPE = DB1TAPE")
```

Example 7

The following example causes the rows to be reconstructed from the backup dump DB1DUMP and tape DB1 with the given serial number specification 493203 and the entire audit since the time of the dump.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= FROM DB1DUMP TAPE = DB1TAPE(SERIALNO = 493203)")
```

Example 8

The following example causes the rows to be reconstructed from the backup dump DB1DUMP and tape DB1TAPE with the given density specification FMTST9840 and the entire audit since the time of the dump.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 RECOVER(ROWS USING BACKUP)
DB1/= FROM DB1DUMP TAPE = DB1TAPE(DENSITY = FMTST9840)")
```

Examples of the RECOVER Statement for Whole Database Recovery

The following examples illustrate the use of the REBUILD function, which moves the entire database forward in time. DMUTILITY begins the REBUILD process by loading the entire database from one or more dumps on multidump tapes.

Example 1

The following command implements the REBUILD THRU AUDIT function, which applies the afterimages from all audit files created since the time of the dump. This process continues until all afterimages in the audit file specified by <integer> have been applied. This command applies to the database QRESCFILE, dump DB1DUMP, which is stored on tape QRESCFILETAPE.

```
RUN *SYSTEM/DMUTILITY("DB = QRESCFILE RECOVER(REBUILD
THRU AUDIT 999) QRESCFILE/= FROM DB1DUMP TAPE = QRESCFILETAPE")
```

Example 2

The following command will recover the entire database QRESCFILE since the most current dump, most current full, and most current accumulated respectively.

```
RUN *SYSTEM/DMUTILITY("DB = QRESCFILE RECOVER(REBUILD
THRU AUDIT 999) QRESCFILE/= FROM MOST CURRENT")
```

```
RUN *SYSTEM/DMUTILITY("DB = QRESCFILE RECOVER(REBUILD
THRU AUDIT 999) QRESCFILE/= FROM MOST CURRENT FULL)
```

```
RUN *SYSTEM/DMUTILITY("DB = QRESCFILE RECOVER(REBUILD
THRU AUDIT 999) QRESCFILE/= FROM MOST CURRENT ACCUM")
```

Example 3

The following command will recover the entire database QRESCFILE up from a specified time point of AUGUST 25 AT 12:00, from dump DB1DUMP, which is stored on tape QRESCFILETAPE.

```
RUN *SYSTEM/DMUTILITY("DB =
QRESCFILE RECOVER(REBUILD TO GEQ AUGUST 25 AT 12:00)
QRESCFILE/= FROM DB1DUMP TAPE = QRESCFILE TAPE")
```

Example 4

The following command will recover the entire database QRESCFILE from dump DB1DUMP, which is stored on tape QRESCFILETAPE, using the NOZIP option, which will allow entire database recovery to be controlled by a single WFL.

```
RUN *SYSTEM/DMUTILITY("DB =
QRESCFILE OPTIONS(NOZIP) RECOVER(REBUILD THRU AUDIT 999)
QRESCFILE/= FROM DB1DUMP TAPE = QRESCFILETAPE")
```

Example 5

The following command recovers the entire database QRESCFILE from dump DB1DUMP, which is stored on tape QRESCFILETAPE using the FLUSHDB option, which helps determine when DMRECOVERY needs to take restart points.

```
RUN *SYSTEM/DMUTILITY("DB =
QRESCFILE OPTIONS(FLUSHDB = 1) RECOVER(REBUILD THRU AUDIT 999)
QRESCFILE/= FROM DB1DUMP TAPE = QRESCFILETAPE")
```

Example 6

The following command recovers files that are stored on family index 0 from the database QRESCFILE. This backup exists on dump DB1DUMP, which is stored on tape QRESCFILETAPE.

```
RUN *SYSTEM/DMUTILITY("DB = QRESCFILE
RECOVER(REBUILD THRU AUDIT 999) QRESCFILE/=
(FAMILYINDEX = 0) FROM DB1DUMP TAPE = QRESCFILEEETAPE")
```

Example 7

The following command recovers the entire database QRESCFILE from dump DB1DUMP, which is stored on tape QRESCFILETAPE with the specified serial number 394039.

```
RUN *SYSTEM/DMUTILITY("DB = QRESCFILE
RECOVER(REBUILD THRU AUDIT 999) QRESCFILE/=
FROM DB1DUMP TAPE = QRESCFILETAPE(SERIALNO = 394039) ")
```

Example 8

The following command recovers the entire database QRESCFILE from dump DB1DUMP, which is stored on tape QRESCFILETAPE with the specified tape density FMTST9840.

```
RUN *SYSTEM/DMUTILITY("DB = QRESCFILE
RECOVER(REBUILD THRU AUDIT 999) QRESCFILE/=
FROM DB1DUMP TAPE = QRESCFILETAPE(DENSITY = FMTST9840) ")
```

Database Recovery Using Incremental and Accumulated Dumps

When using incremental and accumulated dumps in the recovery or copy process, the order of the dumps specified in the recovery source list must adhere to the following guidelines:

- A full dump must be designated as the first dump specification.
- One accumulated dump and one or more incremental dumps follow a full dump.
- An accumulated dump cannot follow an incremental dump.
- Dumps specified must be in the order in which they are performed.
- When DMUTILITY is running the RECOVER or COPY command to recover the database, each tape name must include "CYCLE = <last cycle>" and "VERSION = <last version number>" in the <tape specification> construct unless the dump is stored on a multidump tape.

Note: *Tape processing becomes single threaded when DMUTILITY is loading from a multidump tape.*

If the recovery starts and the dumps are not specified according to the previous guidelines, you must choose whether to abort or initialize the recovery process without using the remaining dump files by processing the necessary audit files.

Although the incremental and accumulated dump feature is available, all recoveries require a full dump. The number of audit files saved remains the same. It is recommended that you save at a minimum the last two to three dumps along with the audit files generated from the same timeframe. If a recovery process using an incremental or accumulated dump fails, the alternative is to recover the database using only the full dump and corresponding audit files.

The DMUTILITY and DMDUMPDIR programs allow recovery using the RECOVER FROM MOST CURRENT construct to automatically access the correct dumps when incremental and accumulated backups exist. The database DUMPSTAMP option enables the creation of incremental and accumulated backups.

The DMUTILITY and DMDUMPDIR programs allow recovery using the RECOVER FROM MOST CURRENT FULL construct to automatically access the most current full backup with the database DUMPSTAMP option enabled. This recovery occurs even when corresponding accumulated backups, incremental backups, or both backups exist.

The DMUTILITY and DMDUMPDIR programs allow recovery using the RECOVER FROM MOST CURRENT ACCUM construct to automatically access the correct dumps when you want the most current full and accumulated backup. This recovery occurs even when corresponding incremental backups exist.

Running Recovery

The format for running partial database recovery is the following:

```
RUN RECONSTRUCT/<database name>
```

The format for running partial database recovery under a permanent directory is the following:

```
RUN <path name>/RECONSTRUCT/<database name>;  
DATAPATH = *DIR/TEST ON TESTPACK;
```

The format for running whole database recovery is the following:

```
RUN SYSTEM/DMRECOVERY ("<db statement>")
```

The format for running whole database recovery under a permanent directory is the following:

```
RUN SYSTEM/DMRECOVERY ("<db statement>");  
DATAPATH = <path name> ON <family name>
```

These are some of the messages that can appear during ABORT recovery or when running RECONSTRUCT or DMRECOVERY:

- AUDITUPDATE LEVEL IS <#>. RECOVERY UPDATE LEVEL IS <#>. The <#> is a number representing the update level. The audit file has an update level that differs from that of recovery. You can either continue the run or discontinue it.
- THE FOLLOWING AUDIT BLOCK IS IN ERROR. There is something wrong with the audit file. The run discontinues.
- AUDIT IO ERROR: "TERMINATE" TO FINISH NORMALLY - OTHERWISE "DS" The database is in sync, but recovery has not progressed to the user-specified point. Enter *TERMINATE* to stop processing or *DS* to discontinue the run. A *TERMINATE* command completes processing to the point before the error occurred. For example,

during a REBUILD THROUGH AUDIT FILE 9 command, if an error occurs at audit file 7, the TERMINATE command completes processing at audit file 6.

- INSERT WRITE RING ON UNIT - <#>; THEN AX TO CONTINUE.

The <#> is a number representing the tape drive number. DMRECOVERY needs to write to tape, but the tape has no write ring. Insert a write ring on the tape; then use the AX (Accept) command on the operator display terminal (ODT) to continue.

- WORKER # <#> TERMINATED ABNORMALLY. AX TO RESTART WORKER <#>.

The <#> construct is a number representing the worker number. Something happened to one of the RECONSTRUCTFILTER workers. Either correct the problem and continue the run with an AX command, or discontinue it with a DS (Discontinue) command.

For more information on the AX and DS commands, refer to the *System Commands Operations Reference Manual*.

Note: DMUTILITY passes the parameters to the recovery programs by way of the files <dbname>/RECONSTRUCTINFO (for row recovery), <dbname>/REBUILDINFO (for REBUILD), and <dbname>/ROLLBACKINFO (for ROLLBACK). The layouts of these files can be found in DATABASE/PROPERTIES.

- NO FILE <database name>/AUDIT<nnn> (MT) #1 <nnn>:0

The requested audit file is not on disk, and the recovery program looks for the audit file on tape.

You can mount the corresponding audit tape or use the FILE ASSIGNMENT command to specify the disk pack on which the audit file resides. If you use the FILE ASSIGNMENT command, specify CYCLE = 1 in the FILE ASSIGNMENT statement. If you do not specify CYCLE = 1, the message "Unmatched Genealogy" appears.

- ACCEPT: RETRY OR FAMILY = <family name> TO CONTINUE
THE ROLLBACK OR "STOP" TO END ROLLBACK
*** CHOOSING STOP WILL FINISH ROLLBACK AT AFN = <n>,
ABSNS = <number>, AUDIT TIME = <date-time> ***

Rollback recovery detected a missing audit file. If you choose the AX STOP option, rollback recovery ends at the indicated AFN, ABSNS, and time. Take one of the following actions:

- Provide the requested audit file and continue the rollback recovery by entering AX RETRY or AX FAMILY = <family name>
- End the rollback recovery at a consistent point by entering AX STOP.

- ACCEPT: RETRY OR FAMILY = <family name> TO CONTINUE THE ROLLBACK

Rollback recovery detected a missing duplicate audit. Provide the audit and continue the run with either AX RETRY or AX FAMILY = <family name>. You can also restart the recovery run after you provide the missing audit and enter the DS command.

TAPESERVER System Option and RoboHost Units

A subtle operational change occurs if the TAPESERVER system option is set. If you designate an audit file that cannot be retrieved by a RoboHost unit because either the RoboHost unit or the tape itself is not available, then perform the following steps:

1. Use the NF (No File) system command to respond to the No File condition.
2. Supply the required AX response to the following user message:

```
RETRY OR FAMILY = <FAMILY NAME>
```

Visible Recovery Commands

The Visible Recovery commands are similar to the Visible DBS commands (described in [Section 12, Communicating with the Database](#),) in that they enable you to change various database factors, display status information, and print statistics. However, the differences between the Visible Recovery commands and Visible DBS commands are as follows:

- The Visible Recovery commands enable you to monitor and optimize recovery operations rather than normal database operations.
- While the Visible DBS commands are issued as SM (Send to MCS or database) system commands to the database stack, the Visible Recovery commands are issued as AX (Accept) system commands to an internal task that is dependent on the recovery task.

If you display all of your jobs in the mix, results similar to the following example appear. The SYSTEM/DMRECOVERY task has a different mix number (162) than its dependent task (165), which follows it in the list. You issue a Visible Recovery command against dependent task 165, not the SYSTEM/DMRECOVERY task of 162.

```
Mix Pri          JOB ENTRIES (ALL) USER=PROD1
161  50      JOB (PROD1)
162  50      ..(PROD1) *SYSTEM/DMRECOVERY ON SYS432
165  50      ....(PROD1) (PROD1)CTRLDB ON ADDTEST
```

To issue a Visible Recovery STATUS command for the preceding example, type

```
165 AX STATUS
```

The two categories of Visible Recovery commands are

1. Tuning commands

Use the tuning commands to tailor the amount of memory used by a recovery operation and how often information is written to disk. The three tuning commands are

- ALLOWEDCORE = <integer>
- OVERLAYGOAL = <decimal value>
- WRITEDELAYFACTOR = <decimal value>

2. Monitoring commands

Use the monitoring commands to check on the status of a recovery operation. The three monitoring commands are

- STATUS
- STATISTICS
- STATISTICS CLEAR

The following subsections explain each command in detail.

ALLOWEDCORE = <integer> Command

Use this Visible Recovery command to designate the amount of core memory available for the recovery process.

Initial Value

The initial ALLOWEDCORE value for a recovery operation is the value set for the ALLOWEDCORE parameter in the database control file. The default value is 50,000 words.

Maximum Value

The maximum ALLOWEDCORE setting for a recovery operation is 549,755,813,887 words.

Effect of Changes

Any changes you make to the ALLOWEDCORE setting are written to the database control file and remain in place after the recovery operation completes.

To optimize the ALLOWEDCORE setting for normal database operations, use the Visible DBS *CHANGE* command `ALLOWEDCORE = <unsigned integer>`. For more information on the Visible DBS commands, refer to [Section 12, Communicating with the Database](#).

Optimizing Normal Database and Recovery Operations

If, in your environment, it is appropriate to use different ALLOWEDCORE settings for recovery and normal database operations, perform the following steps to optimize both operations:

1. Note the normal database operation setting for the ALLOWEDCORE value.
2. Initiate the recovery operation.
3. Optimize the ALLOWEDCORE setting for the recovery operation.
4. Open the database after recovery completes.
5. Use the Visible DBS *CHANGE* command `ALLOWEDCORE = <unsigned integer>` to reset the ALLOWEDCORE setting to the value noted in step 1.

OVERLAYGOAL = <decimal value> Command

Use this Visible Recovery command to control the rate at which buffers are overlaid to disk during a recovery operation.

Initial Value

The initial setting for the overlay rate is the value set for the OVERLAYGOAL parameter in the database control file. By default, the overlay rate is 5 percent of the ALLOWEDCORE parameter setting per minute. For example, if the ALLOWEDCORE setting is 100,000 words, the default overlay rate is 5,000 words per minute.

Allowed Values

You can set the OVERLAYGOAL parameter to any decimal value in the range 0 to 100. If you assign a nonzero value to the OVERLAYGOAL parameter, the maximum number of buffers is limited to the smaller of the following two values:

- 512 buffers or the number of system buffers if that number is larger
- The number of buffers within the limits of the ALLOWEDCORE setting

Effect of Changes

Any changes you make to the OVERLAYGOAL parameter are written to the database control file and remain in place after the recovery operation completes.

To reset the value for normal database operations, use the Visible DBS *CHANGE* command `OVERLAYGOAL = <value>`. For more information on the Visible DBS commands, refer to [Section 12, Communicating with the Database](#).

Optimizing Normal Database and Recovery Operations

If, in your environment, it is appropriate to use different overlay rates for recovery and normal database operations, perform the following steps to optimize both operations:

1. Note the normal database operation setting for the OVERLAYGOAL parameter.
2. Initiate the recovery operation.
3. Optimize the overlay rate for the recovery operation.
4. Open the database after recovery completes.
5. Use the Visible DBS *CHANGE* command `OVERLAYGOAL = <value>` to reset the overlay rate to the value noted in step 1.

WRITEDELAYFACTOR = <decimal value> Command

Use this Visible Recovery command to set the time (in seconds) that must elapse after a buffer is modified and before a writeahead operation is initiated.

Each time a buffer is modified, the changed information must be written to disk. Usually as soon as a buffer is modified, a write operation is initiated to write the changed information to disk. If the same buffer is modified repeatedly, delays in the recovery operation can occur because the buffer cannot be modified again until the change has been written to disk.

Using the `WRITEDELAYFACTOR` command you can limit how often a buffer is written to disk. [Figure 8-1](#) illustrates the standard buffer access/write operation scenario. In this scenario, each buffer access is followed by a write operation.

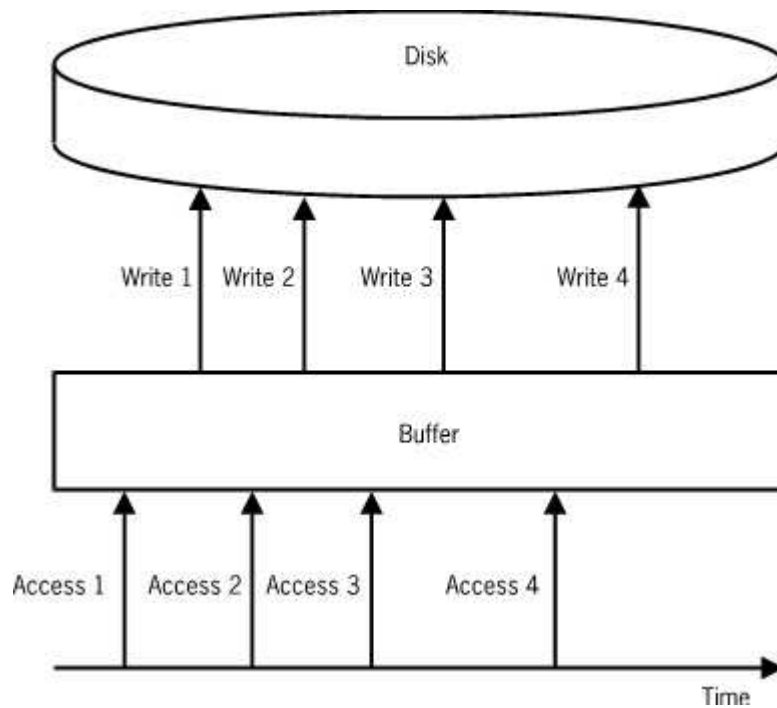


Figure 8-1. Standard Buffer Access/Write Operation Scenario

[Figure 8-2](#) illustrates the effect of the `WRITEDELAYFACTOR` command. In this scenario, several buffer accesses occur before each write operation. The write operation occurs when the time interval you specify in the `WRITEDELAYFACTOR` command has elapsed. If a buffer access occurs before the specified time has elapsed, the time counter is reset.

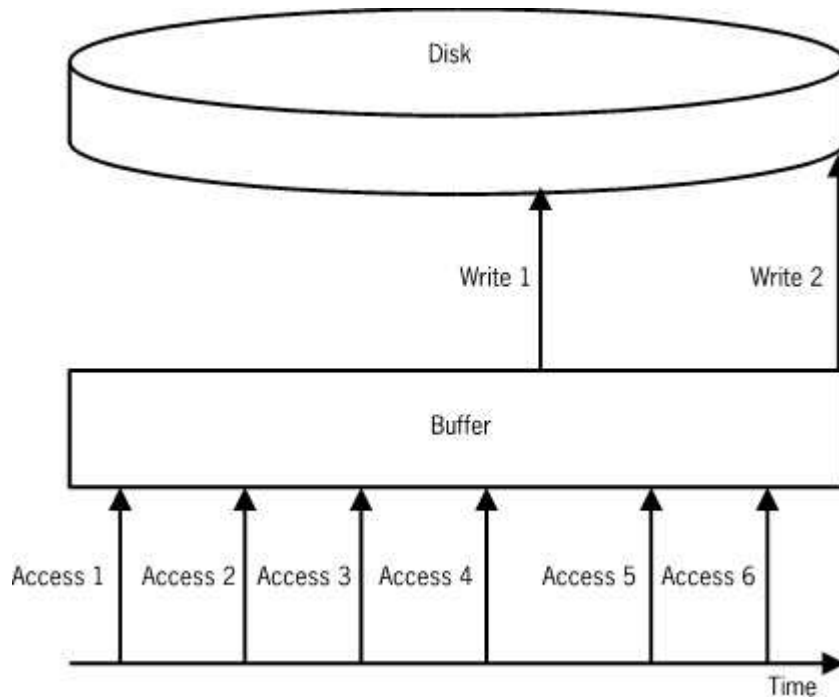


Figure 8-2. Buffer Access/Write Operation Scenario with WRITEDELAYFACTOR Effect

Initial Value

The WRITEDELAYFACTOR value is set initially to 0.2 seconds. The minimum WRITEDELAYFACTOR value you can assign is 0. There is no maximum value.

Do not assign too large a value in the WRITEDELAYFACTOR command because keeping too much information in memory can adversely affect performance. To optimize recovery performance, you must find a balance between performing write operations too frequently and keeping too much information in memory.

STATUS Command

Use this Visible Recovery command to display the current status of the recovery operation. [Figure 8-3](#) is a sample Visible Recovery status display.

```

STATISTICS START: MM/DD/YY 10:35: 2
STATISTICS END:   MM/DD/YY 14:37:29   INTERVAL:      0 DAYS, 0: 2:27
AUDIT START:     MM/DD/YY 20:35:46
AUDIT END:       MM/DD/YY 20:35:46   INTERVAL:      0 DAYS, 0: 0: 0
CURRENT AFN=1, ABSN = 420
ALLOWEDCORE: 3000, TOTALCORE: 2976
OLAYGOAL: 5, OLAYRATE: 5.0472, OVERLAYS:12
READAHEAD FACTOR: 220 AUDIT RECORDS
WRITEDELAYFACTOR: 0.2 SECONDS
READS: 3910, READAHEADS: 3902, AVG IO TIME: .0196788, AVG IO WAIT: 6.0247-4
WRITES: 4568, WRITEAHEADS: 4568, AVG IO TIME: .0203959, AVG IO WAIT: .0012135
    
```

Figure 8-3. Sample Visible Recovery Status Display

Each of the elements in the status display is explained in the following text.

STATISTICS START, STATISTICS END, INTERVAL

These values represent the start, end, and interval time either since the recovery operation started or since you issued the Visible Recovery command STATISTICS CLEAR, whichever is the most recent.

AUDIT START, AUDIT END, INTERVAL

These values represent the times associated with the first and the most recent control points encountered in the audit trail, and the gap between the two control points. The times provided are the original, real times as found in the audit trail. If no end control point is encountered, the audit interval is shown as 0 (zero), and the AUDIT START and AUDIT END values are identical.

CURRENT AFN, ABSN

These values identify the current position, in terms of the audit file number (AFN) and audit block serial number (ABSN), of the recovery operation in the audit trail.

ALLOWEDCORE, TOTALCORE

These values identify the maximum amount of memory the recovery operation is allowed to use (ALLOWEDCORE value) and the amount actually in use (TOTALCORE value).

If the TOTALCORE value is the same as the ALLOWEDCORE value, consider using the Visible Recovery command ALLOWEDCORE to increase the amount of core memory available for the recovery operation.

READAHEADFACTOR, WRITEDELAYFACTOR

The READAHEADFACTOR value identifies the number of audit records to be scanned during a readahead cycle. That is, the READAHEADFACTOR value determines the amount of audit information that has been read from the audit trail and that is currently waiting to be processed.

The value of READAHEADFACTOR is altered as a result of performing readaheads. If readaheads are considered successful, the READAHEADFACTOR value is increased, meaning that more audit records are scanned in the next readahead cycle. Readaheads are considered unsuccessful if the number of physical reads initiated during the readahead cycle fall below a specified percentage. This drop-off in physical reads might happen if the majority of the blocks needed are already in memory or are in the process of being read. If readaheads are unsuccessful, the READAHEADFACTOR value is decreased so that fewer audit records are scanned in the next readahead cycle. The READAHEADFACTOR value is initialized to 20.

To optimize recovery performance, you must maximize the readahead and writeahead operations while minimizing the amount of I/O wait time.

The WRITEDELAYFACTOR value provides the current setting for the WRITEDELAYFACTOR factor. The WRITEDELAYFACTOR value enables several buffer accesses to occur before a write operation is initiated. This action can reduce the time spent waiting for write operations to complete.

The default setting for the WRITEDELAYFACTOR factor is 0.2 seconds.

You can alter the setting for this factor by using the Visible Recovery command WRITEDELAYFACTOR.

For more information, refer to “WRITEDELAYFACTOR = <decimal value> Command” earlier in this guide.

Additional Status Report Information

The Visible Recovery statistics report also contains the following information. In all cases, the values represent the counts either since the recovery operation started or since you issued the last STATISTICS CLEAR command, whichever is the most recent.

- The number of read operations performed
- The number of write operations performed
- The number of readahead operations performed

A low value for the number of readahead operations might indicate that all the ALLOWEDCORE memory is being used. It might also indicate that the setting for the WRITEDELAYFACTOR factor is too high.

- The number of writeahead operations performed
- The average number of seconds for I/O operations
- The average number of seconds spent waiting for I/O operations

STATISTICS and STATISTICS CLEAR Commands

Use the Visible Recovery STATISTICS command to generate a report or listing of recovery operation statistics. Use the Visible Recovery STATISTICS CLEAR command to generate a statistics report, clear the existing statistics, and start a new interval. For both commands, the information is generated as a printer backup (BD) file.

Sample Visible Recovery Statistics Report

Figure 8-4 shows a sample Visible Recovery statistics report.

```

*** HALT/LOAD RECOVERY STATISTICS REPORT FOR JOB 9866/9866 ***
STATISTICS START:   MM/DD/YY  14:22:29
STATISTICS END:     MM/DD/YY  14:23:50   INTERVAL:   0 DAYS, 0: 1:21
AUDIT START:        MM/DD/YY  14:53:37
AUDIT END:          MM/DD/YY
YY 11:58:36   INTERVAL:   0 DAYS, 21: 4:60
CURRENT AFN=1, ABSN = UNDEFINED
ALLOWEDCORE: 50000, TOTALCORE: 1173
OLAYGOAL: 5, OLAYRATE: 0, OVERLAYS:0
READAHEAD FACTOR: 0 AUDIT RECORDS
WRITDELAYFACTOR: 0.2 SECONDS
DMREAD STATISTICS
TOTAL DMREAD CALLS          62
GETBUFFERS                   2    3.22%
ACTUALREADS                   60   96.77%
READAHEADS                     1    1.61%
NON-DISK READS                 61   98.38%
DISK READS                      1    1.61%
LOGICAL READS                   59   95.16%
LOGICAL READS ON READAHEAD      1  100.00%
NUMBER OF WRITE AHEADS         0    0.00%
AUDIT IMAGE APPLICATION STATISTICS
(1) AUDIT RECORDS SCANNED WHEN APPLYING AFTER IMAGES
(2) AUDIT RECORDS ACTUALLY APPLIED AS AFTER IMAGE
(3) AUDIT RECORDS SCANNED WHEN APPLYING BEFORE IMAGES
(4) AUDIT RECORDS ACTUALLY APPLIED AS BEFORE IMAGE
AUDIT RECORD
TYPE#  MNEMONIC          (1)      (2)      (3)      (4)
  2    BCP                21       0       0       0
  3    ECP                21       0       0       0
  4    BTR                20       0       0       0
  5    ETR                20       0       0       0
 10    DSC                20      20       0       0
 16    ADSS              20      20       0       0
 21    DBSI               7        0       0       0
 22    DBST               8        0       6       0
 29    RECOV             12       0       0       0
 32    RDSO              13       0       0       0
 33    RDSC              13       0       0       0
 76    LGRR               1        0       0       0
 79    STRDC              3        0       0       0
 81    SAC                20       0       0       0
 91    SIBO               1        0       0       0
 92    SIBC               1        1       0       0
101    AISE2             20      20       0       0
-----
TOTALS                    221      60      6       0

```

```

DATABASE I/O STATISTICS
(1) NUMBER OF READS
(2) NUMBER OF READ AHEADS
(3) AVERAGE READ I/O TIME (SECONDS)
(4) AVERAGE READ WAIT TIME (SECONDS)
(5) NUMBER OF WRITES
(6) NUMBER OF WRITE AHEADS
(7) AVERAGE WRITE I/O TIME (SECONDS)
(8) AVERAGE WRITE WAIT TIME (SECONDS)
(9) NUMBER OF BUFFERS
STRUCTURE *-----READS-----*-----WRITES-----*--
NO. NAME (1) (2) (3) (4) (5) (6) (7) (8) (9)
3 DATASETONE 0 0 0.00000 0.00000 2 0 0.01849 0.00008 2
4 S2345678901234567 1 0.02570 0.02726 1 0 0.01728 0.00009 1
-----
TOTALS 1 0 0.02570 0.02726 3 0 0.03577 0.00017 3
AUDIT I/O STATISTICS
NUMBER OF READS 15
NUMBER OF READ AHEADS 0
TOTAL READ WAIT TIME 0.0 SECONDS
AVERAGE READ WAIT TIME 6.0 MS.
    
```

Figure 8-4. Sample Visible Recovery Statistics Report

COPY Statement (DMUTILITY)

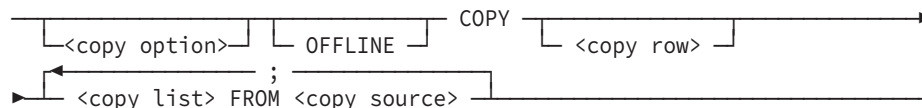
The COPY statement is used to recover unaudited databases. It is also used to copy an offline dump of the entire database, audited or unaudited, from tape to disk. The syntax for this statement is illustrated and explained on the following pages.

You can copy the following from backup dumps:

- Parts of the database
- Selected rows
- Selected family indexes
- Rows from one or more partial dumps

Note: Use the COPY statement only to restore data from a single dump or a set of dumps that were created from a single DUMP statement. Examples 14 and 15 later in this section demonstrate copy operations from multiple dumps.

Syntax



<copy option>

```

— OPTIONS — ( — /1\— WORKERS = <integer> — ) —————|
                | /1\— BYCYCLE —————|
                | /1\— QDCVERIFY —————|
                | /1\— QDCWORKERS = <integer> —|
  
```

<copy row>

```

— (ROWS USING BACKUP) —————|
  
```

<copy list>

```

— <file name> —————|
  | ( <copy list> ) | | <copy selector> | | <copy dest> |
  |—————| |—————| |—————|
  
```



<copy selector>

```

— ( —————|
  | AND |
  | & |
  |—————|
  | /1\— FAMILYINDEX — = —<range> —|
  | /1\— ROW — = —<range> —|
  | /1\— PACKNAME — = —<family name> —|
  | /1\— SECTION — = —<range> —|
  |—————|
  ) —————|
  
```

<copy dest>

```

— <copy as> —————|
  | <copy to> |
  |—————|
  | <copy onto> |
  | <copy to> |
  |—————|
  
```

<copy as>

```

— AS <file name> —————|
  | ON <family name> |
  |—————|
  
```

<copy onto>

```

— ONTO <file name> —————|
  | ON <family name> |
  |—————|
  
```

<copy to>

```

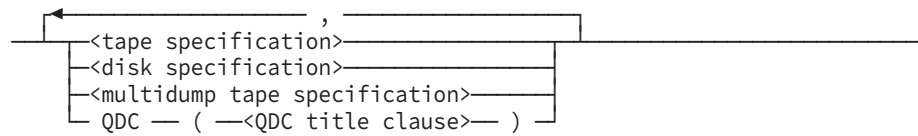
— TO — ( — FAMILYINDEX = —<integer> — ) —————|
                | RETAIN |
  
```

<range>

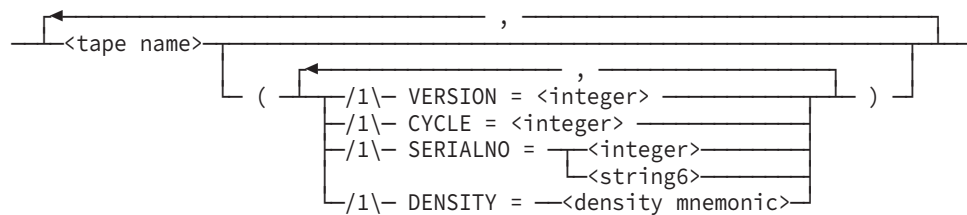
```

— <unsigned integer> —————|
  | — <unsigned integer> |
  |—————|
  
```

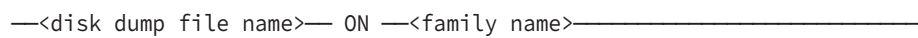

<copy source>



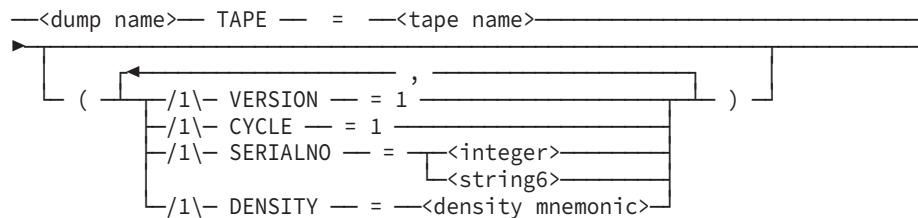
<tape specification>



<disk specification>



<multidump tape specification>



Explanation

The COPY syntax you designate determines if the entire database is copied or if only a portion of the database is copied.

Note: To copy the entire database, ensure that the necessary database dump tapes are specified.

DMUTILITY can perform a COPY without a current level control file being present. If the control file is present in the <copy list>, then the COPY can proceed by reading the control file from the <copy source> and opening this as the control file.

The <copy dest> specifies the location of the control file for the destination database. If the <db statement> does not specify <usercode> or <family name>, then these are inferred and use the information from the <copy dest>. In the absence of an explicit specification in the <copy dest>, the default destination is assumed. All information specified in the <db statement> must match the information implied or explicitly stated in the <copy dest>.

If the control file is contained in the <copy list> and its specification in the <copy dest> matches all information in the <db statement>, then a COPY can be performed without a current level control file. If the control file specification in the <copy dest> does not match the <db statement>, then a fatal error results. If the control file is not in the <copy list> and is not located by the <db statement>, then a fatal error results.

If the <copy selector> is specified or if not all the tape dumps are used in the <copy source>, then only the selected rows or the rows of the specified tapes are copied.

The <copy row> syntax must be specified so that the selected rows or the rows in the partial dump are copied into the destination file. To copy only a portion on the database, the destination files must exist. If the destination files do not exist, DMUTILITY waits on a No File condition.

Note: *Use extreme caution when copying only parts of a database. Data corruption and inconsistencies can easily occur.*

If used for any purpose other than copying the entire database from an offline dump, the COPY statement causes the version timestamps of the files being copied to be changed so that they no longer match the version timestamps in the control file.

It is recommended that you do not override the version timestamps. The most common reasons for a version timestamp mismatch are that you did not specify OFFLINE in the dump statement or that all of the rows of a structure were not reloaded.

If the entire database was copied from an offline dump to a different family pack, and the database has been updated to point to the new pack, you will encounter version timestamp mismatch for both the restart and global data set the next time the database is opened. This is because the version timestamps of the restart and global data sets were updated during the offline dump, and they no longer match the version timestamps in the control file. In this case, it is safe to override the version timestamps.

For audited databases, take care to avoid creating undetected audit discontinuities. Even when the entire database is copied, the audit trail must be taken into consideration and maintained with the copy. The end of the audit must be restored to its position at the time the database was dumped. Thus, when making a dump, the last audit file should be kept with the dump in case the database is to be copied back at a later time. Copying from an online dump is not recommended.

<copy option>

The following options are available for copy:

- WORKERS

This option controls the maximum number of tape drives used to recover files in parallel. The maximum number of workers or tape drives that can be specified is 50. WORKERS is not available for use with multidump tapes.

For single dump tapes, increasing the number of workers can decrease the total time required to complete the database recovery process. Limiting the number of workers can free up tape drives for other purposes. If the same file has been copied to multiple tapes, multiple workers cannot copy the same rows of that file at the same time. Rather, the second worker and any subsequent workers wait for the first worker to finish before continuing. Normally, this occurs if multiple dump tapes were used, each with a different tape name. In this case, the database control file is present on cycle 1, version 1 of each tape.

- BYCYCLE

When reloading the database, DMUTILITY processes one READVOLUME task for each physical task for each physical tape required. The READVOLUME task is processed in the following order if DMUTILITY has access to all the cycles and versions from which the dump is composed: (CYCLE 1, VERSION 1), (CYCLE 1, VERSION 2), . . . (CYCLE 1, VERSION n), (CYCLE 2, VERSION 1), (CYCLE 2, VERSION 2), . . . (CYCLE 2, VERSION m), . . . (CYCLE x, VERSION y). Each READVOLUME task reads its own tape and usually the beginning of the next tape, since the last row normally splits across two tapes.

If you did not include a cycle and version specification in the syntax, DMUTILITY reads the tape directory from the first tape. This tape only has information about the number of cycles, not the number of versions for each cycle. In this case, DMUTILITY processes one READVOLUME task for each cycle because the directory shows only one version exists for each cycle.

If you include a cycle and version number in the syntax, DMUTILITY reads the tape directory from that tape, which likely has more information about all the cycles and the actual number of versions for each cycle. In this case, one READVOLUME task is processed for each version in the order previously described.

This process is designed to locate the desired rows as soon as possible. However, it can appear that the same tape has unnecessarily been requested multiple times by different READVOLUME tasks if all the rows are being reloaded.

When specified, the BYCYCLE option forces DMUTILITY to process one READVOLUME task for each cycle instead of for each version even if DMUTILITY has knowledge of multiple versions existing for some cycles. This action is most efficient when the entire database is being reloaded, since all versions have to be read. One READVOLUME task per cycle means an independent task for each cycle so that tapes do not need to be requested multiple times by different READVOLUME tasks.

When DMUTILITY is copying rows, this option can cause extra searching time for the desired rows that do not reside on the first version of the required cycle.

Note: Because CYCLE is always 1 for multidump tapes, tape processing becomes single threaded when DMUTILITY is loading from a multidump tape.

-  QDCVERIFY

This option initiates the verification of selected data during the COPY process when QDC (<QDC title clause>) is designated in the <copy source> construct. The verification detects CHECKSUM and ADDRESSCHECK errors of the selected data.

- **QDCWORKERS**

This option controls the number of workers to be processed in parallel during a copy that had QDC (<QDC title clause>) designated in the <copy source> construct. From 1 to 50 workers can be specified. If the QDCWORKERS option is not specified, the value 1 is assumed.

OFFLINE

This option prevents other users from accessing the database. DMUTILITY waits for all programs that opened the database to complete processing before it performs the OFFLINE copy. If a database is being copied AS or ONTO another database, OFFLINE also prevents users from accessing the destination database. When the OFFLINE copy is successfully completed, DMUTILITY unlocks the source database control file. It also unlocks the destination database control file, if the COPY was AS or ONTO another database.

If DMUTILITY is discontinued during the OFFLINE copy or fails to unlock the control file, the CANCEL statement can be used to unlock the source database control file. Do not cancel the OFFLINE copy lock of the copied control file.

<copy row>

This option copies selected row or rows from a partial dump when performing a partial database copy. The rest of the rows are retained. If this option is not specified when performing a partial database copy, the selected rows are copied but the remaining rows are discarded. This action could cause database corruption.

This option creates temporary files that are deleted at the end of the copy process. The temporary file names take the following format:

```
<database file name>/TEMP<yyyymmddhhmm>
```

Ensure that there is enough disk space to create these temporary files. The amount of disk space required depends on the size of the structures being copied.

<copy list>

This option designates the files and rows to be loaded. The slash equal sign (/=) can be used to copy a family of files. The equal sign (=) alone designates that all files on the dump tape are to be loaded.

<copy selector>

This option specifies which rows of the file are to be copied. If a copy list is enclosed in parentheses and a copy selector is specified, then all database files in that copy list are restricted by the copy selector. Database files in the copy list that already have a copy selector specification have the outer selection constraints related to the inner selection constraints through the Boolean construct OR. Specifying SECTION in the copy selector causes only those rows that belong to the specified sections to be copied.

FAMILYINDEX

If specified in the copy selector, only those rows that were present on the specified family indexes at the time of the dump are copied from the dump tape. If the dump specified in the copy source does not reflect the current physical placement of the rows on the pack family indexes, you must be careful to ensure that the rows copied are the ones wanted. This situation can arise if, sometime after the family index dump is taken, a COPY or RECOVER is performed without retaining the family index.

ROW

If specified in the <copy selector> construct, only those particular rows are copied. PACKNAME allows the user to limit DMUTILITY to a particular pack family without enumerating the files that were present on the specified PACKNAME at the time of the dump.

<copy dest>

This option identifies where the rows are to be copied. If <copy dest> is not specified, DMUTILITY creates a new file on disk. The title of the new file copied is the same as the title of the file on the dump tape.

<copy as>

This option causes DMUTILITY to copy the rows of a file from the dump tape and create a new file on disk. The title of the newly created file is the file name specified in the <copy as> specification. The file is copied to the pack family specified in the ON <family name> clause, whether or not the original family was the same. If the ON <family name> clause is not specified, the file is copied to its original pack family.

You can also use the <copy as> syntax to move a database to a new location.

These same steps can be used to access data from dumps that are older than the current level of Enterprise Database Server software, which are called archive dumps. The archive retrieval process allows the current level of the Enterprise Database Server software to gain access to data that was backed up under previous levels of Enterprise Database Server software.

Recovering the Database

In the case of archive retrieval, avoid copying the files over the top of current database files.

1. Transfer the dump, DASDL source, and description file from their original location to a new location and usercode.
2. Modify all of the appropriate places in the DASDL source to reflect the new pack location. Make usercode and Enterprise Database Server software title changes at this time.
3. Ensure that the following DASDL options are also included:
 - UPDATE
 - \$SET ZIP
 - \$RESET DMCONTROL

4. Generate an updated description file and compile a new DMSUPPORT library by compiling the modified DASDL source. For example,

```
COMPILE MYNEW/DASDL AS $MYDBNAME
```

5. Create a compatible control file by using the DMCONTROL INITIALIZE option. For example,

```
RUN $SYSTEM/DMCONTROL("DB=MYDBNAME INITIALIZE ")
```

6. Use DMUTILITY to copy the database for traditional databases. For example,

```
RUN $SYSTEM/DMUTILITY("DB=MYDBNAME COPY = AS (NEWUSER) = ON  
NEWPACK FROM MYDBNAMEDUMP ")
```

In addition to the data files, DMUTILITY copies the control file from the dump.

When you use DMUTILITY to copy a permanent directory database, use one of the following forms.

- To replace the files in *DIR/NODE/MYDBNAME, use

```
RUN $SYSTEM/DMUTILITY("DB=MYDBNAME COPY  
= FROM MYDBNAMEDUMP"); DATAPATH = *DIR/NODE ON PACKNAME
```

- To copy the database to the same directory on a different pack, use

```
RUN $SYSTEM/DMUTILITY("DB=MYDBNAME COPY  
MYDBNAME/= AS MYDBNAME/= ON NEWPACK FROM MYDBNAMEDUMP");  
DATAPATH = *DIR/NODE ON PACKNAME
```

- To copy the database from *DIR/NODE to *DIR/NEWNODE, use

```
RUN $SYSTEM/DMUTILITY("DB=MYDBNAME COPY MYDBNAME/= AS  
*DIR/NEWNODE/MYDBNAME/= ON PACK FROM MYDBNAMEDUMP");  
DATAPATH = *DIR/NODE ON PACKNAME
```

7. Update the control file copied from the dump against the new description file. For example,

```
RUN $SYSTEM/DMCONTROL("DB=MYDBNAME UPDATE")
```

Note: Ensure the DMCONTROL program is updating the control file that was copied from the DMUTILITY program dump in step 6.

The database is now ready for use.

For archive retrieval, compile applications for use with this copy of the database. For current databases, database equation can be used. The following example uses the WFL MODIFY command to perform a “permanent” database equation:

```
WFL MODIFY <codefile title> DATABASE MYDBNAME  
(TITLE=(NEWUC)MYDBNAME ON NEWPACK)
```

<copy onto>

This option causes DMUTILITY to copy the rows of a file from the dump tape onto the file specified in the <copy onto> specification. The title of the copied file is the file name specified in the <copy onto> specification. If the ON <family name> clause is specified, the file being copied onto must reside on the specified family name. If the ON <family name> clause is not specified, the file being copied onto must reside on the original pack family of the file being copied. If the file being copied onto is not present, a No File condition results.

If a file is not present on the destination pack, the security type of the copy is set to the security type of the file on the dump tape. If the file was guarded, the security guard title is obtained from the control file on the dump tape.

<copy to>

This option identifies the family index where the rows are to be copied. If RETAIN is specified, the rows are copied to the same family index they occupied when they were dumped. If <copy to> is not specified, the rows are allocated on arbitrary family indexes but on the same pack family as the file to which they correspond.

If <copy to> is specified following <copy as>, and the ON <family name> clause is specified, the FAMILYINDEX pertains to the family name specified in <copy as>.

If the FAMILYINDEX specified in the <copy to> specification does not exist, DMUTILITY waits on a SECTORS REQUIRED condition for that FAMILYINDEX to be present.

<copy source>

The following options are available:

Recovering the Database

- Tape, disk, or multidump tape

This option identifies the tapes and dumps to be used during the copy process. If the WORKERS option is not specified and the database dump being copied was created using the TAPES clause, the WORKERS option is set to the value of TAPES used at the time of the dump. If the TAPES option is greater than 20, the WORKERS value is set to 20. TAPES and WORKERS are not valid for use with multidump tapes.

The tape directories of successive single dump reels are cumulative. The tape directory of the last tape dumped contains information about all rows of the database that were dumped. For example, assume a database was dumped to TAPEX, cycle 1 and cycle 2, and that each cycle has three versions. Also assume that cycle 1, version 3, was the last tape written. Under usual processing, you would specify cycle 1, version 3, in the recovery source. However, if all the rows that need to be recovered are on cycle 2, version 3, then specify cycle 2, version 3, as the recovery source.

Note: *Offline dumps of the secondary database are intended as backup sources for complete or partial database recovery at either the primary or secondary host using the DMUTILITY RECOVER command. The use of an offline dump of the secondary database is not recommended as a source for the DMUTILITY COPY command. For additional information about using an offline dump of the secondary database, refer to the information on managing a Remote Database Backup environment in the Remote Database Backup Operations Guide.*



QDC (<QDC title clause>)

Refer to “Using a Quiesce Database Copy as a Recovery or a Copy Source” in [Section 14, Using a Quiesce Database](#), for information.

Examples

Example 1

This command copies all database files from tape DBDUMP063094 to disk.

```
COPY = FROM DBDUMP063094
```

Example 2

This command copies all rows of the database that were present on family index 3 at the time of the dump. All rows not on family index 3 at the time of the dump are left intact.

```
COPY = (FAMILYINDEX = 3) ONTO = FROM TAPEX
```

Example 3

This command copies all rows of the database to the same pack family and family index on which they resided at the time of the dump.

```
COPY = TO (FAMILYINDEX = RETAIN) FROM TAPEX
```


Example 4

This command copies all rows of DB/D/DATA from the tape TAPEX to SYSPACK. The original pack family need not be SYSPACK.

```
COPY DB/D/DATA AS DB/D/DATA ON SYSPACK FROM TAPEX
```

Example 5

This command copies all rows of DB/D/DATA from the tape TAPEX to family index 2 of pack family SYSPACK. The original pack family need not be SYSPACK.

```
COPY DB/D/DATA AS DB/D/DATA ON  
SYSPACK TO (FAMILYINDEX = 2) FROM TAPEX
```

Example 6

This command copies all rows of the database to the pack family and family index on which they resided at the time of the dump.

```
COPY = TO (FAMILYINDEX = RETAIN) FROM ANOTHERDB063094  
TAPE=DBDUMPS063094
```

Example 7

The following command copies all database files from dump DB1DUMP and tape DB1TAPE to disk.

```
RUN *SYSTEM/DMUTILITY("DB = DB1  
COPY DB1/= FROM DB1DUMP TAPE = DB1TAPE")
```

Example 8

The following command copies all rows of the database DB1 that were present on family index 3 at the time of the dump DB1DUMP. All rows not on family 3 at the time of the dump are left intact.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 COPY DB1/=  
(FAMILYINDEX = 3) FROM DB1DUMP TAPE = DB1TAPE")
```

Example 9

The following command copies all database files from dump DB1DUMP and tape DB1TAPE to disk. Because it is an OFFLINE copy, it prevents other users from accessing the database. DMUTILITY waits for all programs that opened the database to complete processing before it performs the OFFLINE copy.

```
RUN *SYSTEM/DMUTILITY("DB = DB1  
OFFLINE COPY DB1/= FROM DB1DUMP TAPE = DB1TAPE")
```

Example 10

The following command causes DMUTILITY to copy the rows of a file of the dump DB1DUMP and tape DB1TAPE, and create a new file on disk. The title of the newly created file is the file name specified in the <copy as> specification.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 COPY DB1/=
AS TESTDIR/= FROM DB1DUMP
TAPE = DB1TAPE")
```

Example 11

The following command causes DMUTILITY to copy the entire database DB1 from the dump DB1DUMP, which is stored on the tape DB1TAPE with the given serial number specification 394302.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 COPY DB1/=
FROM DB1DUMP TAPE = DB1TAPE(SERIALNO = 394302)")
```

Example 12

The following command causes DMUTILITY to copy the entire database DB1 from the dump DB1DUMP, which is stored on the tape DB1TAPE with the given density specification FMTST9840.

```
RUN *SYSTEM/DMUTILITY("DB = DB1 COPY DB1/=
FROM DB1DUMP TAPE = DB1TAPE(DENSITY = FMTST9840)")
```

Example 13

The following command causes DMUTILITY to copy all database files from dumps T1, T2, and T3 that were created by the listed DUMP statement based on the specified family index 1:

```
RUN *SYSTEM/DMUTILITY ("DB = DB1
DUMP=(FAMILYINDEX=1) TO T1;
      =(FAMILYINDEX=2) TO T2;
      =(FAMILYINDEX=3) TO T3")
RUN *SYSTEM/DMUTILITY("DB = DB1 COPY=FROM T1,T2,T3")
```

Example 14

The following command causes DMUTILITY to copy all database files from dumps T1 and T2 that were created by the listed DUMP statement based on the specified pack name:

```
RUN *SYSTEM/DMUTILITY ("DB = DB1
DUMP=( PACKNAME=DATAPK1) TO T1;
      =( PACKNAME=DATAPK2) TO T2")
RUN *SYSTEM/DMUTILITY("DB = DB1 COPY=FROM T1,T2")
```

Tape Dumps

The tape directories of successive single dump tape reels are cumulative. The tape directory of the last tape dumped contains information about all rows of the database that were dumped. For example, assume a database was dumped to TAPEX, cycle 1 and cycle 2, and that each has three versions. Also assume that cycle 1, version 3, was the last tape written. Under usual processing, you would specify cycle 1, version 3, in the recovery source.

Notes:

- *Because CYCLE is always 1 for multidump tapes, tape processing becomes single threaded when DMUTILITY is loading from a multidump tape.*
- *Whenever you use an existing multidump tape on a system, the fast access directory for that tape must be present on the system. You can copy the directory from another system or create the directory for the tape by using the TAPESET DIRECTORY CREATE command.*
- *Unisys recommends that you do not move multidump tapes between systems to add dumps to them because this can result in inconsistent fast access directory files on the different systems. These inconsistent directory files can cause existing dumps to be overwritten.*

With the exception of incremental and accumulated dumps, if all the rows that need to be recovered are on cycle 1, version 3, then specify cycle 1, version 3 as the single dump tape copy source. When you use incremental and accumulated dumps to copy the database, each tape name must include "CYCLE = last cycle" and "VERSION = last version number" in the tape specification clause.

A dump tape should be named only once in the copy source list, and the specified cycle and version should contain the latest tape directory. DMUTILITY then processes each physical tape in parallel, dependent on the number of workers and the number of tapes specified. The maximum number of workers allowed is 50. If the WORKERS recover option is not specified, the value of the TAPES option at the time the dump is created determines the number of workers.

A DMUTILITY-initiated COPY using the direct data set rows that were in the preallocated region at the time of the dump causes DMUTILITY to simulate the loading of these rows by preallocating them. The effects of the row preallocation cause these rows to appear as though they had actually been written to the dump tape. Refer to "DMUTILITY INITIALIZE Statement" in [Section 5, Initializing and Maintaining](#).

Disk Dumps

The following information explains how to use the COPY statement syntax if you are using disk dumps.

A disk dump should be named only once in the copy source list.

DMUTILITY expects all files named in a copy list to reside on the corresponding copy disk dump source.

A DMUTILITY-initiated COPY using the direct data set rows that were in the preallocated region at the time of the dump causes DMUTILITY to simulate the loading of these rows by preallocating them. The effects of the row preallocation cause these rows to appear as though they had actually been written to the dump.

DMUTILITY TAPECLONE Statement

The TAPECLONE statement, intended only for use in the Remote Database Backup environment, initiates the database clone process on the secondary host.

For more information on the clone process in the Remote Database Backup environment, refer to the *Remote Database Backup Operations Guide*.

STRUCTURECLONE Statement (DMUTILITY)

The STRUCTURECLONE statement, intended for use only in the Remote Database Backup environment, initiates the database structure clone process on the secondary host.

For more information on the structure clone process in the Remote Database Backup environment, refer to the *Remote Database Backup Operations Guide*.

Section 9

Copying Audit Files

Audit files contain a history of the changes made to an audited database and are a necessary component of the database recovery process. If audit files are not available when a database failure occurs, all changes made to a database since the last backup dump was taken might be lost. It is therefore important to manage the audit files for your database. Just as special tools are required to make a backup copy of your database, so a special tool, the COPYAUDIT program, is required to manage audit files and their backups.

The COPYAUDIT program enables you to perform the following audit file management tasks:

- Copy audit files from one medium to another.
- Print or display online directories for audit file tapes.
- Verify the contents of audit files.

Note: *The tasks identified in this section can be initiated through Database Operations Center.*

In This Section

This section describes the COPYAUDIT program and the tasks you can perform with the program. Individual topics include

- Why copy audit files?
- Facilities provided by the COPYAUDIT program
- Initiating the COPYAUDIT program
- Checking the results of a COPYAUDIT run
- Methods for copying audit files
- Using the QUICKCOPY command
- Using the COPY command
- Using the DIRECTORY command to display audit file tape directories
- Using the VERIFY command to verify audit file contents

[Appendix C, COPYAUDIT Error Messages](#), supplements the information provided in this section. At the end of this section, quick-reference information is provided for all the syntax diagrams related to using the COPYAUDIT program.

Conventions

In this section the following conventions are used:

- The term *quickcopy tape* is used to identify a tape to which audit files are copied using the COPYAUDIT *QUICKCOPY* command.
- The terms *quickcopy audit file* and *backup audit file* are used to identify the backup copy of an audit file that has been generated by copying an audit file from disk to tape using the COPYAUDIT *QUICKCOPY* command. The quickcopy, or backup, audit file must be copied back to disk before it can be used for database recovery or used by the PRINTAUDIT program.

Why Copy Audit Files?

Introduction

You need to copy audit files for the following reasons:

- To maintain a backup copy of audit information in case it is needed in a database operation
- To help ensure that there is enough space on disk for the generation of new audit files
You can make space available on your system by backing up the older audit files to tape and then removing the audit files from disk.
- To restore a backup copy of an audit file to disk for database recovery or for use with the PRINTAUDIT program

The COPYAUDIT program is a tool that enables you to safely copy audit files from one medium to another by verifying each audit file as the file is copied.

Note: *Do not copy audit files by using library maintenance. No verification checks are performed, and any tape files produced are not directly usable by the database recovery process.*

To Archive Audit Files for Database Recovery Purposes

The database recovery process uses the information contained in the database audit files to roll back, to rebuild, or to reconstruct portions of a database. Therefore, it is important to maintain backup copies of your audit files in case a database recovery is required. The default SAVEFACTOR for a COPYAUDIT tape is 999 days.

You can recover a database by using audit files that are stored on tape or disk. However, if the audit files were copied or appended to tape using the COPYAUDIT *QUICKCOPY* command, you must copy the audit files back to disk before performing a database recovery. You can use audit files copied to tape with the COPYAUDIT *COPY* command for database recovery without first being copied back to disk.

To Keep Sufficient Space for Audit Files on Your System

Busy databases can generate enough data to fill an audit file every 30 minutes or fewer. If you make primary and secondary copies of all audit files and maintain all audit files on disk, you can rapidly run out of disk space, causing your database to stop and wait until more space is made available for the audits. To avoid disk space problems, use the COPYAUDIT program to transfer your audit files to an archive medium and then automatically remove the audit files from disk. The COPYAUDIT program also provides facilities to copy audit files from the archive medium back to your system.

Facilities Provided by the COPYAUDIT Program

[Table 9-1](#) identifies the tasks you can perform using the COPYAUDIT program and the headings under which the task is described.

Table 9-1. Tasks That Can Be Accomplished by Using the COPYAUDIT Program

To perform this task . . .	Refer to . . .
Copy one or more audit files to a singlereel or to a multiplereel tape.	<ul style="list-style-type: none"> • Methods for Copying Audit Files • Using the QUICKCOPY Command
Copy exactly one audit file to a singlereel tape.	<ul style="list-style-type: none"> • Methods for Copying Audit Files • Using the COPY Command
Copy audit files to disk.	<ul style="list-style-type: none"> • Methods for Copying Audit Files • Using the QUICKCOPY Command • Using the COPY Command
Copy audit files to tapes that already contain audit files.	<ul style="list-style-type: none"> • Using the QUICKCOPY Command • Append Option
Print or display an audit tape directory.	Using the DIRECTORY Command to Display Audit File Tape Directories
Verify the contents of audit files.	Using the VERIFY Command to Verify Audit File Contents

Initiating the COPYAUDIT Program

Initiating COPYAUDIT Automatically

To have the Accessroutines initiate the COPYAUDIT program automatically each time an audit file switch occurs, include the VERIFY clause, the QUICKCOPY TO clause, or the COPY TO clause in the audit trail declaration in the database DASDL source file. Including any of these clauses causes the Accessroutines to automatically initiate a WFL job that in turn initiates the COPYAUDIT program.

When an audit file switch occurs, the Accessroutines performs the following tasks:

1. If the DONTFORCE Visible DBS option is reset (the default setting), the Accessroutines ensures two controlpoints occur as soon as possible after the new audit file is opened.

If the DONTFORCE Visible DBS option is set, the Accessroutines waits until two controlpoints occur naturally.

The controlpoints must occur prior to the COPYAUDIT program being run to ensure that any subsequent ABORT or halt/load recoveries do not use the audit file being copied by the COPYAUDIT program.

2. Using a WFL job, the Accessroutines initiates the COPYAUDIT program.

By default, the name of the WFL job started by the Accessroutines is DATABASE/WFL/COPYAUDIT. This WFL job is provided on the release tape with your data management software.

For information on the DASDL syntax required for designating COPYAUDIT options, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

Initiating COPYAUDIT Manually

You can initiate the COPYAUDIT program manually by issuing the appropriate statement from a message control system (MCS) such as CANDE or through a WFL job.

To initiate the COPYAUDIT program manually on a database, use one of the following statements:

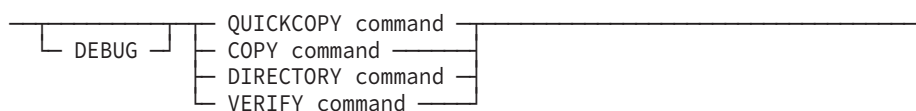
- `RUN SYSTEM/COPYAUDIT("<COPYAUDIT statement>");`
`FILE CF (TITLE=control file title);`

When specified, the COPYAUDIT program obtains the security guard information from the control file and maintains it on the copied audit file when the audit is copied to the disk medium. The file equation of the CF is optional. For additional information, refer to "Effect of the DASDL SECURITYGUARD Attribute" later in this section.

- `START DATABASE/WFL/COPYAUDIT("<COPYAUDIT statement>");`

COPYAUDIT Statement Syntax

The following diagram illustrates the basic syntax of the COPYAUDIT statement.



The DEBUG option causes a program dump to be taken if an error occurs during the COPYAUDIT run. By default the DEBUG option is not set.

The syntax for the individual COPYAUDIT commands is explained under the following headings:

- Using the QUICKCOPY Command
- Using the COPY Command
- Using the DIRECTORY Command to Display Audit File Tape Directories
- Using the VERIFY Command to Verify Audit File Contents

Sample COPYAUDIT WFL Job

The following example illustrates some of the modifications you can make to the standard COPYAUDIT WFL job:

```

BEGIN JOB COPYAUDITS;

% Identifying the name of the printer backup file
for any output messages.
BDNAME=BD/DOC/86000759;

TASK COPYAUDTASK;
RUN *SYSTEM/COPYAUDIT("VERIFY TESTPSVDB/AUDIT1 ON TAPE")
[COPYAUDTASK]; VALUE =0;
IF COPYAUDTASK IS COMPLETEDOK THEN
IF COPYAUDTASK(TASKVALUE) = 1 THEN
    DISPLAY("COPYAUDIT COMPLETED CORRECTLY,
    REMOVE AND STORE TAPE");

IF COPYAUDTASK IS COMPLETEDOK THEN
IF COPYAUDTASK(TASKVALUE) = 2 THEN
    DISPLAY("COPYAUDIT WARNINGS OCCURRED,
    CHECK BEFORE PROCEEDING");

IF COPYAUDTASK ISNT COMPLETEDOK OR
COPYAUDTASK(TASKVALUE) = 0 THEN
    DISPLAY("COPYAUDIT FAILURE OCCURRED, RESOLVE PROBLEM");

END JOB.
    
```

Checking the Results of a COPYAUDIT Run

When the COPYAUDIT program completes, a task value is returned. The task value identifies whether any problems occurred during the run. The job or process that initiates the COPYAUDIT program can interrogate the task value to identify the nature of the problem.

Task Value	Meaning
0	The COPYAUDIT run ended abnormally. Investigate and correct the cause of the failure, and then rerun the COPYAUDIT program.
1	The COPYAUDIT run was successful.

Copying Audit Files

Task Value	Meaning
2	The COPYAUDIT run completed, but a warning was issued. Check the warning and, if appropriate, correct the problem and rerun the COPYAUDIT program.

If the COPYAUDIT run is successful and you included the REMOVE option in the syntax, the original audit file is automatically deleted. If you are operating in a Remote Database Backup environment, you can use the RDB Utility to set an option that delays the removal of the audit file until the audit information has been applied to the secondary database. For more information on delaying the removal of audit files in a Remote Database Backup environment, refer to the *Remote Database Backup Operations Guide*.

If an error occurs during the COPYAUDIT run, you can retry the COPYAUDIT run or terminate the job. The parameters used by the Accessroutines to initiate the COPYAUDIT program are displayed to enable you to rerun the job manually once the cause of the error is corrected.

A falsestopper pattern is a stopper pattern that has been erroneously placed in the audit file. A stopper pattern is present only at the end of an audit file. If the COPYAUDIT program encounters a falsestopper pattern in an audit block, possibly because of a bad disk area, only the good audit blocks before the falsestopper pattern are copied. All audit blocks following the falsestopper pattern are ignored, since DMRECOVERY stops the recovery process as soon as it encounters a falsestopper pattern.

Methods for Copying Audit Files

Introduction

Two basic methods are provided for copying audit files: QUICKCOPY and COPY. The two methods are alternatives to each other.

QUICKCOPY Command

Using the QUICKCOPY command you can

- Copy one or more audit files from disk to tape with a fixedlength block size and optionally verify each file as it is copied.
- Copy audit files to singlereel or multiplereel tapes and optionally verify each file as it is copied.

For this task, the number of tape reels required is dependent on the amount of audit information being copied and is not a function of the number of audit files being copied.

- Copy one or more audit files from tape to disk and optionally verify each file as it is copied.
- Append one or more audit files to an existing quickcopy audit file tape.

- Use tape devices that cannot write data with varying block lengths.
- Use tape devices that do not support a readreverse capability.
- Ensure that data compression either does or does not occur.
- Specify the use of the TAPESET command when using tape drives with the Locate Fast Access capability.

The QUICKCOPY command supports the copying of audit files to

- Multiplereel tapes
- Compressed tapes
- Drives that do not support the readreverse capability

The audit files are copied to tape in fixed-length blocks, and you must copy back the audit files to disk before you can use them for database recovery purposes.

Effect of the DASDL SECURITYGUARD ATTRIBUTE

If the SECURITYGUARD attribute is set for the audit file, it is maintained on the copied audit in the following situations:

- When copying the primary audit as the primary audit, or copying the secondary audit as the secondary audit from one disk medium to another disk medium
- When the COPYAUDIT program is initiated with the file equation of the control file in order to perform one of the following tasks:
 - copy audit from tape/tapeset to disk medium
 - copy primary audit as secondary audit from disk medium to disk medium
 - copy secondary audit as primary audit from disk medium to disk medium

Copy Speed

Using the QUICKCOPY command can be faster than using the COPY command, especially when audit record lengths are small. Audit blocks tend to be smaller when the DASDL options INDEPENDENTTRANS and REAPPLYCOMPLETED are set.

The QUICKCOPY command also supports the use of the TAPESET command to group a series of quickcopy tapes as a single logical tape. Then a disk file can contain information regarding the contents of the tapes in the tape set. Using content information and tape drives with the Locate Fast Access capability can drastically improve the speed at which files are copied to and from tape.

Maximizing Tape Usage

To maximize tape usage, you can

- Use the COMPRESSED option to ensure that audit files are written to compressed tapes.
- Copy more than one audit file to a tape.

You can copy more than one audit file to a tape using either of the following methods:

- Run the COPYAUDIT program only when several audit files need to be copied to tape, and then specify the range of files to be copied in the QUICKCOPY command syntax.
- Use the QUICKCOPY command APPEND option to add one or more audit files to a tape that already contains one or more quickcopy audit files.

Data Compression

All forms of the QUICKCOPY command work with compressed tapes. By default, the COPYAUDIT program uses whatever tape you load. If you load a compressed tape, then data compression occurs. If you load a noncompressed tape, data compression does not occur.

To ensure that data compression occurs on devices that support compression, include the COMPRESSED option when designating the tape medium. To ensure that compressed tapes are not used, include the NONCOMPRESSED option when designating the tape medium.

Note: *Compression options are not supported with the COPY command.*

Audit File Naming Convention

The COPYAUDIT program applies the following naming convention, referred to as the quickcopy naming convention, to audit files on tapes and to the tapes themselves when the QUICKCOPY command is used to copy audit files to tape:

- Primary audit files are copied as <database name>/QCAUDIT<integer>.
- Secondary audit files are copied as <database name>/QC2AUDIT<integer>.
- The tape name is the same as the quickcopy audit file name.
- For multiple-file tapes, the tape name is the same as the name of the first quickcopy audit file on the tape.
- Files copied back to disk have their original name restored.

TAPESET Naming Conventions

The COPYAUDIT program applies the following naming conventions to tapes and audit files on tapes when the QUICKCOPY command with the TAPESET option is used to copy audit files to tape:

- Primary audit files are copied as <database name>/QCAUDIT<integer>.
- Secondary audit files are copied as <database name>/2QCAUDIT<integer>.
- The tape name for a tape set containing primary audit files is <database name>/TAPESET<integer>. The <integer> value is the audit file number of the first audit file in the tape set.
- The tape name for a tape set containing secondary audit files is <database name>/2TAPESET<integer>. The <integer> value is the audit file number of the first audit file in the tape set.

- For multiple-reel tape sets, all reels have the same name as the first reel in the tape set.
- Files copied back to disk have their original name restored.
- In the Remote Database Backup environment, if the TAPASET option is specified on the secondary host, the TAPASET number on that host corresponds to the current AFN on the secondary host but not to the TAPASET number on the primary host.

Database Recovery

If you copy an audit file to tape using the QUICKCOPY command, you must copy back the audit file to disk before the file can be used for database recovery purposes. You can copy back the audit file to disk using either the QUICKCOPY or the COPY command.

Since the audit files must be copied back to disk before they are used with the database recovery process, the QUICKCOPY command supports the following capabilities that are incompatible with the database recovery process:

- Data compression
- Reel switching; that is, an audit file can be larger than one physical tape reel
- Copying more than one audit file to a tape
- Appending audit files to existing audit file tapes

Effect of the DASDL LOCKEDFILE Attribute

If the LOCKEDFILE attribute is set for the database, then the attribute is also set for any audit files. On disk, the LOCKEDFILE attribute can be set on a file-by-file basis. On tapes, the LOCKEDFILE attribute can be set at the tape level only; that is, either all files on the tape have the LOCKEDFILE attribute set or none of the files have the LOCKEDFILE attribute set.

If you try to append an audit file to a tape and the LOCKEDFILE attribute setting for the tape and for the file are different, a warning is issued and you are given the option of copying the audit file to a new tape rather than appending the audit file to an existing tape.

COPY Command

The COPY command supports copying audit files from one medium to another. When you use the COPY command, you must ensure that the audit file fits on a single reel of tape. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for sample audit file size calculations.

Unlike audit files copied to tape using the QUICKCOPY command, audit files copied to tape using the COPY command can be used directly by the database recovery process. However, unlike the QUICKCOPY command, the COPY command does **not** support the following features:

- Data compression
- Reel switching (That is, an audit file can be larger than one physical tape reel.)

Copying Audit Files

- Copying more than one audit file to a tape
- Appending audit files to existing audit file tapes
- Sectioned audit files (You must use the QUICKCOPY command.)
- TAPESET option (You must use the QUICKCOPY command.)

Data Compression

By default, the COPYAUDIT program requests a noncompressed tape if you use the COPY command. Data compression is incompatible with the COPY command for both of the following reasons:

- Data expansion might occur with certain combinations and sequences of data, and the COPYAUDIT run might fail because of insufficient space on the tape.
- Compressed tapes cannot be used directly by the database recovery process and all tapes created using the COPY command must be capable of being used directly by the database recovery process.

Audit File Naming Convention

Audit files copied to tape using the COPY command have the following naming convention:

- Primary audit files are copied as <database name>/AUDIT<integer>.
- Secondary audit files are copied as <database name>/2AUDIT<integer>.

Database Recovery

Audit files that are copied to tape with the COPY command can be used directly for database recovery without first copying back the files to disk if the tape drive used during recovery has read-reverse capability. If the tape drive does not support the read-reverse capability, it might be necessary to copy the file back to disk before the recovery process can complete. To support the database recovery process, audit files copied to tape using the COPY command must fit on a singlereel tape.

Effect of the DASDL LOCKEDFILE Attribute

If the LOCKEDFILE attribute is set for the database, then the attribute is also set for any audit files. The LOCKEDFILE attribute setting is passed to each tape to which you copy an audit file.

Effect of the DASDL SECURITYGUARD Attribute

If the SECURITYGUARD attribute is set for the audit file, it is maintained on the copied audit in the following situations:

- When copying primary audit as primary audit, or when copying secondary audit as secondary audit from disk medium to disk medium
- When the COPYAUDIT program is initiated with the file equation of the control file in order to perform one of the following tasks:

- copy audit from tape to disk medium
- copy primary audit as secondary audit from disk medium to disk medium
- copy secondary audit as primary audit from disk medium to disk medium

Using the QUICKCOPY Command

Introduction

Much of the QUICKCOPY command and COPY command syntax is similar. The QUICKCOPY command syntax differs from the COPY command syntax as follows:

- QUICKCOPY keyword replaces the COPY keyword.
- APPEND keyword is available to support the appending of audit files to existing audit tapes.
- MAXFILESPERTAPE phrase is available to control the number of audit files that can be stored on any one tape.
- Audit file range phrase is available to enable more than one audit file to be copied or appended in a single COPYAUDIT run.
- FROM and TO clauses are limited to allow only disktotape or tapetodisk copies; that is, the QUICKCOPY command cannot be used to copy audit files from tapetotape or from disktodisk.

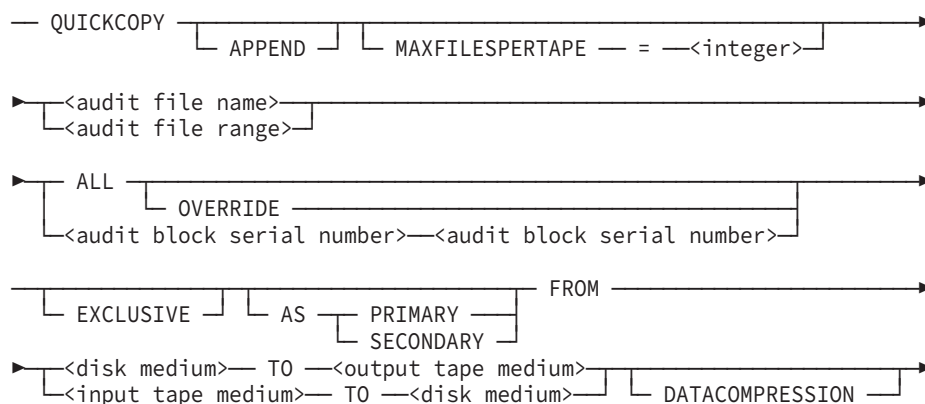
Use the QUICKCOPY operation to copy sectioned audit files between disks and tapes. The QUICKCOPY operation combines the sections of an audit file into a single quickcopy file.

Note: You must use the COPY operation to copy sectioned audit files from disk to disk.

Syntax

The following diagrams illustrate the syntax for the QUICKCOPY command. Explanations of these syntax elements follow the diagrams.

QUICKCOPY Command



Copying Audit Files

CHECK REMOVE COPIES = 1 2 FORWARD COMPARE

<audit file name>

* (usercode) <database name>/ QC 2

AUDIT<integer>

<audit file range>

<audit file name> THRU <audit file name>

<disk medium>

DISK
PACK = <family name>
DBPATH = *DIR/ /7\<node> ON <family name>

<output tape medium>

TAPE
TAPESET <integer>

(/1\<density> ,)
/1\ COMPRESSED
/1\ NONCOMPRESSED
/1\ AUDITENCRYPT = TDES
AES256
AESGCM

SCRATCHPOOL = <scratch pool name>

<input tape medium>

TAPE
TAPESET <integer> (<density>)

<density>

DENSITY = <density mnemonic>

APPEND Option

The APPEND option adds one or more audit files to the end of an existing singlereel or multiplereel tape. The name of the tape is always the name of the first audit file copied to the tape.

The append process requires that

- The audit files to be appended are for the same database as the audit files already on the tape.
For example, you cannot append the audit file DB/AUDIT1 to a tape that contains a copy of the audit file MYDB/AUDIT2.
- When you append more than one audit file at a time, all the audit files being appended must be for the same database.
For example, you cannot append the audit file DB/AUDIT1 and the audit file MYDB/AUDIT2 to the same tape.
- The audit files must be appended in the order in which they were created.
For example, if DB/AUDIT4 is the last audit file copied to a tape, you cannot append audit file DB/AUDIT3 or DB/AUDIT6 to the tape. However, you can append the audit file DB/AUDIT5.
- A logical gap cannot exist in the audit files on a tape.
For example, if a tape contains audit file DB/QCAUDIT5, you can append audit files DB/AUDIT6, DB/AUDIT7, and DB/AUDIT8, but you cannot append audit files DB/AUDIT7 and DB/AUDIT8 without first appending DB/AUDIT6.
- The first audit file for a database, or the first audit file after an audit file number rollover occurs, must always be copied to a new tape. In both cases, the audit file has the name *<database name>/AUDIT1* or *<database name>/2AUDIT1*.
For example, you cannot append audit file DB/AUDIT1 to a tape containing audit file DB/QCAUDIT9999.
- You can only append audit files to new tapes or to tapes that were created using the QUICKCOPY command.
- When you use the APPEND option, the previous audit file must be present to ensure the continuity of the audit files being appended. If the previous audit file is not present, COPYAUDIT waits on a NO FILE condition. Choose NF and use one of the following COPYAUDIT commands:
 - If you want to switch to a different tape, use AX NEW.
 - If you want to try to open the previous audit file again, use AX RETRY.
 - If you are running the COPYAUDIT program manually and want to terminate the COPYAUDIT program, use AX QUIT.If the Accessroutines zipped the COPYAUDIT job, the COPYAUDIT program automatically switches to a new tape.
- The APPEND option is assumed when a TAPESET specification is used and the audit file number is greater than the TAPESET number.
In the following example, DB/AUDIT4 appends to TAPESET 3 because the audit file number is greater than the TAPESET number:

```
QUICKCOPY DB/AUDIT4 ALL FROM PACK=AUDPK TO TAPESET 3
```

Caution

If audit files are released faster than they can be written to the tape, operator intervention might be needed to use a IL (Ignore Label) system command to direct the copy task to search a tape again after a previous audit file has been written to it.

MAXFILESPERTAPE Clause

Use the MAXFILESPERTAPE clause to designate the maximum number of audit files that you can copy to a single-reel or to a multiple-reel tape. The allowed values include any integer in the range 1 through 9999. By default, the MAXFILESPERTAPE clause is assigned the value 9999, and audit files are always appended to the same tape.

The QUICKCOPY and QUICKCOPY APPEND commands provide the ability to store more than one audit file on a tape reel, and to split an audit file over two or more tape reels. Before the audit files can be used for database recovery purposes, the audit files must be copied back to disk. To minimize the time required to locate and copy back the required audit files to disk, it can be advantageous to limit the number of audit files on any singlereel or multiplereel tape. The MAXFILESPERTAPE clause provides this control.

The value you assign in the MAXFILESPERTAPE clause is retained and applied to future append requests for the tape. To change the value, run the COPYAUDIT program and assign a new value.

If you include a MAXFILESPERTAPE clause in the QUICKCOPY or QUICKCOPY APPEND statement, the COPYAUDIT program checks the number of audit files currently on the tape prior to copying or appending an audit file to that tape. If the number of audit files on the tape is greater than the value specified in the MAXFILESPERTAPE clause, the audit file is copied to a new tape.

Example

Assume the following:

- A tape contains quickcopy versions of the audit files DB/AUDIT2 through DB/AUDIT4.
- You want to append the audit files DB/AUDIT5 through DB/AUDIT12.
- You assign a value of 5 in the MAXFILESPERTAPE clause.

Under these circumstances, the following actions occur:

- The files DB/AUDIT5 and DB/AUDIT6 are appended to the tape that contains the files DB/QCAUDIT2 through DB/QCAUDIT4.
- The files DB/AUDIT7 through DB/AUDIT11 are copied to a new tape.
- The file DB/AUDIT12 is copied to a third tape.

At the end of this sample COPYAUDIT run, the following tapes exist. As shown in the following table, the name assigned to each tape is the name of the first audit file copied to that tape.

Tape named . . .	Contains the files . . .
DB/QCAUDIT2	DB/QCAUDIT2 through DB/QCAUDIT6
DB/QCAUDIT7	DB/QCAUDIT7 through DB/QCAUDIT11
DB/QCAUDIT12	DB/QCAUDIT12

Audit File Name and Audit File Range Clauses

Use the audit file name and audit file range clauses to identify the audit file or files you want to copy or append.

Never use quotation marks in these clauses, even if the database name includes a hyphen (-).

When copying audit files to or from tape, the usercode specification applies only to the source or destination audit file on disk. For example, when copying an audit file from tape to disk, the usercode specifies the file usercode that is written to disk. If the disk usercode is the same as that of the usercode used to run the COPYAUDIT program, the usercode specification is unnecessary.

If an audit file was copied to tape using the QUICKCOPY command, you must use the quickcopy naming convention when copying back the audit file to disk.

Use the number 2 to identify a secondary audit file. Secondary audit files are generated when the database description includes the DASDL audit trail DUPLICATED option.

Use the integer in the file name to identify the audit file number. Audit file numbers range from 1 to 9999.

TAPESET Numbers



All information regarding TAPESET numbers pertains to the XE features.

Use the integer in the TAPESET specification to identify the TAPESET number. TAPESET numbers are based on audit file numbers and range from 1 to 9999.

When copying a range of files from a tape set to disk, it is possible the ending file number is beyond the range of files contained in the tape set. In this case, the COPYAUDIT program emits the following warning:

Copying Audit Files

Some of the files specified do not exist in this TAPESET

Only those files that exist in the tape set are copied to disk. The COPYAUDIT program terminates without copying any files not in the tape set.

ALL Option

Use the ALL option to designate that you want to copy the complete audit file. Optionally, you can explicitly designate the first and last audit block serial numbers (ABSNS) of the audit file you want to copy.

OVERRIDE Option

Use the OVERRIDE option to copy the current audit file.

You cannot use the REMOVE option with the OVERRIDE option. This restriction is present to ensure that if you copy an incomplete audit file, the file is not deleted automatically at the end of the COPYAUDIT run.

By default, the COPYAUDIT program validates the last audit block serial number (ABSN) of the audit file you are copying by finding the previous ABSN field of the first block of the next audit file, and by ensuring that the two ABSNs are the same. Using the OVERRIDE option avoids this consistency check. This option is especially useful when copying one complete audit file because the other files are not available for verification of ABSNs.

For example, if you copy the audit file DB/AUDIT1, the last ABSN of the DB/AUDIT1 file is identified by finding the first ABSN in the audit file DB/AUDIT2.

If you do not use the OVERRIDE option and the next audit file is not available, a No File condition occurs, and one of the following No File messages displays:

```
NO FILE <audit file name> ON <family name>
NO FILE <audit file name> ON TAPE
```

After the No File message appears, the following message is displayed:

```
ACCEPT:
  ENTER <task number> AX RETRY OR FAMILY = <familyname>
  ENTER TAPE FOR <family name> IF <audit file name> IS ON TAPE
```

Enter AX RETRY to have the COPYAUDIT program check again in the same location for the next audit file. If you enter a new location, the COPYAUDIT program looks in the new location for the next audit file.

If you use the FA (File Attributes) system command to identify the correct name and location of the audit file, ensure that you identify all the necessary attributes of the audit file. It is especially important to identify the correct CYCLE and VERSION file attributes. If you supply incomplete or incorrect information in the FA system command, an UNMATCHED GENEALOGY error can occur.

The OVERRIDE option is not required when copying audit files from TAPE or TAPASET to DISK using the QUICKCOPY command.

EXCLUSIVE Option

Use the EXCLUSIVE option to open the input audit file in exclusive mode.

The EXCLUSIVE option is ignored if the input audit file is on tape.

Audit Block Serial Number Clauses

Use the audit block serial number clauses to identify the first and last ABSNs in the audit file you want to copy. If you use this method of designating the start and end point of the audit file, you cannot use the OVERRIDE option. Using the ALL option is the preferred method for identifying the start and end point of the audit file.

AS PRIMARY and AS SECONDARY Options

Use the AS PRIMARY option to copy a secondary audit file as a primary audit file. And use the AS SECONDARY option to copy a primary audit file as a secondary audit file.

The audit files on a multiple-file tape must be either all primary audit files or all secondary audit files. You cannot mix primary and secondary audit files on a single tape. If you attempt to mix audit file types when appending audit files, the type of the audit files being copied is automatically changed to match the type of audit files already on the tape.

Disk Medium, Input Tape Medium, and Output Tape Medium Clauses

Use the disk medium, and the input and output tape medium clauses to identify the source and destination locations for the copy operation. Using the QUICKCOPY command you can copy from disk to tape or from tape to disk. You cannot copy audit files from disk to disk or from tape to tape.

TAPASET Specification

Use the TAPASET specification to improve QUICKCOPY performance when using tape drives with the Locate Fast Access capability. If the TAPASET number is not specified, the TAPASET number defaults to the same value as the audit file number.

To use the TAPASET specification with the APPEND option, a directory file for the TAPASET specification must exist. The APPEND option is assumed when a TAPASET specification is used and the audit file number is greater than the TAPASET number.

The COPYAUDIT program automatically creates a directory file when the first reel of a set of tapes is created by the TAPASET option. Directory files are created on the pack identified by the DL LIBMAINTDIR command. If no location is specified, directory files are located on the system halt/load unit.

Copying Audit Files

Directory file titles use the following format:

```
<database name>/<audit pack name>/<tape set number>  
<primary or secondary audit file>  
<first or second copy>/<yyyymmdd><hhmmss>
```

For example, if the audit files for the PARTSDB database are located on the AUDPACK, and the first audit file that appears in a set of tapes is PARTSDB/AUDIT1523 (created at 9:26:08 a.m. on March 21, 2000), the directory file for the second copy of the primary audit tape set is the following:

```
PARTSDB/AUDPACK/152312/20000321092608
```

A TAPESET directory file is automatically removed (following a response to a waiting entry) when a volume, or physical reel, from the set of tapes is purged using the ODT PG or SN command. Refer to the *System Commands Operations Reference Manual* for more information.

At times, you might want to copy the directory file manually prior to purging the tape so that the directory file can be reinstated following the purge.

If a COPYAUDIT command specifies a TAPESET option and the associated directory file cannot be found, the COPYAUDIT program emits the following warning:

```
TAPESET directory file <file name> not found
```

If the file is found but it is corrupted, the COPYAUDIT program emits the following warning:

```
TAPESET  
directory file <file name> contains errors and is not usable
```

In both instances, the COPYAUDIT program proceeds as if the TAPESET specification had not been given. That is, the COPYAUDIT program uses the MCP file search mechanism to locate the necessary files.

If the directory file for the TAPESET option is lost or corrupted, use the DIRECTORY command with the CREATE option to re-create the directing file.

Use the integer in the TAPESET specification to identify the TAPESET number. TAPESET numbers are based on audit file numbers and range from 1 to 9999.

When copying or verifying a range of files from a set of tapes created by the TAPESET option, it is possible the ending file number is beyond the range of files contained in this set of tapes. In this instance, the COPYAUDIT program emits the following warning:

```
Some of the files specified do not exist in this TAPESET
```

Only those files that exist in the set of tape are copied to disk. The COPYAUDIT program terminates without copying any files not in the tape set.

Examples

The following examples illustrate the use of the TAPESET option and the results of using different audit file number and TAPESET specification values.

Example 1

The following command copies the audit file DB/AUDIT817 to TAPESET817 even though the TAPESET number is not specified:

```
QUICKCOPY DB/AUDIT817 ALL FROM PACK=AUDPK TO TAPESET
```

Example 2

The following commands copy an audit file to a new set of tapes. DB/AUDIT3 copies to TAPESET3 because the audit file number is equal to the TAPESET number. DB/AUDIT4 appends to TAPESET3 because the audit file number is greater than the TAPESET number and the APPEND option is implied. In the last example, a syntax error occurs because the audit file number is less than the TAPESET number.

```
QUICKCOPY DB/AUDIT3 ALL FROM PACK=AUDPK TO TAPESET 3
QUICKCOPY DB/AUDIT4 ALL FROM PACK=AUDPK TO TAPESET 3
QUICKCOPY DB/AUDIT4 ALL FROM PACK=AUDPK TO TAPESET 5
```

Example 3

The following command appends the audit file DB/AUDIT2 to TAPESET1 if the audit file number is greater than the TAPESET number. If the audit file number is equal to or less than the TAPESET number, a syntax error occurs.

```
QUICKCOPY APPEND DB/AUDIT2 ALL FROM PACK=AUDPK TO TAPESET 1
```

Example 4

The following command makes two audit file copies of the primary audit file for a permanent directory database. The complete audit file name is *DIR/MEGAMART/EX1/TEST-DB/AUDIT6789. The copies are checked and the original file is deleted.

```
COPY TEST-DB/AUDIT6789 ALL FROM DBPATH=*DIR/MEGAMART/EX1
ON AUDITPACK TO TAPE CHECK REMOVE COPIES=2
```

Density Specification

Use the density specification to designate the type of tape drive to be used. If you do not supply a density value, the I/O subsystem uses the system default rules to determine the type of tape device to use.

REQUIRES MT Condition

The COPYAUDIT program waits on a REQUIRES MT condition under any of the following circumstances:

- If you designate a density value, and a tape with that density is not available
- If a scratch tape is unavailable
- If the system option 27 (serial number) is set

In this instance, the system expects the tape to have a specific serial number. You cannot supply a serial number using the COPYAUDIT syntax. Perform either of the following tasks as a solution:

- Supply the serial number by using the FA (File Attributes) system command when the REQUIRES MT condition occurs.
- Use the SCRATCHPOOL option to designate the output tape.

COMPRESSED and NONCOMPRESSED Options

When using the QUICKCOPY command you can explicitly designate whether data compression is to occur. By default, data compression occurs if you mount a compressed tape, and no data compression occurs if you mount a noncompressed tape.

To explicitly require data compression on devices that support data compression, use the COMPRESSED option. To explicitly require that data compression does not occur, use the NONCOMPRESSED option.

You can use the COMPRESSED and NONCOMPRESSED option only when designating the output tape medium. If you include either option when designating the input tape medium, the option is ignored.

Note: *If you specify both the COMPRESSED and DATACOMPRESSION options at the same time, a double compression occurs (which might not be desirable).*

AUDITENCRYPT Option

When using the QUICKCOPY command you can explicitly designate whether encryption is to occur. By default, encryption does not occur.

You can indicate either the TDES, AES256, or AESGCM encryption algorithm when requesting encryption in a QUICKCOPY operation or let the TDES algorithm be used by default.

Notes:

- *Before an audit file that was encrypted when copied to tape can be processed by MCP or Enterprise Database Server software—such as DMRECOVERY, PRINTAUDIT, and others—the tape must be copied back to disk by using the COPYAUDIT software.*
- *Files are automatically decrypted by the COPYAUDIT software.*

Refer to [Section 15, Using Database Tape Encryption](#), for specific examples and additional information about the AUDITENCRYPT option.

SCRATCHPOOL Option

Use the SCRATCHPOOL option to restrict the output to a tape from the specified scratch pool. The scratch pool name is a 1 character to 17 character identifier.

You can use the SCRATCHPOOL option only when designating the output tape medium. If you include the option when designating the input tape medium, the request is ignored.

DATA COMPRESSION Option

Use the DATA COMPRESSION option to enable the QUICKCOPY operation to compress the data during the copy to tape process.

CHECK Option

Use the CHECK option to have the COPYAUDIT program check the internal integrity of the audit file copy; the audit file contents are not checked against the contents of the original audit file. If you use the CHECK option, the COPYAUDIT program performs the following tasks:

- Reads each new audit file copy in the forward direction and checks for integrity errors.

Note: *If you use the COPY command, by default, the integrity check occurs using a readreverse technique. To check the copy of the audit file using a forward comparison technique, you must explicitly include the FORWARD COMPARE option in the COPYAUDIT statement.*

- Verifies the ABSNs and time stamps for continuity.
- Verifies the checksum, if the DASDL audit trail CHECKSUM option was designated in the database description file when the audit file was created.

If an error occurs, an error message displays. Refer to [Appendix D, Using Mirrored Disks for Disaster Recovery](#), for an explanation of the error message.

Use the CHECK option to verify that the copied audit file is an accurate copy of the source audit file.

When using the CHECK option with the COPY command, the audit file image on the TO <medium> is compared to the audit file on the FROM <medium>. This verification includes, but is not limited to, block-to-block ABSN and timestamp checks and checksum validation (where applicable).

When using the QUICKCOPY command to copy an audit file to tape, the QUICKCOPY algorithm places additional data onto the tape to ensure the integrity of the tape files. As long as tape integrity can be verified, it is presumed that the data on the tape is identical to that of the original audit file. The error checking invoked by the CHECK option or the VERIFY command for quickcopy tapes uses only the internal integrity checks and does not compare against the data in the original audit file.

REMOVE Option

Use the REMOVE option to delete the original audit file as soon as the quickcopy operation completes successfully.

You cannot use both the OVERRIDE option and the REMOVE option in the same COPYAUDIT statement. This restriction prevents the accidental deletion of an audit file without a complete backup copy being available.

COPIES Option

Use the COPIES option to identify the number of copies of the audit file you want to make. At most you can request that two copies of the audit file are to be made simultaneously. By default, only one copy is made.

All copies of the audit file are given the same name.

Requesting two copies of an audit file is valid only when the output medium is tape. If you request two copies of the audit file when the output medium is disk, COPYAUDIT error 76 occurs.

FORWARD COMPARE Option

All quickcopy operations use a forwardcomparison technique. Including the FORWARD COMPARE option in the syntax of your QUICKCOPY statement is optional and for documentation purposes only. Use of the FORWARD COMPARE option is valid only if you also include the CHECK option in your statement.

Examples

The following examples illustrate correct QUICKCOPY command syntax.

Example 1

The following command copies the primary audit files TESTDB/AUDIT1 through TESTDB/AUDIT5 on pack AUDPK to a tape. After each audit file is copied, the tape is repositioned and the audit file is read in the forward direction to check for correctness.

```
QUICKCOPY TESTDB/AUDIT1 THRU TESTDB/AUDIT5 ALL FROM PACK=AUDPK  
TO TAPE CHECK FORWARD COMPARE
```

Example 2

The following command appends the secondary audit files TESTDB/2AUDIT6 through TESTDB/2AUDIT8 on pack SECAUDPK to the tape created in Example 1:

```
QUICKCOPY APPEND TESTDB/2AUDIT6 - TESTDB/2AUDIT8 ALL AS PRIMARY  
FROM PACK=SECAUDPK TO TAPE CHECK
```

In this example, the tape is assumed to already contain some primary audit files, and because a tape can contain only primary or only secondary audit files, the audit files are copied as primary audit files.

After each audit file is copied, the tape is repositioned and the audit file is read in the forward direction to check for correctness. The forward comparison occurs even though the FORWARD COMPARE option is omitted from the statement, because the quickcopy operation does not support the readreverse comparison technique.

Using the COPY Command

Syntax

The following diagrams illustrate the syntax for the COPY command. The text following the diagrams explains the elements of the COPY command syntax diagrams that differ from the elements in the QUICKCOPY command syntax. Use the information that follows with the information provided earlier in this section under the heading "Using the QUICKCOPY Command."



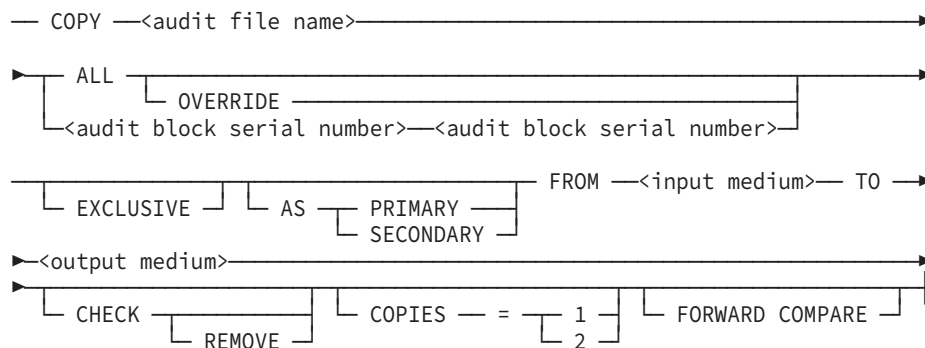
For sectioned audits, the COPY command restores all sections of the audit file from the archive medium. XE features do not provide the capability to copy specific audit sections.

The COPY command can be used when the source file is a sectioned audit file on disk, only if the destination is also on disk. If the destination is not on disk, the QUICKCOPY command must be used.

Using the COPY command while specifying a disk-based sectioned audit file as the source and the destination as a tape results in the following error message:

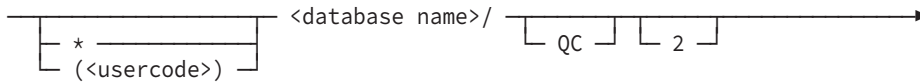
```
The QUICKCOPY command must be used to copy a
sectioned audit file from disk to tape.
```

COPY Command



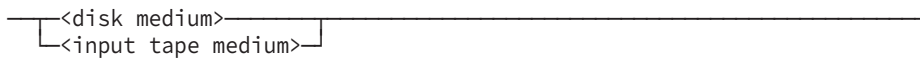
Copying Audit Files

<audit file name>

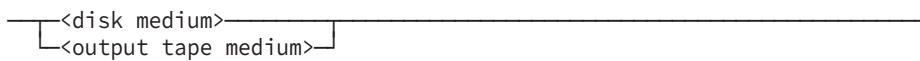


▶ AUDIT<integer>

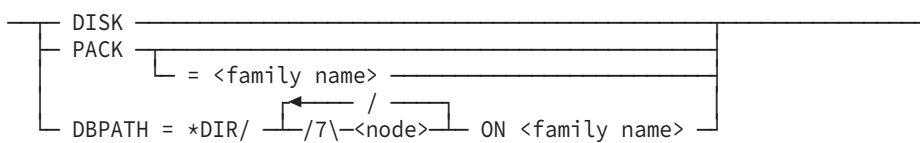
<input medium>



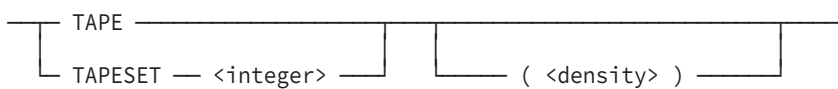
<output medium>



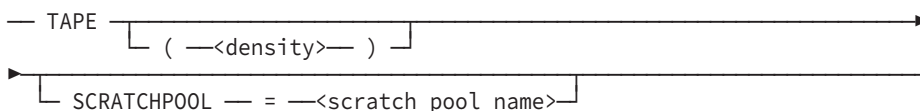
<disk medium>



<input tape medium>



<output tape medium>



<density>

— DENSITY = —<density mnemonic>

Audit File Name Clause

Use the audit file name clause to identify the audit file you want to copy.

If the audit file was copied to tape using the QUICKCOPY command, you must use the quickcopy naming convention when copying back the audit file to disk.

Medium Option

Using the COPY command you can copy audit files

- From disk to tape
- From tape to disk
- From tape to tape

- From disk to disk

Density Specification

Use the density specification to designate the type of tape drive to be used. If you do not supply a density specification, the I/O subsystem uses the system default rules to determine the type of tape drive to use.

FORWARD COMPARE Option

Use the FORWARD COMPARE option to request that the tape be rewound and that the copy of the audit file be checked using a forwardcomparison technique rather than the default readreverse comparison technique. This option is valid only if the CHECK option is also designated. If you use the FORWARD COMPARE option without also using the CHECK option, the request is ignored.

Use the FORWARD COMPARE option with tape drives that do not support the readreverse comparison technique.

Examples

The following examples illustrate correct COPY command syntax.

Example 1

The following command makes two secondary audit file copies of the primary audit file (ADM1)TEST-DB/AUDIT6789. The copies are checked and then the original file is deleted.

```
COPY (ADM1)TEST-DB/AUDIT6789 ALL AS SECONDARY
FROM PACK=AUDITPACK TO TAPE CHECK REMOVE COPIES=2
```

Example 2

The following command copies the primary audit file Z/AUDIT1 to a tape using a quickcopy operation. After the audit file is copied, the tape is rewound and the copy checked using a forwardcomparison technique. The copy of the audit file is automatically renamed Z/QCAUDIT1 and must be copied back to disk before it can be used with any Enterprise Database Server software.

```
COPY Z/AUDIT1 ALL FROM PACK=DMTEXT TO
TAPE CHECK FORWARD COMPARE
```

Example 3

The following commands copy the primary audit file Z/AUDIT1 from a tape to the USER pack:

```
COPY Z/AUDIT1 ALL FROM TAPE TO PACK = USER
```

Example 4

The following command copies all sections of the primary sectioned audit file Z/AUDIT1 from USER1 PACK to USER2 PACK:

Note: You only need to specify the audit file name. Do not include the section number.

```
COPY Z/AUDIT1 ALL FROM PACK=USER1 TO PACK=USER2
```

For example, if the audit file has five sections, five audit files are copied to USER2:

```
Z/AUDIT1  
Z/AUDIT1/1  
Z/AUDIT1/2  
Z/AUDIT1/3  
Z/AUDIT1/4
```

Using the DIRECTORY Command to Display Audit File Tape Directories

Introduction

The DIRECTORY command enables you to identify the audit files contained on each quickcopy tape. You can view the directory online or print the directory. If you use the PRINT option, the directory prints to a session printer backup file.

To identify the tape for which you want a directory, you must supply the tape name. The tape name is always the same as the name of the first file copied to the tape.

The CREATE option is valid only when the audit tape name specifies a TAPESSET name. Use the CREATE option to create the disk file that contains the directory information for files on the tape set. It is recommended that you use this command to create the directory file if the directory file is lost or corrupted, or if the audit TAPESSET command originated from a different host system.

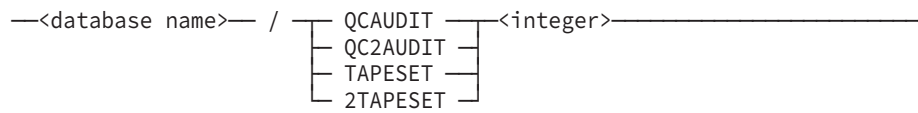
The CREATE option requires access to all tape set reels. If your response to a NO FILE condition indicates that a reel is not available when needed, the DIRECTORY command terminates but completed entries written to the directory are retained.

Syntax

The following diagram illustrates the syntax for the DIRECTORY command:

```
— DIRECTORY —<audit tape name>—  
┌── PRINT ──┐  
└── CREATE ─┘
```

<audit tape name>



The following example will create a new directory file on pack for the TAPESET:

```
DIRECTORY PVDB/TAPESET21 CREATE
```

Example

The following statement generates a printer backup file containing the names of the audit files included on a tape called PVDB/QCAUDIT21. The directory shows the audit file names with the quickcopy naming convention, which identifies that the audit files were copied to tape using the QUICKCOPY command, and that the files must be copied back to disk before they can be used for database recovery purposes.

```
DIR PVDB/QCAUDIT21 PRINT
```

The file generated by the command might contain the following entries:

```
DIRECTORY FOR TAPE PVDB/QCAUDIT21(SERIALNO=101010) ON UNIT#12
FILE#1 = PVDB/QCAUDIT21 (03/15/95);
FIRST ABSN=2860 LAST ABSN=5589
FILE#2 = PVDB/QCAUDIT22 (03/15/95);
FIRST ABSN=5589 LAST ABSN=8385
FILE#3 = PVDB/QCAUDIT23 (03/15/95);
FIRST ABSN=8385 LAST ABSN=11678
FILE#4 = PVDB/QCAUDIT24 (03/15/95);
FIRST ABSN=11678 LAST ABSN=15543
```

Using the VERIFY Command to Verify Audit File Contents

Introduction

Use the VERIFY command to ascertain if an audit file can be used by the data management software without error. You can verify one audit file or a range of audit files. One audit file can be verified on disk or on tape. A range of audit files can be verified only on a quickcopy tape. You cannot verify a range of audit files on disk.

The VERIFY command tests individual logical audit files. It does not test for consistency between audit files.

For nonsectioned audits, the semantics of the VERIFY command are unchanged.

Copying Audit Files



For sectioned audits, the VERIFY command validates the entire audit file. The Enterprise XE features do not provide the capability to verify specific audit sections.

The verification process includes the following checks:

- All audit file data blocks are checked to ensure that they are present and in the correct order.
- The ABSNs and timestamps are verified.
- The audit block serial numbers (ABSNs) are checked to make sure that they increase by one from block to block.
- The timestamp in the previous block is compared with the copy of the previous timestamp in the current block.
- The checksum is verified if the DASDL audit trail option CHECKSUM is designated in the database description file.

If an error is detected during the verification process, the COPYAUDIT program returns an error.

Syntax

The following diagrams illustrate the syntax for the VERIFY command.

VERIFY Command

— VERIFY <audit file name> ON <medium> EXCLUSIVE <audit file range>

<audit file name>

* <database name>/ QC 2
(<usercode>)

▶ AUDIT<integer>

<audit file range>

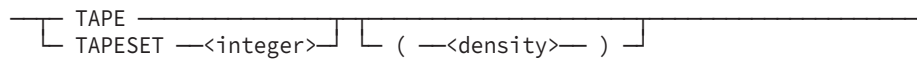
— <audit file name> THRU <audit file name>

<medium>

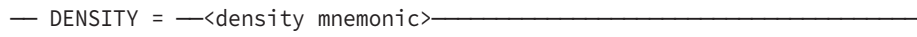
<disk medium>
<tape medium>

DISK
PACK = <family name>
DBPATH = *DIR/ /7\-<node> ON <family name>

<tape medium>



<density>



Examples

The following command verifies the audit file (ADMIN)BANKDB/AUDIT723, which is located on a pack called PRIMAUD:

```

    VERIFY (ADMIN)BANKDB/AUDIT723 ON PACK=PRIMAUD
  
```

The following command verifies the audit files (ADMIN)BANKDB/QCAUDIT24 through (ADMIN)BANKDB/QCAUDIT28, which are located on tape:

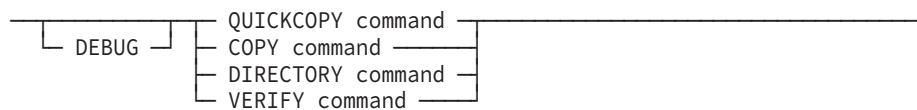
```

    VERIFY (ADMIN)BANKDB/QCAUDIT24 THRU (ADMIN)BANKDB/QCAUDIT28
    ON TAPE
  
```

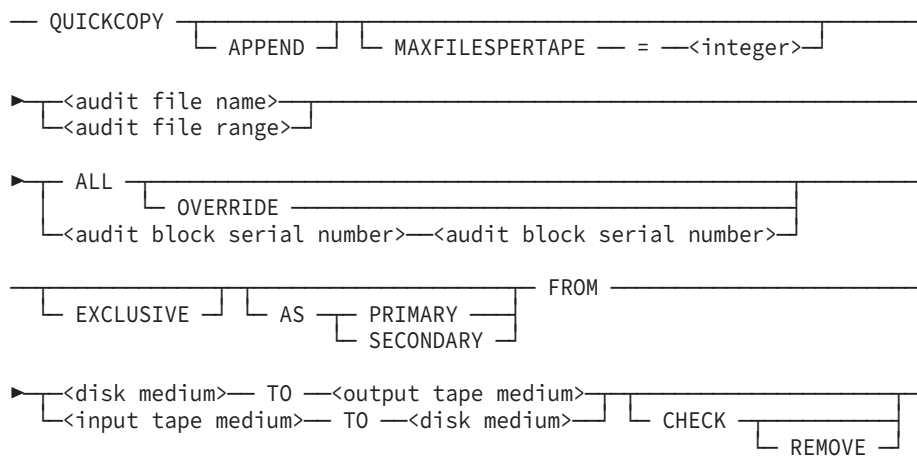
Quick-Reference Information

The information presented here is for quick-reference purposes only. For an explanation of any element of a syntax diagram, refer to the appropriate information presented earlier in this section.

COPYAUDIT Statement



QUICKCOPY Command



Copying Audit Files

COPIES = 1
2 FORWARD COMPARE

<audit file name>

* (<usercode>) <database name> / QC 2

AUDIT <integer>

<audit file range>

<audit file name> THRU <audit file name>

<disk medium>

DISK
PACK = <family name>
DBPATH = *DIR/ /7\ <node> ON <family name>

<input tape medium>

TAPE
TAPESET <integer> (<density>)

<output tape medium>

TAPE
TAPESET <integer>

(/1\ <density> ,)
/1\ COMPRESSED
/1\ NONCOMPRESSED
/1\ AUDITENCRYPT = TDES
AES256
AESGCM

SCRATCHPOOL = <scratch pool name>

<density>

DENSITY = <density mnemonic>

COPY Command

COPY <audit file name>

ALL
OVERRIDE <audit block serial number> <audit block serial number>

FROM —<input medium>— TO —>
 [EXCLUSIVE] [AS] [PRIMARY]
 [SECONDARY]

—<output medium>—
 [CHECK] [REMOVE] [COPIES — = —] [1] [2] [FORWARD COMPARE]

<audit file name>

[*] [<usercode>] [<database name>/] [QC] [2]
 —>
 — AUDIT<integer> —

<input medium>

[<disk medium>]
 [<input tape medium>]

<output medium>

[<disk medium>]
 [<output tape medium>]

<disk medium>

[DISK]
 [PACK] [= <family name>]
 [DBPATH = *DIR/] [/7\] [<node>] [ON <family name>]

<input tape medium>

[TAPE]
 [TAPESET —<integer>] [(— DENSITY = <density> —)]

<output tape medium>

— TAPE [(—<density>—)] —>
 — [SCRATCHPOOL — = —<scratch pool name>] —

<density>

— (— DENSITY = —<density mnemonic>—) —

DIRECTORY Command

— DIRECTORY —<audit tape name>— [PRINT] [CREATE]

Copying Audit Files

<audit tape name>

—<database name> / { QCAUDIT | QC2AUDIT | TAPESET | 2TAPESET } <integer> —————

VERIFY Command

— VERIFY { <audit file name> | <audit file range> } ON <medium> { EXCLUSIVE }

<audit file name>

{ * | <usercode> } <database name> / { QC | 2 }

▶ AUDIT<integer> —————

<audit file range>

—<audit file name> { THRU | _ _ } <audit file name> —————

<medium>

{ <disk medium> | <tape medium> }

<disk medium>

{ DISK | PACK { = <family name> | DBPATH = *DIR/ /7\-<node> } ON <family name> }

<tape medium>

{ TAPE | TAPESET <integer> (<density>) }

<density>

— DENSITY = <density mnemonic> —————

Section 10

Printing, Viewing, and Extracting Audit Information

Overview

The PRINTAUDIT program is a tool that enables you to print, view, and extract to a data file all or part of an audit file.

Note: *The tasks identified in this section can be initiated through Database Operations Center.*

In This Section

The information on using the PRINTAUDIT program is divided into the following topics:

- PRINTAUDIT program overview
- Initiating the PRINTAUDIT program
- Overview of PRINTAUDIT commands
- Selecting audit data
- Generating a customized version of the PRINTAUDIT program
- Quick-reference information for all the syntax diagrams related to using the PRINTAUDIT program

PRINTAUDIT Program Overview

Introduction

The PRINTAUDIT program enables you to perform any of the following tasks:

- Print all or part of an audit file (PRINT command).
- Display online all or part of an audit file (DISPLAY command).
- Write to a disk file all or part of an audit file (EXTRACT command).
- Create a tailored version of the PRINTAUDIT program for specialized data extraction (SELECT command).

The PRINTAUDIT program works with both primary and secondary audit files, and can be run interactively or in batch mode. You can also use the PRINTAUDIT program with the audit files on both the Remote Database Backup primary and secondary hosts.

Selecting Audit Information

Using the PRINTAUDIT program, you can select

- The complete audit file
- A range of audit records (referred to in this section as an interval)
- The type of audit information
- The type of audit information within an interval

Types of Interval

You can specify an interval based on any of the following criteria:

- Time interval by month, day, year, hour, minute, and second
- Serial interval by audit block serial number (ABSN)
- Relative block interval by relative block numbers

Criteria for Selecting Audit Records

You can select audit records based on any of the following criteria:

- Stack number or a range of stack numbers
- Program mix number or a range of program mix numbers
- Program identifier—that is, the program file title
- Structure number, a range of structure numbers, blocks within a structure, or by a word offset within a block
- Record type selection by audit record type
- Field selection by data within a specified field

If the data selection choices available in PRINTAUDIT do not suit your needs, you can use the PRINTAUDIT *SELECT* command to create a tailored PRINTAUDIT program that does suit your needs.

Initiating the PRINTAUDIT Program

Introduction

You can run the PRINTAUDIT program interactively or in batch mode. In either mode the syntax is similar. The primary differences between the two PRINTAUDIT modes are that

- In interactive mode, you are prompted for the PRINTAUDIT commands that you want to issue.
- In interactive mode, you cannot use the SELECT command to create a tailored PRINTAUDIT program.
- In batch mode, you must supply a card file that details the audit information you want to access.

Interactive Mode

To initiate an interactive PRINTAUDIT session, use the following procedure:

1. Enter the following CANDE command:

```
RUN *SYSTEM/PRINTAUDIT;  
FILE AUDIT (TITLE=<audit file title>);  
FILE DASDL (TITLE=<description file title>);
```

The PRINTAUDIT program displays "PLEASE ENTER REQUEST:" to indicate that the program is ready for additional commands and options.

2. Enter the PRINTAUDIT commands.

Normally, you can enter the commands on a single line; however, commands can be longer than the space available on one line width of the terminal screen. If your command exceeds one line, use the following procedure to enter a multiple-line command:

- a. Type the first line of the command and type a percent sign (%) at the end of the line to indicate that more input follows.
- b. Press the transmit key.
The PRINTAUDIT program responds with #% (a number sign followed by a percent sign).
- c. Repeat steps a and b until you are ready to enter the last line of the command.
- d. Type the last line of the command and do not put a percent sign (%) at the end of the line.
- e. Press the transmit key.

The PRINTAUDIT program processes the entire command. The requested information is either printed or displayed online. Once the command has been processed and the results generated, the PRINTAUDIT program displays the following messages:

```
REQUEST COMPLETE  
PLEASE ENTER REQUEST
```

Batch Mode

To initiate the PRINTAUDIT program in batch mode, you must include a card file in the run statement. Use the card file to identify the PRINTAUDIT commands you want to process. The internal name of the card file is CARD.

Printing, Viewing, and Extracting Audit Information

If you run the PRINTAUDIT program, and a card file is not present and the task attribute STATION is not 0 (zero), an interactive PRINTAUDIT session starts.

If the card file is not present and the task attribute STATION is 0 (zero), the program prints all records of the specified audit file in hexadecimal digit format.

You can use a WFL file equation to equate the card file to a disk file.

Using a WFL Job

Use a WFL job with the following layout to initiate a PRINTAUDIT run in batch mode:

```
BEGIN JOB PRINTAUDIT;
RUN SYSTEM/PRINTAUDIT;
FILE AUDIT (TITLE=<audit file title>);
FILE DASDL (TITLE=<description file title>);
DATA CARD
<one or more PRINTAUDIT commands>
? % end DATA CARD
END JOB
```

Using CANDE

Use a CANDE statement with the following format to initiate a PRINTAUDIT run in batch mode:

```
RUN *SYSTEM/PRINTAUDIT;
FILE AUDIT(TITLE=<audit file title>);
FILE DASDL (TITLE=<description file title>);
FILE CARD (TITLE=<card file name> ON <pack name>);
```

Entering Information in the Card File

Normally, commands are entered in a single record; however, some commands can exceed the allowable record width. To enter a multiple-record command, insert a percent sign (%) at the end of the record to indicate that more input follows. Repeat the process as many times as necessary. Do not put a percent sign (%) at the end of the last line of the command.

Identifying the Audit File to be Analyzed

Use the FILE AUDIT file equation statement to identify the audit file to be analyzed.

The audit file can be either a primary or a secondary audit file, and can reside on disk or tape. The naming convention for audit files is

- Primary audit files have the title <database name>/AUDIT<audit number>.
- Secondary audit files have the title <database name>/2AUDIT<audit number>.

You cannot use the PRINTAUDIT program with quickcopy audit files. If you want to use the PRINTAUDIT program with audit files copied to tape using the QUICKCOPY command, you must first copy back the audit files to disk.

Examples

The following statement causes the PRINTAUDIT program to run interactively and to process the secondary audit file (ADMIN)BANKDB/2AUDIT765, which is located on tape. In this example, the database description file is titled (ADMIN)DESCRIPTION/BANKDB and is located on a pack called ISYS.

```
RUN *SYSTEM/PRINTAUDIT;  
FILE AUDIT (TITLE=(ADMIN)BANKDB/2AUDIT765, KIND=TAPE);  
FILE DASDL(TITLE=(ADMIN)DESCRIPTION/BANKDB ON ISYS);
```

The following statement causes the PRINTAUDIT program to run interactively and to process the audit file *DIR/TEST/BANKDB/AUDIT1, which is located within a permanent directory. In this example, the database description file is not used.

```
RUN *SYSTEM/PRINTAUDIT;  
FILE AUDIT (TITLE=*DIR/TEST/BANKDB/AUDIT1 ON MYAUDITPK);
```

TAPESERVER System Option and RoboHost Units

A subtle operational change occurs if the TAPESERVER system option is set. If you designate an audit file that cannot be retrieved by a RoboHost unit because either the RoboHost unit or the tape itself is not available, then perform the following steps:

1. Use the NF (No File) system command to respond to the No File condition.
2. Supply the required AX response to the following user message:

```
RETRY OR FAMILY = <FAMILY NAME>
```

Relating Structure Names to Structure Numbers

Using the PRINTAUDIT program you can view audit information that relates to specific database structures. The description file title supplied in the PRINTAUDIT run statement is used to correlate structure names to structure numbers. By default the description file is assumed to have the following title:

```
DESCRIPTION/<database name> ON <myself.family>
```

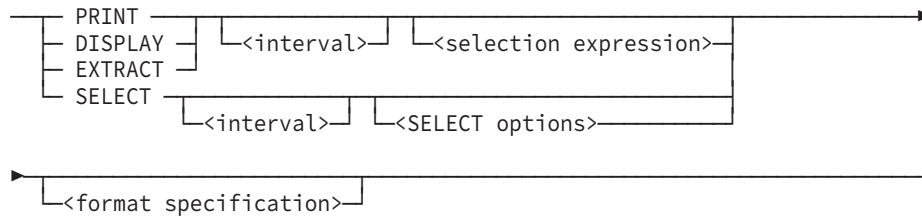
The myself.family construct identifies the default primary or secondary pack associated with the usercode from which you run the PRINTAUDIT program.

If the description file is not available and you want to select information based on a database structure, use the structure number and not the structure name to identify the structure about which you want information.

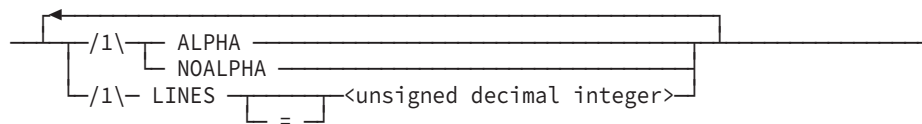
Example

The following statement causes the PRINTAUDIT program to run interactively and identifies the name of the description file as *DESCRIPTION/BANKDB and the location as the pack SYSPACK. In this example, the audit file being processed is named (ADMIN)BANKDB/AUDIT82 and is located on the pack AUDPACK.

<PRINTAUDIT output request>



<format specification>



PRINTAUDIT Output Request

The PRINTAUDIT output request identifies the information you want to select and directs the output to one of the following locations.

Request	Directs the selected audit information to . . .
PRINT	The system printer.
DISPLAY	The terminal, so that the information can be viewed online.
EXTRACT	<p>A disk file called PRINTAUDIT/REPORT/<task number>.</p> <p>To supply a different disk file name, file-equate the name you want to use to the internal file name DK when you initiate the PRINTAUDIT program.</p> <p>For example, the following CANDE statement initiates the PRINTAUDIT program in interactive mode to analyze the audit file (DMPROD)DB/AUDIT5 ON ISYS. The statement uses the file equation statement to direct extracted information to a file called (SRL)MYDB/PRINTAUDIT/REPORT ON REPORT. For this example to work, the usercode from which the PRINTAUDIT program is run must have access to both the SRL and DMPROD usercodes.</p> <pre> RUN *SYSTEM/PRINTAUDIT; FILE AUDIT (TITLE=(DMPROD)DB/AUDIT5 ON ISYS); FILE DK (TITLE=(SRL)MYDB/PRINTAUDIT/REPORT ON REPORT); </pre>

Request	Directs the selected audit information to . . .
SELECT	<p>A location determined by a tailored version of the PRINTAUDIT program you develop.</p> <p>The PRINTAUDIT program enables you to choose audit information. However, if the information you require cannot be identified with the options provided in the basic PRINTAUDIT program, use the PRINTAUDIT command SELECT to create a tailored version of the PRINTAUDIT program so that you can choose audit information based on criteria you define.</p> <p>You can use the SELECT command only when running the PRINTAUDIT program in batch mode.</p>

QUIT Command

Use the QUIT command to end an interactive PRINTAUDIT session.

Interval Clause

Use the interval clause to identify the portion of the audit file from which information is to be selected.

Refer to “Designating Intervals” later in this section for detailed information on this topic.

Selection Expression

Use the selection expression to identify the audit records you want to print, display, or extract.

If you supply a selection expression without defining an interval clause, all audit records in the audit file that meet the selection expression criteria are chosen.

If you supply both a selection expression and an interval clause, both conditions must be met before a record is chosen.

Refer to “Selecting Audit Data” later in this section for detailed information on this topic.

SELECT Options

Use the SELECT options when creating a tailored version of the PRINTAUDIT program. For more information, refer to “Generating a Customized Version of the PRINTAUDIT Program” later in this section.

Format Specification

Use the format specification to further refine the format of the output from a PRINTAUDIT run. The options available in the format specification are described in the following text.

Option	Explanation
ALPHA	Directs the PRINTAUDIT program to format information in both hexadecimal and alphanumeric formats. By default, the output is in hexadecimal format only.
NOALPHA	Directs the PRINTAUDIT program to format information in hexadecimal format only. This is the default setting.
LINES = <unsigned decimal integer>	Defines the maximum number of lines of hexadecimal information printed for each audit record. Audit record heading information is not included in the line count. The default number of lines printed for each audit record is 99999.

Designating Intervals

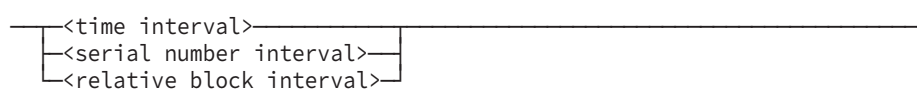
Introduction

You can use any of the following intervals to identify the portion of the audit file you want to examine. By default, the PRINTAUDIT program examines the complete audit file.

- Time interval
- Serial number interval
- Relative block interval

The following diagram illustrates the relationship between the three types of interval. Each of the interval types is described in more detail in the text that follows.

<interval>



Block Zero and Partial Audit Record Information

In addition to the audit information you specifically request, the PRINTAUDIT program always selects the contents of block 0 (zero) and any partial audit records at the beginning or at the end of an audit file.

An audit record is a logical entity that is often larger than one physical block or record. When an audit file switch occurs, the logical audit record might be split into two physical parts because of physical space limitations. The first part is stored in the old audit file, and

the second part in the new audit file. The two physical halves of the audit record are referred to as partial audit records. Partial audit records are always selected because the PRINTAUDIT program cannot determine if the records meet the requested selection criteria.

The information displayed for block 0 (zero) contains the following two fields:

- AUDITEOF (word 1)

The AUDITEOF field identifies the location of the end of the audit file. The frequency with which the information in the AUDITEOF field is updated depends on the setting of the DASDL audit trail option UPDATE EOF.

For more information on the UPDATE EOF audit trail option, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

- AUDITTIMESTAMP (word 2)

The AUDITTIMESTAMP field identifies when the audit file was created.

Designating a Time Interval

Introduction

Use the time interval to identify the portion of the audit file to be examined based on the date and time at which the audit information was created. Using the time interval you can designate two timestamps. One timestamp designates a starting point and the other timestamp designates a stopping point for the selection. As part of the timestamp you can designate both a date and a time.

Getting More or Less Data Than You Expect

There are many different types of audit records. Some audit records contain timestamps and others do not. The PRINTAUDIT program verifies which audit records fit a time interval based on a timestamp that is contained only in a subset of the audit records. Therefore, using a time interval as your selection criterion might produce a different number of audit records than you expect.

First Audit Record Selected

The first audit record encountered that meets the time interval criteria is the first record that is selected. If no audit record is found with a suitable timestamp, then no records are selected even if audit records were generated during the designated time interval.

Once a suitable audit record is located all subsequent audit records are selected whether or not they contain a timestamp.

For example, under the following circumstances, audit records are returned only for the period starting at 15:01 on May 22, 2005:

- You designate a starting point of 13:22 on May 22, 2005.
- There is no record with the timestamp 13:22.

- The first record with a timestamp greater than the one requested is 15:01 on May 22, 2005.

If the stopping point you designate is less than 15:01 on May 22, 2005, then no audit records are selected.

Last Audit Record Selected

If an audit record with the exact ending timestamp is available, then that is the last audit record selected. If there is not an audit record with the exact ending timestamp, the last audit record selected is the one before the audit record that contains a timestamp greater than the stopping point for the selection criteria. As a result, some of the audited records selected might have been generated after the designated stopping point.

For example, under the following circumstances, all the audit records up to the record containing the 11:05 on July 21 timestamp are selected:

- You designate a stopping point of 10:09 on June 26, 2005.
- There is no record with that timestamp.
- The first record with a timestamp greater than 10:09 on June 26 is 11:05 on July 21.

Effect of Time Changes

When using a time interval, if the system clock of the machine has been moved backward (for example, to switch from daylight saving time to standard time), do not designate as the starting or stopping point of the time interval a point in time during the time change. For instance, if the system clock has been set backward from 2 a.m. to 1 a.m., do not use a stopping point between 1 a.m. and 2 a.m. Instead, choose a time after 2 a.m. or before 1 a.m.

Designating One Timestamp in the Time Interval

If you supply only one timestamp, the timestamp identifies the starting time; the stopping point is the end of the specified audit file.

Designating Two Timestamps in the Time Interval

If you supply two timestamps, the first timestamp is the starting point; the second timestamp is the stopping point. The first timestamp must be earlier (less) than, or the same as, the second timestamp.

If you designate two timestamps, and the audit file ends before the second timestamp is reached, the PRINTAUDIT program automatically looks for the next audit file in the series.

Designating a Date but Not a Time

If the first timestamp you supply contains no time of day, the interval is assumed to start at the beginning of the specified day. If the second timestamp you supply contains no time of day, the interval is assumed to terminate at the end of the specified day.

Printing, Viewing, and Extracting Audit Information

For example, if the starting date and the ending dates are both set to 12/03/94, then all the audit records for 12/03/94 are selected.

Designating a Time but Not a Date

If the first timestamp you supply contains no date, today's date is assumed. If the second timestamp you supply contains no date, the date of the first timestamp is assumed.

Syntax

The following diagrams illustrate the syntax for designating a time interval.

<time interval>

— TIME —<date stamp>—
 └── TO —<date stamp>—┘

<date stamp>

┌──<date>— @ —<time>—┐
├──<date>— ┐
└── @ —<time>— ┘

<date>

—<month>— / —<day>— / —<year>—

<time>

—<hour>— : —<minute>—
 └── : —<second>—┘

Syntax Explanation

The following information explains the elements of the syntax diagrams.

Option	Valid Values
<month>	An integer between 1 and 12 that identifies the month.
<date>	An integer between 1 and 31 that identifies the day.
<year>	An integer of either 2 digits or 4 digits that identifies the year. When specified in 4 digits, the year must be between 1970 and 2035. When specified in 2 digits with a value in the range 00 to 35, the year is treated as 20xx (that is, after A.D. 2000). When specified in 2 digits with a value of 70 to 99, the year is treated as 19xx (that is, before A.D. 2000). Twodigit years between 35 and 70 are invalid.
<hour>	An integer between 0 and 23 that identifies the hour according to a 24hour clock. The value 0 (zero) represents midnight.

Option	Valid Values
<minute>	<p>An integer between 0 and 60 that represents the number of minutes past the hour.</p> <p>If you use 60 for the minutes, then the time is converted to the next hour. For example, supplying a time of 1:60:00 is equivalent to supplying a time of 2:00:00.</p>
<second>	<p>An integer between 0 and 60 that represents the number of seconds past the minute and hour.</p> <p>If you use 60 for the seconds, then the time is converted to the next minute. For example, supplying a time of 1:30:60 is equivalent to supplying a time of 1:31:00. If you do not supply a value for the seconds, 0 (zero) seconds is assumed.</p>

Designating a Serial Number Interval

Introduction

Use the serial number interval to select a portion of the audit file based on the audit block serial number (ABSN) associated with an audit record. Note that there can be more than one audit record associated with each ABSN.

Designating Starting and Ending Points

If you use the serial number interval to select a portion of the audit file, you must explicitly designate both the starting and ending ABSN. The first ABSN must be greater than 1 and less than or equal to the second ABSN. The ending point for the interval can be in the same audit file as the starting point, or it can be in a different audit file. Any required audit file switching is performed automatically.

You can designate the serial numbers as either unsigned decimal integers or unsigned hexadecimal integers.

Syntax

The following diagrams illustrate the syntax for designating a serial number interval.

<serial interval>

— SERIAL —<ABSN>—<ABSN>—————|

<ABSN>

—<unsigned decimal integer>—————|
 (—<unsigned hexadecimal integer>—) —|

Designating a Relative Block Interval

Introduction

Use the relative block interval to identify a portion of the audit file by identifying the position of the blocks in relation to the start or end of the audit file.

Designating Starting and Ending Points

You must designate both a starting and an ending block number. The relative block interval can be used to designate blocks in one audit file only; audit file switching is not supported with this option.

The first relative block number you designate must be less than or equal to the second block number you designate. You can designate the block numbers using unsigned decimal integers, unsigned hexadecimal integers, or an * (asterisk).

Identifying Blocks in Relation to the Start of the Audit File

Use 1 to identify the first block in the audit file, use 2 to identify the second block, use 3 to identify the third block, and so on.

Identifying Blocks in Relation to the End of the Audit File

Use * (asterisk) to identify the last block in the audit file, use *-1 to identify one block from the end of the audit file, use *-2 to identify the second block from the end of the audit file, and so on.

Note: If you use this method to identify blocks on a tape file, the entire tape is read first to determine the number of the last block. Then the tape is rewound and read again while being processed.

Syntax

The following diagrams illustrate the syntax for designating a relative block interval.

<relative block interval>

—<block number>—<block number>—————|

<block number>

—<unsigned decimal integer>—————|
| (—<unsigned hexadecimal integer>—) |
| * —————|
| * — — —<unsigned decimal integer>—————|

Selecting Audit Data

Introduction

You can select audit records based on any of the following criteria:

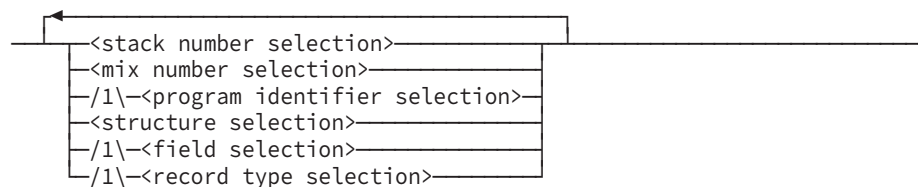
- Stack number or a range of stack numbers
- Program mix number or a range of program mix numbers
- Program identifier—that is, the program file title
- Structure identifier, a range of structure numbers, blocks within a structure, or a word offset within a block
- Field selection by data within a specified field
- Record type selection by audit record type

By default, all audit information is selected. If you designate an audit interval and define the type of data you want, only data that meets both conditions is selected.

Syntax

As shown in the following syntax diagram, you can use some selection criteria once only, you can use some selection criteria more than once, and you can combine different selection criteria. Each of the selection criteria is described in more detail in the text that follows.

<selection expression>



Selecting Records by Stack Number

Introduction

Use the stack number selection to choose records based on the stack numbers of the programs that caused the database changes.

You can use both unsigned decimal integers and unsigned hexadecimal integers to identify stack numbers.

Supplying a Range of Stack Numbers

You can provide a list of stack numbers or ranges of stack numbers. If you supply a range, the smaller stack number must be provided first. For example, the range 10010-10017 is valid, while the range 16026-15429 is invalid.

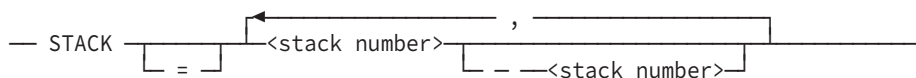
Combining Stack Number Selections with Other Selection Criteria

If you use a stack number selection with any other selection criteria, only audit records that have the correct stack number and meet the other selection criteria are selected.

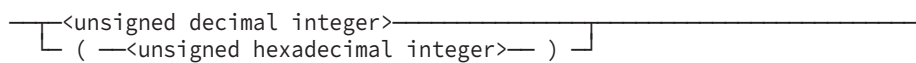
Syntax

The following diagrams illustrate the syntax for designating a stack number selection.

<stack number selection>



<stack number>



Selecting Records by Program Mix Number

Introduction

Use the mix number selection to choose records based on the mix numbers of the programs that caused the database changes.

The required audit information is located by identifying the stack number related to the designated mix number and then using the stack number to identify the appropriate audit records. The restart data set open (RDSO) audit records contain the mix number to stack number association information.

Supplying a Range of Mix Numbers

You can provide a list of mix numbers or ranges of mix numbers. If you supply a range, the smaller mix number must be provided first. For example, the range 1834-1897 is valid, while the range 8672-7234 is invalid.

Selecting Data Near the Beginning of an Audit File

If you select data by mix number, data toward the beginning of an audit file might not be selected. If the RDSO record associated with a mix number occurs in a prior audit file, the data in the current audit file affected by the mix number is not selected.

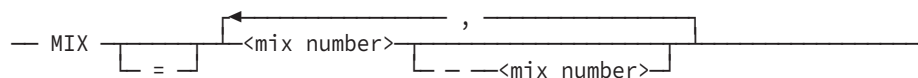
Combining Mix Number Selections with Other Selection Criteria

If you use a mix number selection with any other selection criteria, only audit records that have the correct mix number and meet the other selection criteria are selected.

Syntax

The following diagrams illustrate the syntax for designating a mix number selection.

<mix number selection>



<mix number>



Selecting Records by Program Identifier

Introduction

Use the program identifier to choose records based on the title of the program that caused the database changes. The program identifier is matched against information in the restart data set open (RDSO) and restart data set close (RDSC) records. If a match occurs, the mix number information in the record is added to the list of mix numbers being selected. Thus, if more than one execution of the program causes entries in the audit file, all records for all executions are processed.

Selecting Data Near the Beginning of an Audit File

If you select data by program identifier, data toward the beginning of an audit file might not be selected. If the RDSO record associated with the program occurs in a prior audit file, the data in the current audit file affected by the program is not selected.

Identifying the Full Program Title

If you do not include a usercode with the program name, then the usercode from which you are running the PRINTAUDIT program is used. If you do not include a pack name, then all audit records that contain the program name are processed.

For more information on designating file titles, refer to the *WFL Reference Manual*.

Syntax

The following diagram illustrates the syntax for designating a program identifier selection.

<program identifier selection>

— PROGRAM = <file title>—————|

Selecting Records by Structure Identifier or Block Number

Introduction

Use the structure selection to choose records based on any of the following criteria:

- Structure names or numbers
- Ranges of structure numbers
- Blocks, or ranges of blocks, within a structure
- Records located at a designated word offset within a block within a structure

Combining Structure Selections with Other Selection Criteria

If you use a structure selection with any other selection criteria, only audit records that affect the correct structure, block, or record and meet the other selection criteria are selected.

Selecting Structures by Name

If you select a structure by name, the PRINTAUDIT program uses the database description file identified in the PRINTAUDIT run statement to correlate a structure name to a structure number. By default, the database description file is assumed to be

```
DESCRIPTION/<database name> ON <myself.family>
```

Selecting More Than One Structure or Block

If you select more than one structure or block,

- The second structure name must designate a structure whose structure number is higher than that designated by the first structure name.
- The second structure number must be greater than or equal to the first structure number.
- The second block number must be greater than or equal to the first block number.

Using the Block Address Field (BAF) and Word Address Field (WAF) Values

Use the block address field (BAF) and word address field (WAF) values to identify particular records in the database. If a beforeimage or an afterimage database record was audited, then the identified audit record is selected. The BAF identifies the block that contains the desired record. The WAF identifies the word offset of the record in the block.

Specifying an Alias Name in a Structure Identifier

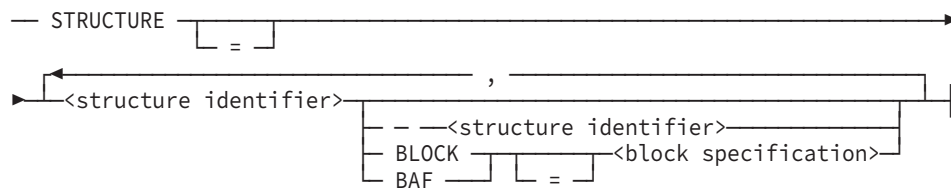
You can specify an alias name for a structure identifier. This means that you can refer to a structure identifier using 16-bit character structure names through COBOL85 programs and Enterprise Database Server utility programs. The alias name is treated the same as a regular structure name.

For more information about alias names, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

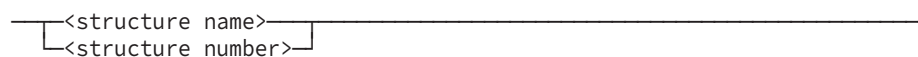
Syntax

The following diagrams illustrate the syntax for designating a structure selection.

<structure selection>



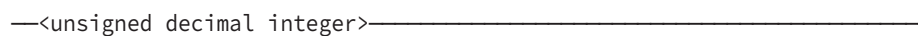
<structure identifier>



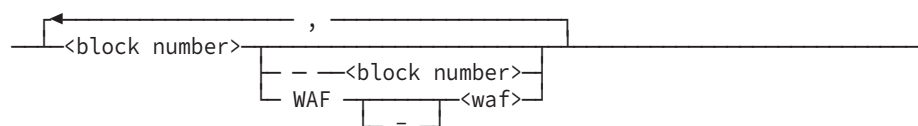
<structure name>



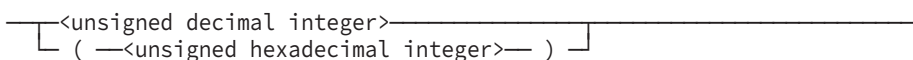
<structure number>



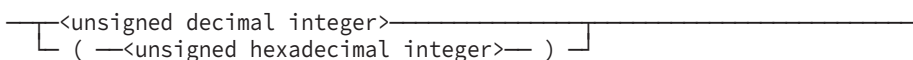
<block specification>



<block number>



<waf>



Selecting Records by Field

Introduction

Use the field selection to choose audit records that contain beforeimage or afterimage database records with a specific value at a particular position within the database record.

Compile your database with the compiler control option LAYOUT set so that you can generate a list of the items, and key item offsets and sizes for the database record. The offsets provided by the LAYOUT option are for the memory record layout format. The data in an audit is in disk record layout format. Offsets for items other than key items might be different between the memory and disk record layout formats.

Limiting the Search Automatically to Specific Record Types

When you use a field selection, the PRINTAUDIT program selects records only from the following list of record types. All other record types are excluded from the selection process.

- Afterimage only (AIO)
- Beforeimage only (BIO)
- Change compact data (CCD)
- Data set delete (DSD)
- Data set modify (DSM)
- Data set create (DSC)

Combining Field Selections with Other Selection Criteria

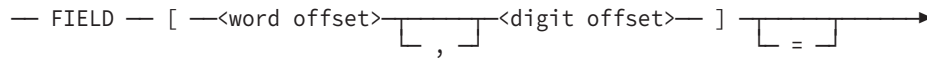
If you use a field selection with any other selection criteria, only audit records that contain the appropriate field selection value and meet the other selection criteria are selected.

To obtain meaningful results, include exactly one structure selection with the field selection.

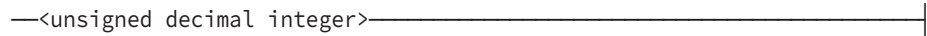
Syntax

The following diagrams illustrate the syntax for designating a field selection.

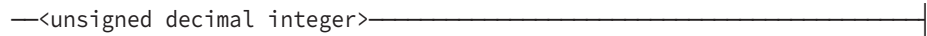
<field selection>



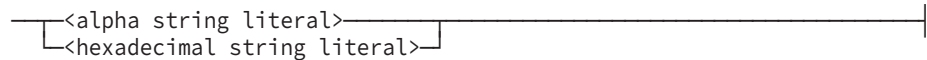
<word offset>



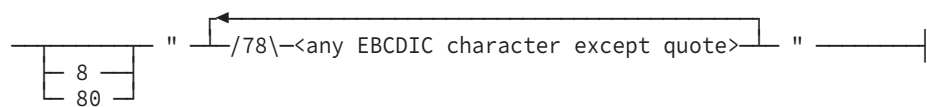
<digit offset>



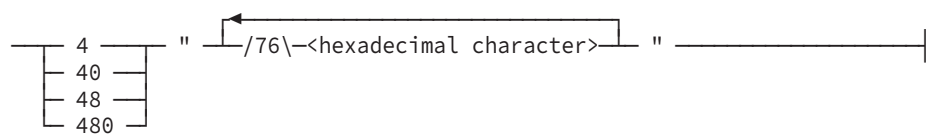
<value>



<alpha string literal>



<hexadecimal string literal>



Syntax Explanation

You can use any EBCDIC character in the alpha string literal except for quotation marks (").

You can use the digits 0 through 9, and the uppercase letters A through F in the hexadecimal string literal.

For more information on alpha and hexadecimal string literals, refer to the discussion on string literals in the *Application Development Solutions ALGOL Programming Reference Manual, Volume 1: Basic Implementation*.

Selecting Records by Record Type

Introduction

The audit file is composed of records of various types. Each audit record type has an associated number and mnemonic name. Use the record type selection to retrieve information for a specific record type number or mnemonic.

Combining Record Type Selections with Other Selection Criteria

If you use a record type selection with any other selection criteria, only audit records that are the correct type and meet the other selection criteria are selected.

Matching Audit Record Types, Names, and Mnemonics

The record type mnemonics and their meanings are given in [Table 10–1](#), which follows the record type selection syntax diagram. In [Table 10–1](#), the record types are listed by mnemonic in alphabetic order.

Syntax

The following diagram illustrates the syntax for designating a record type selection.

<record type selection>

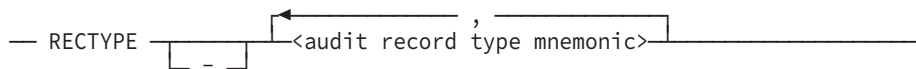


Table 10–1. Record Type Mnemonics

Record Type Mnemonic	Record Type Number	Description
C		Control records Use C to designate all of the following records: SPT, BCP, ECP, DBSI, DBST, FILEDC, STRDC, and RECOV
D		Data change records Use D to designate all of the following records: DSC, DSD, DSM, AIO, BIO, and CCD.
ADSS	16	Allocate data set space
AGCI	89	Aggregate change image
AINI	23	Audit initial block values
AIO	30	Afterimage only
AIRE	39	Add index random entry
AISE2	101	Add index sequential entry
ALN	00	Audit line number (in Accessroutines)
ARNE	51	Add random entry
AUDMIS	90	Miscellaneous audit
AUDTB2	67	Audit change to word Y,X in block

Table 10-1. Record Type Mnemonics (cont.)

Record Type Mnemonic	Record Type Number	Description
BCP	02	Begin control point
BIO	31	Beforeimage only
BLKIMG	59	Block image
BTR	04	Begin transaction
BUSAIO	115	DATAEXCHANGE audit image
BVEOF	36	Bit vector end-of-file
B4ROOT	75	Ordered data set root beforeimage
CCD	62	Change compact data
CDI	104	Change dynamic information
CIRE	41	Change index random entry
CISE	15	Change index sequential entry
CPID	56	Change partition ID
CPNT	45	Change partition names table
CUOL	26	Change unordered data set storage links
DBSI	21	Database stack initiate
DBST	22	Database stack terminate
DDCD	80	Direct data set change of data end-offile
DDSEOF	54	Direct data set end-of-file
DDSIEF	53	Direct data set increment end-of-file
DDSIEOF	53	Direct data set increment end-of-file (same as DDSIEF)
DIRE	40	Delete index random entry
DISE	14	Delete index sequential entry
DISE2	102	Delete index sequential entry on Mark 3.9 and later releases
DLIRT	43	Delink index random table
DLKBLK	48	Delink block
DLLIST	38	Delink ordered/unordered list table
DRNE	52	Delete random entry
DSC	10	Data set create
DSD	11	Data set delete
DSM	12	Data set modify

Table 10-1. Record Type Mnemonics (cont.)

Record Type Mnemonic	Record Type Number	Description
DSSD	17	Deallocate data set space
ECP	03	End control point
ETR	05	End transaction
FGTBLK	28	Forget data block
FILEDC	74	File discontinuity
GCAISE	106	Garbage collection add index sequential entry
GCB	63	Get compact block
GCBLK	105	Garbage collection block
GCCISE	108	Garbage collection change index sequential entry
GCDISE	107	Garbage collection delete index sequential entry
GCGIST	109	Garbage collection get index sequential entry
GCGROW	111	Garbage collection grow index sequential table
GCISPT	112	Garbage collection index sequential split table
GCRIST	110	Garbage collection return index sequential table
GETBLK	27	Get data block
GIST	19	Get index sequential table
GRNS	49	Get random space
GRWIST	25	Grow index sequential table
GTRL	103	Global transaction link record
IDSBLK	61	Insert data set block
INSBLK	47	Insert block
ISPT	24	Index sequential split table
KIOA	90	KEYEDIOII allocation change
LGRA	34	Last good restart array
LGRR	76	Last good restart record
LTAR	93	Long transaction address record
MIDTR	78	Midtransaction
ODDSC	73	Ordered data set create
ODEOF	68	Get ordered data set block
ODRTBL	69	Return ordered data set block

Table 10–1. Record Type Mnemonics (cont.)

Record Type Mnemonic	Record Type Number	Description
ODSFDN	70	Shift ordered data set records down
ODSFUP	71	Shift ordered data set records up
ODSPBL	72	Split ordered data set block
OINZP	55	Open initialize partition
PFIX	94	Remote Database Backup path fixup record
PNT	44	Partition names table
QDCREC	114	Quiesce database copy audit record
RCB	64	Return compact block
RDERR	77	Block image of read errors
RDSC	33	Restart data set close
RDSO	32	Restart data set open
RECOV	29	Recovery point
RFIX	98	Online reorganization fixup
RFLUSH	100	Online reorganization FLUSHDB
RFMT	96	Online reorganization structure FMTSTAMP change
RIST	20	Return index sequential table
RLOCK	65	Row lockout
RMOVE	97	Online reorganization data move
RPATH	99	Online reorganization path
RRNS	50	Return random space
RSTATE	95	Online reorganization state change
RUODSS	28	Return unordered data set space (same as FGTBLK)
SAA	86	Single abort assign record
SAAG	88	Single abort aggregate record
SAC	81	Single abort create record
SAD	82	Single abort delete record
SAI	84	Single abort insert record
SAM	83	Single abort modify record
SAR	85	Single abort remove record
SDSEOF	35	Standard data set end-of-file

Table 10–1. Record Type Mnemonics (cont.)

Record Type Mnemonic	Record Type Number	Description
SIBC	92	Structure information block (SIB) close
SIBO	91	Structure information block (SIB) open
SPIRT	42	Split index random table
SPLIST	37	Split ordered/unordered list table
SPT	01	Syncpoint
STRDC	79	Structure discontinuity
SVPT	87	Savepoint record
TABAI	58	Table afterimage
TABBA	66	Table beforeimage and afterimage
TABBI	57	Table beforeimage
TBLXCH	18	Table exchange

Generating a Customized Version of the PRINTAUDIT Program

Introduction

If you have data selection requirements that differ from the capabilities provided by the standard PRINTAUDIT program, you can create a tailored version of the PRINTAUDIT program.

This PRINTAUDIT facility is also called *dynamic recompilation* of the PRINTAUDIT program.

Basic Steps

The basic steps in creating a tailored version of the PRINTAUDIT program are as follows:

- Write ALGOL code to define your data selection criteria.
- Embed the ALGOL code in your PRINTAUDIT statement as part of the SELECT statement.
- Run the PRINTAUDIT program in batch mode to compile and run a tailored version of the PRINTAUDIT program.

The ALGOL code you supply is inserted between the last declaration and the first executable code of the basic PRINTAUDIT program. You can include declarations, initialization code, inner loop code, and wrap-up code so that you can process as well as print the audit trail information you want with the tailored PRINTAUDIT program.

These steps are discussed under the following headings later in this section:

- Developing the ALGOL code
- Using the SELECT statement with the ALGOL code

Compiling the Tailored PRINTAUDIT Program

The tailored version of the PRINTAUDIT program is compiled automatically when you run the PRINTAUDIT program.

[Table 10–2](#) identifies the file equations you can use when you run the PRINTAUDIT program. The files are used at compilation time, and they are located by following the standard file search rules.

Table 10–2. PRINTAUDIT File Equations

Internal Name	Function and External Name
SOURCE	Identifies the name and location of the PRINTAUDIT symbol file. The default source file is DATABASE/PRINTAUDIT ON DISK.
PROPERTIES	Identifies the name and location of the database properties file. The default properties file is DATABASE/PROPERTIES ON DISK.
INTL	Identifies the name and location of the internationalization features file. The default internationalization features file is SYMBOL/INTL/ALGOL/DMS/PROPERTIES ON DISK.

Naming Convention for Compiled Object Code

The compiled object code is named according to the following convention:

```
PRINTAUDIT/PROG/<job mix no><task mix no>
```

Naming Convention for Generated WFL Job

The WFL job generated for the compilation is also saved. The generated WFL job is named according to the following convention:

```
PRINTAUDIT/PATCH/<job mix no><task mix no>
```

Developing the ALGOL Code

Required Elements

In the ALGOL code you supply, you must include a procedure (or define) with no parameters, called USERPROCEDURE. Use the USERPROCEDURE procedure to provide the audit data selection logic.

The USERPROCEDURE procedure is called automatically once for each audit record in the interval you designate. To include the scanned record in the program output, assign the value TRUE to the global Boolean variable PRINTIT. If you do not want to include the record in the program output, assign a value of FALSE to the PRINTIT variable.

Optional Elements

You can include the following two optional elements in the ALGOL code:

- USERWRAPUP procedure
- User-declared variables and procedures

USERWRAPUP Procedure

Use the untyped procedure with no parameters, called USERWRAPUP, to define additional tasks you want the tailored PRINTAUDIT program to complete after all the required audit records have been selected. For example, use the USERWRAPUP procedure to display or print the number of audit records selected.

The USERWRAPUP procedure is called automatically after the last audit record has been processed, just before the end of the task.

Variable Elements

[Table 10–3](#) presents a partial list of the variables you can use in the USERPROCEDURE and USERWRAPUP procedures. Refer to the examples later in this section for additional information on using these variables in your program.

Table 10–3. Variables Available to the USERPROCEDURE and USERWRAPUP Procedures

Variable	Type
AUDIT[n]	Array
AUDITBLK0[n]	Array
AUDITSN	Real
AUDITSNR	Real
AUDITSZ	Real

Table 10-3. Variables Available to the USERPROCEDURE and USERWRAPUP Procedures (cont.)

Variable	Type
AUDREC(n)	Word
AUDTYPE	Real
CONTROLRECORD	Boolean
PRINTIT	Boolean
USERAUDINX	Real

Audit Data Block Information

[Table 10-4](#) describes the fields in an audit data block. You can refer to these fields in an ALGOL program by using the format AUDIT [<field name>]. The AUDIT parameter refers to the array containing the image of the current audit block. See the variable AUDIT[n] in [Table 10-3](#).

Table 10-4. Audit Data Block Information

Field	Meaning
AUDITSERIALNUMF	Provides the ABSN associated with the audit block.
AUDRECSPLIT	Assigns the integer 1 if the audit record is split across more than one physical block.
LASTAUDRECSPLIT	Assigns the integer 1 if the last audit record is split across more than one physical block.
LASTAUDITCWF	Provides an index to the last control word in the audit block.
LASTUSEDRAW	Provides an index to the last word in the audit block.
MYAUDITBLOCKSZ	Provides the size of the audit block in words.
AUDCHKSUMLOC	Provides the location of the word containing the checksum value.
PREVAUDSEG	Provides the segment address of the previous audit block.
DMIOTIME(AUDIT)	Provides the amount of database input/output time used to generate the information in the audit block. The time is provided in units of 2.4 microseconds.
NONDMPTIME(AUDIT)	Provides the amount of nondatabase processor time used to generate the information in the audit block. The time is provided in units of 2.4 microseconds.
DMUPDPTIME(AUDIT)	Provides the amount of processor time used for database updates to generate the information in the audit block. The time is provided in units of 2.4 microseconds.

Table 10-4. Audit Data Block Information (cont.)

Field	Meaning
DMINQPTIME(AUDIT)	Provides the amount of processor time used for database inquiries to generate the information in the audit block. The time is provided in units of 2.4 microseconds.
MYTIMESTAMP(AUDIT)	Provides the time at which the last piece of information was written in the audit block. This value identifies the timestamp of the audit block.
PREVTIMESTAMP(AUDIT)	Provides the timestamp associated with the previous audit block.

Example 1

Use `AUDIT[AUDITSERIALNUMF]` to provide the ABSN associated with the audit block.

Example 2

Use `AUDIT[DMIOTIME(AUDIT)]` to provide the amount of database input/output time used to generate the information in the audit block.

Stopper Pattern Information

When an audit block other than the last block in a row is written to an audit file on disk, a stopper pattern is added to the end of the last audit file. This stopper pattern is used by the recovery process to find the end of a disk type audit. The available stopper pattern information is described in [Table 10-5](#). The `AUDIT` parameter refers to the array containing the image of the current audit block. See the variable `AUDIT[n]` in [Table 10-3](#).

Table 10-5. Stopper Pattern Information

Field	Meaning
STOPPERABSN(AUDIT)	Provides the ABSN of the previous audit block.
STOPPERTHEEND(AUDIT)	Marks the end of the audit file.
STOPPERDBTS(AUDIT)	Provides the database timestamp.
STOPPERMTS(AUDIT)	Provides the timestamp associated with the previous audit block.
AUDITBLOCKSZWITHSTOPPER	Provides the size of the audit block. This value takes into account the size of the stopper pattern.

Example

Use `STOPPERABSN(AUDIT)` to provide the ABSN associated with the previous audit block.

DMAuditLib Interface

All the information in the DMAuditLib interface array AUDIT_INFO is available to the USERPROCEDURE and USERWRAPUP procedures. For more information on DMAuditLib, refer to [Section 20, Using the Audit Reader Library Interface](#).

Using the SELECT Statement with the ALGOL Code

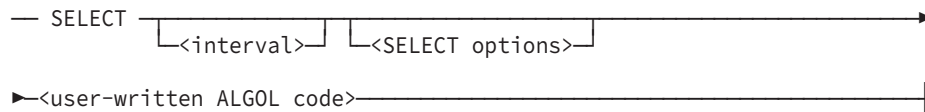
Introduction

Use the PRINTAUDIT *SELECT* command to process your ALGOL code and generate a tailored version of the PRINTAUDIT program.

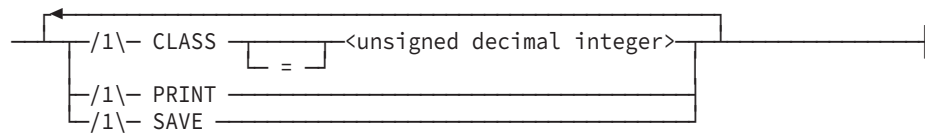
Syntax

The following diagrams illustrate the syntax for the SELECT command.

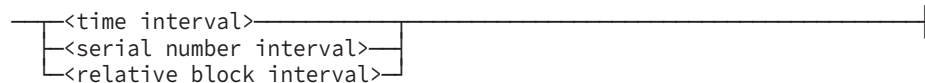
SELECT Command



<SELECT options>



<interval>



Explanation

The three main elements to the SELECT command are the interval, the SELECT options, and the userwritten ALGOL code. Samples of user-written ALGOL code are included in the examples later in the section.

Defining the Interval

Use the interval to identify the portion of the audit file that is to be examined. The three types of interval are as follows:

- Time interval
- Serial number interval

Printing, Viewing, and Extracting Audit Information

- Relative block interval

The purpose of these intervals and the exact syntax for defining any interval is discussed fully under “Designating Intervals” earlier in this section.

Defining the SELECT Options

The following three SELECT options are available.

Option	Meaning
CLASS	Denotes that the WFL job created to compile the tailored PRINTAUDIT program contains a WFL <i>CLASS</i> task attribute specification. Use the WFL <i>CLASS</i> task attribute specification to identify the queue in which the job is to run.
PRINT	Prints the WFL job, including the ALGOL code statements you supply.
SAVE	Saves the compiled, tailored version of the PRINTAUDIT program on disk. Once the program is saved, the program is initiated automatically. Note that the tailored PRINTAUDIT program is compiled, saved on disk, and then run. If you do not specify the SAVE option, the tailored PRINTAUDIT program is compiled and run without being saved on disk.

Examples of PRINTAUDIT Commands

Introduction

Three types of examples are provided as follows:

- Sample PRINTAUDIT commands
- Sample interactive PRINTAUDIT session
- Sample code for generating a tailored PRINTAUDIT program

Sample PRINTAUDIT Commands

The following examples illustrate correct PRINTAUDIT command syntax.

Example 1

This example command directs that all of the records from an audit file be printed in hexadecimal digits.

```
PRINT
```

Example 2

This example command prints the audit record information in hexadecimal digits. The command specifies only a 2-digit year for the date. If you specify a value in the range of 70 to 99 for a 2-digit year, the system interprets the 4-digit year to be 19xx. If you specify a value in the range 00 to 35 for a 2-digit year, the system interprets the 4-digit year to be 20xx.

```
PRINT TIME 11/15/95 8:23:36 TO 13:37:55 STR=17 RECTYPE=DSM
```

In this example, the information is limited to data that fits all of the following criteria:

- Audit record type data set modify (DSM) for structure 17
- Data generated between the time of 8:23:36 and 13:37:55 on November 15, 1995

Example 3

This example command prints the audit record information in hexadecimal digits. The command specifies a 4-digit year for the date. When you specify a 4-digit year, the year must be between 1970 and 2035.

```
PRINT TIME 11/15/2005 8:23:36 TO 13:37:55 STR=17 RECTYPE=DSM
```

Example 4

This example command displays on the terminal screen all the audit records for structure number 4. The audit record information is displayed as both hexadecimal digits and alphanumeric characters.

```
DISPLAY STR=4 ALPHA
```

Example 5

This example command displays on the terminal screen all audit records caused by the program with the mix number 3293. The audit record information selected is displayed as hexadecimal digits.

```
DISPLAY MIX=3293
```

Example 6

This example command prints all audit records caused by the program (PROJA)UPDATE/PRODDB ON SYSPACK. The audit record information selected is printed as both hexadecimal digits and alphanumeric characters.

```
PRINT PROG=(PROJA)UPDATE/PRODDB ON SYSPACK ALPHA
```

Example 7

This example command prints to a disk file all audit records in the current audit file that were created at or after 11:47:58 on November 15, 2005. The audit record information is written to disk as hexadecimal digits. No more than six lines of hexadecimal output is written for each audit record.

Printing, Viewing, and Extracting Audit Information

Note: Read the descriptions of 2- and 4-digit years for Examples 2 and 3.

```
EXTRACT TIME 11/15/2005 11:47:58 LINES=6 NOALPHA
```

Example 8

This example command displays all audit records that contain the value "SMITH" in the specified field of the structure called MASTER-DATA. The audit information selected is displayed as hexadecimal digits.

```
DISPLAY STR=MASTER-DATA FIELD [12,4]="SMITH"
```

Sample Interactive PRINTAUDIT Session

The following example illustrates the information displayed on your screen after you initiate an interactive PRINTAUDIT session.

In the example, the user enters the command `DISPLAY MIX = 3293` when prompted for a PRINTAUDIT request. To clarify the example, user input is shown in bold characters.

```
PLEASE ENTER REQUEST:
DISPLAY MIX = 3293

** REPORT ON CONSTRAINTS: **
  OUTPUT TO REMOTE
RANGE:
  0 999999999
AUDIT RECORD ABBREVIATIONS:
  ** ALL **
MAX LINES OF HEX DUMP PER AUDIT RECORD:
  99999
STACK NUMBERS:
  ** ALL **
MIX NUMBERS:
  3293
STRUCTURE NUMBERS:
  ** ALL **

RUNNING.
TITLE           =(ADD71)DATADICTIONARY/AUDIT174 ON DMS71
PACKNAME        =DMS71.
KIND            =PACK
MAXRECSIZE     =    900 WORDS
BLOCKSIZE      =    900 WORDS
AREASIZE       =    100 BLOCKS
AREAS          =    481
LASTRECORD     =    2
***** BLOCK           0 OF FILE   174 *****
AUDIT EOF=      2(000000000002)
AUDIT TIME STAMP = 11/05/2005 12:24:04.620
 0(0000) 00000000BA1E 000000000002 000454C2DA24 002400820005 000000000000
 5(0005) 000000000039 000000000000 190000000064 3AEE4F432510 000000000002
10(000A) 3BCD454C2DA5 000000000000 000000000000 000000000000 000000000000
15(000F) 000000000000 FOR 10 WORDS (2 LINES)
25(0019) 000000000000 000000000000 000000000000 000000000000 972E7610E20D
```

```
***** BLOCK                2 OF FILE 174 *****
**** SER= 476648(00000000BA20) LCW 593(0251) LWD 593(0251) SPLIT=0
**** MY TS=13:05:23.960(000492561269) PR TS=12:24:12.451(000454F4A36A)
RDSO  = 32 STR=   2SNR=(226) INX=   10(000A) SZ=   14
      DATETIMESTAMP = 11/05/2005 12:24:16.605
      3293/3293 (ADD71)SYSTEM/DATADICIONARY/MANAGER ON DMS71

0(0000) 226002000E20 000000000001 000000000000 3BCD4550F0DD 0CDD0CDD002E
5(0005) 4DC1C4C4F3F6 5DE2E8E2E3C5 D461C4C1E3C1 C4C9C3E3C9D6 D5C1D9E861D4
10(000A)C1D5C1C7C5D9 40D6D540C4D4 E2F3F64B0000 226002000E20

REQUEST COMPLETED
```

Examples Illustrating the Generation of Tailored Versions of the PRINTAUDIT Program

The following examples illustrate the use of the SELECT command for generating tailored versions of the PRINTAUDIT program.

Example 1

In this example, a tailored PRINTAUDIT program is generated to print all the Data Set Create (DSC) audit records for structure number 5. In the USERWRAPUP procedure, the program

- Generates a count of the audit records that are printed
- Prints the total number of audit records in the file
- Prints the average block size of the audit records in the file

Following the WFL job is a sample of the type of information that might be generated by the compilation of the tailored PRINTAUDIT program.

```
BEGIN JOB RUNPRINTAUDIT;
USER = MY;
BDNAME = BD/UCF/86000795/50;
CLASS = 10;
FAMILY DISK = MYPACK OTHERWISE PACK1;
RUN SYSTEM/PRINTAUDIT;
FILE AUDIT (KIND=PACK, TITLE=(MY)ALLSETDB/AUDIT1 ON MYPACK);
FILE DASDL (KIND=DISK, TITLE=(MY)DESCRIPTION/ALLSETDB ON MYPACK);
FILE SOURCE (TITLE=DATABASE/PRINTAUDIT);
FILE PROPERTIES (TITLE=DATABASE/PROPERTIES);
DATA CARD
SELECT %
PRINT SAVE CLASS=10 %
NOALPHA

%%% START OF USER SPECIFIED DECLARATIONS %%%
%%% SELECT OPTIONS ENDED WITH NOALPHA %%%

INTEGER COUNTEMUP,TOTALCOUNT,MEANBLOCKSZ ; % USER VARIABLES

PROCEDURE USERPROCEDURE; % FOR SELECTING AUDIT RECORDS (REQUIRED)
BEGIN % THIS EXAMPLE LOOKS FOR DSC RECORDS
```

Printing, Viewing, and Extracting Audit Information

```

                                % IN STRUCTURE 5 AND PRINTS THEM
PRINTIT := FALSE;             % DO NOT PRINT AUDIT RECORD
IF AUDITSN = 5 THEN          % FOUND STRUCTURE 5
IF AUDTYPE = DSC THEN        % FOUND DATA SET CREATE
BEGIN
    COUNTEMUP := * + 1; % COUNT DSC RECORDS
    PRINTIT := TRUE;    % PRINT THIS AUDIT RECORD
END;
TOTALCOUNT := * + 1;      % COUNT ALL AUDIT RECORDS

MEANBLOCKSZ := * +
(AUDIT [MYAUDITBLOCKSZ]    % USE DEFINE OF DATABASE/PROPERTIES
- MEANBLOCKSZ)/TOTALCOUNT;

END USERPROCEDURE;

PROCEDURE USERWRAPUP;        % USER-SPECIFIED FINAL PROCEDURE
BEGIN                       % EXAMPLE PRINTS RECORD COUNTS
    WRITE(PRINTER
, <J7, " DATA SET CREATE RECORDS FOUND (STRUCTURE 5), "
, "TOTAL RECORDS LOOKED AT = ", J7
, ", AVERAGE AUDIT BLOCK SIZE = ", J7>,
    COUNTEMUP, TOTALCOUNT, MEANBLOCKSZ );
    IF TAPE.OPEN THEN DISPLAY("*** AUDIT STILL OPEN IN WRAPUP");
    WRITE(PRINTER, <"*** END OF OUTPUT ***">);
END USERWRAPUP;

PROCEDURE EXAMPLE_USER_PROCEDURE; % (OPTIONAL)
BEGIN % EXAMPLE OF A USER DECLARED PROCEDURE
    % DISPLAYS STARTUP AND AUDIT STATE MESSAGES
    % AND INITIALIZES USER DECLARED VARIABLES.

    DISPLAY("*** CUSTOMIZED PRINTAUDIT EXAMPLE ***");
TOTALCOUNT:=MEANBLOCKSZ:=COUNTEMUP:=0; % RESET RECORD COUNTS

END EXAMPLE_USER_PROCEDURE;
%%% FIRST STATEMENT OF INNER BLOCK OF PRINTAUDIT %%%
%%% PLACE TO PUT USER SETUP AND INITIALIZATION %%%
EXAMPLE_USER_PROCEDURE; % CALL USER PROCEDURE (OPTIONAL)

DISPLAY("*** START OF SELECT ***");
%%% SELECT CODE GENERATED BY PRINTAUDIT FOLLOWS %%%
?END JOB;
```

Following is the information generated by the compilation of the tailored PRINTAUDIT program. The output from the compilation includes a copy of the code used to generate the program.

```
** REPORT ON CONSTRAINTS: **
    OUTPUT TO PRINTER
RANGE:
    0 999999999
AUDIT RECORD ABBREVIATIONS:
    ** ALL **
MAX LINES OF HEX DUMP PER AUDIT RECORD:
    99999
```



```

STACK NUMBERS
  ** ALL **
MIX NUMBERS:
  ** ALL **
STRUCTURE NUMBERS:
  ** ALL *
  ?BEGIN JOB PRTAUD5978;
CLASS = 0010;
COMPILE PRINTAUDIT/PROG/59755978 WITH DMALGOL LIBRARY;
FILE AUDIT(TITLE = (MY)ALLSETSDB/AUDIT1 ON MYPACK
           , PACKNAME=MYPACK
           , KIND=DISKPACK);

COMPILER FILE PROPERTIES(TITLE = *DATABASE/PROPERTIES ON DISK
                        , PACKNAME=DISK
                        , KIND=DISKPACK);
COMPILER FILE INTL(TITLE = *SYMBOL/INTL/ALGOL/DMS/PROPERTIES ON DISK
                  , PACKNAME=DISK
                  , KIND=DISKPACK);
COMPILER FILE TAPE(TITLE = *DATABASE/PRINTAUDIT ON DISK
                  , PACKNAME=DISK
                  , KIND=DISKPACK);
COMPILER DATA CARD
INTEGER COUNTMUP,TOTALCOUNT,MEANBLOCKSZ ; % USER VARIABLES
PROCEDURE USERPROCEDURE; % FOR SELECTING AUDIT RECORDS (REQUIRED)
BEGIN % THIS EXAMPLE LOOKS FOR DSC RECORDS
      % IN STRUCTURE 5 AND PRINTS THEM
      PRINTIT := FALSE; % DO NOT PRINT AUDIT RECORD
      IF AUDITSN = 5 THEN % FOUND STRUCTURE 5
      IF AUDTYPE = DSC THEN % FOUND DATA SET CREATE
      BEGIN
        COUNTMUP := * + 1; % COUNT DSC RECORDS
        PRINTIT := TRUE; % PRINT THIS AUDIT RECORD
      END;
      TOTALCOUNT := * + 1; % COUNT ALL AUDIT RECORDS
      MEANBLOCKSZ := * +
        (AUDIT [MYAUDITBLOCKSZ] % USE DEFINE OF DATABASE/PROPERTIES
         - MEANBLOCKSZ)/TOTALCOUNT;
END USERPROCEDURE;
PROCEDURE USERWRAPUP; % USER-SPECIFIED FINAL PROCEDURE
BEGIN % EXAMPLE PRINTS RECORD COUNTS
  WRITE(PRINTER
        ,<J7," DATA SET CREATE RECORDS FOUND (STRUCTURE 5), "
        ,"TOTAL RECORDS LOOKED AT = ", J7
        ," , AVERAGE AUDIT BLOCK SIZE = ", J7>,
        COUNTMUP, TOTALCOUNT, MEANBLOCKSZ );
  IF TAPE.OPEN THEN DISPLAY("*** AUDIT STILL OPEN IN WRAPUP");
  WRITE(PRINTER,<"*** END OF OUTPUT ***">);
END USERWRAPUP;
PROCEDURE EXAMPLE_USER_PROCEDURE; % (OPTIONAL)
BEGIN % EXAMPLE OF A USER DECLARED PROCEDURE
      % DISPLAYS STARTUP AND AUDIT STATE MESSAGES
      % AND INITIALIZES USER DECLARED VARIABLES.
      DISPLAY("*** CUSTOMIZED PRINTAUDIT EXAMPLE ***");
      TOTALCOUNT:=MEANBLOCKSZ:=COUNTMUP:=0; % RESET RECORD COUNTS
END EXAMPLE_USER_PROCEDURE;

```

Printing, Viewing, and Extracting Audit Information

```
%%% FIRST STATEMENT OF INNER BLOCK OF PRINTAUDIT %%%
%%% PLACE TO PUT USER SETUP AND INITIALIZATION   %%%
EXAMPLE_USER_PROCEDURE;          % CALL USER PROCEDURE (OPTIONAL)
DISPLAY("*** START OF SELECT ***");

%%% SELECT CODE GENERATED BY PRINTAUDIT FOLLOWS %%%
BEGIN
LOWER:= 000000000000; UPPER:= 000999999999; REELSWITCHOK:= FALSE;
LOWERLREC:=FALSE; UPPERLREC:=FALSE;
ALPHADUMP:= TRUE;
HEXLN:= 000000099999;
END;
$ POP VOIDT SEQ
?
IF T IS COMPILEDOK THEN
  RUN PRINTAUDIT/PROG/10191021;
END JOB.
```

Example 2

The following example generates a tailored PRINTAUDIT program that prints every quiet point in an audit file, and displays the ABSN and word offset of the last quiet point.

```
BEGIN JOB PRINT/AUDIT;

  RUN SYSTEM/PRINTAUDIT;
    FILE AUDIT (KIND=PACK, TITLE=CLASSDB/AUDIT1);
    FILE DASDL (TITLE=DESCRIPTION/CLASSDB);

  DATA CARD
    SELECT PRINT SAVE ALPHA

    %%%% BEGIN ALGOL PORTION %%%%

    REAL LASTQPABSN,
    LASTQPOFFSET;

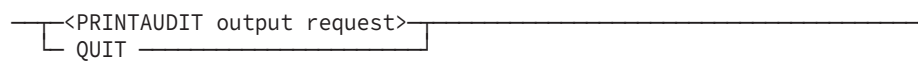
  PROCEDURE USERPROCEDURE;
  BEGIN
    PRINTIT:=FALSE;
    IF CONTROLRECORD
      OR (AUDTYPE = BTR AND AUDREC(2) = 1) THEN
      BEGIN
        PRINTIT      :=TRUE;
        LASTQPABSN   :=AUDIT[0];
        LASTQPOFFSET:=USERAUDINX;
      END;
  END USERPROCEDURE;

  PROCEDURE USERWRAPUP;
  BEGIN
    DISPLAY ("LAST QUIET POINT AT ABSN = " CAT
      STRING(LASTQPABSN,*) CAT ", OFFSET = "
      CAT STRING(LASTQPOFFSET,*));
  END USERWRAPUP;
?END JOB PRINT/AUDIT;
```

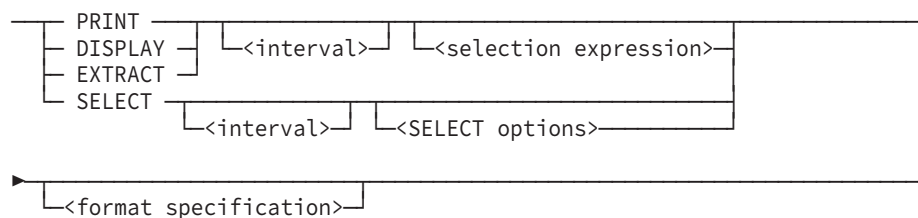
Quick-Reference Information

The information presented here is for quick-reference purposes only. For an explanation of any element of a syntax diagram, refer to the appropriate information presented earlier in this section.

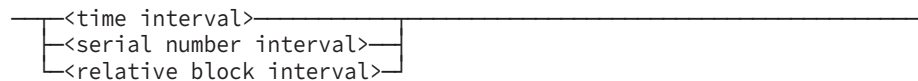
PRINTAUDIT Statement



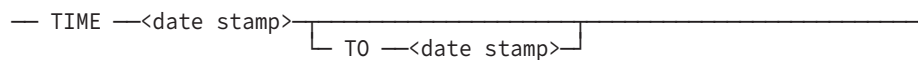
<PRINTAUDIT output request>



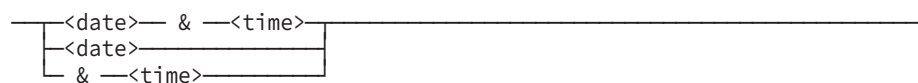
<interval>



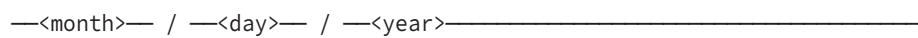
<time interval>



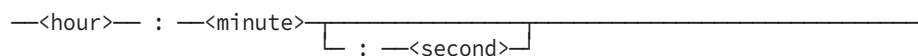
<date stamp>



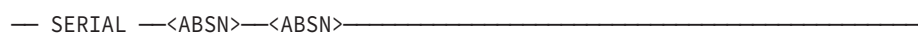
<date>



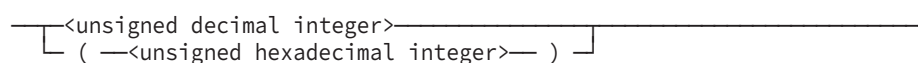
<time>



<serial number interval>



<ABSN>



<relative block interval>

—<block number>—<block number>—————|

<block number>

—<unsigned decimal integer>—————|
 — (—<unsigned hexadecimal integer>—) —|
 * —————|
 * — — —<unsigned decimal integer>—————|

<selection expression>

—<stack number selection>—————|
 —<mix number selection>—————|
 —/1\—<program identifier selection>——|
 —<structure selection>—————|
 —/1\—<field selection>—————|
 —/1\—<record type selection>—————|

<stack number selection>

— STACK — [=] —<stack number>— , —<stack number>—|

<stack number>

—<unsigned decimal integer>—————|
 — (—<unsigned hexadecimal integer>—) —|

<mix number selection>

— MIX — [=] —<mix number>— , —<mix number>—|

<mix number>

—<unsigned decimal integer>—————|

<program identifier selection>

— PROGRAM — [=] —<file title>—————|

<structure selection>

— STRUCTURE — [=] —————>|
 —<structure identifier>— , —<structure identifier>—|
 — BLOCK — [=] —<block specification>—|
 — BAF — [=] —————|

<structure identifier>

—<structure name>—————|
 —<structure number>—————|

<structure name>

—<identifier>—

<structure number>

—<unsigned decimal integer>—

<block specification>

—<block number> — , —<block number> —
 [—<block number> —]
 WAF [=] <waf>

<block number>

—<unsigned decimal integer>—
 [(—<unsigned hexadecimal integer>—)]

<waf>

—<unsigned decimal integer>—
 [(—<unsigned hexadecimal integer>—)]

<field selection>

— FIELD — [—<word offset> — , —<digit offset>—] [=]
 ▶<value>—

<word offset>

—<unsigned decimal integer>—

<digit offset>

—<unsigned decimal integer>—

<value>

—<alpha string literal>—
 [<hexadecimal code>— " —<hexadecimal string>— "]

<record type selection>

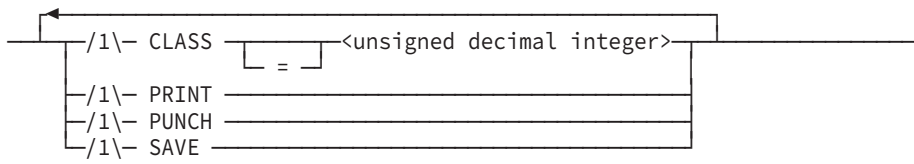
— RECTYPE [=] —<audit record type mnemonic>—

SELECT Command

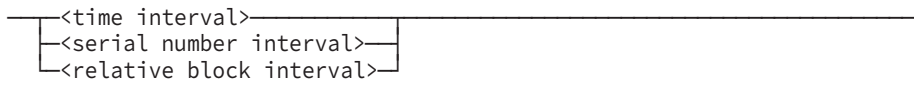
— SELECT [<interval>] [<SELECT options>]

Printing, Viewing, and Extracting Audit Information

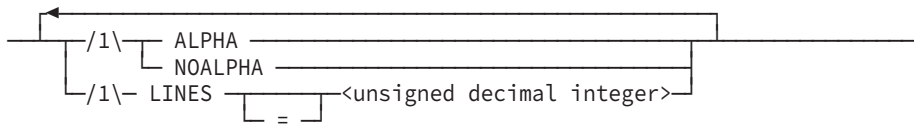
<SELECT options>



<interval>



<format specification>



Section 11

Checking Integrity and Performance

This section describes the Database Certification. Database Certification provides a means of checking for file and structure integrity.

This section also describes the following DMUTILITY statements: DBDIRECTORY, DISABLE, ENABLE, LIST, and WRITE.

Note: *The tasks identified in this section can be initiated through Database Operations Center.*

Database Certification

Database Certification is a utility program that ensures the integrity of structures in an Enterprise Database Server database. The alias name is treated the same as a data structures. The Database Certification utility program is compiled as SYSTEM/DBCERTIFICATION. The utility can then be used interactively from a remote terminal or initiated as a batch job. In either case, free-format user input commands are used to determine the level of certification to be provided for a structure.

Three levels of certification are provided by the utility.

- Physical integrity. This level ensures that a file is physically intact and accessible, and isolates problems at the data block level.
- Internal (intrastructure). This level verifies that data in a single structure is consistent, storage control information is valid, and internal control information is valid.
- Crosschecking (interstructure). This level verifies that relationships between data structures are correct.

Database Certification enables the user to specify the certification tests at the various levels. Except for the READONLY test from the Physical Integrity level, all certification request tests are typically performed while a database is not in use. By default, Database Certification has exclusive use of the database. While certification is in progress, any program that attempts to access the database is locked out. However, provisions can be made to allow users to operate Database Certification in a nonexclusive mode.

The following paragraphs contain instructions for operating the Database Certification program. An output description is included in this section.

Running Database Certification

Database Certification is run interactively from a remote terminal by using the Command and Edit (CANDE) system, or as a batch job through a Work Flow Language (WFL) file.

The Database Certification program uses the tailored support library DMSUPPORT/<database name>. When this library is not present, the Database Certification program is suspended and the following message is displayed on the terminal screen:

```
NO FILE DMSUPPORT/<database name>
```

Interactive Mode

In interactive mode, all input commands are entered at the remote terminal. Each command is executed as it is entered. Operation of the Database Certification program in the interactive mode consists of the following steps:

1. Log on to CANDE.
2. Enter the following statements:

```
RUN $SYSTEM/DBCERTIFICATION;  
FILE DASDL(TITLE=DESCRIPTION/<database name>  
ON <family name>);
```

Note: *The Data and Structure Definition Language (DASDL) file must be equated to the current description file for the database, because Database Certification reads and interprets the DASDL description file.*

Database Certification displays a number sign (#) to indicate that the program is ready for additional input.

Typically, commands are entered on a single line; however, some commands can exceed the line width of the terminal screen. To enter a multiline command, insert a percent sign (%) at the end of the line to indicate that more input follows, and then transmit. Database Certification responds with a number sign and a percent sign (#%). Additional input can then be entered. The process can be repeated as many times as necessary. When the command input is completed, the last line is transmitted without a percent sign and the entire command is executed.

Several commands can be entered in a single input message if each complete command is terminated by a semicolon (;).

When Database Certification is run interactively, the output is directed to the terminal and the line printer. The Database Certification program interrupts the output and displays the word PAGE on the terminal after displaying a full page of output. The program then waits for the user to transmit one or more blanks before displaying the next page. If a nonblank character is transmitted, the REMOTEOUT option is assigned the value RESET and the remaining output is directed to the line printer.

The REMOTEOUT option can be assigned the value RESET at any time with the OPTIONS command.

Batch Mode

In batch mode, all input commands are entered as a batch job. The Database Certification output is transferred to a printer file with the internal name LINE.

Operation of the Database Certification program in batch mode consists of creating a card deck or JOBSYMBOL file. The following WFL statements can be used to run Database Certification:

```
? BEGIN JOB DBCERTIFICATION;
  RUN SYSTEM/DBCERTIFICATION;
  FILE DASDL(TITLE=DESCRIPTION/<database name>
    ON <family name>);
  DATA CARD
    <one or more DBCERTIFICATION input statements>
? END JOB
```

Note: The DASDL file must be equated to the current description file for the database, because Database Certification reads and interprets the DASDL description file.

When Database Certification commands are entered from cards or disk, the command information must not extend beyond the first 72 characters of each record. The remaining columns are reserved for sequence numbers or comments. A command can extend over as many records as desired; however, a word must not be split between two records.

Every command must end with a semicolon (;). Several commands can be placed on a single input record, provided that each command is delimited by a semicolon.

A percent sign (%) in the input record denotes a comment. Anything between the percent sign and the end of the record is ignored, unless the percent sign is enclosed in quotation marks. For example, in "ABC%DE" the percent sign is interpreted as part of the quoted string. The Database Certification program also returns a TASKVALUE indicating whether there were any warnings or errors reported in the certification process. If there were errors, the TASKVALUE would be passing the value -1. If there were warnings, the TASKVALUE would be assigned the value 2. This TASKVALUE can be read by a WFL job. The following WFL job example certifies a database and checks the result before updating the database.

```
? BEGIN JOB CERTIFYBEFOREUPDATE;
  TASK T;
  RUN SYSTEM/DBCERTIFICATION [T];
  FILE DASDL(TITLE=DESCRIPTION/TESTDB ON DMS35);
  DATA CARD
    CERTIFY ALL;
? %END OF DATA CARD
  IF T(TASKVALUE) LESS 1
  THEN
  BEGIN
    DISPLAY "WARNING: CERTIFICATION ERRORS REPORTED.";
    GO TO EXITJOB;
  END;
  ELSE DISPLAY "OK TO UPDATE.";
%UPDATING DATABASE ...
```

```
EXITJOB:  
? END JOB
```

Restarting Database Certification

Database Certification can be restarted if it is discontinued, or if a halt/load occurs during a certification run. To restart the program, reenter the job.

The Database Certification program checks for the presence of `DBCERTIFICATION/RECOVERY/<database name>` at the beginning of each run. If the recovery file is present, then the previous job ended abnormally and the program attempts to read the recovery file. A valid recovery file indicates where processing is to resume. Database Certification finishes certifying the structure being processed at the time the program discontinued or a halt/load occurred. Database Certification then goes to the end of the task.

An invalid recovery file causes a message to be displayed that instructs you to reenter the entire job.

The `DMUTILITY CANCEL` function must be used to unlock the control file if the aborted run of Database Certification was an exclusive-use run and the program is not going to be restarted. Unless the control file is unlocked, access to the database is denied to other users.

Output Description of Database Certification

The output from Database Certification is hardcopy analysis listings. These listings include appropriate error messages, addresses of detected errors, and a hexadecimal dump of the blocks where errors were detected.

Users executing the program from a remote terminal receive only the appropriate error messages at the terminal. The error messages, hexadecimal dumps, and error addresses are also provided in the hard copy listings generated at the end of the session. If the Database Certification job discontinues or a halt/load occurs before the end of the session, the printer file is saved. This printer file is printed only if Database Certification is restarted and allowed to complete.

The user can assign the value `RESET` to the `DUMPBLOCKS` option (refer to “`OPTIONS Command`” later in this section) in order to suppress the printing of the hexadecimal dump output. The option can be used in either batch or interactive mode.

Example

This example shows a partial Database Certification report in which the data set is sectioned, and the report specifies the section that is being certified.

DBCERTIFICATION SSR 51.1 (51.116.0017)
MONDAY, DECEMBER 8, 2005, 6:18 PM.
DASDL:(NAGEL)DESCRIPTION/FORSDB ON DMTEST
CERTIFICATION FOR DATABASE FORSDB

INPUT SPECIFICATIONS:
CERTIFY ALL;
STRUCTURE D # 4 DISJOINT STANDARD DATA SET
SECTION 0 of D

NO PHYSICAL ERRORS DETECTED
NO STRUCTURE CHECK ERRORS DETECTED
NO AVAILABLE SPACE ERRORS DETECTED
NO CONTENTS ERRORS DETECTED
NO VERIFYSTORE ERRORS DETECTED

ANALYSIS TIMES:
PROCESS 00:00:00.65
IO 00:00:01.42
ELAPSED 00:00:07.10

DBCERTIFICATION SSR 51.1 (51.116.0017)
MONDAY, DECEMBER 8, 2005, 6:18 PM.
DASDL:(NAGEL)DESCRIPTION/FORSDB ON DMTEST
CROSSCHECKING CERTIFICATION FOR DATABASE FORSDB

CROSSCHECKING STRUCTURE E #E5 AND:

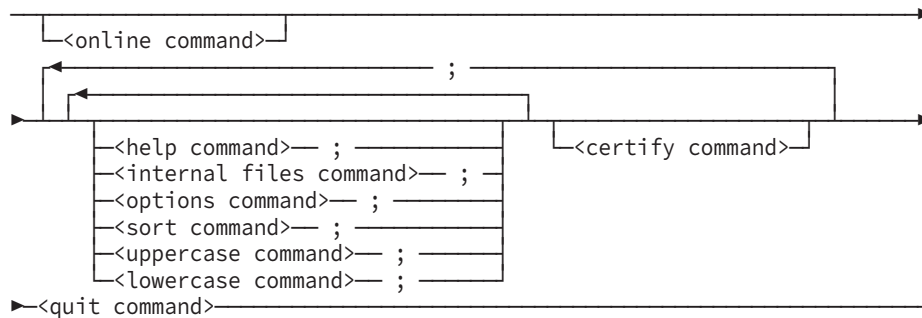
D #4 FOR EMBEDDED STRUCTURE TO ITS OWNER CROSSCHECKING
=====
====> NO ERRORS DETECTED

ANALYSIS TIMES:
PROCESS 00:00:00.58
IO 00:00:01.20
ELAPSED 00:00:04.77

DBCERTIFICATION Command

The DBCERTIFICATION command allows you to initiate and control the certification of the database.

Syntax



Explanation

Following are the syntax and explanation for each command used in operating the Database Certification program. The applicable syntax options, program examples, and extent of each command are included in this section. Commands other than the CERTIFY command, such as the SORT and OPTIONS commands, affects only the CERTIFY command that they precede in the command input.

ONLINE Command

The ONLINE command allows access to the database by any valid user during the time that Database Certification is in progress.

Syntax

— ONLINE _____|

Explanation

The ONLINE command allows access to the database while Database Certification is in progress. The ONLINE command must be the first Database Certification command entered when exclusive use of the database is not required during certification. When ONLINE is not specified, the database is locked and can be accessed only by the Database Certification program.

ONLINE should be specified only if the structure being certified is not updated while Database Certification is running. This means, for example, that the following Enterprise Database Server verbs should not be used: STORE, DELETE, INSERT, REMOVE, and ASSIGN. If a structure being certified is modified, false certification errors can result.

HELP Command

The HELP command displays the syntax for a specified DBCERTIFICATION input command.

Syntax

— HELP —<dbcertification command> _____|
 └<command variable>┘

Explanation

The following information explains the elements of the HELP command syntax diagram.

Option	Explanation
<dbcertification command>	Composed of the following keywords: CERTIFY, INTERNAL FILES, OPTIONS, QUIT, UPPERCASE, LOWERCASE, or SORT.

Option	Explanation
<command variable>	Includes any of the following DBCERTIFICATION command variables: <certify command>, <internal files command>, <options command>, <quit command>, and <sort command>.

Examples

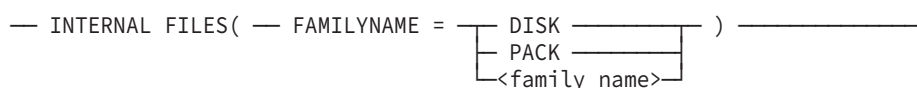
- HELP CERTIFY
This command causes Database Certification to display the syntax for the CERTIFY command.
- HELP <certify command>
This command causes Database Certification to display the syntax for the <certify command> construct.

INTERNAL FILES Command

The INTERNAL FILES command must precede the CERTIFY command and is in effect only for the one CERTIFY command that follows.

The INTERNAL FILES command specifies a default family name where the temporary extracted data files are to be stored. If an INTERNAL FILES specification is not declared, the family name of the database control file is assumed. An INTERNAL FILES command is used when a family name is not specified in the SORT command.

Syntax



Explanation

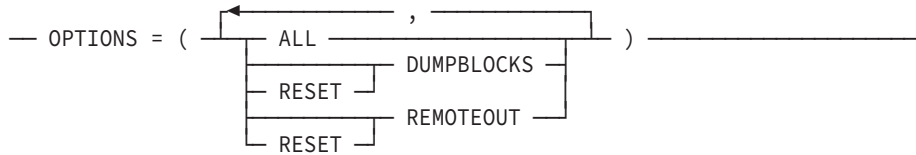
The INTERNAL FILES command indicates that file-storage specifications follow. The FAMILYNAME specification indicates that the file-storage family name follows. The DISK specification indicates that the internal files are to be stored on FAMILYNAME DISK. The PACK specification indicates that the internal files are to be stored on FAMILYNAME PACK. The <family name> construct indicates that the internal files are to be stored on FAMILYNAME <family name>.

An internal file is created for every data set, set, and subset structure that is certified. To calculate an internal file size, follow the rule that the file might contain four words plus the key length for every record in the database structure. So, if you have a structure with 4,000,000 records, and the key length is two words, the internal file is about (4 + 2) * 4,000,000 or 24,000,000 words long.

OPTIONS Command

The OPTIONS command specifies whether the output contains error data and whether the output is displayed on the remote terminal, the line printer, or both.

Syntax



Explanation

The following information explains the elements of the OPTIONS command syntax diagram.

Option	Explanation
ALL	Specifies both the DUMPBLOCKS and REMOTEOUT options.
DUMPBLOCKS	Causes hexadecimal dump printing of data blocks where errors have been detected during the certification process. The DUMPBLOCKS option is set by default.
REMOTEOUT	Causes error messages to be displayed at the terminal as well as printed on the line printer. The REMOTEOUT option is set by default. When this option is assigned the value RESET, the REMOTEOUT option restricts the error output to the line printer. The last active REMOTEOUT setting in the user input is the request that is used by the Database Certification program.
RESET	Sets DUMPBLOCKS or REMOTEOUT to FALSE.

SORT Command

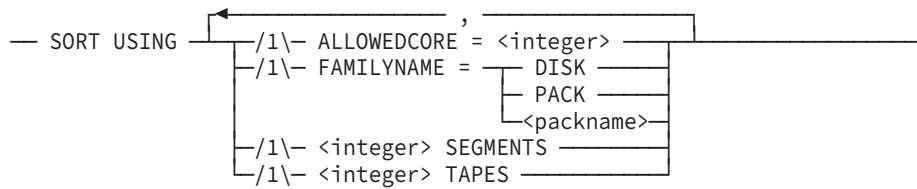
The Database Certification program uses the SORT intrinsic to place the temporary extracted data files in the proper order for internal and crosscheck certification. SORT phrases allow the user to specify the resources (disk, pack, or tape) to be used by the SORT intrinsic.

If a SORT command is not given, Database Certification uses the following default values:

- The ALLOWEDCORE value is set to 12000.
- The family name is set to DISK.
- The SEGMENT value is set to the physical file size multiplied by 2.25.
- The number of tapes is set to zero (0).

Any user-supplied sort option or options can override each corresponding default sort option.

Syntax



Explanation

The following information explains the elements of the SORT command syntax diagram.

Option	Explanation
SORT USING	Initiates the SORT command.
ALLOWEDCORE	Specifies the amount of sort core that is to be used by the SORT intrinsic. If ALLOWEDCORE is not specified, 12000 words is assumed.
FAMILYNAME	Specifies the pack family where the internal sort files are to be maintained. You can specify DISK, PACK, or a pack name. If FAMILYNAME is not specified, then the family name declared in the INTERNAL FILES command is assumed by default. If an INTERNAL FILES specification is not declared, the family name of the database control file is assumed.
SEGMENTS	Specifies the number of segments to be sorted by the SORT intrinsic. The default setting is 2.25 multiplied by the physical file size.
TAPES	Specifies the number of tapes to be used by the SORT intrinsic. The default is 0 TAPES.

UPPERCASE Command

The UPPERCASE command allows you to specify translation of message output to uppercase.

Syntax



Explanation

By default, the text is lowercase. If uppercase is desired then enter *UPPERCASE*. To return to lowercase output, enter *LOWERCASE*.

LOWERCASE Command

The LOWERCASE command allows you to specify translation of message output to lowercase.

Checking Integrity and Performance

Syntax

— LOWERCASE —————|

Explanation

By default, the text is translated to lowercase. If uppercase is desired then enter *UPPERCASE*. To return to lowercase output, enter *LOWERCASE*.

QUIT Command

The QUIT command is used to terminate the Database Certification program.

Syntax

— QUIT —————|
— BYE —————|
— END —————|
— STOP —————|

Explanation

QUIT, BYE, END, STOP are keywords that terminate the Database Certification program.

CERTIFY Command

The CERTIFY command specifies the data structures that are to be verified by the Database Certification program, and the certification options used to test them.

Syntax

— CERTIFY — ALL —————|
 | <structure list> | | <certify options> |

<structure list>

— <structure name> , <structure name> —————|
 | . <partition ID> —————|
 | <structure number> —————|
 | . <partition ID> —————|
 | <structure number> - (hyphen) - <structure number> —————|

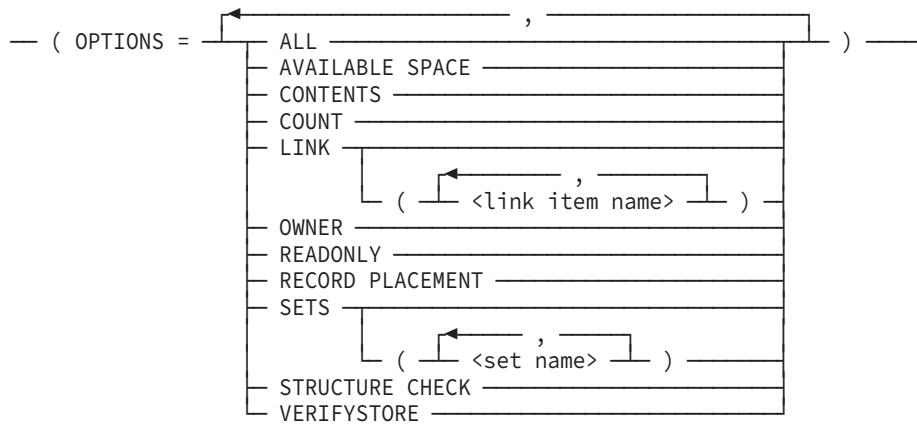
<link item name>

— <simple identifier> —————|

<set name>

— <simple identifier> —————|

<certify options>



If a CERTIFY command is not preceded by a SORT command, Database Certification uses the default SORT command options.

Explanation

The CERTIFY command is a keyword indicating that the CERTIFICATION option is requested for the structures that follow. Following is a brief description of the CERTIFY commands and constructs.

Option	Explanation
CERTIFY ALL	Selects every structure in the database for verification.
<structure name>	Represents the name of a structure in the database as specified in DASDL.
<partition ID>	Represents the value of the key that selects a partition of the database.
<structure number>	Represents the integer number of a structure as assigned by the DASDL compiler.
<certify options>	The following options are available with the CERTIFY command. The syntax is OPTIONS = <command>
ALL	This option selects all certification options as follows: AVAILABLE SPACE, CONTENTS, RECORD PLACEMENT, STRUCTURE CHECK and VERIFYSTORE are the internal (intrastructure) certification options. COUNT, LINK, OWNER, and SETS are the crosschecking (interstructure) certification options. READONLY is the physical integrity certification option.

Checking Integrity and Performance

Option	Explanation
AVAILABLE SPACE	This is an internal (intrastructure) certification option. It verifies the available-space control information and DKTABLE consistency in both standard and standard variable format data sets. The certification that is performed depends on the structure.
CONTENTS	This is an internal (intrastructure) certification option. It verifies that control items are within acceptable ranges (for example, that link addresses in a data set do not exceed the bounds of the referenced structure). The CONTENTS option also checks duplicate resolver words. The certification that is performed depends on the data structure.
COUNT	This is a cross-checking (interstructure) certification option. It verifies that the number of counted links in the structure being certified equals the value stored in the referenced record-count item field.
LINK	This is a cross-checking (interstructure) certification option. It initiates checks for all link items that are declared in the selected structures. Counted links are not included in the checks. Use the COUNT option to certify counted links. You can check specific links by entering a link item name for each link to be tested.
OWNER	This is a cross-checking (interstructure) certification option. It performs checks to ensure that the mapping between an embedded structure and its owner is correct.
READONLY	This is a physical integrity certification option. It provides checking for checksum and addresscheck words and for DMROWLOCK. This option must be specified only when the physical integrity checks are the only checks the user requires.
RECORD PLACEMENT	This is an internal (intrastructure) certification option. It performs checks to ensure that records in the structure are in the proper location based on key values, hash checks, or record directory words.
SETS	<p>This is a cross-checking (interstructure) certification option. It performs checks to ensure that the mapping between a set and the data set it spans is valid. Specific sets are checked by entering a set name for each set to be tested.</p> <p>You can specify an alias name for a set name. This means that you can refer to a set name using 16-bit character structure names through COBOL85 programs and Enterprise Database Server utility programs.</p> <p>For more information about alias names, refer to the <i>Data and Structure Definition Language (DASDL) Programming Reference Manual</i>.</p>

Option	Explanation
STRUCTURE CHECK	<p>This is an internal certification option. It provides internal structure checking. The checks can vary with the structure and can include:</p> <ul style="list-style-type: none"> • Check of control words in block 0 • Data control word checks (for compact data sets) and block control word checks (for ordered data sets) • Table control word checks in index sets • Data block and table block chain checking • Empty block checking • Check of control information in the last record of the file <p>The STRUCTURE CHECK option is implicitly set when an internal certification option (AVAILABLE SPACE, CONTENTS, RECORD PLACEMENT, STRUCTURE CHECK, VERIFYSTORE) is specified.</p>
VERIFYSTORE	<p>This is an internal (intrastructure) certification option. It ensures that every valid record satisfies the test conditions.</p>

Examples

- CERTIFY ALL
 This statement causes Database Certification to perform all certification checks for all structures.
- CERTIFY 5-16(OPTIONS=AVAILABLE SPACE,CONTENTS,RECORD PLACEMENT,% STRUCTURE CHECK,VERIFYSTORE)
 This example causes Database Certification to perform all the internal checks for structures 5 through 16.
- CERTIFY 7(OPTIONS=AVAILABLE SPACE, CONTENTS)
 This example causes Database Certification to perform available space checking and contents checking for structure 7. The structure check and readonly checks are also provided.
- OPTIONS = (RESET DUMPBLOCKS); SORT USING 30000 SEGMENTS,% ALLOWEDCORE = 20000; CERTIFY 9(OPTIONS=STRUCTURE CHECK)
 This example causes Database Certification to perform the structure check option for structure 9. Readonly checks are also provided. Hexadecimal dumps are not provided in the output. Sorting of the extracted files is given 20,000 words of core, and the sort mode is disk using 30,000 segments.

Restrictions

The following restrictions apply to the use of the CERTIFY command:

- Partitioned structures cannot be cross-checked.
- The crosscheck options that can be selected for disjoint data sets are LINK, COUNT, and SETS. The options that can be selected for embedded data sets are LINK, COUNT,

SETS, and OWNER. Functions performed by these options are described in “Certify Options for Structure Types” later in this section.

- The OWNER option is used with embedded structures only when certifying ordered data sets, index sequential sets, ordered list sets, and unordered list sets.
- For structures using the DATAENCRYPT option, if encrypted data items are used as key items in a set with logical or physical sections, DBCERTIFICATION will not cross-check the section number. Currently, DBCERTIFICATION extracts keys from set entries and uses the keys to search the section table to retrieve a section number. Since encrypted key items are represented as hashed values, searching the section table will not work. For this reason, cross-checking is skipped for structures that contain encrypted key items and logical or physical sections.

CERTIFY Options for Structure Types

The tables on the following pages contain the valid options and certification functions that can be specified in the CERTIFY command for data sets and sets. Options for data sets and sets are grouped separately. Alphabetic listings of the Enterprise Database Server structure types and valid certification keywords for the structures are included in each section.

The type of valid syntax options and certification functions depend on the structure being certified. A brief description of the specific checks made for each option is included in each table.

The function of the various control fields, words, records, and blocks that the Database Certification program checks in a structure are not described. The *Data and Structure Definition Language (DASDL) Programming Reference Manual* provides this information.

The READONLY option performs the same function on all data structures. This option verifies the checksum word, the addresscheck word, and performs a DMROWLOCK test.

Data Sets

A data set is a collection of related data records that are stored in a file on a random-access storage device. A data set is similar to a conventional file in that it contains data items and has logical and physical properties similar to those of files. However, unlike conventional files, data sets can contain other data sets, sets, and subsets.

Compact Data Sets

Compact data set records vary in size because the data set items vary in size or in number of occurrences, or are stored conditionally. Records in compact data sets are not maintained in logical order. Compact data sets can be disjoint or embedded.

The certification options for compact data sets and the functions that they perform are described in the following table.

Option	Function
AVAILABLE SPACE	<p>Ensures that key entries (available space) in fine tables are in ascending order.</p> <p>Ensures that available space information in fine tables corresponds to the available space in the data control word of the data block.</p>
CONTENTS	<p>Ensures that absolute address (AA) words in fine tables are valid.</p> <p>Verifies that link items have valid block-address and word-address field values (that is, that the reference physical specifications for the structure are not exceeded).</p> <p>Verifies that the count item value does not exceed the DASDL specification.</p> <p>Verifies that OCCURS DEPENDING ON and SIZE DEPENDING ON items do not exceed the DASDL specifications.</p>
RECORD PLACEMENT	<p>Ensures that record directory words for records that have been relocated are valid (that is, that the block address and word address of the moved record do not exceed the physical specifications for the structure).</p>
STRUCTURE CHECK	<p>Verifies that the control words in block 0 (zero) are valid (that is, the value in the LASTBLOCKADDRESS file is equal to FILE.LASTRECORD + 1 and the NEXTBLOCKADDRESS word is an address for an empty block).</p> <p>Verifies that the second block of a file is a table block.</p> <p>Ensures that empty blocks are in a one-way linked chain.</p> <p>Ensures that table control words in table blocks are valid. Ensures that coarse-table and fine-table entries contain valid information.</p> <p>Ensures that the data control word in the data block is valid (words available + slots allocated + 3 LEQ BLOCKSIZE).</p> <p>Checks each in-use directory entry word to ensure that information is valid (that is, that the start of each record does not exceed one BLOCKSIZE of the structure and is in the data record portion of the data block).</p>
VERIFYSTORE	<p>Ensures that every valid record satisfies the VERIFY conditions.</p>

The following table describes crosscheck certification options for compact data sets.

Option	Function
COUNT	<p>Verifies that the number of counted links pointing to another data set is equal to the count item value in the referenced data set record.</p>
LINK <link item name>	<p>Verifies that all links in the compact data set point to valid records in the referenced structure.</p>

Checking Integrity and Performance

Option	Function
SETS <set name>	Verifies that mappings between the data set and the sets that span it are valid.

Direct Data Sets

Direct data set records are stored in key value order. One unsigned numeric data item in the record is designated as the key item. The value of the key item is the file-relative record address. Direct data sets can only be disjoint.

The certification options for direct data sets and the functions that they perform are described in the following table.

Option	Function
CONTENTS	Verifies that link items have valid block-address and word-address field values (that is, that the referenced physical specifications for the structure are not exceeded). Verifies that the count item value does not exceed the DASDL specification.
RECORD PLACEMENT	Verifies that the key field in each valid record is equal to the relative record number.
VERIFYSTORE	Ensures that every valid record satisfies the VERIFY conditions.

The following table describes crosscheck certification options for direct data sets.

Option	Function
COUNT	Verifies that the number of counted links pointing to another data set is equal to the count item value in the referenced data set record.
LINK <link item name>	Verifies that all links in the direct data set point to valid records in the referenced structure.
SET <set name>	Verifies that mappings between the data set and the sets that span it are valid.

Ordered Data Sets

Ordered data set records are kept in a physical sequence based on a user-specified key without using a set. Ordered data sets can be either disjoint or embedded, but typically are embedded.

The certification options for ordered data sets and the functions that they perform are described in the following table.

Option	Function
CONTENTS	<p>Verifies that link items have valid block-address and word-address field values (that is, that the reference physical specifications for the structure are not exceeded).</p> <p>For AA words in table blocks, verifies that</p> <ul style="list-style-type: none"> • The block address does not exceed the number of blocks in the file. • The word size is less than the file-block size. • The four bits between the block address and word address are 0000.
RECORD PLACEMENT	Verifies that key items in the data block are in ascending order.
STRUCTURE CHECK	<p>For block 0 (zero) control words, verifies that</p> <p>The block referenced by the left-off block field in block 0 (zero) is a data block.</p> <p>The block-address field in the LASTBLOCK field of block 0 (zero) is 1 more than the number of blocks in the file.</p> <p>Verifies that data blocks are in a two-way linked chain if the structure is disjoint.</p> <p>Verifies that empty blocks are in a one-way linked chain.</p> <p>Verifies that table blocks are in a one-way linked chain.</p> <p>Checks the block control words in data blocks to ensure that all fields are valid.</p> <p>Verifies that the NUMSUBBLOCKS field in the data block equals 1 if the data set is disjoint and the block is data block.</p>
VERIFYSTORE	Ensures that every valid record satisfies the VERIFY conditions.

The following table describes crosscheck certification options for ordered data sets.

Option	Function
COUNT	Verifies that the number of counted links pointing to another data set is equal to the count item value in the referenced data set record.
LINK <link item name>	Verifies that all links in the ordered data set point to valid records in the referenced structure.
OWNER	Ensures that mapping between the embedded ordered data set and its owner is valid.

Random Data Sets

Random data set records are not maintained in logical order, but are allocated to particular blocks based on a hashing function of the record key. Random data sets can only be disjoint.

The certification options for random data sets and the functions that they perform are described in the following tables.

Option	Function
CONTENTS	Verifies that link items have valid block-address and word-address field values (that is, that the reference physical specifications for the structure are not exceeded). Verifies that the count item value does not exceed the DASDL specification.
RECORD PLACEMENT	Verifies that the hash value of an overflow block hashes to the block address of its basic block. Verifies that the value obtained by folding the key in a valid record is equal to the fold word that points to the record.
STRUCTURE CHECK	Verifies that the block number stored in block 0 (zero) represents an empty block. Verifies that the empty block chain is a valid one-way linked chain. Verifies that the NUMBEREMPTY field represents the actual number of available record slots in that block.
VERIFYSTORE	Ensures that every valid record satisfies the VERIFY conditions.

The following table describes crosscheck certification options for random data sets.

Option	Function
COUNT	Verifies that the number of counted links pointing to another data set is equal to the count item value in the referenced data set record.
LINK <link item name>	Verifies that all links in the random data set point to valid records in the referenced structure.
SETS <set name>	Verifies that mappings between the data set and the sets that span it are valid.

Restart Data Sets

Restart data sets are similar to disjoint standard data sets with no variable records. Restart data sets can only be disjoint.

The certification options for restart data sets and the functions that they perform are described in the following tables.

Option	Function
AVAILABLE SPACE	Verifies that each AA word in the DKTABLE points to an invalid record in a data block.
STRUCTURE CHECK	Verifies that the last record is valid (that is, that $DKNUMSEGMENTS + DKBASE + 2 = FILE.LASTRECORD$). Verifies that all DKTABLE segments contain valid segments and valid AA words. Verifies that the last segment is partially full (contains fewer than 29 entries) if more than one segment is in DKTABLE.
VERIFYSTORE	Ensures that every valid record satisfies the VERIFY conditions.

The following table describes the crosscheck certification option for restart data sets.

Option	Function
SETS <set name>	Verifies that mappings between the data set and the sets that span it are valid.

Standard (Fixed-Format) Data Sets

All data blocks in a standard (fixed-format) data set contain records of the same type and size. Records are not maintained in logical order. Standard (fixed-format) data sets can be disjoint or embedded.

The certification options for standard (fixed-format) data sets and the functions that they perform are described in the following tables.

Option	Function
AVAILABLE SPACE	Verifies that each AA word in the DKTABLE points to an invalid record in a data block.
CONTENTS	Verifies that link items have valid block-address and word-address field values (that is, that the reference physical specifications for the structure are not exceeded). Verifies that the count item value does not exceed the DASDL specification.

Checking Integrity and Performance

Option	Function
STRUCTURE CHECK	<p>Verifies that the last record is valid (that is, that $DKNUMSEGMENTS + DKBASE + 2 = FILE.LASTRECORD$).</p> <p>Verifies that all DKTABLE segments contain valid segments and valid AA words.</p> <p>Verifies that only the last segment is partially full (less than 29 entries) if more than one segment is in the DKTABLE.</p>
VERIFYSTORE	Ensures that every valid record satisfies the VERIFY conditions.

The following table describes crosscheck certification options for standard (fixed-format) data sets.

Option	Function
COUNT	Verifies that the number of counted links pointing to another data set is equal to the count item value in the referenced data set record.
LINK <link item name>	Verifies that all links in the standard data set point to valid records in the referenced structure.
SETS <set name>	Verifies that mappings between the data set and the sets that span it are valid.

Standard (Variable-Format) Data Sets

Data blocks in a standard (variable-format) data set contain records of different types and sizes. Records are not maintained in logical order. Standard (variable-format) data sets can be disjoint or embedded.

The certification options for standard (variable-format) data sets and the functions that they perform are described in the following tables.

Option	Function
AVAILABLE SPACE	Ensures that AA words in block 0 (zero) point to invalid records.
CONTENTS	<p>Verifies that link items have valid block-address and word-address field values (that is, that the reference physical specifications for the structure are not exceeded).</p> <p>Verifies that the count item value does not exceed the DASDL specification.</p>

Option	Function
STRUCTURE CHECK	<p>Ensures that block 0 (zero) information is valid. Ensures that empty blocks are in a one-way linked chain.</p> <p>Ensures that table blocks are in a one-way linked chain.</p> <p>Ensures that absolute addresses in the table blocks are valid.</p> <p>Ensures that data control words are valid (that is, that LEFTOFFFIELD does not exceed the structure block size).</p>
VERIFYSTORE	Ensures that every valid record satisfies the VERIFY conditions.

The following table describes crosscheck certification options for standard (variable-format) data sets.

Option	Function
COUNT	Verifies that the number of counted links pointing to another data set is equal to the count item value in the referenced data set record.
LINK <link item name>	Verifies that all links in the standard data set point to valid records in the referenced structure.
SETS <set name>	Verifies that mappings between the data set and the sets that span it are valid.

Unordered Data Sets

Unordered data sets have either a fixed format or a variable format. Records are not maintained in logical order. Unordered data sets can be disjoint or embedded, but typically are embedded.

The certification options for unordered data sets and the functions that they perform are described in the following tables.

Option	Function
CONTENTS	<p>Verifies that link items have valid block-address and word-address field values (that is, that the reference physical specifications for the structure are not exceeded).</p> <p>Verifies that count item value does not exceed the DASDL specification.</p>

Checking Integrity and Performance

Option	Function
STRUCTURE CHECK	Ensures that block 0 (zero) control words are valid. Ensures that the empty blocks are in a one-way linked chain. Ensures that the data blocks are in a two-way linked chain. Ensures that the control words in the data blocks are valid. Ensures that the link word maintained in the available area of a data block is valid.
VERIFYSTORE	Ensures that every valid record satisfies the VERIFY conditions.

The following table describes crosscheck certification options for unordered data sets.

Option	Function
COUNT	Verifies that the number of counted links pointing to another data set is equal to the count item value in the referenced data set record.
LINK <link item name>	Verifies that all links in the unordered data set point to valid records in the referenced structure.
SETS <set name>	Verifies that mappings between the data set and the sets that span it are valid.

Sets and Subsets

A set or subset is a file of indexes that refer to all or some of the records of a single data set. Sets and automatic subsets are automatically maintained by the system. Sets and subsets permit access to the records of a data set in some logical sequence. Sets and subsets are typically used to optimize certain types of retrievals of the data set records.

Bit Vector Sets

One bit exists in a bit vector set for each record in a data set. Each bit has an implicit positional relationship to one data set record. A bit has a value of 1 if the corresponding data set record belongs to the set or subset; it has a value of 0 if the corresponding record is deleted or does not belong to the set or subset. Bit vector sets can only be disjoint and can only refer to standard (fixed-format) data sets.

The certification options for bit vector sets and the functions that they perform are described in the following tables.

Option	Function
CONTENTS	Verifies that the last bit set in the last block of the in-use tables does not have a positional relationship that exceeds the number of records in the spanned data set.
STRUCTURE CHECK	Verifies that bit 47 of word 0 (zero) of table 0 (zero) is set. Verifies that the EOFBLOCK absolute address (located in the last segment of the bit vector) does not exceed the bounds of the referenced file. Verifies that the EOFBLOCK does not point inside the in-use tables.

Index Random Sets

Index random set or subset records are not maintained in key order but are allocated to particular tables based on a hashing function of the key. Index random sets can only be disjoint.

The certification options for index random sets and the functions that they perform are described in the following table.

Option	Function
CONTENTS	Ensures that the AA words are in the range of the data set that spans them.
RECORD PLACEMENT	Verifies that each table key equals the existing fold word when the table key is folded.
STRUCTURE CHECK	Verifies that block 0 (zero) information is valid. Verifies that table control words in blocks are valid. Verifies empty blocks are in a one-way linked chain. Verifies that each block in any given overflow chain hashes to the block address of the root block of the chain.

Index Sequential Sets

An index sequential set or subset is a collection of tables arranged hierarchically. Fine tables are at the lowest level of the hierarchy. One entry exists in the fine tables for each data set record contained in the set or subset. In a table, entries are kept in increasing binary collating sequence. Fine tables are ordered by another level of tables (coarse tables). One entry exists in the coarse tables for each fine table. Index sequential sets can be disjoint or embedded.

The certification options for index sequential sets and the functions that they perform are described in the following tables.

Checking Integrity and Performance

Option	Function
CONTENTS	Ensures that each AA word in a fine table is in the range of the reference data set.
RECORD PLACEMENT	Ensures that the coarse table key entries are in ascending order. Verifies that each coarse table key value for a disjoint structure references the highest key value in the referenced fine table. Verifies that all fine table key entries are in ascending order.
STRUCTURE CHECK	Verifies that block 0 (zero) information is valid. Verifies that table control words in blocks are valid. Verifies that empty blocks are in a one-way linked chain. If duplicates are allowed, ensures that the duplicate resolver is the same as the AA word in the key entry of the record.

The following table describes the crosscheck certification option for index sequential sets.

Option	Function
OWNER	Ensures that mapping between an embedded index sequential set and its owner is valid.

Ordered List Sets

Ordered list sets are maintained in order by key. Entries reside in tables. One entry exists in the tables for each data set record in the set or subset. Ordered list sets can be disjoint or embedded.

The certification options for ordered list sets and the functions that they perform are described in the following tables.

Option	Function
CONTENTS	Ensures that the AA words are in the range of the data set that spans them.
RECORD PLACEMENT	Verifies that the table entry keys of ordered lists are in ascending order.
STRUCTURE CHECK	Verifies that block 0 (zero) information is valid. Verifies that table control words in blocks are valid. Verifies that empty blocks are in a one-way linked chain.

The following table describes the crosscheck certification option for ordered list sets.

Option	Function
OWNER	Ensures that mapping between an embedded ordered list set and its owner is valid.

Unordered List Sets

Unordered list sets retrieve data set records in physical order. No key items are present. Instead, the AA words that point to the data set records serve as the keys. The unordered list sets are maintained in order of data set record addresses. Unordered list sets can be disjoint or embedded.

The certification options for unordered list sets and the functions that they perform are described in the following tables.

Option	Function
CONTENTS	Ensures that the AA words are in the range of the data set that spans them.
RECORD PLACEMENT	Verifies that the table entry AA words of unordered lists are in ascending order.
STRUCTURE CHECK	Verifies that block 0 (zero) information is valid. Verifies that table control words in blocks are valid. Verifies that empty blocks are in a one-way linked chain.

The following table describes the crosscheck certification option for the unordered list sets.

Option	Function
OWNER	Ensures that mapping between an embedded unordered list set and its owner is valid.

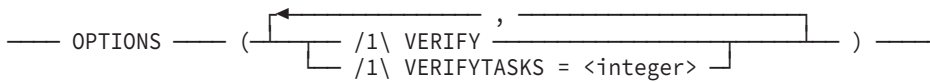
DMUTILITY DBDIRECTORY Statement

The DBDIRECTORY statement produces a report on the status of the files and rows designated in the <dbdirectory list>. The following diagram illustrates the syntax for the DBDIRECTORY command.

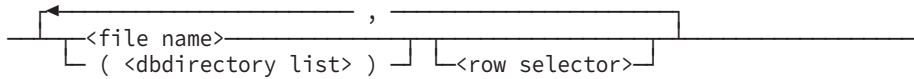
Syntax



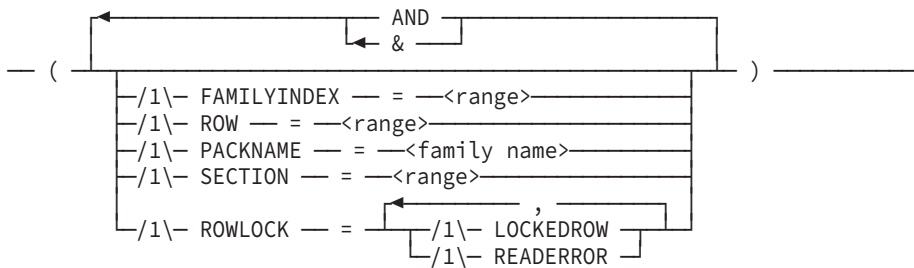
<dbdirectory option>



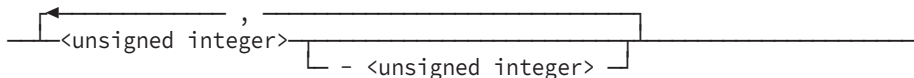
<dbdirectory list>



<row selector>




<range>



Explanation

The following table explains the elements of the syntax diagram.

Option	Explanation
 VERIFY	Controls preverification of the <dbdirectory list> when the specified database is in a quiesced state. Preverification includes integrity checks for CHECKSUM and ADDRESSCHECK errors. Integrity errors are identified.
 VERIFYTASKS	Specifies the number of tasks allowed to perform a VERIFY process in parallel. The VERIFY process cannot be restarted.

Option	Explanation
<dbdirectory list>	Identifies the files and rows include in the report. The slash equal sign (/=) produces a report on the status of all rows in a family of files. The equal sign (=) alone reports on the status of all rows in the database
<row selector>	Specifies the rows of the file to report. If a DBDIRECTORY list is enclosed in parentheses and a row selector is specified, all database files in the <dbdirectory list> are restricted by that row selector. Database files in the <dbdirectory list> that already have a row selector specification have the outer selection constraints related to the inner selection constraints with the Boolean construct OR.
FAMILYINDEX	If specified in the row selector, only those rows that currently reside on the specified family indexes are to be listed on the report
ROW	If specified, only those rows are listed.
PACKNAME	Enables you to limit DMUTILITY to a particular pack family without enumerating the files that exist on that pack family. PACKNAME is typically used in conjunction with FAMILYINDEX.
 SECTION	If specified, only rows that belong to the sections are specified.
ROWLOCK = LOCKEDROW	If specified, all locked out rows are reported.
ROWLOCK = READERROR	If specified, all rows having read operation errors (but not those locked out) are reported.

Example 1

This command produces a report on the status of all rows in the database.

```
DBDIRECTORY =
```

Example 2

This command produces a report of all rows in the database that are locked out or have read operation errors. Row recovery can be used to restore the damaged rows. Refer to “DMUTILITY RECOVER Statement” in [Section 8, Recovering the Database](#), for directions on how to recover rows.

```
DBDIRECTORY = (ROWLOCK=LOCKEDROW , READERROR)
```

Example 3

This command performs a verification of a Quiesce database and produces a report of all rows in the database that have integrity errors.

Explanation

The following information explains the elements of the syntax diagram.

Option	Explanation
DISABLE	<p>Disables the database after all the current database users leave the mix. After the DISABLE command is issued, DMUTILITY displays the following message every five minutes until all users leave the mix:</p> <pre>WAITING FOR ALL USERS TO LEAVE , DATABASE DISABLE REQUESTED</pre> <p>If you issue a DISABLE request in a Remote Database Backup environment, Tracker and the database are brought down at the next restart point.</p> <p>The disable task stays in the mix until the database has been disabled.</p>
DISABLE NOW	<p>Disables the database immediately. All current user programs are terminated and system error 18 occurs.</p> <p>If you issue a DISABLE NOW request in a Remote Database Backup environment, Tracker as well as the database is brought down immediately. When the database is opened for the first time following the enable request, halt/load recovery runs.</p>
DISABLE <minutes>	<p>Disables the database in the designated number of minutes. After the designated time period, all user programs are terminated and system error 18 occurs.</p> <p>The value supplied for <minutes> must be an unsigned integer.</p> <p>Once the disable request has been issued, DMUTILITY periodically displays messages indicating the amount of time until the disable request takes effect.</p> <p>The disable task stays in the mix until the database has been disabled.</p> <p>If you use this option in a Remote Database Backup environment, then Tracker is brought down at the next restart point if a restart point occurs before the designated time period elapses. If a restart point does not occur in this timeframe, then Tracker and the database are brought down after the designated time period elapses.</p>
ENABLE	<p>Cancels the disable request.</p> <p>If the database is currently disabled, the ENABLE command cancels the lock on the database control file and makes the database available.</p> <p>If the disable is in progress, then the enable request must be entered from a different session than the disable request currently in progress. When the enable request is made, the DMUTILITY run that issued the disable request displays the message</p> <pre>DISABLE REQUEST HAS BEEN CANCELED</pre>

Example 1

The following statement immediately disables the database called PI:

```
R $SYSTEM/DMUTILITY("DB=PI ON ISYS DISABLE NOW")
```

Example 2

The following statement disables the database ORGDB after 15 minutes:

```
R $SYSTEM/DMUTILITY("DB=ORGDB ON DMTEST DISABLE 15")
```

Example 3

The following statement disables the database BANKAPP after all the current users of the database leave the mix:

```
R $SYSTEM/DMUTILITY("DB=BANKAPP DISABLE")
```

Example 4

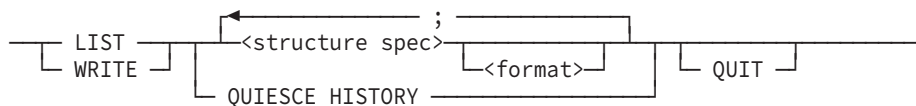
The following statement enables the database LFMOVE:

```
R $SYSTEM/DMUTILITY("DB=LFMOVE ON HYBRID ENABLE")
```

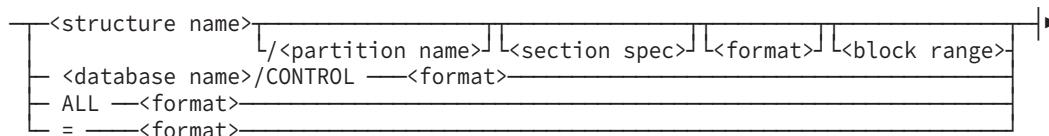
DMUTILITY LIST/WRITE Statement

The LIST/WRITE statement allows you to display the contents of database files. LIST directs output to the terminal; WRITE directs output to the printer. The syntax for these statements is illustrated and explained in the following text.

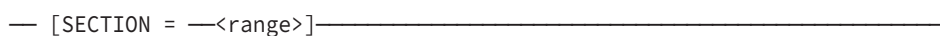
Syntax



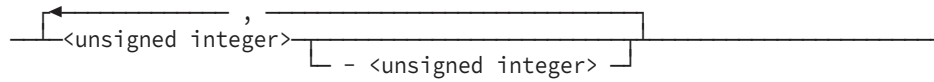
<structure spec>



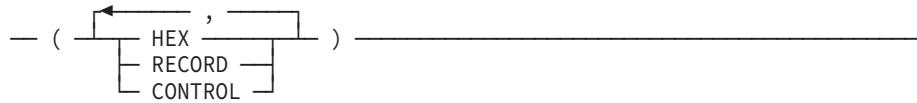
<section spec>



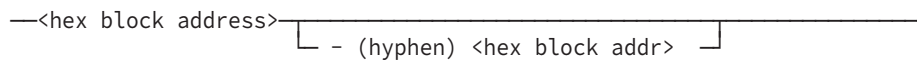
<range>



<format>




<block range>




Explanation

The following information explains the elements of the LIST/WRITE statement syntax diagrams.

Option	Explanation
<structure spec>	Identifies the database files to be displayed. If ALL or an equal sign (=) is specified, the contents of all database files, including the database control file, are displayed. ALL and the equal sign are synonymous in every respect.
 <section spec>	If specified, indicates that only those records belonging to the designated sections are listed. The range for a sectioned data set is 0 through 510. This element is valid only when a structure name is specified
<partition name>	Consists of from 1 to 17 letters and digits that identify the partition.
<format>	Indicates the format in which database information is displayed. If HEX is specified, DMUTILITY displays the entire contents of each block. RECORD causes DMUTILITY to display the records contained in each block. If CONTROL is specified, only the control information for each block in the file is displayed. If <format> is not specified, RECORD format is assumed by default.
<block range>	Identifies the block or range of blocks to be displayed. If <block range> is not specified, all blocks in the file are displayed. This element is valid only when a structure name is specified.

Checking Integrity and Performance

Option	Explanation
<hex block address>	Identifies the address of the block in hexadecimal characters. If a range of blocks is to be displayed, the second hexadecimal block address must be larger than the first. The QUIT command is used if DMUTILITY LIST/WRITE was run with an asterisk (*) parameter. QUIT signifies end of input and terminates DMUTILITY.
 QUIESCE HISTORY	Displays the quiesce history of a database. Refer to "QUIESCE HISTORY Option of the WRITE Command" in Section 14, Using a Quiesce Database , for information about the QUIESCE HISTORY option.

Example 1

The following command displays in RECORD format the contents of all database files:

```
WRITE =
```

Example 2

The following command displays only the control information for each block in every database file:

```
WRITE = (CONTROL)
```

Example 3

The following command displays in HEX format the contents of blocks 0 through A for all database files:

```
WRITE = (HEX) 0-A
```

Example 4

The following command displays in RECORD format the contents of D1, and in HEX format the contents of D2:

```
WRITE D1;D2 (HEX)
```

Example 5

The following command displays in RECORD format the contents of the database control file:

```
WRITE TESTDB/CONTROL
```

Example 6



The following command displays in RECORD format the contents of section 0 of data set D1:

```
WRITE D1 [SECTION = 0]
```

Example 7



The following command displays the quiesce history of the database:

```
WRITE QUIESCE HISTORY
```


Section 12

Communicating with the Database

The Visible DBS commands allow direct communication with the database. Direct communication permits dynamic control over system resources on a global or structure-by-structure basis. This section also discusses database events management.

Using the Visible DBS commands you can perform tasks as follows:

- Interrogate the database status, including the values of ALLOWEDCORE, the amount of buffer core currently in use, the values of SYNCPOINT and CONTROLPOINT, and information about the audit file.
- Change the values of ALLOWEDCORE, SYNCPOINT, OVERLAYGOAL, and CONTROLPOINT.
- Force an audit file switch for audited databases.
- Initiate, terminate, and monitor the status of online garbage collections of sectioned or nonsectioned disjoint index sequential sets.
- Cause the current statistics to be printed if the STATISTICS option is set in the Data and Structure Definition Language (DASDL).
- Interrogate the status of individual structures, including the number of random and serial users, REBLOCK setting, buffer specifications, and REBLOCKFACTOR values.
- Change the values of REBLOCK, REBLOCKFACTOR, and the buffer specifications for a structure.
- Determine what tasks are currently in transaction state and what tasks are waiting for locked records or a syncpoint.
- Determine the current values of or change the values of the TRACKERFLUSHDB and TRACKERQPFACOR options for databases in a Remote Database Backup environment.

Entering Visible DBS Commands

You enter commands to the Visible DBS program using the following system command:

```
<dbms mix number> SM <Visible DBS command>
```

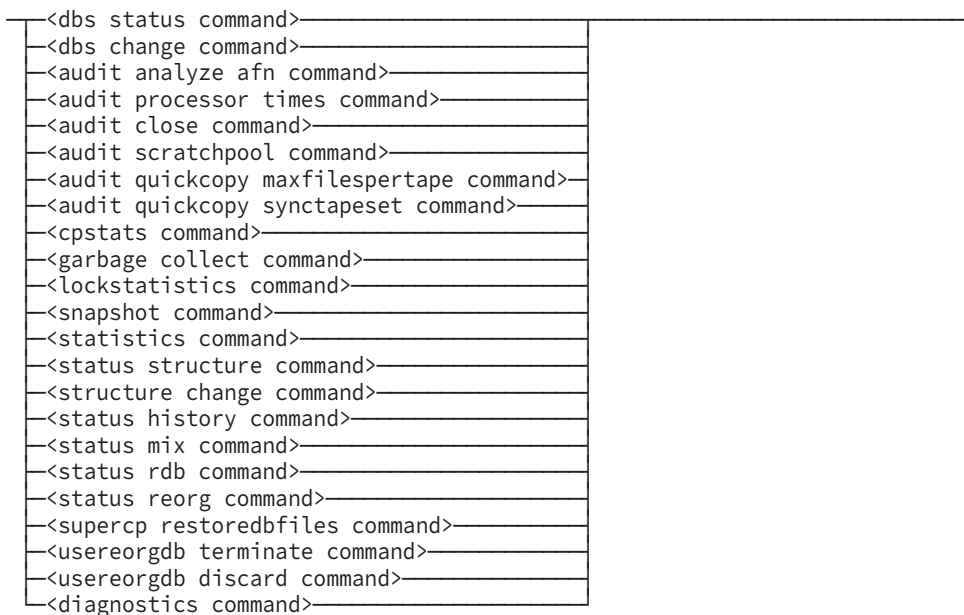
The <dbms mix number> must be that of a database stack. You can determine the <dbms mix number> by entering the system command DBS.

You can also enter commands using Database Operations Center.

Visible DBS Commands

The input text of the Visible DBS command can be a maximum of 126 characters, and is composed of any one of several commands. The syntax for each command is illustrated and explained on the following pages. The following diagram illustrates the syntax for the program.

Syntax



Errors and Warnings

If the input to the Visible DBS command contains a syntax error, an error message prefixed by Syntax is displayed and the input is not processed.

If the input to the Visible DBS command is syntactically correct, but certain specifications are illegal, a warning message prefixed by WARNING is displayed and the remainder of the input is processed. For example, if REBLOCK is specified for a structure for which REBLOCK is FALSE in DASDL, a warning is returned and the remaining input is applied.

DBS STATUS Command

The DBS STATUS command causes the following information to display:

- The current state of the database. One of five possible conditions displays:
 - WAITING FOR ABORT
 - WAITING FOR RECOVERY
 - TERMINATION IN PROGRESS

- OPEN INITIALIZE
- OPEN COUNTS: INQUIRY = <integer>, UPDATE = <integer>
- The current status, SET or RESET, of database events. The current events supported are:
 - DEADLOCK
 - ADMIN
 - FATALERROR
 - SECURITYERROR
 - PUBLICIO

For additional information, refer to [Section 21, Database Events Management](#).

- The current value of OVERLAYGOAL and the current overlay rate achieved.
- The current value of MEMORY RESIDENT and the current number of MEMORY RESIDENT buffers.

If the amount of MEMORY RESIDENT buffer usage is greater than the RESIDENT LIMIT value, a warning message warns the user that no more MEMORY RESIDENT buffers can be allocated to the system unless the user takes some action. Such action includes resetting the MEMORY RESIDENT option for some structures through the DBS CHANGE command.

- The value of ALLOWEDCORE and the amount of buffer core currently in use.
- The current values of SYNCPOINT and CONTROLPOINT.
- The database status following a QUIESCE command. One of two possible conditions appears:
 - WAITING FOR ALL TRANSACTIONS TO COMPLETE, DATABASE QUIESCE REQUESTED
 - DATABASE IS QUIESCED, WAITING FOR RESUME

There is no QUIESCE status output for the Visible DBS STATUS command following a successful RESUME command.

- Various information concerning the audit file, including
 - AREAS, AREASIZE, and BLOCKSIZE.
 - The current audit file number and audit block serial number.
 - Information about the relative position of the Accessroutines in the current audit file.
- Information identifying if audit processor data is being collected.
- Information identifying if statistics are currently being printed.
- Information identifying if capture buffers information is being collected.
- The current value, in minutes, assigned to the TRACKERFLUSHDB option and the current integer value assigned to the TRACKERQPFACOR option.

The TRACKERFLUSHDB and TRACKERQPFACOR options are valid only in a Remote Database Backup environment. For more information, refer to the *Remote Database Backup Operations Guide*. The release level of the Accessroutines being used by the database.

- Garbage collection status information if an online garbage collection is in progress. For more information, see the GARBAGE COLLECT command later in this section.

Syntax

— STATUS _____|

Example

This example displays the current state of the database; the values of ALLOWEDCORE, SYNCWAIT, and the amount of buffer core currently in use; the values of SYNCPOINT and CONTROLPOINT; and various information about the audit file.

```
4637 SM STATUS
```

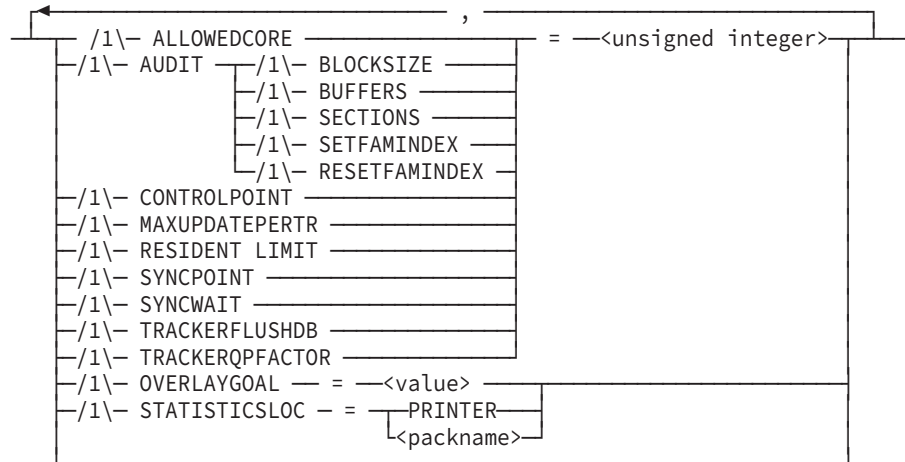
A report similar to the following displays on your terminal:

```
OPEN COUNTS:  INQUIRY  = 2, UPDATE = 0
FORCED OVERLAYS = 1
SYNC WAIT IS 0 SECONDS
EVENTLOGGING PUBLICIO: RESET
EVENTLOGGING SECURITYERROR: RESET
EVENTLOGGING ADMIN: SET
EVENTLOGGING FATALERROR: SET
EVENTLOGGING DEADLOCK: RESET
OVERLAYGOAL =      5  %  ALLOWEDCORE / MINUTE
RESIDENT TOTAL: ALLOWED = 25000, IN USE =22576
CORE TOTAL: ALLOWED  =  50000, IN USE  =  47974,
OLAYRATE=  18.5490%
CONTROL POINT AGEING AFTER AUDIT SWITCHES = FORCE
SYNCPOINT = 100, CONTROLPOINT = 2
CURRENT AUDIT SECTIONS = 1, CF AUDIT SECTIONS = DEFAULT.
CURRENT AUDIT BUFFERS  = 9, CF AUDIT BUFFERS = AUTOMATIC.
AUDIT BLOCKSIZE = 900, AREASIZE = 3000, AREAS  = 100
AUDIT PROCESSOR TIMES OFF
STATISTICSLOC      = PRINTER.
LAST TRACKED ADDRESS: AFN = 5,  ABSN = 6240, OFFSET = 16
PRINT STATISTICS  = ON
4637 08:26 DISPLAY:(ALMA)LOCKSTATSDB: LOCK STATISTICS OFF.
TRACKERQPFACOR   = 10
TRACKERFLUSHDB   = 10
DMSII ACCESSROUTINES 43.301.109
```

DBS CHANGE Command

The DBS CHANGE command is used to change the values of ALLOWEDCORE, AUDIT BLOCKSIZE, AUDIT BUFFERS, AUDIT SECTIONS, AUDIT SETFAMINDEX, AUDIT RESETFAMINDEX, CONTROLPOINT, MAXUPDATEPERTR, OVERLAYGOAL, RESIDENT LIMIT, STATISTICSLOC, SYNCPOINT, SYNCWAIT, TRACKERFLUSHDB, and TRACKERQPFACOR.

Syntax



Explanation

ALLOWEDCORE

ALLOWEDCORE changes the maximum amount of core memory that can be used by the database buffers. The maximum amount that can be assigned is $2^{39}-1$.

AUDIT BLOCKSIZE

AUDIT BLOCKSIZE changes the audit buffer size. Its value is designated in words. This option can be used only when no one is updating the database. Otherwise, the following message is displayed:

```
Update users active --- blocksize not changed.
```

In addition, the current audit file must be closed with the AUDIT CLOSE command.

AUDIT BUFFERS



AUDIT BUFFERS changes the number of audit buffers available to XE structures. In general, a larger number of buffers improves audit throughput and smoothes the intermittent delays associated with short periods of intense audit generation.

The minimum number of audit buffers allowed is 3 and the maximum number is 256. The default number of audit buffers is $\langle \text{number of audit sections} \rangle * ((\text{blocks per area}) - 1)$.

If you decide to change the value for audit buffers by way of a DASDL update after you change the value by using the AUDIT BUFFERS command, you must perform a control file override. Refer to [Section 5, Initializing and Maintaining](#), for additional information about performing a control file override. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified using the DBS CHANGE Visible DBS command with the AUDIT BUFFERS option.

AUDIT SECTIONS



AUDIT SECTIONS changes the number of audit sections that make up the logical audit file. In general, using sectioned audits improves audit throughput and smoothes the intermittent delays associated with short periods of intense audit generation. Sectioned audit files, and therefore the AUDIT SECTIONS command, are only available to sites that have installed the XE license.

Changes to the number of audit sections take effect when the next audit file is opened. Refer to the AUDIT CLOSE command later in this section for additional information.

Changing the number of audit sections can change the number of audit buffers. When the number of audit sections is changed and the specification for audit buffers is automatic, the number of audit buffers is changed to ensure that the number of audit buffers per section is less than the number of blocks per area.

If you decide to change the value for audit sections by way of a DASDL update after you change the value by using the AUDIT SECTIONS command, you must perform a control file override. Refer to [Section 5, Initializing and Maintaining](#), for additional information about performing a control file override. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified using the DBS CHANGE Visible DBS command with the AUDIT SECTIONS option.

Note: *The total size of an audit file is the nonsectioned audit file size multiplied by the number of audit sections. The minimum number of audit sections allowed is 1 and the maximum number is 63. The optimal number of audit sections varies with both the audit trail and application system configuration. In most cases, the optimal number of audit sections is less than or equal to the number of physical disk drives in the audit pack family. Be sure there is adequate disk space available when increasing audit sections.*

SETFAMINDEX

SETFAMINDEX forces all rows of a sectioned audit file to be assigned to the same family index.

RESETFAMINDEX

RESETFAMINDEX enables the MCP to assign rows with the criteria used by MCP.

OVERLAYGOAL

The value assigned to OVERLAYGOAL can be any decimal in the range 0 to 100, inclusively. When you enter this command, the values for the specified parameters are changed immediately. The new values are retained until the next DBS CHANGE command is entered to change the values, or until the next control file update.

MAXUPDATEPERTR

MAXUPDATEPERTR allows you to control the maximum number of updates per transaction. The value of MAXUPDATEPERTR must be greater than 0 and not exceed 50,000. The update value of MAXUPDATEPERTR is affected at the next syncpoint. When the limit is exceeded, all of the previous updates are backed-out and the program receives a LIMITERROR 8. This feature can only be turned-off by performing a DASDL update.

RESIDENT LIMIT

RESIDENT LIMIT changes the maximum amount of core memory that can be used by memory resident buffers. This amount cannot exceed the value assigned to the ALLOWEDCORE option. If the ALLOWEDCORE setting is changed to a value below that of the current RESIDENT LIMIT value, the RESIDENT LIMIT value is automatically adjusted to be the same as the ALLOWEDCORE value.

STATISTICSLOC

The STATISTICSLOC allows the user to designate the location of the statistics report. It requires the STATISTICS option to be set to TRUE. If you choose to specify the pack name, the statistics output is reported as:

```
<dbusercode>DBSTATS/<dbname>/YYYYMMDD/HHMMSS ON <packname>
```

If it is a permanent directory database, the DBPATH is assumed and stored as

```
<dbpath>/DBSTATS/<dbname>/YYYYMMDD/HHMMSS ON <packname>
```

When the statistics are generated, YYYYMMDD and HHMMSS represent the day and time respectfully.

The following example changes the designated location of the Statistics report:

```
9748 SM STATISTICSLOC = DMTEST
```

SYNCPOINT

The SYNCPOINT value controls the maximum number of transactions that can occur between syncpoints. This value limits the amount of recovery time that is required if the database terminates abnormally. The default is 100, and the maximum number is 16,777,215. When the syncpoint value is set through DASDL, the maximum number is 4095.

When you enter the DBS CHANGE command, the syncpoint value is retained until another DBS CHANGE command modifies the value, the next DASDL update occurs, or some type of control file recovery is performed.

Note: *Infrequent syncpoints can adversely affect recovery. Large SYNCPOINT values can increase the number of transactions that must be reprocessed following halt/load or abort recovery. Large SYNCPOINT values can, therefore, cause long recovery times if not used wisely.*

SYNCWAIT

The value for SYNCWAIT cannot be interchanged between the integer 0 and integers other than 0. The user must activate or deactivate SYNCWAIT through DASDL. When SYNCWAIT is not specified in DASDL, the Visible DBS command indicates that the SYNCWAIT value is 0. The SYNCWAIT command is used to change the SYNCWAIT value when the parameter has been set to a valid value in DASDL.

TRACKERFLUSHDB

Use the TRACKERFLUSHDB option with a Remote Database Backup database to set, in minutes, the minimum frequency with which the Tracker flushes the TRACKERINFO file to disk. The actual time between flushes is dependent upon the TRACKERFLUSHDB value and the contents of the audit trail. (The flush occurs only at the end of a controlpoint.) The default value for the TRACKERFLUSHDB option is 10 minutes. Valid values are 1 to 255.

When an abnormal database stack termination occurs on the secondary system, the frequency with which the TRACKERINFO file is flushed to disk determines the amount of audit image reapplication work that must be done by the Tracker.

In a Remote Database Backup database using the default TRACKERFLUSHDB option of 10 minutes, many audit records can accumulate and many controlpoints might be encountered before the 10 minutes elapse and the TRACKERINFO file is flushed.

In general, to minimize the secondary system recovery time, decrease the TRACKERFLUSHDB value as the audit activity increases. Use the DBS CHANGE command to alter the TRACKERFLUSHDB value.

TRACKERQPFACOR

Use the TRACKERQPFACOR option with a Remote Database Backup database to regulate the number of quiet points that the Tracker prescans. The default value of TRACKERQPFACOR is 1. Valid values are 1 to 255.

When you change the value of TRACKERQPFACOR, the new value is stored in the database control file. You can only change the TRACKERQPFACOR value by using a Visible DBS CHANGE command. You can determine the current value of TRACKERQPFACOR by performing a Visible DBS STATUS command.

Example

This example changes the values of SYNCPOINT, CONTROLPOINT, SYNCWAIT, and ALLOWEDCORE to the specified values and designates an audit buffer size of 32000 words.

```
2468 SM SYNCPOINT=100, CONTROLPOINT=10, SYNCWAIT=10,  
ALLOWEDCORE=25000, AUDIT BLOCKSIZE = 32000
```

AUDIT ANALYZE AFN Command

Use the AUDIT ANALYZE AFN command to generate a report of average and peak audit generation rates. As input, you must supply the number of a closed audit file. The information in the report is calculated by analyzing the contents of the audit file.

By default, the audit generation rates are calculated over 60 second time intervals. Consider using a different time interval if any of the following apply:

- The audit files are very large or very small.
- The audit generation rate is very high or very low.

For each time interval from the first audit block timestamp, one line of output is printed that contains the amount of audit generated (expressed in bits of audit per second) and a bar line reflecting that amount.

When you use the AUDIT ANALYZE AFN command, a job called ANALYZEAUDIT starts under the usercode of the database. Once the ANALYZEAUDIT job completes, the report prints automatically on the default printer for your system.

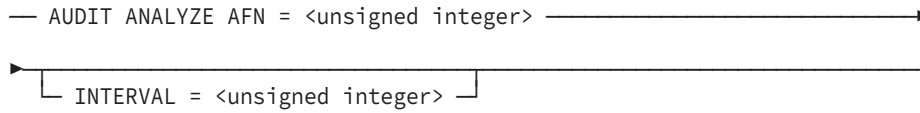
The analysis report also contains processor information if the Visible DBS AUDIT *PROCESSOR TIMES* command was enabled during the time the audit file was open. The AUDIT PROCESSOR TIMES command is described later in this section.

Use the analysis report to aid any network- or host-sizing work you are performing. The report can be especially useful when you are sizing your database environment as part of implementing Remote Database Backup. For more information on network- and hostsizing for the Remote Database Backup environment, refer to the *Remote Database Backup Operations Guide*.

When you are performing host sizing, consider generating reports using several audit files. By selecting audit files from different time periods, you can obtain a picture of how your database activity looks over peak, average, and slack time periods.

The following diagram illustrates the syntax of the AUDIT ANALYZE AFN command:

Syntax



Explanation

The following information describes the elements of the syntax diagram.

Option	Explanation
<unsigned integer> (for AFN)	Designates the audit file number to be analyzed.
INTERVAL <unsigned integer>	Designates, in seconds, the time interval for the report. The default is 60 seconds; that is, there is one line of output printed for each 60-second time interval of the audit.

Example

Following is sample output from a report generated using the AUDIT ANALYZE AFN command. In this example, the time interval is set to 60 seconds, and the AUDIT PROCESSOR TIMES command was not enabled when the audit file was created.

```

TIME                AUDIT BITS / SECOND (* = 1000)
16:17:19            17034 *****
16:18:19             1327 *
16:19:19             1356 *
16:20:19             1301 *
16:21:19             1320 *
16:22:19             1372 *
16:23:19             1063 *
  
```

AUDIT PROCESSOR TIMES Command

The AUDIT PROCESSOR TIMES command begins or terminates the accumulation of processor times in Enterprise Database Server audit blocks. Each time the state of the AUDIT PROCESSOR TIMES option is changed (ON or OFF), an audit file switch occurs. Therefore, either all blocks in an audit file contain processor times or none of the blocks in the audit file contain processor times. When the ON option is used, each audit block contains four time accumulators. These accumulators contain

- The processor time accumulated by Enterprise Database Server applications for Enterprise Database Server inquiry purposes
- The processor time accumulated by Enterprise Database Server applications for Enterprise Database Server update purposes
- The processor time accumulated by Enterprise Database Server applications while outside of the Enterprise Database Server

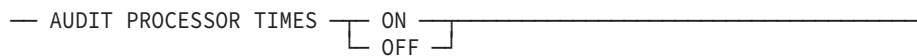
- The I/O time accumulated by the database stack

Use the time information in the report column DMS UPDATE to size your database and network requirements. If you are planning to install Remote Database Backup, the information in the report provides guidelines for host and network sizing. For more information on planning a Remote Database Backup environment, refer to the *Remote Database Backup Operations Guide*.

To print the audit processor information, use the AUDIT ANALYZE AFN command on a suitable audit file. The default time interval for the report is 60 seconds; that is, there is one line of output printed for each 60-second time interval of the audit. For more information on printing the report, refer to "AUDIT ANALYZE AFN Command" earlier in this section.

The following diagram illustrates the syntax for the AUDIT PROCESSOR TIMES command.

Syntax



Example

Following is sample output from a report generated using the AUDIT ANALYZE AFN command. In this instance, processor information was collected while the audit file was in use. Compare this report with the report shown under "AUDIT ANALYZE AFN Command" earlier in this section to see the effect of collecting processor information.

Note: The line in the report starting "TIME PROCESSOR..." has been truncated because of physical width limitations for the document. In the audit analysis report the line reads TIME PROCESSOR TIME IN SECONDS AND % OF INTERVAL (*=10%) I/O TIME AND % OF INTERVAL.

TIME	PROCESSOR TIME IN SECONDS AND % OF INTERVAL (*=10%)						I/O TIME AND		
	DMS	INQUIRY		DMS	UPDATE		NON	DMS	I/O
			%			%	DMS		
12:48:36	0.00	0.00	%	0.28	0.47	%	0.13	0.22	1.02 1.71 %
12:49:36	0.00	0.00	%	0.24	0.40	%	0.13	0.21	0.24 0.39 %
12:50:36	0.00	0.00	%	0.24	0.40	%	0.13	0.21	0.21 0.35 %
12:51:36	0.00	0.00	%	0.23	0.38	%	0.12	0.21	0.20 0.34 %
12:52:36	0.00	0.00	%	0.24	0.39	%	0.13	0.21	0.15 0.24 %
12:53:36	0.00	0.00	%	0.23	0.39	%	0.12	0.20	0.17 0.28 %
12:54:36	0.03	0.06	%	0.36	0.60	%	0.14	0.23	0.22 0.37 %
12:55:36	1.90	3.17	%	10.73	17.88	%	3.40	5.67	4.85 8.08 %

AUDIT CLOSE Command

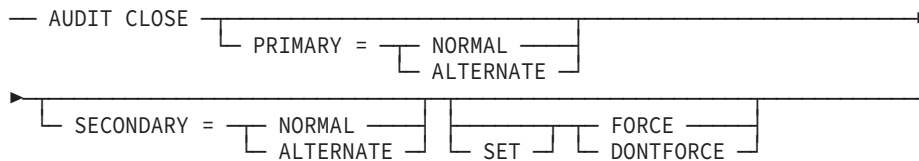
The AUDIT CLOSE command forces an audit file switch and is only valid for audited databases that are currently open for update purposes.

If the AUDIT CLOSE command is used when the database is open for inquiry only, an audit file switch does not occur.

If the database has been open for update during the current session, but there are no update users when the AUDIT CLOSE command is issued, the audit file is released and closed, but an audit file switch does not occur.

The following diagram illustrates the syntax for the AUDIT CLOSE command.

Syntax



Explanation

The following information explains the elements of the AUDIT CLOSE command syntax diagram.

Option	Explanation
PRIMARY = and SECONDARY =	Designates the location of the next primary or secondary (duplicated) audit file. If neither PRIMARY = nor SECONDARY = is specified, the Accessroutines acts as if a normal audit file switch had occurred.
NORMAL	Causes the next audit file to be placed on the normal audit family specified in DASDL. If the normal audit media specified is disk or pack, Accessroutines automatically switches to the alternate media if a SECTORS REQUIRED condition occurs. Automatic switching is not done in the event of a TIMEOUT condition.
ALTERNATE	If the normal audit media specified in DASDL is disk or pack, designating ALTERNATE causes the next location to be the alternate audit media specified in DASDL. The Accessroutines is prevented from switching back to the normal audit location after every audit file is closed. All subsequent audit files are placed on the alternate audit file media until another AUDIT CLOSE command is entered or until the Accessroutines goes away and is invoked again. When all updates of a database close the database and inquiry users remain, the audit files are left in use. However, if release of the audits is necessary, an SM AUDIT CLOSE message causes the audit files to be released.
FORCE	Forces the closure of the current audit file and sets up the conditions necessary for two controlpoints to occur as soon as possible. This allows the zip of COPYAUDIT to take place.

Option	Explanation
DONTFORCE	Causes the zip of COPYAUDIT to be delayed until the normal SYNCPOINT/CONTROLPOINT activity forces two controlpoints to take place. Using this option can improve the overall database performance, especially if the database uses a large ALLOWEDCORE setting.
SET	Causes the FORCE or DONTFORCE setting to be changed. Using the SET option retains the setting change across a halt/load. Note: If this option is included in the command string, an audit switch does not occur.

Example

Assume the following:

- Audit file 65 is in use.
- There are two update applications running.
- There is one inquiry application running.

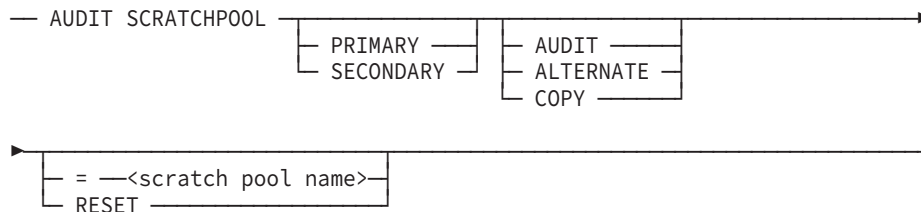
If the AUDIT CLOSE command is issued under these circumstances, audit file 65 is closed and a new audit file 66 is opened.

If the update applications close the database and then the AUDIT CLOSE command is issued, audit file 65 is no longer marked as in-use, but no audit file switch occurs. The next time the database is open for update, audit file 65 is used.

AUDIT SCRATCHPOOL Command

The AUDIT SCRATCHPOOL command identifies the name of the scratch pool from which a tape is selected.

Syntax



Explanation

The following information describes the elements of the AUDIT SCRATCHPOOL command syntax diagram.

Option	Explanation
PRIMARY or SECONDARY	Indicates that the scratch pool is selected for use with the primary or the secondary version of the audit file, the alternate audit file, or the copy of the audit file.
AUDIT, ALTERNATE, or COPY	<p>Designates that the scratch pool is selected for the audit file, the alternate audit file, or the copy of the audit file.</p> <p>If none of these options are designated, the scratch pool is assigned to all audit files even if a different scratch pool was previously defined for a particular audit file. For example, assume the following command previously assigned the scratch pool for primary audits to XTEXT:</p> <pre data-bbox="669 688 1230 709">AUDIT SCRATCHPOOL PRIMARY AUDIT = XTEXT</pre> <p>Then the following command would define the scratch pool names to YTEST, including the primary audits previously defined to XTEXT:</p> <pre data-bbox="669 806 1029 827">AUDIT SCRATCHPOOL = YTEST</pre>
<scratch pool name>	Identifies the scratch pool from which a tape is selected. The scratch pool name is an identifier of at most 17 characters.
RESET	Designates no scratch pool attribute. If a scratch pool name was previously assigned, that name is deleted.

AUDIT QUICKCOPY MAXFILESPERTAPE Command

Use the AUDIT QUICKCOPY MAXFILESPERTAPE command to change the number of audit files that can be appended to an audit tape or to view the current setting of the MAXFILESPERTAPE option. You can use the AUDIT QUICKCOPY MAXFILESPERTAPE command only when the audit trail specification in the DASDL source file

- Includes the QUICKCOPY APPEND command
- Does not include the NOZIP option

If you use the AUDIT QUICKCOPY MAXFILESPERTAPE command and either or both of the preceding conditions are not true, an error is returned.

Syntax

```
— AUDIT QUICKCOPY MAXFILESPERTAPE [ PRIMARY | SECONDARY ] = <integer>
```

Explanation

The following information explains the elements of the AUDIT QUICKCOPY MAXFILESPERTAPE command syntax diagram.

Option	Explanation
PRIMARY, SECONDARY	Indicates whether the command applies to the copying of the primary or the secondary audit files. By default, the command applies to copying both primary and secondary audit files.
<integer>	<p>Designates an integer in the range 1 through 9999. Assigning a value of 1 restricts the number of audit files on a tape to one. Assigning a value of 9999 causes audit files to be continuously appended to the same tape.</p> <p>A tape can consist of any number of reels.</p> <p>To check the current setting for the number of audit files that can be appended to any tape, use the AUDIT QUICKCOPY MAXFILESPERTAPE command and do not include the = <integer> clause.</p>

For more information on the MAXFILESPERTAPE option, refer to [Section 9, Copying Audit Files](#). For more information on using the audit trail specification in the DASDL source file, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

AUDIT QUICKCOPY SYNCTAPESET Command

All information regarding the AUDIT QUICKCOPY SYNCTAPESET command pertains to the XE features.



Use the AUDIT QUICKCOPY SYNCTAPESET command to reset the TAPESET number used by the Accessroutines when initiating the COPYAUDIT command to copy closed audit files to tape.

Syntax

— AUDIT QUICKCOPY SYNCTAPESET _____

Explanation

When you use the AUDIT QUICKCOPY SYNCTAPESET command, the TAPESET numbers for both the primary and secondary audit trails are set so that the next time the Accessroutines initiates a COPYAUDIT command, the TAPESET number used is the same as the copied audit file.

If you do not specify the TAPESET option in DASDL, the AUDIT QUICKCOPY SYNCTAPESET command results in the following error message:

```
AUDIT TAPESET NOT SPECIFIED IN DASDL
```

CPSTATS Command

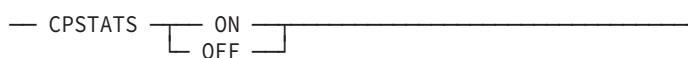
The CPSTATS command causes buffer information to be captured between and at control points. This information includes:

- Number of writeaheads performed between control points
- Number of buffers the writeahead process skipped between control points due to audit constrained
- Number of buffers the writeahead process skipped between control points due to I/O pending
- Number of flushed buffers at control points
- Number of modified buffers (not flushed) at control points

To use the CPSTATS command:

- To set CPSTATS ON, the STATISTICS command must be set in the DASDL. Refer to STATISTICS in the *DASDL Programming and Reference Manual* for additional information.
- Use the VDBS command SM CPSTATS ON and CPSTATS OFF.
- By default, CPSTATS is OFF. It should only be turned ON during the representative timing intervals, and then turned OFF again at the final closing of the database.
- Immediately after turning ON the CPSTATS command, perform a STATISTICS RESTART. Once this is completed, run the STATISTICS command at the end of the interval and turn OFF the CPSTATS. Then perform one last STATISTICS RESTART.
- Check the current settings of CPSTATS by using the SM STATUS command.

Syntax



Explanation

The following information explains the elements of the CPSTATS diagram.

Option	Explanation
ON	If designated, buffer statistics captured between and at control points are printed at the final close of the database.
OFF	Specifying OFF disables the printing of buffer statistics at the final closing of the database.

GARBAGE COLLECT Command



Use the GARBAGE COLLECT command to initiate, terminate, and monitor the status of an online garbage collection of sectioned or nonsectioned disjoint index sequential sets that exist in a database.

The characteristics of the GARBAGE COLLECT command operation are that it

- Runs online
- Runs up to 10 garbage collections in parallel
- Does not lock out users
- Requires that INDEPENDENTTRANS option be set for the database
- Either succeeds completely or makes no change to the structure

For example, if a halt/load occurs while the garbage collection task is running, the system discards the results of the operation, making no change to the set.

- Cannot run at the same time as
 - An online or offline dump
 - A database reorganization
 - A reconstruction on any structure in the database
 - A Remote Database Backup takeover

For example, if you execute the GARBAGE COLLECT command during a reorganization, the system does not initiate the garbage collection and displays an error message.

Heavy user activity on the structure could adversely affect the performance of the garbage collection. In extreme circumstances, user activity can prevent the Enterprise Database Server from completing the garbage collection. In these extreme cases, application program processing takes priority, and the Enterprise Database Server terminates the garbage collection. If this action occurs, the garbage collection must be restarted.

Every block written to the new structure during the garbage collection is audited. Any updates that are applied to the new structure during the garbage collection are also audited. The audit information allows the new structure to be rebuilt or reconstructed if the new structure is lost or damaged during the period between the removal of the old structure and a successful dump of the new structure.

For the preceding reasons, it is recommended that garbage collection for index sets be used only during periods of light user activity on the structure.

If the LOCKEDFILE attribute has been set on the structure outside of DASDL, reset the attribute before entering the GARBAGE COLLECT command. Otherwise, the garbage collection displays error messages and terminates without garbage collecting the structure.

If the LOCKEDFILE attribute is set outside of DASDL on a structure that is part of a garbage collection process, attempts are made to detect the condition and leave the structure as it was before the garbage collection started. However, even with these checks, it might still be possible to manually alter the LOCKEDFILE attribute at a time that is not detected by the garbage collection process, resulting in a corrupted structure. Because detection is not always possible, use the LOCKEDFILE option in DASDL if protection is needed.

Alternative to a Reorganization

The GARBAGE COLLECT command is an advantageous alternative to a reorganization when you need to

- Consolidate unused space in sets or subsets.
- Rebalance index structures to optimize access through sets.

The benefits are

- The database remains online and available.
- Any failure of the garbage collection has no impact on the database.

Disk Storage Requirements

The online garbage collection creates the files it uses on the same pack as the set being collected. Therefore, each index set being collected requires at least enough pack space for the files generated by the GARBAGE COLLECT operation.

How the GARBAGE COLLECT Command Works

The garbage collection process includes the following steps:

1. Create and populate the following two temporary files:

- <set name>/NEW

The set name is the title of the set on which garbage collection is to be performed. The file possesses the same attributes, such as area size and pack name, as the set to be collected. During the online garbage collection, the system writes a new balanced index to this file.

- KEY_UPDATES

KEY_UPDATES is a sequential file in which the system captures all user updates to the set while the garbage collection is performed.

The KEY_UPDATES file is limited to 10,000,000 entries. The error, "KEY_UPDATES FILE IS FULL" is returned when the limit is exceeded and the garbage collect operation is terminated.

2. Apply the updates in the KEY_UPDATES file to the <set name>/NEW file.

When the application is complete, the <set name>/NEW file contains both the balanced index set and all user updates.

3. Swap the <set name>/NEW file with the original set file.

The system stops access to the original file momentarily. The system changes the original set file name to <set name>/OLD so that the file is available in case of a halt/load. The system changes the <set name>/NEW file name to <set name>. User access to a database resumes, using the new set file.

4. Remove the file <set name>/OLD after the two control points.

A message appears indicating that the garbage collection for the set was successful and all associated file handling is complete.

By using the two temporary files described in the previous procedure, the Enterprise Database Server generates the garbage-collected set file while the original set file is in use. Consequently, uninterrupted user access for both inquiries and updates on the set continues during garbage collection.

After the GARBAGE COLLECT Operation

After you perform a successful garbage collection,

- You must perform a database dump of the structures on which a garbage collection process was initiated. If possible, include the entire data set family in the dump. This dump is useful for performing later rebuild or reconstruct operations.

You can perform a rebuild operation though a garbage collection, but it is recommended that you perform a database dump after the garbage collection and use that dump for later rebuild recoveries. This action results in a faster rebuild recovery. You cannot perform a row recovery (reconstruct operation) of the set that was garbage collected if you are using a dump that was created prior to the garbage collection.

- You cannot perform a rollback recovery of the database to a time prior to the time of a successfully completed garbage collection.

Syntax

Initiating Garbage Collection

```

_____
|                                     |
|      GARBAGE COLLECT                 |
|      GC                             |
|                                     |
|_____<GC str list>_____|

```

Terminating Garbage Collection

```

_____
|      TERMINATE                       |
|      DS                               |
|      GARBAGE COLLECT                 |
|      GC                             |
|_____<str list>_____|

```

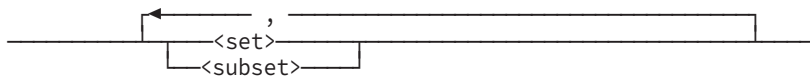
Monitoring Garbage Collection:

```

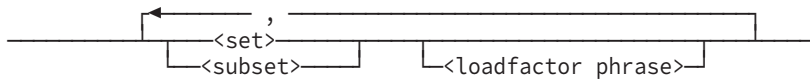
_____
|      STATUS                           |
|      GARBAGE COLLECT                 |
|      GC                             |
|_____<str list>_____|

```

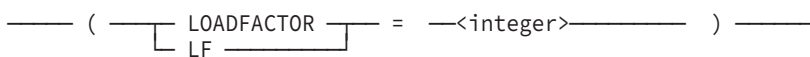
<str list>



<GC str list>



<loadfactor phrase>



Explanation

The <GC str list> variable is a list of sectioned or nonsectioned disjoint index sequential sets and their subsets. The sets must exist and be invoked in the database.

The <str list> variable is a list of sectioned or nonsectioned disjoint index sequential sets and their subsets that are being garbage collected.

A LOADFACTOR phrase is used with the set or subset names that are specified directly before it. If there is no LOADFACTOR phrase specified after a set or subset name, the default load factor is used. For instance, in the following example, a load factor of 75 percent is used for A3 and A4, and the default load factor is used for A6.

```
SM GC A2 (LF=80), A3, A4 (LF=75), A5 (LF=60), A6
```

Example

You can use the STATUS GC <GC str list> command to obtain the status of a structure for which garbage collection has been initiated. The following status information appears when using the syntax for monitoring garbage collection:

```
JOB#(<no>), GC/<str. name>/<string number>
<status>
JOB#(<no>), GC/<str. name>/<string number>
BLOCKS PROCESSED=(<number>)(%)
JOB#(<no>), GC/<str. name>/<string number>
UPDATES TO BE APPLIED=(<number>)
```

The following status message can appear if there is low user activity on a structure:

```
GC/<SET NAME>/<structure #> COMPLETED OK, WAITING FOR TWO CP'S
```

If this situation occurs and the previous message appears, perform one of the following tasks:

- Wait until enough updates have been performed on the database to cause control points to occur and garbage collection will complete automatically.
- Run a program to force syncpoints/control points to occur.

- Close the database.

LOCKSTATISTICS Command

The LOCKSTATISTICS command causes the current LOCK statistics to be printed.

To use the LOCKSTATISTICS command:

- To set the LOCKSTATISTICS command to ON, the STATISTICS command must be set in the DASDL. Refer to the STATISTICS option in the *DASDL Programming and Reference Manual* for additional information.
- Use the VDBS commands SM LOCKSTATISTICS ON and SM LOCKSTATISTICS OFF to turn the command on and off.
- By default, LOCKSTATISTICS is set to OFF. It should only be turned ON during the represented timing intervals, and then turned OFF again at the final closing of the database.
- Immediately after turning ON the LOCKSTATISTICS command, perform a STATISTICS RESTART. Once this completes, run the STATISTICS command at the end of the interval, and turn OFF the LOCKSTATISTICS command. Then perform one last STATISTICS RESTART.
- Check the current settings of LOCKSTATISTICS by using the SM STATUS command.

Note: Using the ON parameter with the LOCKSTATISTICS command can adversely impact performance. It is recommended that you use LOCKSTATISTICS ON in short, controlled time intervals. Use this setting for diagnostic purposes only.

Syntax

```
— LOCKSTATISTICS [ ON | OFF ]
```

Explanation

The following information explains the elements of the LOCKSTATISTICS command syntax diagram.

Option	Explanation
ON	Prints LOCK statistics at the final closing of the database.
OFF	By default, LOCKSTATISTICS is OFF. Specifying OFF disables the printing of statistics at the final closing of the database.

SNAPSHOT Command

The SNAPSHOT command displays information about database applications.

Syntax

— SNAPSHOT —————|

Explanation

The following information explains the element of the SNAPSHOT command syntax diagram.

Option	Explanation
SNAPSHOT	<p>A SNAPSHOT will show database statistics of active applications that are accessing the database. This option generates cumulative figures from the point the database was initiated or from when the STATISTICS option was started or restarted up to the time the SNAPSHOT command was submitted. This option is only valid when STATISTICS is set in DASDL. The location for the SNAPSHOT output is the same as the STATISTICS output.</p> <p>For example, DBSTATS/<database>/SNAPSHOT/<date>/<timestamp></p>

The following is an example output:

```

SNAPSHOT          DATE          TIME
  STARTED        <date>        <timestamp>

(1) MIX NUMBER
(2) TRANSACTION COUNT
(3) TRANSACTION DURATION TIME (SECONDS)
(4) USER FIND DATA RECORD
(5) STORE AFTER CREATE/INSERT KEY INTO SET
(6) STORE AFTER LOCK/CHANGE DATA IN KEY
(7) DELETE DATA RECORD/DELETE ENTRY FROM SET
(8) TASK NAME

(1)  (2)  (3)  (4)  (5)  (6)  (7)  (8)
4525  1  11.622  0  10000  0  0  (<USERCODE>)CANDE/CODE4290 ON <PACK>

FILES DBSTATS/=:S
      DBSTATS/= ON <PACK>

File Name          Filekind      Records      Sectors      CreationTime
-----+-----+-----+-----+-----
DBSTATS/TEST/SNAPSHOT/<DATE>/<TIMESTAMP> BACKUPPRINTE  1  10  <DATE>
DBSTATS/TEST/SNAPSHOT/<DATE>/<TIMESTAMP> BACKUPPRINTE  1  10  <DATE>
2 FILES FOUND

UNISYS Unisys e-@ction Enterprise Database Server for ClearPath MCP -
DATABASE STATISTICS      ** DATABASE (<USERCODE>)<DATABASE> **
    
```

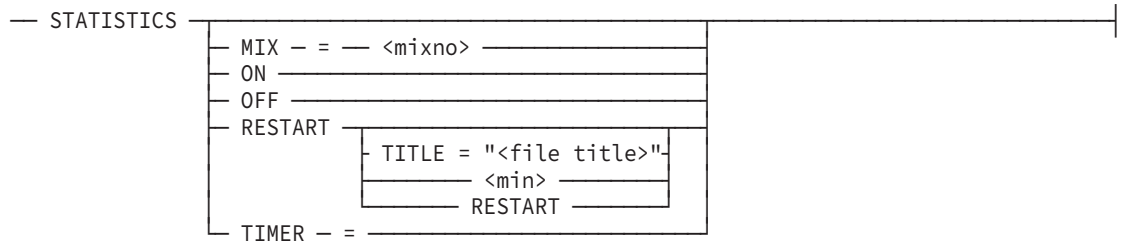
JOB: <JOB> MIX: <MIX> STACK: <STACK>

RELEASE: SSR 59.1 (59.141.0091)
 SYSTEM: CS4290 SYSTEM SERIAL NUMBER: 7426

STATISTICS Command

The STATISTICS command causes the current statistics to be printed.

Syntax



Explanation

The following information explains the elements of the STATISTICS command syntax diagram.

Option	Explanation
MIX	Prints statistics for a task. The <mixno> variable is the task number.
ON	If designated, statistics are printed at the final closing of the database.
OFF	Disables printing of statistics at the final closing. The state of the option is retained in the control file. Statistics can still be printed with the STATISTICS or STATISTICS RESTART commands. This does not affect the state of the option.
RESTART	If designated, all internal counters and timers are cleared after printing, thus statistics are restarted as if the database had just been opened. When <file title> is specified with a family name, the file type of the statistics report will be DATA. If a family name is not specified, the statistics file will be on the family defined with the system DL BACKUP command and the file type will be BACKUPPRINTER. If the STATISTICSLOC database option is also specified, this VDBS command has precedence.

Option	Explanation
TIMER	Specifies how long a statistics will be printed. The <min> variable represents the time in unit of minutes. The range is 0 to 1440. An entry of 0 indicates no periodic printing. Use the RESTART option to clear statistic data after the periodic printing.

STATUS STRUCTURE Command

The STATUS STRUCTURE command displays information about individual structures.

The STATUS STRUCTURE command causes the following information to be displayed for every structure in the physical structure list, or for all physical structures if an asterisk (*) is specified:

- General information about the structure, including
 - Whether the structure is opened or closed.
 - The number of random and serial users of the structure.
- Current settings for the structure, including
 - Whether REBLOCK is assigned the value SET or RESET for the structure.
 - Buffer specifications.
 - The number of small or large buffers allocated.
 - The current REBLOCKFACTOR.
 - The next REBLOCKFACTOR to be used when the structure is closed and reopened.
 - Whether the DUMPENCRYPT option is set for the structure, and the algorithm used.
- Garbage collection status information if an online garbage collection is in progress. For more information, see the GARBAGE COLLECT command earlier in this section.

Note: There is no reference to REBLOCK information if REBLOCK is not set in DASDL.

Syntax

```
— STATUS STRUCTURE [ * | <physical structure list> ]
```

<physical structure list>

```
— <structure specification> , <structure specification>
```

<structure specification>

```
— <structure number> . <structure name>
```


Explanation

Only data sets, sets, and subsets can be specified in the <physical structure list>. Accesses cannot be specified in the physical structure list. When an asterisk (*) is specified, the command applies to all physical structures in the database.

Qualification of structures in the physical structure list is necessary only when the structure name is not unique. Qualification is performed by specifying each structure name in the hierarchy, beginning with the outermost unique structure name and proceeding to the structure name of the structure being qualified.

You can specify an alias name for a physical structure list. This means that you can refer to a physical structure list using 16-bit character structure names through COBOL85 same as a regular structure name.

For more information about alias names, refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

Examples

- The following example displays the current status of PARTSET and PARTDATA:

```
1234 SM STATUS STRUCTURE PARTSET, PARTDATA
```

- The following example displays the current status for a hierarchical family of structures:

```
2222 SM STATUS STRUCTURE PARTINFO,
PARTINFO.NAMES, PARTINFO.NAMES.SPECS
```

- The following output example is displayed in response to STATUS STRUCTURE 40. The number 10 at the end of line 2 of the output represents the REBLOCKFACTOR, and 0 IN USE at the end of line 3 of the output represents the number of buffers currently in use.

```
(#40) DATASET1: (CLOSED)
(#40) REBLOCK SET, REBLOCKFACTOR= 10.
(#40) BUFFERS: (4+1 OR 2) 0 IN USE.
```

- The following output example is displayed in response to the command STATUS STRUCTURE S1, DS1, where S1 is a set and DS1 is a data set. Line numbers have been added for ease of reference

```
Line 1(#5) USERS=5 (5 UPDATE + 0 INQ,
1 RANDOM + 4 SERIAL).
Line 2(#5) MEMORY RESIDENT = RESET.
Line 3(#5)S1: BUFFERS: (100+100 OR 20) 5 2006.
Line 4(#4) USERS=5 (5 UPDATE + 0 INQ,
5 RANDOM + 0 SERIAL).
Line 5(#4) REBLOCK SET, REBLOCKFACTOR = 4
Line 6(#4) MEMORY RESIDENT = RESET.
Line 7(#4)DS1: BUFFERS: (100+100 OR 20)
3 4002, 0 16062.
```

The output includes information that differs from the preceding example because there are update users in this case, whereas the data set was not in use in the previous example.

The explanation for this example is as follows:

Line 1

Structure #5 (set S1) has five update users and zero inquiry users. Of those users, one is a random user, and four are serial users.

Line 2

Structure #5 (set S1) has the MEMORY RESIDENT option reset.

Line 3

Structure #5 (set S1) has 100 system buffers and 100 random buffers. Twenty serial buffers are available. Currently 5 buffers, each consisting of 2006 words, are in use.

Line 4

Structure #4 (data set DS1) has five update users and zero inquiry users. Of those users, five are random users and none are serial users.

Line 5

Structure #4 (data set DS1) has the REBLOCK option turned on and a REBLOCKFACTOR value of 4.

Line 6

Structure #4 (data set DS1) has the MEMORY RESIDENT option reset.

Line 7

Structure #4 (data set DS1) has 100 system buffers and 100 random buffers. Twenty serial buffers are available. Currently three small blocks, each consisting of 4002 words, are in use. No large blocks are in use.

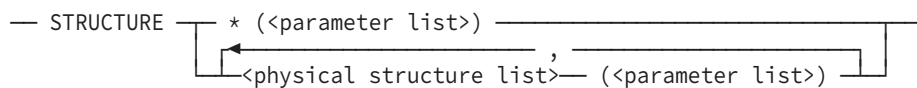
In the information displayed for data set DS1, note that REBLOCKFACTOR is set at line 5. The first set of numbers (3 @ 4002) after the buffer specification is the number and size of small blocks, and the second set of numbers (0 @ 16062) is the number and size of large blocks.

If the same example is run with the REBLOCK value of RESET in DASDL, then line 5 is omitted and only the number and size of small blocks appear after the buffer specification in the last line of output.

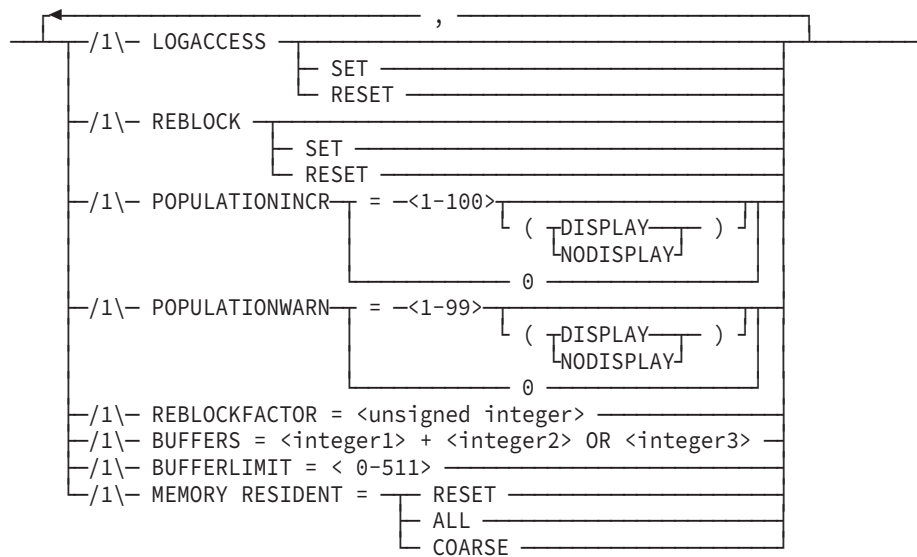
STRUCTURE CHANGE Command

The STRUCTURE CHANGE command is used to set the current parameters of a structure. The parameters specified through this command are stored in the control file and are maintained on subsequent database stack (DBS) runs. When this command is used with a physical structure list, a status display for every structure specified results. No status display occurs if an asterisk (*) is specified.

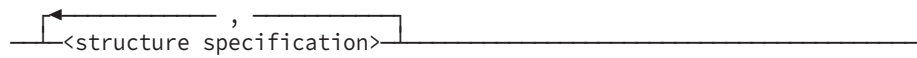
Syntax



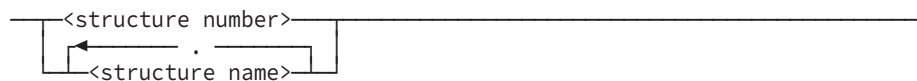
<parameter list>



<physical structure list>



<structure specification>



Explanation

The STRUCTURE CHANGE command is composed of at least one physical structure list, or an asterisk (*) if the command applies to all physical structures in the database, and a corresponding parameter list. Each parameter list is applied only to the structures

specified in the immediately preceding physical structure list. If the asterisk is specified, the corresponding parameter list applies to all physical structures in the database. The rules for specification of structures in the physical structure list are identical to those given in the STRUCTURE STATUS command.

LOGACCESS

Provides database access information. It cannot be used for recovery purposes. Refer to the “Logging Data Access” section later in this guide for additional information.

REBLOCK and REBLOCK SET

Informs the Access routines to employ the REBLOCK mechanism for serial users of a structure. This only takes effect if REBLOCK is TRUE for the structure in DASDL. REBLOCK can never be SET when the serial buffers specification is less than 2.

REBLOCK RESET

Prevents all reblocking on a structure. Any current serial users discontinue reblocking, and the large buffers are overlaid.

REBLOCKFACTOR

Allows the user to change the REBLOCKFACTOR for a structure, provided REBLOCK is TRUE in DASDL. REBLOCKFACTOR does not change until the structure is completely closed and reopened.

POPULATIONINCR

Use this option to enable the Enterprise Database Server to automatically increase the size of a structure beyond the limits declared in the database DASDL source file.

Expansion is accomplished by allowing the number of areas in the structure to increase to the system maximum.

The actual number of records that can be stored within a structure is based upon the number of areas designated for the structure and the structure type. For instance, the record calculation for

- Variable-format data sets is based on the maximum allowed record size
- Compact data sets is based on the average record size

Therefore, even if a population is specified, the actual runtime evaluation of the allowed number of records might differ from the specified number. This effect is particularly noticeable when the specified population is small compared to the number of records that can be contained in an area.

The following switches are provided with the POPULATIONINCR option:

- 0. Switches off the option. To explicitly turn off the POPULATIONINCR option, include the following statement in the database DASDL source file:

```
POPULATIONINCR = 0
```

- 1-100. Defines the percentage of the automatic population increase. The percentage must be an integer. By default, the allowed structure size is increased by 10 percent. To set the automatic population increase to another value, follow the POPULATIONINCR keyword with the percentage increase required. For instance, to set the percentage increase to 20 percent, include the following statement in the database DASDL source file:

```
POPULATIONINCR = 20
```

- (DISPLAY). Updates the control file and enables the delivery of a message when an automatic population increase has occurred. DISPLAY is the default.
- (NODISPLAY). Updates the control file only. No message is sent.

Two possible side effects of increasing population through the POPULATIONINCR mechanism (rather than by performing a reorganization to increase the area size) are as follows:

- The spanning sets for the structures also grow, but without the benefit of table balancing. Thus, the number of table levels might eventually increase and cause some performance degradation.
- The container size of the POPULATION item for the structure can overflow. This causes truncation of the POPULATION item value and might lead to erroneous application results.

When the POPULATIONINCR option is set for a data set, and that data set reaches the quantity of records at which an application would normally receive a limit error, the following actions occur unless system limitations prohibit them:

- The allowed areas for the data set increases by the specified percentage.
- The STORE application of the operation is allowed to proceed.
- A message is sent to the operator display terminal (ODT) in the form of a task that is placed in the waiting mix.

Use the IB (Instruction Block) system command to display the message text.

- The control file is updated to reflect the new population limit for the structure and the time that the message was delivered.

This control file information is removed when a DASDL UPDATE is performed that requires a reorganization of the structure or explicitly increases the AREAS option setting for the structure to a value greater than the information stored in the control file.

If the full area increase of the structure can be accommodated, a message similar to the following is emitted:

```
MYBD: NOTICE - CUSTOMERS have been granted an automatic
population increase of 10%. The target for its maximum
number of rows has increased from 200 to 220. The structure
can now hold 18324 records; based upon the specified record
size.
```

Physical file limitations can prevent the full requested increase from occurring. Under those circumstances, the following additional message is emitted:

```
MYBD: WARNING - Rows 975 through 1000 have been made available
to CUSTOMERS. 1000 AREAS is the maximum number allowed by the
system. Automatic expansion is no longer available to this
structure until it has been reorganized.
```

When this warning occurs, avoid limit errors and unplanned database downtime by reorganizing the structure and increasing the AREASIZE option value as soon as practicable. Once the area size is increased, the population increase mechanism can again be used to increase the structure automatically.

Note: *The POPULATIONINCR option is set when you generate or update a database, unless you explicitly disabled the option.*

While updating the AREAS value for a data set where the POPULATIONINCR option is set, note that an automatic population increase might have increased the control file AREAS value. The DMCONTROL program determines whether to use the old control file value or the new DASDL value.

If a reorganization of the structure is required, the user-specified value from the DASDL is inserted into the new control file, regardless of whether the specified value is enough to hold all records in the data set. The population increment timestamp is reset.

If a reorganization is not required, then the larger of the two values (user-specified DASDL value or control file value) is used for the new control file AREAS value.

If the user-specified value is used, the timestamp is reset; otherwise, it is retained.

If the user-specified AREAS value is not enough and the population increase mechanism is enabled, then the REORGANIZATION program itself causes the automatic population increase mechanism to be invoked, increasing the AREAS specification from the user-specified value.

If the population increase mechanism is not enabled, then the REORGANIZATION program receives a fatal limit error.

If the automatic population increase mechanism has never been invoked, the DASDL AREAS specification is always used.

If you decide to change the POPULATIONINCR value by way of a DASDL update after you change the value by using the POPULATIONINCR option, you must perform a control file override. Refer to [Section 5, Initializing and Maintaining](#), for additional information about performing a control file override. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified using the STRUCTURE CHANGE Visible DBS command with the POPULATIONINCR option.

POPULATIONWARN

Use this option to have the Enterprise Database Server issue a notification when a data set reaches a percentage of its allowed population. (The population calculations are based on the AREAS value.) Notification is placed in the control file and, by default, is also sent to the operator display terminal (ODT) in the form of a task that is placed in the waiting mix.

In general, knowing when structures are approaching their population limit enables steps to be taken to avoid having applications receive limit errors and to avoid the resulting unplanned database downtime.

The POPULATIONWARN option is off by default. That is, the control file is not updated and no notifications are issued.

The following switches are provided with the POPULATIONWARN option:

- 0. Switches off the option. To explicitly turn off the POPULATIONWARN option, include the following statement in the database DASDL source file:

```
POPULATIONWARN = 0
```

- 1-99. Specifies the percentage of total structure capacity that triggers a warning message. The percentage is expressed as an integer. For instance, to trigger a warning when a structure is 85 percent full, include the following statement in the database DASDL source file:

```
POPULATIONWARN = 85
```

- (DISPLAY). Updates the control file and enables the delivery of a warning message when a structure reaches its warning level capacity. DISPLAY is the default.
- (NODISPLAY). Updates the control file only. The warning message is not sent.

For instance, if the following statement is included in the database DASDL source file, the control file is updated, but a warning message is not issued when a structure becomes 75 percent full:

```
POPULATIONWARN = 75 (NODISPLAY)
```

Note: *The total structure capacity might be different from that specified in the database description if automatic limit increases have occurred because of the use of the POPULATIONINCR option.*

You can use the POPULATIONWARN option at both the database and the structure level. Specifications applied at the structure level take precedence over those at the database level.

When the designated percentage of total structure capacity is reached, the following actions occur:

- The control file is updated to reflect the time of the event.
- Unless the NODISPLAY switch is set, a message is sent to the operator display terminal (ODT) in the form of a task that is placed in the waiting mix.

Use the IB (Instruction Block) system command to display the message text.

Once the warning condition has been met—even if automatic population increases occur—further warnings are not issued for that structure until the related control file information has been reset.

To reset the control file information for a structure, perform one of the following actions:

- DASDL UPDATE to increase the allowed number of areas
- Reorganization or garbage collection to reduce the number of in-use areas (ROWSINUSE) below the target percentage

The actual number of records that can be stored within a structure is based upon the number of areas designated for the structure and the structure type. For instance, the record calculation for variable-format data sets is based on the maximum allowed record size, while the record calculation for the compact data sets is based on the average record size.

Therefore, even if a population is specified, the actual runtime evaluation of the allowed number of records might differ from the specified number. This effect is particularly noticeable when the specified population is small compared to the number of records that can be contained in an area.

If you decide to change the POPULATIONWARN value by way of a DASDL update after you change the value by using the POPULATIONWARN option, you must perform a control file override. Refer to [Section 5, Initializing and Maintaining](#), for additional information about performing a control file override. The override enables the system to recognize that the DASDL update value takes precedence over the value you specified using the STRUCTURE CHANGE Visible DBS command with the POPULATIONWARN option.

BUFFERS

Allows the user to change the number of buffers allocated and deallocated for the specified structures. This specification corresponds to the DASDL specification:

```
BUFFERS = <integer1> + <integer2> PER RANDOM USER
          OR <integer3> PER SERIAL USER
```

The <integer1> must be an integer between 0 and 1048575, inclusive. The <integer2>, and <integer3> constructs must be integers from 0 to 254, inclusive. The <integer1> specifies the number of base buffers (small) to allocate for the structure. The <integer2> specifies the number of additional buffers (small) to allocate for each random user. The <integer3> specifies the number of additional buffers to allocate for each serial user. If REBLOCK is assigned the value SET, <integer3> specifies the number of large buffers, and must be greater than or equal to 2. If REBLOCK is assigned the value RESET and <integer3> is greater than or equal to 2, the Access routines performs readahead operations (with small blocks) for serial users of the structure. Readahead operations do not occur if <integer3> is less than 2.

When setting buffers for sectioned structures, you can avoid using excessive amounts of memory if you take the following information into consideration: the number of buffers available for the structure is the number of specified buffers multiplied by the number of sections in the structure

Changing buffer specifications might cause overlays to adjust the number of buffers in the buffer pool. The number of buffers in the buffer pool can exceed the buffer specifications. This is especially true of audited databases running well below ALLOWEDCORE.

BUFFERLIMIT



Allows the user to change the limit of the buffers used for each section of an XE structure. The command is only valid for XE structures. Setting the buffer limit to zero sets the buffer limit to the default value which is 512.

MEMORY RESIDENT = ALL and MEMORY RESIDENT = COARSE

Informs the Accessroutines to employ the MEMORY RESIDENT mechanism on a structure. Setting MEMORY RESIDENT to COARSE is meaningful only for an INDEX SEQUENTIAL set. With this option, only the coarse tables are kept in memory. For any other structures, the effect of using this option is the same as setting MEMORY RESIDENT to ALL.

MEMORY RESIDENT = RESET

Unlocks any buffers currently in memory.

Examples

Example 1

The following example invokes the REBLOCK capability for serial users of the structures specified in the physical structure list. In this example, the structure number assigned by the DASDL compiler is used to denote the structures:

```
7654 SM STRUCTURE 22,25 (REBLOCK)
```

Example 2

The following example changes the number of buffers allocated and deallocated for the specified structures. The BUFFERS specification for structures PARTDATA and PARTSET is altered to two system buffers plus zero buffers for both random and serial users. The BUFFERS specification for structures assigned the structure number 22 or 25 is altered to one system buffer, plus one buffer per random user, or three buffers per serial user.

```
9988 SM STRUCTURE PARTDATA, PARTSET  
      (REBLOCK RESET, BUFFERS = 2 + 0 OR 0),  
      22,25 (BUFFERS = 1 + 1 OR 3, REBLOCK)
```

In addition, the example changes the REBLOCK capability for PARTDATA and PARTSET to RESET, and sets the REBLOCK capability for structures assigned the structure numbers 22 and 25.

Example 3

The following example invokes the REBLOCK capability and changes the number of buffers allocated and deallocated for all physical structures in the database:

```
9182 SM STRUCTURE * (REBLOCK, BUFFERS = 1 + 1 OR 2)
```

The following example changes the MEMORY RESIDENT option of PARTDATA to RESET if the MEMORY RESIDENT option was initialized previously. The MEMORY RESIDENT buffers used by PARTDATA are now available for deallocation. The MEMORY RESIDENT option of PARTSET is initialized through this command as well. If PARTSET is an INDEX SEQUENTIAL set, then coarse tables stay in memory.

```
9123 SM STRUCTURE PARTDATA (MEMORY RESIDENT=RESET),  
      PARTSET(MEMORY RESIDENT=COARSE)
```

STATUS HISTORY Command

The STATUS HISTORY command provides the stack history of all active applications accessing the designated database. The program name is displayed followed by the stack state and the procedure call history. The procedure call history lists the sequence numbers, procedure names, and code file names for each procedure that has been entered but not yet exited at the time of the request.

Syntax

```
— STATUS HISTORY —————|  
| ALL |  
| <mix number> |  
| INTRAN |
```

Explanation

Use the ALL option to display information about all the tasks related to the database. This information not only includes tasks and libraries that declare the database, but also includes tasks that do not declare the database but are currently attached to the database stack through a library or other mechanism.

The <mix number> option displays trace back stack history about only the requested task.

The INTRAN option limits reporting to include only those applications and libraries that are in transaction state. It also reports the start time of the transaction.

The meaning of each column in the resulting display is as follows:

1. The transaction status of the task as it affects the designated database. The following status can be displayed:
 - *T* means the task is in transaction state.
 - Blank means the task is not in transaction state. *L* means the task is a library task and a user is in transaction state through this library.
 - *R* means the task has declared a database and is executing a long transaction, and the SYNCWAIT option is set for the database.

In this case, a TR start time and perhaps an LTR start time are displayed.

The TR start time is the time the transaction started. This time does not change.

The LTR start time reflects the starting time of a long transaction since the last implicit begin transaction (BTR). The time can change. Each instance of an implicit end transaction (ETR) sets the LTR start time to 0 (zero). If another implicit BTR occurs, the LTR start time is set to the current time. This behavior continues until the long transaction ends with an actual ETR. So if the LTR start time is recent, it means the SYNCWAIT option is working and allowing other transactions to complete while a long transaction continues. If the LTR start time is not recent, it could mean there is a long transaction preventing other transactions from completing.

Refer to the description of the SYNCWAIT option in the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for information about an implicit BTR or ETR.

- *N* means the task is a library task and is in a long transaction, and the SYNCWAIT option is set for the database.
 - *A* means the task is in the process of a single abort when INDEPENDENTTRANS is set for the database.
2. The mix number of the task.
 3. The access type of the task. The access type identifies the mode in which the database has been opened by the task. The access type is either update (U) or inquiry (I).
 4. The priority of the task.
 5. The title of the task.
 6. The waiting status of the task. The waiting status can take any of the following values:
 - SYNC means the task is waiting for a syncpoint.
 - The meaning of the mix number depends on the transaction status of the task as follows:
 - If *L* is not displayed in the first column, the task is waiting for a record that is currently locked by the task identified by the mix number.
 - If *L* is displayed in the first column, the mix number identifies that a task is in transaction state through this library. The library task is identified in the second column, and the task that is in transaction is identified in column 6.
 - GONE means that the database is still in transaction state even though the task

initiated through the library has completed. A waiting status of GONE can be displayed only if L or A is displayed in the first column.

- Blank means either that a non-library task is not waiting for locked records or that a library task does not have any callers in transaction state.
- U means that the user program is running.

Examples

- The following command displays the stack history of all the tasks against the database identified by the mix number 3277.

```
3277 SM STATUS HISTORY
```

The following is the sample output listing:

```
17:12:52 3277 -> 3277 OPERATOR ENTERED:3277 SM:STATUS HISTORY.
17:12:52 3277 MSRDISP14:DISPLAY: 0003281 U 50
          (UC)OBJECT/PROG/LOCK ON DMSII
17:12:52 3277 MSRDISP14:DISPLAY:1021:0485:3
          (14088680/14608440) in >> Current MCP <<.
17:12:53 3277 MSRDISP14:DISPLAY:2B:009A:1 (35604000)
          BEGINTRANSACTION in (UC)SYSTEM/ACCESSROUTINES.
17:12:53 3277 MSRDISP14:DISPLAY:2F:0306:4 (35876500)
          TRANSACTIONBEGIN in (UC)SYSTEM/ACCESSROUTINES.
17:12:53 3277 MSRDISP14:DISPLAY: 3:0020:3 (00005000)
          BLOCK#1 in (UC)OBJECT/PROG/LOCK
17:12:53 3277 MSRDISP14:DISPLAY:T0003276 U 50
          (UC)OBJECT/PROG/TESTDB/CREATE ON D .
17:12:53 3277 MSRDISP14:DISPLAY:101B:0525:4
          (14088680/14384860)
          in >> Current MCP <<.
17:12:53 3277 MSRDISP14:DISPLAY:1B:053B:3 (14387100) in
          >> Current MCP <<.
17:12:53 3277 MSRDISP14:DISPLAY:76:0337:1 (10470990)
          WAITFORAUDITIOCOMPLETE in
          (UC)SYSTEM/ACCESSROUTINES.
17:12:53 3277 MSRDISP14:DISPLAY:75:0340:5
          (10436700) WRITEAUDITBUFFER in
          (UC)SYSTEM/ACCESSROUTINES
17:12:53 3277 MSRDISP14:DISPLAY:FB:019F:4 (41806700)
          CREATE in (UC)SYSTEM/
17:12:53 3277 MSRDISP14:DISPLAY:8A:0011:5 (38044000)
          CREATE in (UC)SYSTEM/
17:12:53 3277 MSRDISP14:DISPLAY: 3:0026:5 (00013000)
          BLOCK#1 in (UC)OBJECT/PROG/TESTDB/CREATE
17:12:53 3277 MSRDISP14:DISPLAY:
          ----- 2 ACTIVE ENTRIES -----.
```

- The following command displays the stack history of the task with mix number 3276 with respect to the database identified by mix number 3277. Information about other inquiry or update tasks against the database is not displayed.

```
3277 SM STATUS HISTORY 3276
```

The following is the sample output listing:

```
17:13:17 -> 3277 OPERATOR ENTERED:3277 SM:STATUS HISTORY
```

```

3276.
17:13:18 3277 MSRDISP14:DISPLAY:T0003276 U 50
          (UC)OBJECT/PROG/TESTDB/CREATE ON DMSII.
17:13:18 3277 MSRDISP14:DISPLAY:101B:06F6:2
          (14088680/14415730) in >> Current MCP <<.
17:13:18 3277 MSRDISP14:DISPLAY:1B:0542:3 (14387520)
          in >> Current MCP<<.
17:13:18 3277 MSRDISP14:DISPLAY:76:01C4:4 (10451900)
          WAITFORAUDITIOCOMPLETE in
          (UC)SYSTEM/ACCESSROUTINES.
17:13:18 3277 MSRDISP14:DISPLAY:90:0062:2 (11530500)
          FORCEAUDITAWCP in
          (UC)SYSTEM/ACCESSROUTINES .
17:13:18 3277 MSRDISP14:DISPLAY:1B:0083:3 (30807000)
          DMSREADP in (UC)SYSTEM/ACCESSROUTINES.
17:13:18 3277 MSRDISP14:DISPLAY:58:05CB:1 (72471500)
          PATHFINDER in (UC)SYSTEM/ACCESSROUTINES .
17:13:18 3277 MSRDISP14:DISPLAY:FD:03B7:4 (41815000)
          STORE in (UC)SYSTEM/ACCESSROUTINES .
17:13:18 3277 MSRDISP14:DISPLAY:8E:0147:0 (38052900)
          STORE in (UC)SYSTEM/ACCESSROUTINES.
17:13:18 3277 MSRDISP14:DISPLAY: 3:0040:5 (00020500)
          BLOCK#1 in (UC)OBJECT/PROG/TESTDB/CREATE

```

- The STATUS HISTORY command displays information about running tasks that have database declarations within the application or library source code and the stack history of the tasks. Use the ALL option to display information about all the tasks related to the database.

```
3277 SM STATUS HISTORY ALL
```

The following is the sample output listing:

```

17:12:58 -> 3277 OPERATOR ENTERED:3277
          SM:STATUS HISTORY ALL.
17:12:58 3277 MSRDISP14:DISPLAY: 0003281 U 50
          (UC)OBJECT/PROG/LOCK ON DMSII SYNC.
17:12:58 3277 MSRDISP14:DISPLAY:1021:0485:3
          (14088680/14608440)
          in >> Current MCP<<.
17:12:58 3277 MSRDISP14:DISPLAY:2B:009A:1
          (35604000) BEGINTRANSACTION in
          (UC)SYSTEM/ACCESSROUTINES.
17:12:58 3277 MSRDISP14:DISPLAY:2F:0306:4
          (35876500) TRANSACTIONBEGIN in
          (UC)SYSTEM/ACCESSROUTINES.
17:12:58 3277 MSRDISP14:DISPLAY: 3:0020:3
          (00005000) BLOCK#1 in
          (UC)OBJECT/PROG/LOCK .
17:12:58 3277 MSRDISP14:DISPLAY:T0003276 U 50
          (UC)OBJECT/PROG/TESTDB/CREATE ON D .
17:12:58 3277 MSRDISP14:DISPLAY:101B:0525:4
          (14088680/14384860)
          in >> Current MCP <<.
17:12:58 3277 MSRDISP14:DISPLAY:1B:053B:3
          (14387100) in >> Current MCP << .
17:12:58 3277 MSRDISP14:DISPLAY:76:0337:1

```

Communicating with the Database

```

(10470990) WAITFORAUDITIOCOMPLETE in
(UC)SYSTEM/ACCESSROUTINES.
17:12:58 3277 MSRDISP14:DISPLAY:90:0062:2
(11530500) FORCEAUDITAWCP in
(UC)SYSTEM/ACCESSROUTINES .
17:12:58 3277 MSRDISP14:DISPLAY:1B:0083:3
(30807000) DMSREADP in
(UC)SYSTEM/ACCESSROUTINES .
17:12:58 3277 MSRDISP14:DISPLAY:58:05CB:1
(72471500) PATHFINDER in
(UC)SYSTEM/ACCESSROUTINES .
17:12:58 3277 MSRDISP14:DISPLAY:FD:03B7:4
(41815000) STORE in
(UC)SYSTEM/ACCESSROUTINES .
17:12:58 3277 MSRDISP14:DISPLAY:8E:0147:0
(38052900) STORE in
(UC)SYSTEM/ACCESSROUTINES .
17:12:58 3277 MSRDISP14:DISPLAY: 3:0040:5
(00020500) BLOCK#1 in
(UC)OBJECT/PROG/TESTDB/CREATE .
17:12:58 3277 MSRDISP14:DISPLAY:
--- 2 ACTIVE ENTRIES ---.
```

- The following examples demonstrate use of the STATUS HISTORY command with the INTRAN option. This will provide the stack history of all the tasks that belong to the declared database currently in transaction state.

```
3277 SM STATUS HISTORY INTRAN
```

The following is a sample output listing:

```
17:13:30 -> 3277 OPERATOR ENTERED:3277
SM:STATUS HISTORY INTRAN.
17:13:30 3277 MSRDISP14:DISPLAY:1021:0485:3
(14088680/14608440) in >> Current MCP <<.
17:13:30 3277 MSRDISP14:DISPLAY:2B:009A:1 (35604000)
BEGINTRANSACTION in (UC)SYSTEM/ACCESSROUTINES.
17:13:30 3277 MSRDISP14:DISPLAY:2F:0306:4 (35876500)
TRANSACTIONBEGIN in (UC)SYSTEM/ACCESSROUTINES.
17:13:30 3277 MSRDISP14:DISPLAY: 3:0020:3 (00005000)
BLOCK#1 in (UC)OBJECT/PROG/LOCK .
17:13:30 3277 MSRDISP14:DISPLAY:T0003276 U 50
(UC)OBJECT/PROG/TESTDB/CREATE ON D .
17:13:30 3277 MSRDISP14:DISPLAY:T0003276
TR start: 1/13/2010 AT 17:12:22.45.
17:13:30 3277 MSRDISP14:DISPLAY:101B:0525:4
(14088680/14384860) in >> Current MCP <<.
17:13:30 3277 MSRDISP14:DISPLAY:1B:053B:3 (14387100)
in >> Current MCP <<.
17:13:30 3277 MSRDISP14:DISPLAY:76:0337:1 (10470990)
WAITFORAUDITIOCOMPLETE in
(UC)SYSTEM/ACCESSROUTINES.
17:13:30 3277 MSRDISP14:DISPLAY:75:0340:5 (10436700)
WRITEAUDITBUFFER in
(UC)SYSTEM/ACCESSROUTINES .
17:13:30 3277 MSRDISP14:DISPLAY:FB:019F:4 (41806700)
CREATE in (UC)SYSTEM/ACCESSROUTINES ,
17:13:30 3277 MSRDISP14:DISPLAY:8A:0011:5 (38044000)
```

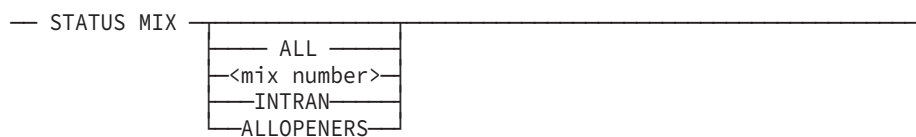
```

CREATE in (UC)SYSTEM/ACCESSROUTINES .
17:13:30 3277 MSRDISP14:DISPLAY: 3:0026:5 (00013000)
BLOCK#1 in (UC)OBJECT/PROG/TESTDB/CREATE .
17:13:30 3277 MSRDISP14:DISPLAY: 1 OF THE 2 ACTIVE ENTRIES
IS IN TRANSACTION STATE.
    
```

STATUS MIX Command

The STATUS MIX command is used to determine the tasks that are currently in transaction state against the designated database and the tasks that are waiting for locked records or a syncpoint.

Syntax



Explanation

The STATUS MIX command displays information about running tasks that have database declarations within the application or library source code. The command returns information on both tasks and libraries that have database declarations but does not always include tasks that access the database through a library or other mechanism without declaring the database. For example, if a task is accessing the database through a library and the library is not in transaction state, the task is not included in the returned information.

The STATUS MIX command is designed to communicate the status of one database only. For example, one response to a STATUS MIX command might indicate that a program is trying to lock a record in another database.

Use the ALL option to display information about all the tasks related to the database. This information not only includes tasks and libraries that declare the database, but also includes tasks that do not declare the database but are currently attached to the database stack through a library or other mechanism.

The INTRAN option limits reporting to include only those applications and libraries that are in transaction state. It also reports the start time of the transaction.

If you specify the mix number in the command, only information for the requested task is displayed.

The meaning of each column in the resulting display is as follows:

1. The transaction status of the task as it affects the designated database. The following status can be displayed:
 - *T* means the task is in transaction state.
 - Blank means the task is not in transaction state.
 - *L* means the task is a library task and a user is in transaction state through this library.
 - *R* means the task has declared a database and is executing a long transaction, and the SYNCWAIT option is set for the database.

In this case, a TR start time and perhaps an LTR start time are displayed.

The TR start time is the time the transaction started. This time does not change.

The LTR start time reflects the starting time of a long transaction since the last implicit begin transaction (BTR). The time can change. Each instance of an implicit end transaction (ETR) sets the LTR start time to 0 (zero). If another implicit BTR occurs, the LTR start time is set to the current time. This behavior continues until the long transaction ends with an actual ETR. So if the LTR start time is recent, it means the SYNCWAIT option is working and allowing other transactions to complete while a long transaction continues. If the LTR start time is not recent, it could mean there is a long transaction preventing other transactions from completing.

Refer to the description of the SYNCWAIT option in the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for information about an implicit BTR or ETR.

- *N* means the task is a library task and is in a long transaction, and the SYNCWAIT option is set for the database.
 - *A* means the task is in the process of a single abort when INDEPENDENTTRANS is set for the database.
2. The mix number of the task.
 3. The access type of the task. The access type identifies the mode in which the database has been opened by the task. The access type is either update (U) or inquiry (I).
 4. The priority of the task.
 5. The title of the task.
 6. The waiting status of the task. The waiting status can take any of the following values:
 - SYNC means the task is waiting for a syncpoint.
 - The meaning of the mix number depends on the transaction status of the task as follows:
 - If *L* is not displayed in the first column, the task is waiting for a record that is currently locked by the task identified by the mix number.
 - If *L* or *N* is displayed in the first column, the mix number identifies that a task is in transaction state through this library. The library task is identified in the second column, and the task that is in transaction is identified in column 6.

- GONE means that the database is still in transaction state even though the task initiated through the library has completed. A waiting status of GONE can be displayed only if L or A is displayed in the first column.
- Blank means either that a non-library task is not waiting for locked records or that a library task does not have any callers in transaction state.

The ALLOPENERS option displays the status of all database openers. An opener is an application that has opened the database. For a program that has multiple concurrent opens of the same database, the ALLOPENERS option displays information on every open.

Examples

- The following command displays the transaction status of the tasks against the database identified by the mix number 2313:

```
2313 SM STATUS MIX
```

- The following command displays the transaction status of the task with mix number 2134 with respect to the database identified by mix number 2315. Information about other inquiry or update tasks against the database is not displayed.

```
2315 SM STATUS MIX 2134
```

- *The STATUS MIX command displays information about running tasks that have database declarations within the application or library source code. Use the ALL option to display information about all the tasks related to the database. The following is a sample STATUS MIX output listing:*

```

----- ACTIVE ENTRIES -----
T0001023 U 50 (DMSII)OBJECT/TEST/LOCKING      2135
 0001127 I 50 OBJECT/SALES/VOLUME           SYNC
L0002135 U 50 TRANSACTION/PROCESSOR

Column:  1    2    3    4          5          6
         -    -    -    -          -          -

```

- The following examples demonstrate use of the STATUS MIX command with the INTRAN option:

```
L0007530 U 50 (ABCD) OBJECT/TEST/DMINQ/STATS ON MYPACK 7535
Transaction start: MM/DD/YYYY AT 13:51:39.90
0007535 U 50 (ABCD) DBUPDATE/0
1 OF THE 2 ACTIVE ENTRIES IS IN TRANSACTION STATE
```

```
R0008788 U 50 (ABCD)DBUPDATE/4
R0008788 TR start: MM/DD/YYYY AT 15:25:19.64
R0008788 LTR start: MM/DD/YYYY AT 15:30:33.41
T0008785 U 50 (ABCD)DBUPDATE/2 8788
T0008785 TR start: MM/DD/YYYY AT 15:25:19.64
2 OF THE 4 ACTIVE ENTRIES ARE IN TRANSACTION STATE
```

- If a task is in the process of aborting, then one extra line following the task entry lists the reason for the termination, as shown in the following example:

```
A<task #> ABORT REASON: : STR:<str#> CAT:<cat#>
SUBCAT:<subcat#>
```

In this example:

- <task#> is the number of the task that is aborting.
 - <str#> is the structure number on which the termination is occurring.
 - <cat#> is the category of the reason for aborting the task.
 - <subcat#> is the subcategory of the error.
- The following examples demonstrate use of the STATUS MIX command with the ALLOPENERS option. In the example, the line L0000895 OPENER:889 shows the mix that opens the database through the library. The lines above and below that line indicate that mix 889 opens the database through the library (mix 895) and is in transaction state.

```
% Program opens same database twice,  
% one with INQUIRY mode and one  
% with the UPDATE mode  
0001534 I 50 (DMSII)TEST/ALLOPENERS ON MYPACK  
0001534 U 50 (DMSII)TEST/ALLOPENERS ON MYPACK  
----- 2 ACTIVE ENTRIES -----.
```

```
% Program opens same database twice,  
% one with INQUIRY mode and the  
% other one open through library with  
% UPDATE mode and is in transaction state.  
L0000895 U 50 (DMSII)TEST/ALLOPENERS/LIB  
ON MYPACK 889.  
L0000895 OPENER: 889.  
0000889 I 50 (DMSII)TEST/ALLOPENERS/CALL  
ON MYPACK  
----- 2 ACTIVE ENTRIES -----.
```

STATUS RDB Command

The STATUS RDB command displays statistics that are accumulated and reported for port I/O operations in Remote Database Backup.

Note: *These statistics are also reported with the normal Enterprise Database Server statistics at the database end of job.*

Syntax

— STATUS RDB _____|

Example

The following is an example of the output created when this command is initiated:

TOTAL SEND CALLS: 16
 TOTAL AUDIT WORDS GENERATED: 28 K WORDS
 TOTAL ACR WAIT TIME: 0: 0:35
 AVERAGE BLOCK SIZE: 1771.9 WORDS
 AVERAGE ACR WAIT TIME (SEC): 2.2088

SECONDARIES:	NUMBER OF SENDS	WORDS SENT	SEND TIME	AVERAGE SEND (SECONDS)
CS7201: ACTIVE	19	28 KW	0:0:59	3.108

The first set of statistics represents activity in the Accessroutines at the primary server.

TOTAL SEND CALLS	Total number of times that the Accessroutines called RDBSUPPORT to send an audit block to the secondary database.
TOTAL AUDIT WORDS GENERATED	Total number of words of audit generated by the database.
TOTAL ACR WAIT TIME	Total amount of time that the Accessroutines had to wait for an acknowledgement from the secondary database. The port I/O and disk I/O of audit blocks are overlapped.
AVERAGE BLOCK SIZE	Average audit block size in words.
AVERAGE ACR WAIT TIME	Average wait time in seconds for an Accessroutines acknowledgment.

Similar statistics are reported for the secondary host. The send time for a secondary host represents the total port I/O send and acknowledgement time. The difference between the ACR wait time and the secondary send time represents the overlap of audit disk I/O and fill time for the next audit block.

STATUS REORG Command

The STATUS REORG command is used to interrogate the progress of a Reorganization of the database. The mix number used with this command must be the job number of the database stack.

Syntax

```
— STATUS REORG [ PRINT ]
```

Explanation

While a database is being reorganized, you can use this command to check the progress of each reorganization task—generate or fixup—associated with the reorganization.

If you are checking the progress of a generate task, the number of records reorganized is displayed. Where possible, the percent of the total records to be reorganized is displayed. The total number of records to be reorganized is not known for variable format or for compact data sets. For a set or subset, the total number of records used for calculating the percentage is the number of records in the spanned dataset. If you are checking the progress of a fixup task, the number of blocks fixed up, and a percent of the total blocks to be fixed up is displayed.

Only one status can be displayed for a structure. Therefore, if there is both a generate task and a fixup task for one structure, the status of the generate task is displayed, except during the time the fixup task is in progress. While the fixup task is in progress, the STATUS REORG command displays the number of blocks fixed up for that structure, and the generate task does not appear. After the fixup task has completed, the generate task appears completed, and nothing appears for the fixup task.

If the reorganization process is not currently running, the progress displayed indicates that the reorganization process has not yet started, or that it is already completed. Entering PRINT causes the status information to be printed as well as displayed.

Examples

- The following example displays the status of the reorganization that is currently in process:

```
2313 SM STATUS REORG
```

- The following is an example of the displayed and printed output:

```
2313 14:25 DISPLAY: REORG NOT RUNNING
```

- The following example displays and prints the status of the reorganization that is currently in process:

```
2313 SM STATUS REORG PRINT
```

- The following is an example of the displayed and printed output:

```
2313 14:31 DISPLAY: GEN/ELECTORS-1/4 952 RECORDS REORGED  
2313 14:31 DISPLAY: FIX/E1BYURID/5 50 BLOCKS FIXED UP (25%)
```

SUPERCP RESTOREDBFILES Command

The SUPERCP RESTOREDBFILES command enables users to recover database files that were removed while the database was open.

Syntax

```
— SUPERCP RESTOREDBFILES —————|
```

Explanation

Issuing the SUPERCP RESTOREDBFILES command forces two control points (super control point) and locks the database files; ACCESSROUTINES then recovers any removed files, unless the files have been untouched.

USEREORGDB TERMINATE Command

The USEREORGDB TERMINATE command terminates and cleans up from a previously recessed REORGDB reorganization. The RECESS command is processed by the main REORGANIZATION program.

Syntax

— USEREORGDB TERMINATE _____|

Explanation

If the USEREORGDB TERMINATE command is issued during the initial copy phase, the copy buffers are deallocated, copied files removed, and the reorganization is terminated.

If the command is executed after the copy phase, then all of the reorganization work files as well as the files belonging to the temporary database are removed.

When the command completes, the control file for the production database is returned to its normal state and the database status is displayed.

If the USEREORGDB TERMINATE command is executed and a reorganization has not been recessed, the following message is displayed with the status information:

```
USEREORGDB IS NOT RECESSED, TERMINATE NOT NEEDED.
```

When the command completes, all files belonging to the temporary database (such as the data files, control file, and audit files) as well as all internal files created by the background offline reorganization and the CAUDIT files are removed.

USEREORGDB DISCARD Command

The USEREORGDB DISCARD command cleans up left over information from a previously failed REORGDB reorganization.

Syntax

— USEREORGDB DISCARD _____|

Explanation

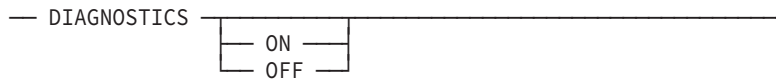
The USEREORGDB DISCARD command can be used without bringing down the database. Using the OVERRIDE USEREORGDB parameter with SYSTEM/DMCONTROL requires the database to be down.

The USEREORG TERMINATE command cannot be used for failed REORGDB reorganizations because it requires that the reorganization first be recessed, and a non-running reorganization cannot be recessed.

DIAGNOSTICS Command

The DIAGNOSTICS command controls what diagnostic information will be collected for certain unexpected errors.

Syntax



Explanation

The following information explains the elements of the DIAGNOSTICS command syntax diagram. When you change the setting of the DIAGNOSTICS option, the new value is stored in the database control file. The default value for the DIAGNOSTICS option is OFF.

Option	Explanation
ON	If designated, additional diagnostic information, such as a full memory dump, will be collected if certain specific errors occur. No more than one memory dump will be taken per database instantiation.
OFF	Disables the collection of additional diagnostic information.

Section 13

Maintaining Databases Containing Large Objects



If a database has internal large objects defined in DASDL, several tank data sets are established by the software. Those tanks need to be maintained to ensure that

- Large objects are deleted if DASDL changes are made that are associated with the large objects.
- Disk space of deleted large objects is consolidated.

The location of the tank data sets is identified in DASDL. Refer to DASDL for the database to determine the location of the files and to the *DASDL Programming Reference Manual* to determine the tank names. Knowing this information is important when doing database backups and using the following commands.

Tank Sizes Available for LOBS

DMSII maintains 3 different tanks for LOBS and determines which tank to use based on the size of the LOB.

Note the record size for each tank:

- SMALL TANK = 0.6 KB/REC
- MEDIUM TANK = 6.0 KB/REC
- LARGE TANK = 24 KB/REC

A LOB can occupy up to 10 records before it shifts over to the next larger tank.

For example, a 6.0 KB LOB will enter the small tank and will occupy 10 records, whereas a 6.01 KB LOB will enter the medium tank. Similarly, a 60.0 KB LOB will enter the medium tank and take up 10 records whereas a larger LOB will enter the large tank.

LOB Utility functions can be used while a database is up and running.

LOB Utility functions will not return the normal DMUTILITY task values if an action requested fails or has warnings.

Maintaining Databases Containing Large Objects

If the tank is corrupted during a LOBUTILITY run, rebuild recovery can be used to retrieve the data. It is recommended to use the following syntax to back up only the structures which contain LOB items prior to running LOBUTILITY:

```
RUN $SYSTEM/DMUTILITY ("DB=<database name>
ON <pack name> OFFLINE
DUMP <database name>/LOB-STR-DIR/= ,
<database name>/SMALL-LOBS/= ,
<database name>/MED-LOBS/= ,
<database name>/LARGE-LOBS/= TO
<dump file name> ON <pack name>")
```

Note: The tasks identified in this section can be initiated through Database Operations Center.

LOBANALYZE Command (DMUTILITY)

Purpose

Use the LOBANALYZE command to produce a report of the disk usage of the large object (LOB) tank data sets. Based on the report, you can decide when to perform a garbage collection and the appropriate garbage collection method to use to consolidate LOB tank data sets.

Refer to "Interpreting the LOBANALYZE Report" for more information.

Syntax

— LOBANALYZE _____|

Example

The following command initiates an analysis of the large objects of the EMP database on the DMCP disk drive, which ends with the creation of a report:

```
SYSTEM/DMUTILITY ("DB=EMP ON DMCP LOBANALYZE")
```

This command produces a report in the following format:

```
***** OUTPUT OF LOBANALYZER *****
```

	SMALL TANK (0.6 KB/REC)	MEDIUM TANK (6 KB/REC)	LARGE TANK (24 KB/REC)
AVAILABLE FOR REUSE(KB) :	292.968	292.968	820.312
> 50 RECORDS :	20.00%	0.00%	0.00%
< 50 RECORDS :	50.00%	59.73%	14.28%
< 10 RECORDS :	30.00%	40.27%	85.71%
IN USE (KB) :	292.968	585.937	1523.437
IN USE/TOTAL CAPACITY :	0.00%	0.00%	0.00%
NUMBER OF DELETED LOBS :	40	12	10
NUMBER OF ADJACENT :	6	2	2

ADJACENT/DELETE LOBS	:	15.00%	16.66%	20.00%
TOTAL CAPACITY (KB)	:	15804492.187	17326406.250	17326406.250

LOBCLEANUP Command (DMUTILITY)

Purpose

Use the LOBCLEANUP command to delete

- Large objects that belonged to a deleted structure
- Large object items that have been deleted from an existing structure

LOBCLEANUP is automatically executed when a program performs an OPEN UPDATE on a database after a Reorganization. LOBCLEANUP will remove LOBS from the tank only when a DASDL update was performed that deleted either a structure or a LOB item.

Executing LOBCLEANUP manually is not recommended since it is automatically executed when necessary.

Syntax

— LOBCLEANUP _____|

Example

The following command deletes large objects that have been deleted from an existing structure and deletes any large objects that belonged to a deleted structure of the BLOBS database on the DMCP disk drive:

```
SYSTEM/DMUTILITY ("DB=BLOBS ON DMCP LOBCLEANUP")
```

LOBCOMBINE/LOBSQUASH Command (DMUTILITY)

Purpose

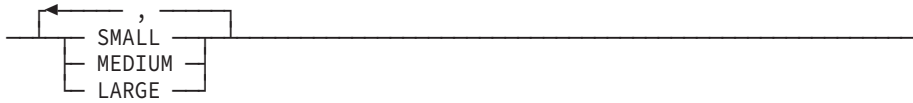
There are two methods of consolidating spaces left when large objects have been deleted programmatically or when the LOBCLEANUP command has been used:

- Use the LOBCOMBINE command to merge adjacent spaces that have been left after large objects have been deleted. This consolidation method completes quickly.
- Use the LOBSQUASH command to merge into one location the scattered spaces that have been left after large objects have been deleted. This consolidation method might take considerable system resources to accomplish.

Syntax

— [LOBCOMBINE] [<tank spec>] [PARALLEL] _____|
 [LOBSQUASH] [ALL]

<tank spec>



<tank spec>

Use the tank spec to indicate the tank sizes against which to run the consolidation. One to three of the specifications can be used in a command.

PARALLEL

Use PARALLEL to indicate that multiple tasks are processed on each tank simultaneously. If only one tank is specified, this option is ignored.

Examples

Example 1

The following command initiates a consolidation of adjacent spaces in the small and large tanks of the BLOBS database on the DMCP disk drive. Both consolidations occur in parallel.

```
SYSTEM/DMUTILITY  
("DB=BLOBS ON DMCP LOBCOMBINE SMALL, LARGE PARALLEL")
```

Example 2

The following command initiates a consolidation of scattered spaces in all the tanks of the BLOBS database on the DMCP disk drive:

```
SYSTEM/DMUTILITY ("DB=BLOBS ON DMCP LOBSQUASH ALL")
```

Interpreting the LOBANALYZE Report

AVAILABLE FOR REUSE (KB)

The total space that once was occupied by LOB records that have since been deleted.

> 50 RECORDS

Percentage of available space consisting of more than 50 adjacent records.

< 50 RECORDS

Percentage of available space consisting of less than 50 but at least 10 adjacent records.

< 10 RECORDS

Percentage of available space consisting of less than 10 adjacent records.

Example

There are 100 LOB items of size 0.6 KB in the small tank and the first 20 LOBS are deleted. (Small tank: 1 Record = 0.6 KB) Prior to LOBCLEANUP, the AVAILABLE FOR REUSE will be 100 percent for < 10 RECORDS. After running LOBCLEANUP, the AVAILABLE FOR REUSE value will be 100 percent for < 50 RECORDS.

IN USE (KB)

Total size of records that are occupied by LOBS for each tank.

IN USE/TOTAL CAPACITY

Ratio of IN USE to TOTAL CAPACITY for each tank.

TOTAL CAPACITY (KB)

Overall capacity for each tank.

NUMBER OF DELETED LOBS

This is the number of regions of available space left by deleting LOBS. It can be affected by LOBCOMBINE and LOBSQUASH.

If this value is high but NUMBER OF ADJACENT is not, then the deleted LOB regions are scattered throughout the tank and LOBSQUASH can be used to consolidate space.

Example

If NUMBER OF DELETED LOBS is 100 and NUMBER OF ADJACENT is 20, then LOBSQUASH will be effective because the majority of the deleted LOB regions are scattered throughout the tank.

NUMBER OF ADJACENT

This is the number of adjacent deleted LOB regions which are now available for reuse. When this value is high, LOBCOMBINE can be used to consolidate space.

Example

If NUMBER OF DELETED LOBS is 100 and NUMBER OF ADJACENT is 90, then LOBCOMBINE will be effective because the majority of the deleted LOB regions in the tank are adjacent.

ADJACENT/DELETED LOBS

This is the ratio of ADJACENT to DELETED LOBS.

A higher ratio indicates that LOBCOMBINE will be more effective while a lower ratio indicates that LOBSQUASH will be more effective in consolidating tank space.

Section 14

Using a Quiesce Database



When using the Enterprise Database Server software in a mirrored environment, you can suspend the updating process of a database using the QUIESCE command and then continue updating a database by using the RESUME command. While the database is in the quiesced state, you can do one of the following tasks:

- Move the mirrored disk to another host.
- Create a quiesce database copy on the same host.

If you have created a quiesce database copy on the same host, you can then use that copy with audit files to recover a database or copy a database.

For information about mirrored disks, refer to the *System Operations Guide*.

Note: The tasks identified in this section can be initiated through Database Operations Center.

Tasks Related to Quiesce Databases

[Table 14-1](#) identifies the tasks you can perform in relation to using a quiesce database and the headings in this section under which these tasks are described.

Table 14-1. Tasks Related to Quiesce Databases or Quiesce Database Copies

To perform this task . . .	Refer to ...
Suspend the updating of a database to enable the process of creating a physically consistent copy of an online database.	QUIESCE Command (DMUTILITY)
Resume updating the database.	RESUME Command (DMUTILITY)
Start creating a quiesce database copy of your live database on the same host.	QUIESCE QDC Command (DMUTILITY)
Create a quiesce database copy.	CREATE QDC Command (DMUTILITY)

Table 14–1. Tasks Related to Quiesce Databases or Quiesce Database Copies (cont.)

To perform this task . . .	Refer to ...
Restore the original configuration of a quiesce database copy.	RESTORE FROM QDC Command (DMUTILITY)
Use a quiesce database copy as a recovery or copy source.	Using a Quiesce Database Copy as a Recovery or a Copy Source
Display the quiesce history of a database.	QUIESCE HISTORY Option of the WRITE Command
Restore the control file of a live database from its quiesce database copy.	CFRESTORE Command (DMUTILITY)

QUIESCE Command (DMUTILITY)

Purpose



Use the QUIESCE command to create a physically consistent copy of an online database. It is recommended that you use the QUIESCE command in a mirrored disk environment in which the copy can be exported to a different host to offload activities such as backup, certification, and data warehousing.

Note: *The QUIESCE command can be used only with an audited database.*

DMUTILITY backups created from an imported database can be used for all forms of recovery at the source host, provided the original family names are retained when you import the database copy.

A DMUTILITY DUMP operation on a database copy created with the QUIESCE command records the time the QUIESCE procedure takes place.

A database copy created with the QUIESCE command is also recognized by an Enterprise Database Server system as a complete, valid database source to begin audit applications. For a Remote Database Backup system, a remote database copy at the secondary host is recognized as a complete, valid database clone source to begin audit application. All physical disks that comprise a QUIESCE database copy must be brought online by using MCP operational commands in the same manner that existing physical disks are brought online.

When you use the QUIESCE command, the results are as follows:

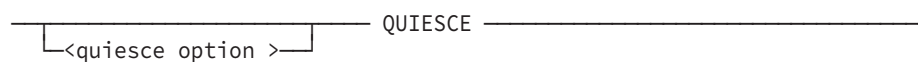
1. The DMUTILITY program waits until all current transactions are complete.
2. All applications in a transaction state complete their current transactions.

3. An application attempting to enter transaction state is suspended with the message "DATABASE IS QUIESCED, WAITING TO RESUME."
4. All data and audit buffers are flushed to disk during the creation of two audited control points.
5. The database control file is marked as being in a quiesced state.
6. The control file stores the QUIESCE timestamp.
7. The DMUTILITY program completes with the message "DATABASE QUIESCED."
8. The database remains in a quiesced state until you use the DMUTILITY RESUME command.

Notes:

- For a database enabled with the Remote Database Backup capability, use the DMCONTROL QUIESCERDBRESET command to disable the capability on the database copy created through the QUIESCE command. This operation must be performed before the copy can be used for processing. Refer to [Section 5, Initializing and Maintaining](#), of this guide for syntax information.
- In a Remote Database Backup environment, a quiesced database is in a "resumed" state after a halt/load recovery process completes.
- Discontinuing an application when the database has been quiesced might not take effect until the database is resumed. This behavior occurs because the Accessroutines might need to store the restart information in the restart dataset as the application closes down. The Accessroutines can store the restart information and complete the close down only after the database resumes. If you enter a DS command on the suspended application, the application is discontinued and the restart information is not saved.

Syntax



Quiesce Option

Refer to [High Availability QUIESCE](#) in this section for information about the <quiesce option>.

Example

The following example and figure illustrates the use of the QUIESCE command:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB QUIESCE")
```

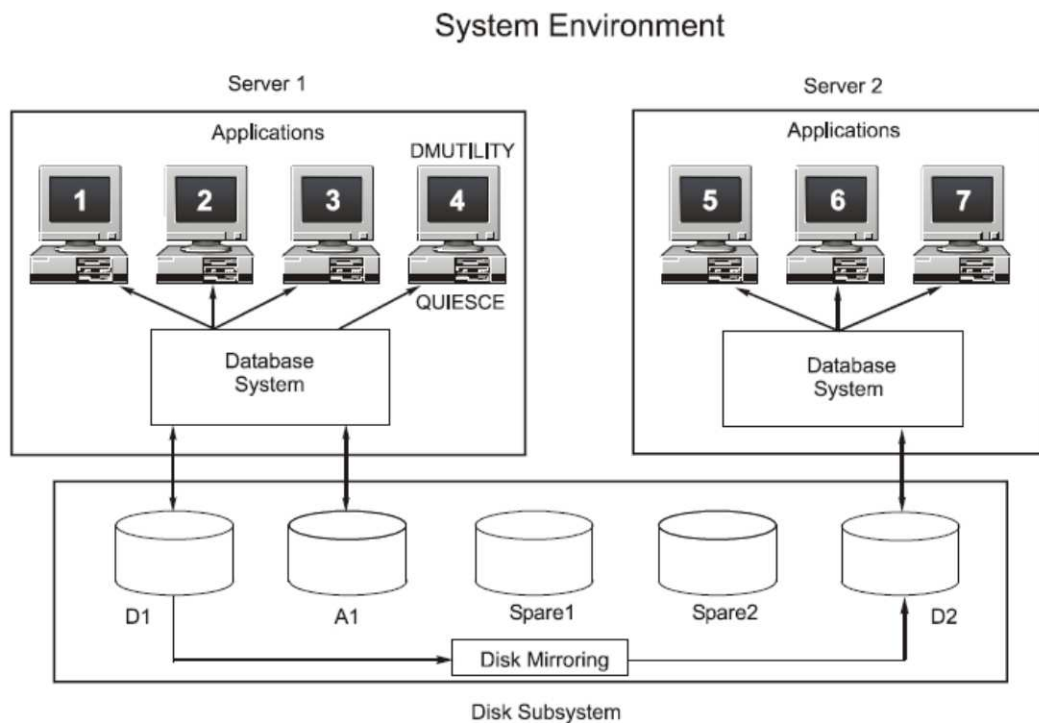
The following example illustrates the use of the QUIESCE command for a permanent directory database:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB QUIESCE");
DATAPATH = <path name> ON <family name>
```

[Figure 14–1](#) shows an example of the database system environment and how the QUIESCE command works in disk subsystems. The database system at Server 1 contains four applications. Applications 1 and 2 are read/write capable, Application 3 is read-only capable, and Application 4 is the DMUTILITY program, which performs the QUIESCE function.

The database system at Server 2 contains three applications, numbered 5 through 7. Each application is read-only capable.

In the disk subsystem, D1 represents a disk containing data files, and A1 is a disk containing audit files, both of which are written to and read by the database system at Server 1. Spare1 and Spare2 are spare disks available to both Server 1 and Server 2. D2 is a physically mirrored copy of D1 that is read by the database system at Server 2.



009128

Figure 14–1. QUIESCE Command in a Database System Environment

RESUME Command (DMUTILITY)

Purpose



Use the RESUME command to return an online database to its normal state following a QUIESCE command. It is recommended that you use the RESUME command in a mirrored disk environment in which detaching a mirrored copy can offload activities such as backup, certification, and data warehousing.

When you use the RESUME command, the results are as follows:

- The quiesced state for the database control file is reset.
- The QUIESCE timestamp is marked with null values.
- The DMUTILITY program completes with the message "DATABASE RESUMED."
- An application that was suspended with the message "DATABASE IS QUIESCED, WAITING FOR RESUME" continues.

Note: A quiesce database copy cannot be resumed while it is being used as a recovery source or a copy source.

Syntax

```
— RESUME _____|
```

Example 1

The following example illustrates the use of the RESUME command:

```
RUN *SYSTEM/DMUTILITY("DB=TESTDB RESUME")
```

Example 2

[Figure 14–1](#) illustrates the scenario on which this example is based, where D1 and D2 are named LIVEPK. To recover the backup database system so that you can initiate transactional updates, perform the following steps:

1. Quiesce the live database by entering the following command at the live system, Server 1:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK QUIESCE")
```

2. Split your mirrored disks that contain all of the database files on the live pack LIVEPK.
3. Copy the audit files needed for a recovery to the appropriate audit pack as configured in the database control file at Server 2.
4. Resume the live database at Server 1:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

5. Acquire the split mirrored disks as LIVEPK at Server 2.
6. Resume the database at Server 2:

```
RUN *SYSTEM/DMUTILITY ("DB=(LIVE)LIVEDB ON LIVEPK RESUME" )
```

7. Recover the database at Server 2.

Example 3

The following example illustrates the use of the RESUME command for a permanent directory database:

```
RUN *SYSTEM/DMUTILITY ("DB=TESTDB RESUME" );  
DATAPATH = <path name> ON <family name>
```

Example 4

The following example illustrates the use of the DMUTILITY RECOVER command to rebuild a database image that is in a state of QUIESCE:

1. Rename and acquire the split mirrored disks.
2. Run the following DMUTILITY RECOVER command:

```
RUN $SYSTEM/DMUTILITY ("DB=TESTDB RECOVER  
(REBUILD THROUGH AUDIT 1990)FROMQUIESCE DB" )
```

QUIESCE QDC Command (DMUTILITY)

The QUIESCE QDC command is a keyed feature. Refer to the Software Product Catalog for information about Quiesce Database Copy (QDC)

- For Single Host Systems
- As a Backup Source
- As a Recovery Source

Purpose



Use the QUIESCE QDC command to start creating a quiesce database copy of your live database on the same host. A quiesce database copy is a mirrored disk copy of a live database that is in a quiesced state. This command places your live database in a quiesced state, just as if you had executed a DMUTILITY QUIESCE command. The QUIESCE QDC command stores the title of the intended quiesce database copy as a registered quiesce database copy in the control file of the live database.

Note: The QUIESCE QDC command is not supported for databases that contain partitioned structures.

You can register and create multiple consistent copies of a live database at the same host by following these steps:

1. Run the QUIESCE QDC command using the QDC title clause.
2. Split your mirrored disks containing all of your database files.
3. Run the RESUME command to reactivate your live database.
4. Rename and acquire the split mirrored disks.
5. Run the DMCONTROL CREATE QDC command to create the quiesce database copy using the same QDC title clause and specifying the new family packs.

Use the DMCONTROL CREATE QDC command to create each consistent copy of your online database at the same host. These copies of your online database can be used to offload activities such as backup generation and data warehousing. DMUTILITY dumps created from a quiesce database copy can be used for all forms of recovery of the live database that require a dump.

The following important issues pertain to a quiesce database copy:

- It must be configured to use the same level of software that the live database uses.
- It can be configured to use its own database software code files by using the DMCONTROL statement with the <code file title change> command option and file-equating CF and CFOLD to the control file of the quiesce database copy.
- Once you have used the RESUME command to reactivate a quiesce database copy, it is no longer a quiesce database copy and operates as an independent nonrelated database. Dumps created from this resumed quiesce database copy cannot be used for recovery of the live database.
- A quiesce database copy cannot be resumed when being used as a recovery source or a copy source.

Refer to the DMUTILITY QUIESCE command and the DMCONTROL CREATE QDC command for a complete explanation of these commands.

Syntax

```

┌──────────────────────────┐ QUIESCE ─── QDC (──<QDC title clause>──) ───┐
└──────────────────────────┘
└──────────────────────────┘
<quiesce option >

```

<QDC title clause>

```

── TITLE= ─── * ─── ───<database name>── ON ───<family name>──┐
┌──────────┐ ┌──────────┐ ┌──────────┐
└──<usercode>──┘ └──<path name>──┘

```

<path name>

```

──*DIR/ ─── ─── /7\ ─── ───<node>──┐
┌──────────┐ ┌──────────┐ ┌──────────┐

```

Quiesce Option

Refer to [High Availability QUIESCE](#) for information about the <quiesce option>.

QDC Title Clause

Use the QDC title clause to register a quiesce database copy with the live database specified in the database statement. Use the DMCONTROL CREATE QDC command and specify the same QDC title clause to create the actual quiesce database copy.

A quiesce database copy title must use the same database name as the database statement, but must use a different family name and usercode. For a permanent directory database, a quiesce database copy title must use the same database name as the database statement, but must use a different family name and different path name.

The control file of the live database specified by the database statement is updated with the registration of the quiesce database copy. You can have up to 15 quiesce database copy registrations at a time.

DMSUPPORT Library for Quiesce Database Copy

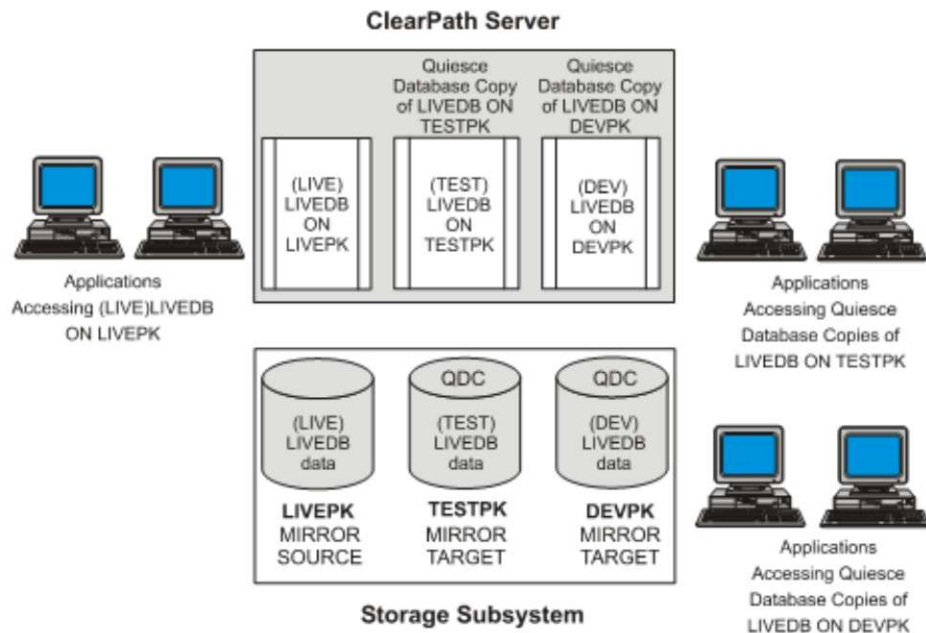
The control file for the quiesce database copy has the same specification for the DMSUPPORT title as in the live database. The following conditions apply:

- If the DMSUPPORT specification includes a usercode and family name, then the quiesce database copy shares the same DMSUPPORT code file.
- If the DMSUPPORT specification does not contain a usercode or family name, then the quiesce database copy looks for the DMSUPPORT code file under the usercode and family name of the quiesce database copy.

The DMSUPPORT title for the quiesce database copy can also be changed using the DMCONTROL syntax shown in example 2.

Examples

[Figure 14-2](#) illustrates the scenario on which examples 1 and 2 are based.



008395

Figure 14–2. Single Server, One Live Database, Two Quiesce Database Copies

Example 1

To create a quiesce database copy instance of LIVEDB on TESTPK for testing a new application before bringing it in to the live database environment, perform the following steps:

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
  QUIESCE QDC(TITLE=(TEST)LIVEDB ON TESTPK)")
```

2. Split your mirrored disks containing all of your database files.

3. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Rename and acquire the split mirrored disks.

5. Enter the following command:

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON
  LIVEPK CREATE QDC TITLE=(TEST)LIVEDB ON
  TESTPK FAMILY LIVEPK=TESTPK")
```

The DMCONTROL CREATE QDC command must be executed from a privileged usercode or the DMCONTROL code file must be marked as a privileged code file.

6. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(TEST)LIVEDB ON TESTPK RESUME")
```

7. Enter the following command:

```
RUN TEST/APPLICATION;  
DATABASE LIVEDB(TITLE=(TEST)LIVEDB ON TESTPK)
```

The application is database-equated to a quiesce database copy at run time and performs updates to the resumed quiesce database copy.

Example 2

To create a second quiesce database copy instance of LIVEDB on DEVPK that uses a DMSUPPORT library specification different from the DMSUPPORT library specification of the LIVEDB for inquiry processing of the quiesce database copy data, perform the following steps:

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=LIVE)LIVEDB ON LIVEPK  
QUIESCE QDC(TITLE=(DEV)LIVEDB ON DEVPK")
```

2. Split your mirrored disks containing all of your database files.

3. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Rename and acquire the split mirrored disks.

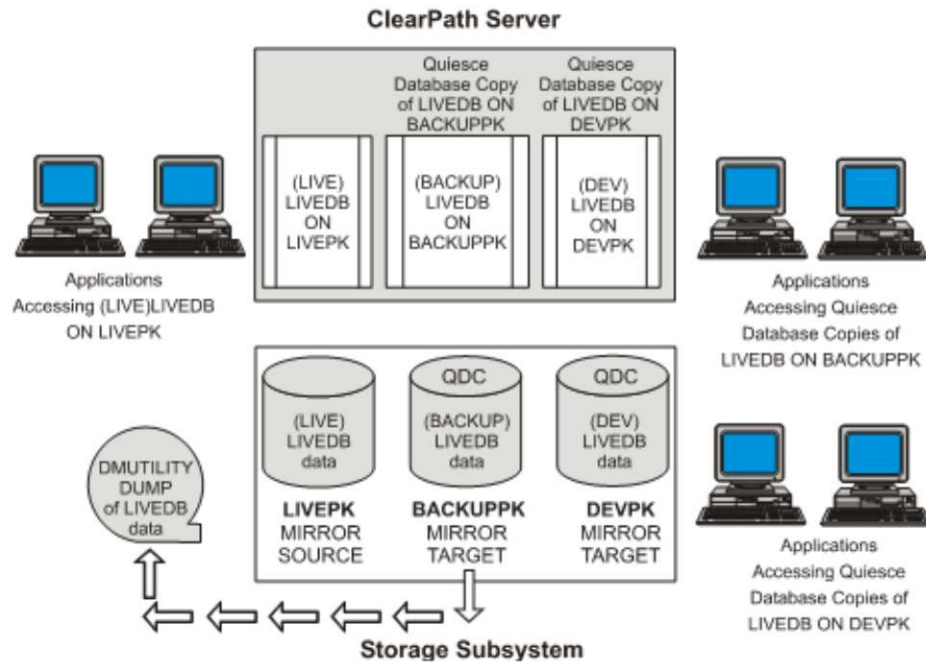
5. Enter the following commands:

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON LIVEPK CREATE  
QDC TITLE=(DEV)LIVEDB ON DEVPK FAMILY LIVEPK=DEVPK")
```

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON LIVEPK DMSUPPORT  
TITLE=(DEV)DMSUPPORT/LIVEDB ON DEVPK");FILE  
CF(TITLE=(DEV)LIVEDB/CONTROL ON DEVPK);FILE  
CFOLD(TITLE=(DEV)LIVEDB/CONTROL ON DEVPK)
```

The DMCONTROL CREATE QDC command must be executed from a privileged usercode or the DMCONTROL code file must be marked as a privileged code file.

[Figure 14-3](#) illustrates the scenario on which examples 3 through 6 are based.



008399

Figure 14-3. Single Server, One Live Database, Two Quiesce Database Copies, Live Database Backed Up from QDC ON BACKUPPK

Example 3

To create a quiesce database copy instance of LIVEDB on BACKUPPK as a source to perform an offline backup of LIVEDB, perform the following steps:

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
  QUIESCE QDC(TITLE=(BACKUP)LIVEDB ON BACKUPPK)")
```

2. Split your mirrored disks containing all of your database files

3. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Rename and acquire the split mirrored disks.

5. Enter the following command:

```
RUN *SYSTEM/DMCONTROL(DB=(LIVE)LIVEDB ON LIVEPK
  CREATE QDC TITLE=(BACKUP)LIVEDB ON BACKUPPK FAMILY
  LIVEPK=BACKUPPK")
```

This command must be executed from a privileged usercode.

6. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(BACKUP)LIVEDB ON
  BACKUPPK OFFLINE DUMP=TO LIVEDBBACKUP")
```

Example 4

To use LIVEDBBACKUP to recover (LIVE)LIVEDB, enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON
LIVEPK RECOVER(REBUILD THRU AUDIT 1959) FROM
LIVEDBBACKUP")
```

Example 5

To use LIVEDBBACKUP to rebuild (LIVE)LIVEDB when the control file of the live database has been lost, enter the following commands:

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB
ON LIVEPK INITIALIZE")

RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK COPY LIVEDB/CONTROL AS
(LIVE)LIVEDB/CONTROL ON LIVEPK FROM LIVEDBBACKUP")

RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON LIVEPK FAMILY
BACKUPPK=LIVEPK");FILE CF=(LIVE)LIVEDB/CONTROL ON LIVEPK;
FILE CFOLD=(LIVE)LIVEDB/CONTROL ON LIVEPK

RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
RECOVER (REBUILD THRU AUDIT 1959) FROM LIVEDBBACKUP")
```

Example 6

To use LIVEDBBACKUP to copy (LIVE)LIVEDB, enter the following commands:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK COPY = AS
(LIVE)= ON LIVEPK FROM LIVEDBBACKUP")

RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON LIVEPK FAMILY
BACKUPPK=LIVEPK");FILE CF=(LIVE)LIVEDB/CONTROL ON LIVEPK;
FILE CFOLD=(LIVE)LIVEDB/CONTROL ON LIVEPK
```

CREATE QDC Command (DMCONTROL)

The CREATE QDC command is a keyed feature. Refer to the *Software Product Catalog* for information about Quiesce Database Copy (QDC) for Single Host Systems.

Purpose

Use the DMCONTROL CREATE QDC command to create a quiesce database copy. This command verifies that the quiesce database copy specified in the QDC title clause is a registered quiesce database copy of the live database specified in the <db statement> from a previously performed DMUTILITY QUIESCE QDC command.

The DMCONTROL CREATE QDC command then performs the following tasks:

- Creates the quiesce database copy.
- Changes the usercode and pack name in the quiesce database control file. If it is a permanent directory database, it changes the path name and pack name in the quiesce database control file
- Performs the specified usercode or path name changes.

The following integrity checks are performed:

- Verification that all database files exist
- Verification that every row of a database file exists and that the control file is present

Notes:

- *Before you create your quiesce database copy, refer to the DMUTILITY QUIESCE QDC command for instructions on how to register your quiesce database copy with the live database. Also, the DMCONTROL CREATE QDC command must be executed from a privileged usercode or the DMCONTROL code file must be marked as a privileged code file.*
- *Unisys recommends that the usercode and disk attributes of the control file specification be defined in the DASDL definition of the live database and that a complete DMSUPPORT title including usercode and disk also be specified in the schema.*

If these recommendations are not followed, the CF and CFOLD titles must be file-equated when a DMCONTROL CREATE QDC command is initiated using a usercode that is different than the usercode of the live database. Because the usercode of the live database is not specified in the schema, DMCONTROL assumes that all data files of the live database along with the live database control file reside under the usercode that is specified in the file-equated CF and CFOLD titles. Refer to example 7 later in this topic for a demonstration of this procedure.

For a live database enabled with the RDB database option, the Remote Database Backup capability of the quiesce database copy is automatically disabled by the CREATE QDC command.

For a live database enabled with the DMDUMPDIRECTORY database option, the DUMPDIR capability of the quiesce database copy is automatically disabled by the CREATE QDC command.

For a live database enabled with the TPS database option, the TPS capability of the quiesce database copy is automatically disabled by the CREATE QDC command.

For a live database enabled with the EVENTS subscription, the EVENTS subscriptions of the quiesce database copy are automatically turned off by the CREATE QDC command.

A quiesce database copy must be configured to use the same level of software that the live database uses. It cannot be configured for a database containing partitioned structures.

Using a Quiesce Database

Add the usercode of the quiesce database copy title to the guard file of a security guarded database. Optionally, create a new guard file for the quiesce database copy and then initiate a DMCONTROL command to change the name and location of the guard file to access the newly created file.

For more information about running DMCONTROL, refer to “Running DMCONTROL” in [Section 5, Initializing and Maintaining](#).

Syntax

```
- CREATE — QDC <QDC title clause> <data file family change>
```

Examples

[Figure 14-4](#) illustrates the scenario on which examples 1 and 2 are based.

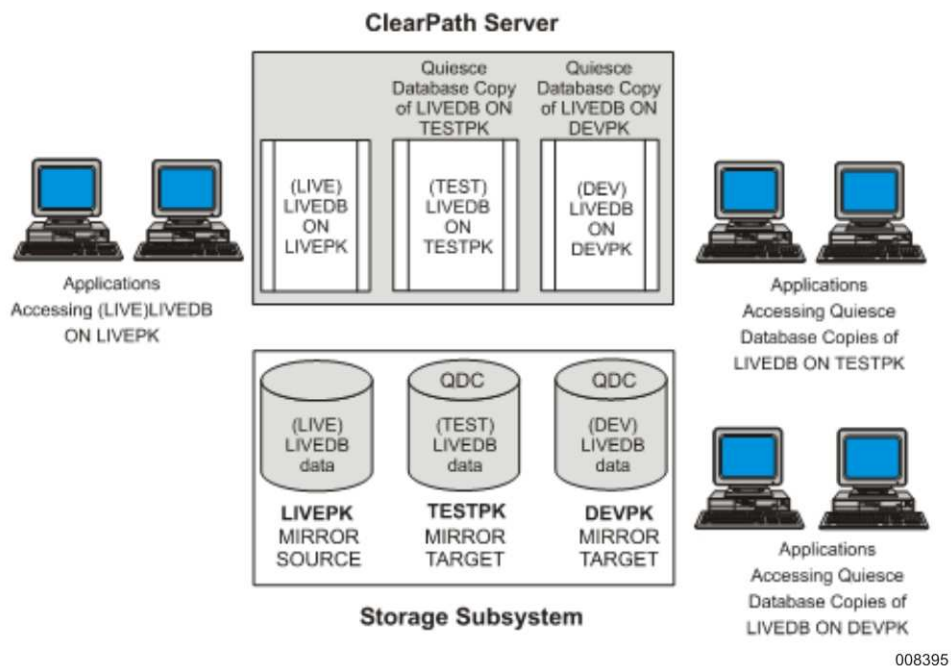


Figure 14-4. Single Server, One Live Database, Two Quiesce Database Copies

Example 1

To create a quiesce database copy instance of LIVEDB on TESTPK for testing a new application before bringing it in to the live database environment, perform the following steps:

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON
LIVEPK QUIESCE QDC(TITLE=(TEST)LIVEDB ON TESTPK)")
```

2. Split your mirrored disks containing all of your database files.
3. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Rename and acquire the split mirrored disks.
5. Enter the following command:

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON
LIVEPK CREATE QDC TITLE=(TEST)LIVEDB ON
TESTPK FAMILY LIVEPK=TESTPK")
```

The DMCONTROL CREATE QDC command must be executed from a privileged usercode or the DMCONTROL code file must be marked as a privileged code file.

6. Enter the following command:

```
RUN *SYSTEM/DMUTILITY ("DB=(TEST)LIVEDB ON TESTPK RESUME")
```

7. Enter the following command:

```
RUN TEST/APPLICATION;
DATABASE LIVEDB(TITLE=(TEST)LIVEDB ON TESTPK)
```

The application is database-equated to a quiesce database copy at run time and performs updates to the resumed quiesce database copy.

Example 2

To create a second quiesce database copy instance of LIVEDB on DEVPK that uses a DMSUPPORT library specification different from the DMSUPPORT library specification of the LIVEDB for inquiry processing of the quiesce database copy data, perform the following steps:

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=LIVE)LIVEDB
ON LIVEPK QUIESCE QDC(TITLE=(DEV)LIVEDB ON DEVPK)")
```

2. Split your mirrored disks containing all of your database files.
3. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Rename and acquire the split mirrored disks.
5. Enter the following commands:

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON LIVEPK
CREATE QDC TITLE=(DEV)LIVEDB ON DEVPK FAMILY LIVEPK=DEVPK")
```

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON
LIVEPK DMSUPPORT TITLE=(DEV)DMSUPPORT/LIVEDB ON
DEVPK");FILE
CF(TITLE=(DEV)LIVEDB/CONTROL ON DEVPK);FILE
```

```
CFOLD(TITLE=(DEV)LIVEDB/CONTROL ON DEVPK)
```

The DMCONTROL CREATE QDC command must be executed from a privileged usercode or the DMCONTROL code file must be marked as a privileged code file.

[Figure 14-5](#) illustrates the scenario on which examples 3 through 5 are based.

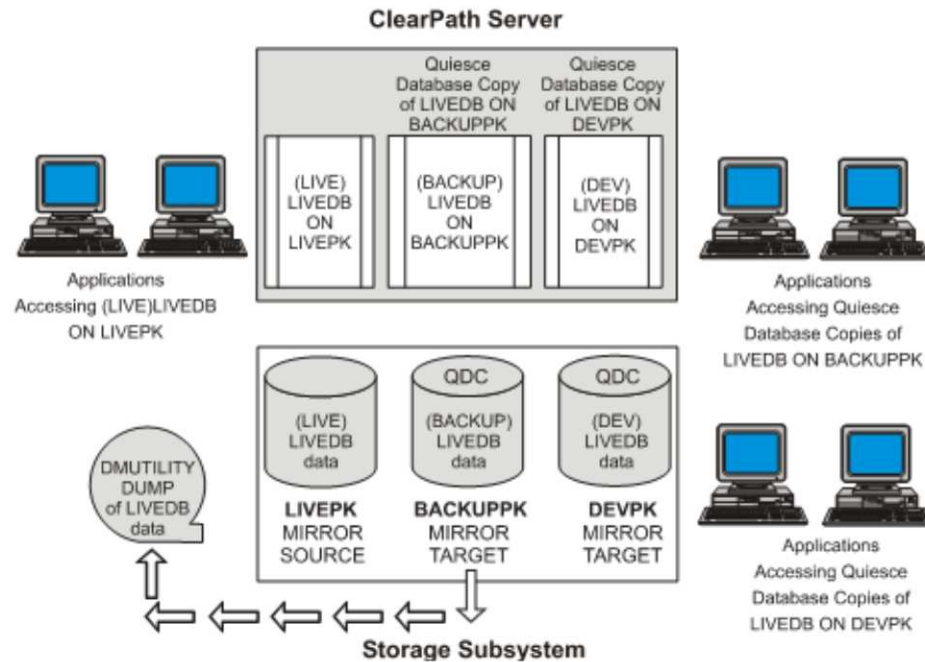


Figure 14-5. Single Server, One Live Database, Two Quiesce Database Copies, Live Database Backed Up from QDC ON BACKUPPK

Example 3

To create a quiesce database copy instance of LIVEDB on BACKUPPK as a source to perform an offline backup of LIVEDB, perform the following steps:

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK QUIESCE QDC(TITLE=(BACKUP)LIVEDB ON BACKUPPK)")
```

2. Split your mirrored disks containing all of your database files.
3. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Rename and acquire the split mirrored disks.
5. Enter the following command:

```
RUN *SYSTEM/DMCONTROL(DB=(LIVE)LIVEDB ON LIVEPK
```

```
CREATE QDC TITLE=(BACKUP)LIVEDB ON BACKUPPK FAMILY
LIVEPK=BACKUPPK")
```

This command must be executed from a privileged usercode.

6. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(BACKUP)LIVEDB
ON BACKUPPK OFFLINE DUMP=TO LIVEDDBACKUP")
```

Example 4

To use LIVEDDBACKUP to recover (LIVE)LIVEDB, enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
RECOVER(REBUILD THRU AUDIT 1959) FROM LIVEDDBACKUP")
```

Example 5

To use LIVEDDBACKUP to recover (LIVE)LIVEDB when the control file of the live database has been lost, enter the following commands:

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON LIVEPK INITIALIZE")
```

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
COPY LIVEDB/CONTROL AS (LIVE)LIVEDB/CONTROL
ON LIVEPK FROM LIVEDDBACKUP")
```

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON LIVEPK FAMILY
BACKUPPK=LIVEPK");FILE CF=(LIVEDB) ON LIVEPK;
FILE CFOLD=(LIVE)LIVEDB ON LIVEPK
```

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK RECOVER (REBUILD THRU AUDIT 1959) FROM LIVEDDBACKUP")
```

Example 6

This example creates a quiesce database copy instance of LIVEDB. In this example, all the LIVEDB data files reside on LIVEPK, but LIVEDB/CONTROL resides on a separate family named CTLPK. The quiesce database copy that is created has data files that reside on TESTPK and a control file that resides on QDCCTLPK.

1. Enter the following command:

```
RUN *SYSTEM/DMUTILILTY("DB=(LIVE)LIVEDB ON CTLPK QUIESCE QDC
(TITLE=(TEST)LIVEDB ON QDCCTLPK)")
```

Note: The DB statement indicates the family on which the LIVEDB control file resides. The QDC TITLE statement indicates the family on which the copy of the quiesce database control file resides.

2. Split the mirrored disks that contain all of the database files.
3. Enter the following command to resume the LIVEDB:

```
RUN SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON CTLPK RESUME")
```

4. Rename and acquire the split mirrored disks. Make sure that the QDC control file (TEST) LIVEDB/CONTROL is located on QDCCTLPK pack.

5. Enter the following command

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON CTLPK CREATE QDC  
TITLE=(TEST)LIVEDB ON QDCCTLPK FAMILY LIVEPK = TESTPK")
```

Notes:

- The *FAMILY* statement indicates that the data files of *LIVEDB* are located on *LIVEPK* and the data files of *QDC* are located on *TESTPK*.
- You must have a privileged usercode to execute the *DMCONTROL CREATE QDC* command.

Example 7

This example creates a quiesce database copy instance of *LIVEDB*. In this example, the *DASDL* definition does not contain a usercode attribute in the control file specification or include a usercode attribute for the *DMSUPPORT* title.

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB = (LIVE)LIVEDB ON LIVEPK  
QUIESCE QDC (TITLE = (BACKUP)LIVEDB ON BACKUPPK)")
```

2. Split the mirrored disks that contain all of the database files.
3. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB = (LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Rename and acquire the split mirrored disks.
5. From a privileged usercode, enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB = (LIVE)LIVEDB ON LIVEPK  
CREATE QDC TITLE = (BACKUP)LIVEDB ON BACKUPPK  
FAMILY LIVEPK = BACKUPPK")  
FILE CF (TITLE = (LIVE)LIVEDB/CONTROL ON LIVEPK);  
FILE CFOLD (TITLE = (LIVE)LIVEDB/CONTROL ON LIVEPK)
```

Example 8

This example creates a quiesce database copy instance of *LIVEDB* under a permanent directory on *LIVEPK*. In this example, the copy is to a different path name located on *BACKUPPK* as a source to perform an offline backup of *LIVEDB*:

1. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB = LIVEDB QUIESCE  
QDC(TITLE=*DIR/BACKUP/LIVEDB ON BACKUPPK)");  
DATAPATH = *DIR/LIVE ON LIVEPK
```

2. Split your mirrored disks containing all of your database files.
3. Enter the following command.

```
RUN *SYSTEM/DMUTILITY("DB = LIVEDB RESUME");  
DATAPATH = *DIR/LIVE ON LIVEPK
```

4. Rename and acquire the split mirrored disks.

5. Enter the following command. This command must be executed from a privileged usercode:

```
RUN *SYSTEM/DMCONTROL("DB=LIVEDB  
CREATE QDC TITLE = *DIR/BACKUP/LIVEB  
ON BACKUPPK FAMILY LIVEPK = BACKUPPK")
```

6. Enter the following command:

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB  
OFFLINE DUMP=TO LIVEDDBACKUP");  
DATAPATH = *DIR/BACKUP ON BACKUPPK
```

RESTORE FROM QDC Command (DMCONTROL)

The RESTORE FROM QDC command is a keyed feature. Refer to the *Software Product Catalog* for information about Quiesce Database Copy (QDC) as a Recovery Source.

Purpose

Use the RESTORE FROM QDC command to restore the original configuration of a quiesce database copy. You can use the restored copy as a recovery source for a database REBUILD operation of the live database. The DMCONTROL RESTORE FROM QDC command

- Verifies that all database files exist under the quiesce database copy usercode on the family locations designated in the live database control file.
- Restores the live database control file from the saved copy of the control file.
- Restores the original usercodes of the live database data files or, if it is a permanent directory database, the command will restore the original path name of the live database data files.
- Must be used with CF file equation. Refer to the example for additional information.

Note: You must issue the DMCONTROL RESTORE FROM QDC command from a privileged usercode. Executing the command restores the original settings of the live database options (Remote Database Backup, DMDUMPDIRECTORY, and TPS) to their original state at the time of the quiesce database copy creation.

To rebuild your live database using a quiesce database copy, perform the following steps:

1. Take the original live database disks offline.
2. Relabel the quiesce database copy disks.
3. Run the DMCONTROL RESTORE FROM QDC command.
4. Run the DMUTILITY REBUILD FROM QUIESCE DB command.

Note: Disk configuration, disk mirroring, and disk management are external activities to an Enterprise Database Server system.

Syntax

— RESTORE FROM QDC <QDC title clause> —————|

Example 1

[Figure 14-6](#) illustrates the results of performing the following procedure.

To rebuild (LIVE)LIVEDB using the quiesce database copy (TEST)LIVEDB, perform the following steps:

1. Take the LIVEPK disks offline.
2. Relabel the TESTPK disks to the LIVEPK family of disks.
3. Enter the following commands:

```
RUN $SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB ON
LIVEPK RESTORE FROM QDC TITLE=(TEST)LIVEDB ON TESTPK");
FILE CF(TITLE=(TEST)LIVEDB/CONTROL ON LIVEPK);
```

```
RUN $SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
RECOVER (REBUILD THRU AUDIT 1990) FROM QUIESCE DB")
```

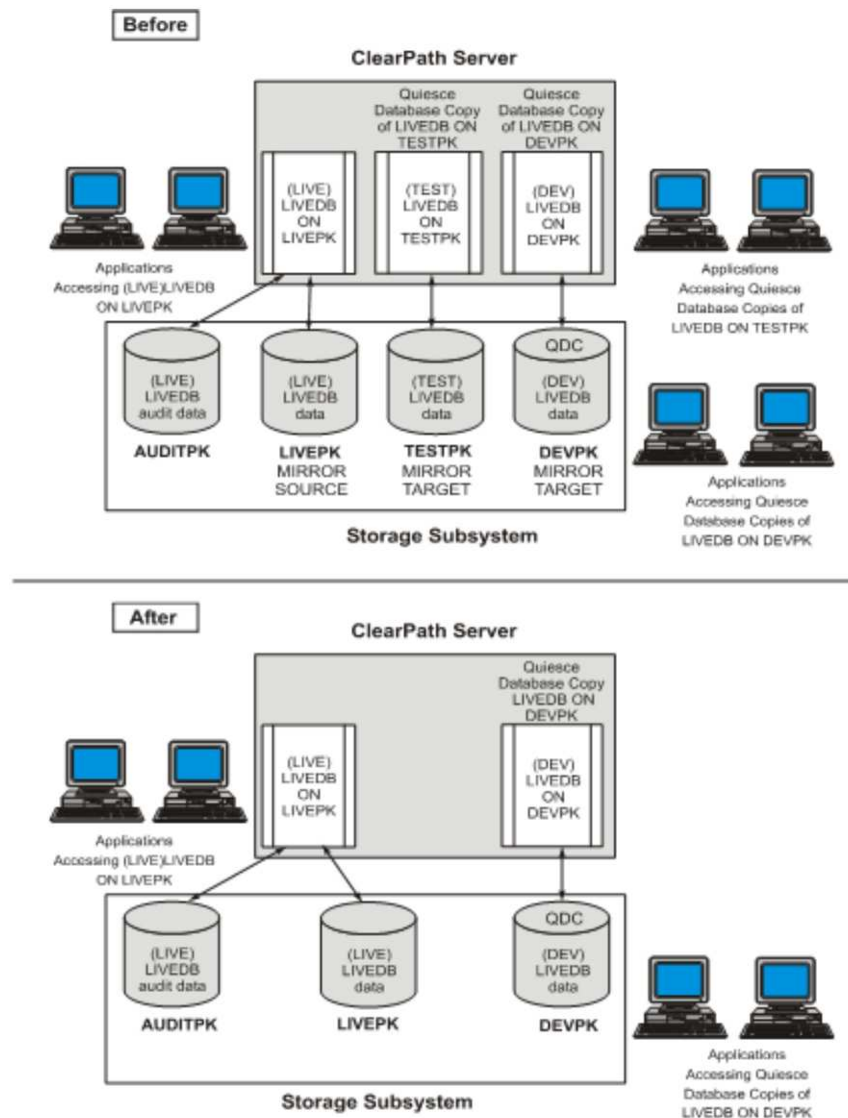
Example 2

To rebuild *DIR/LIVE/LIVEDB on LIVEPK using the quiesce database copy *DIR/BACKUP/LIVEDB on BACKUPPK, perform the following steps:

1. Take the LIVEPK disks offline.
2. Relabel the TESTPK disks to the LIVEPK family of disks.
3. Enter the following commands:

```
RUN $SYSTEM/DMCONTROL("DB=LIVEDB RESTORE FROM QDC
TITLE=*DIR/BACKUP/LIVEDB ON TESTPK");
FILE CF(TITLE=*DIR/BACKUP/LIVEDB/CONTROL ON LIVEPK);
```

```
RUN $SYSTEM/DMUTILITY("DB=LIVEDB RECOVER
(REBUILD THRU AUDIT 1990) FROM QUIESCE DB");
DATAPATH = *DIR/LIVE ON LIVEPK
```

008389

Figure 14–6. Rebuild of a Live Database Using a Quiesce Database Copy

Creating Incremental/Accumulated Dumps from a Quiesce Database

Use the following DMUTILITY QUIESCE options to configure a database image to be used for incremental and accumulated dumps for recovering the live database:

```
FOR FULLDUMP
FOR ACUDUMP
FOR INCUDUMP
```

Syntax

```
QUIESCE FOR FULLDUMP
QUIESCE FOR ACUDUMP
QUIESCE FOR INCDUMP
QUIESCE QDC <qdc specification> FOR FULLDUMP
QUIESCE QDC <qdc specification> FOR ACUDUMP
QUIESCE QDC <qdc specification> FOR INCDUMP
```

Using a Quiesce Database Copy as a Recovery or a Copy Source

A quiesce database copy can be used as a <recover source> construct in the RECOVER statement or as a <copy source> construct in the COPY statement. Refer to [Section 8, Recovering the Database](#), for the syntax of the RECOVER statement or the COPY statement.

When the (QDC <QDC title clause>) is specified as a recovery source, consider the following:

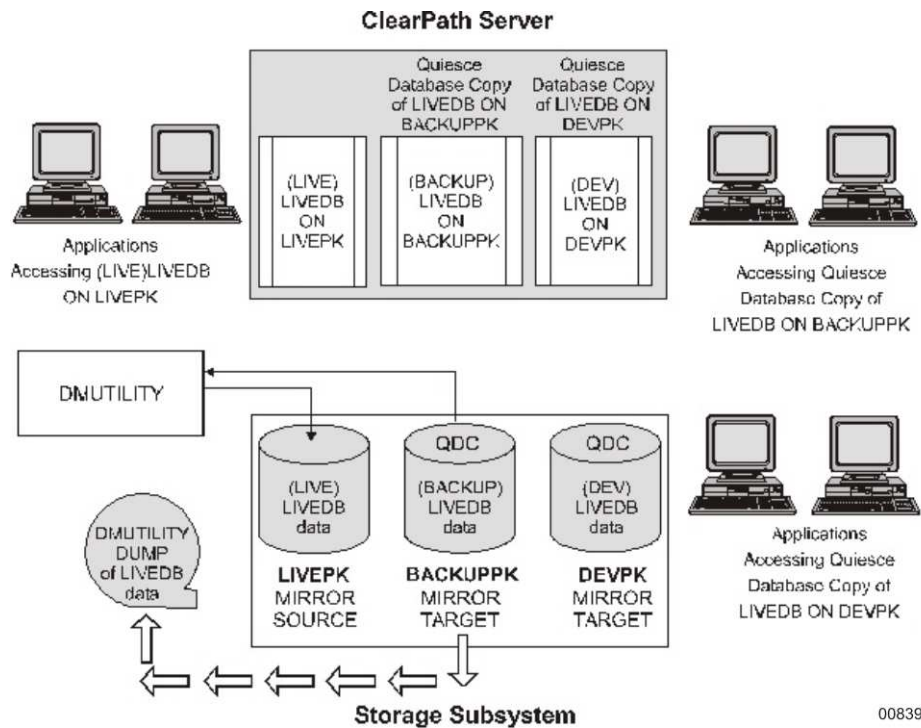
- Other applications can be using the quiesce database copy as the recovery is being performed.
- The RECOVER statement can specify REBUILD or RECONSTRUCT.
- No other recovery source can be used when QDC (<QDC title clause>) is specified.
- Incremental or accumulated backups cannot be used when QDC (<QDC title clause>) is specified.
- The QDCVERIFY option detects CHECKSUM and ADDRESSCHECK errors on the selected data during the recovery process.
- The QDCWORKERS option indicates that multiple processes can occur in parallel during the recovery process.
- The recovery can be restarted.
- A quiesce database copy cannot be resumed when being used as a recovery source or a copy source.

When the (QDC <QDC title clause>) is specified as a copy source, consider the following:

- No other copy source can be used when QDC (<QDC title clause>) is specified.
- The QDCVERIFY option detects CHECKSUM and ADDRESSCHECK errors on the selected data during the copy process.
- The QDCWORKERS option indicates that multiple processes can occur in parallel during the copy process.
- A quiesce database copy cannot be used as a source for copying the control file by itself. Use the CFRESTORE command to copy a control file.

Note: A quiesce database copy cannot be used in the REPLICATE command.

[Figure 14–7](#) illustrates the scenario on which examples 1 through 8 are based.



008393

Figure 14-7. Using a Quiesce Database Copy as a Recovery Source

Example 1

```

RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK OPTIONS
(QDCVERIFY, QDCWORKERS=30)RECOVER (ROWS USING BACKUP)=
(PACKNAME=LIVEPK @ FAMILYINDEX=3)FROM QDC(TITLE=(BACKUP)
LIVEDB ON BACKUPPK)")

```

Or, if it is a permanent directory database

```

RUN *SYSTEM/DMUTILITY("DB=LIVEDB OPTIONS(QDCVERIFY,
QDCWORKERS=30)RECOVER (ROWS USING BACKUP)=(PACKNAME=LIVEPK
@ FAMILYINDEX=3)FROM QDC(TITLE=*DIR/BACKUP/LIVEDB
ON BACKUPPK)");DATAPATH=*DIR/LIVE ON LIVEPK

```

These commands recover all data that resides on FAMILYINDEX 3 of pack family LIVEDB from quiesce database copy (BACKUP)LIVEDB ON BACKUPPK. During the recovery, 30 parallel tasks are processed and CHECKSUM and ADDRESSCHECK errors are detected.

Example 2

```

RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
OPTIONS(QDCVERIFY, QDCWORKERS=50)RECOVER(REBUILD
THRU AUDIT 1959) FROM QDC (TITLE=(BACKUP)LIVEDB
ON BACKUPPK)")

```

Or, if it is a permanent directory database

Using a Quiesce Database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB OPTIONS(QDCVERIFY,
QDCWORKERS=50)RECOVER (REBUILD THRU AUDIT 1959)FROM
QDC(TITLE=*DIR/BACKUP/LIVEDB ON BACKUPPK)");
DATAPATH=*DIR/LIVE ON LIVEPK
```

These commands recover all the data on (LIVE)LIVEDB ON LIVEPK from quiesce database copy (BACKUP)LIVEDB ON BACKUPPK. During the recovery, 50 parallel tasks are processed and CHECKSUM and ADDRESSCHECK errors are detected. In addition, audit files through 1959 are used to update the data on (LIVE)LIVEDB ON LIVEPK.

Example 3

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK OPTIONS
(QDCVERIFY, QDCWORKERS=50)COPY=AS(LIVE)=ON LIVEPK FROM
QDC(TITLE=(BACKUP)LIVEDB ON BACKUPPK) ")
```

Or, if it is a permanent directory database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB OPTIONS(QDCVERIFY, QDCWORKERS=50)
COPY=AS *DIR/LIVE/= ON LIVEPK FROM QDC (TITLE=*DIR/BACKUP/LIVEDB
ON BACKUPPK) "); DATAPATH=*DIR/LIVE ON LIVEPK
```

These commands copy all the data from quiesce database copy (BACKUP)LIVEDB ON BACKUPPK to (LIVE)=ON LIVEPK. During the copy, 50 parallel tasks are processed and CHECKSUM and ADDRESSCHECK errors are detected.

Example 4

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
RECOVER (ROWS USING BACKUP)=(ROWLOCK=LOCKEDROW,READERROR)
FROM QDC(TITLE=(BACKUP)LIVEDB ON BACKUPPK) ")
```

Or, if it is a permanent directory database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB RECOVER (ROWS USING BACKUP)
=(ROWLOCK=LOCKEDROW,READERROR)FROM QDC(TITLE=*DIR/BACKUP/LIVEDB
ON BACKUPPK) "); DATAPATH=*DIR/LIVE ON LIVEPK
```

These commands recover all rows having write operation errors (ROWLOCK = LOCKEDROW) or read operation errors (ROWLOCK = READERROR), using the data on the specified quiesce database copy plus the changes recorded in the audit since the time of the quiesce database copy creation. Because the QDCWORKERS clause is not specified, one process is assumed.

Example 5

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RECOVER
(ROWS USING BACKUP)LIVEDB/D/DATA FROM QDC(TITLE=(BACKUP)LIVEDB
ON BACKUPPK) ")
```

Or, if it is a permanent directory database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB RECOVER (ROWS USING BACKUP)
*DIR/LIVE/LIVEDB/D/DATA FROM QDC(TITLE=*DIR/BACKUP/LIVEDB
ON BACKUPPK) "); DATAPATH=*DIR/LIVE ON LIVEPK
```

These commands recover only rows in the file LIVEDB/D/DATA that have copies on the quiesce database copy. Resident rows allocated since the time of the quiesce database copy was created are also recovered. Because the QDCWORKERS clause is not specified, one process is assumed.

Example 6

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RECOVER (ROWS
USING BACKUP WITH FILTERING(WORKERS=3)) = (ROWLOCK=READERROR)
FROM QDC(TITLE=(BACKUP)LIVEDB ON BACKUPPK)")
```

Or, if it is a permanent directory database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB RECOVER (ROWS USING BACKUP WITH
FILTERING(WORKERS=3)) = (ROWLOCK=READERROR) FROM
QDC(TITLE=*DIR/BACKUP/LIVEDB ON BACKUPPK)");
DATAPATH = *DIR/LIVE ON LIVEPK
```

This command recovers all rows marked as having read operation errors and invokes the DMRECONFILTER utility to filter the audit information necessary for the reconstruction. Three filter workers are used in parallel to filter the audits generated since the quiesce database copy was created. Because the QDCWORKERS clause is not specified, one process is assumed.

Example 7

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RECOVER
(ROWS USING BACKUP) LIVEDB/D/DATA (RESTORE) FROM
QDC(TITLE=(BACKUP)LIVEDB ON BACKUPPK)")
```

Or, if it is a permanent directory database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB
RECOVER (ROWS USING BACKUP)
LIVEDB/D/DATA (RESTORE)
FROM QDC(TITLE=*DIR/BACKUP/LIVEDB ON BACKUPPK)");
DATAPATH = *DIR/LIVE ON LIVEPK
```

These commands restore all rows of the file LIVEDB/D/DATA, using the data on the quiesce database copy, and any changes recorded in the audit through the most current audit file. Because the QDCWORKERS clause is not specified, one process is assumed.

Example 8

The following commands restore the control of the live database from its quiesce database copy and initiate a REBUILD process.

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK CFRESTORE FROM
QDC(TITLE = (BACKUP)LIVEDB ON BACKUPPK)")
```

Or, if it is a permanent directory database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB CFRESTORE FROM
QDC(TITLE=*DIR/BACKUP/LIVEDB ON BACKUPPK)");
DATAPATH=*DIR/LIVE ON LIVEPK
```

Using a Quiesce Database

The preceding commands copy the control file of the quiesce database copy to the live database.

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK RECOVER(REBUILD THRU
AUDIT 1959 FROM QDC(TITLE = (BACKUP)LIVEDB
ON BACKUPPK) ")
```

Or, if it is a permanent directory database

```
RUN *SYSTEM/DMUTILITY("DB=LIVEDB RECOVER(REBUILD THRU
AUDIT 1959 FROM QDC(TITLE = *DIR/BACKUP/LIVEDB ON BACKUPPK) ");
DATAPATH=*DIR/LIVE ON LIVEPK
```

The preceding commands initiate a whole database recovery using the REBUILD process and the quiesce database copy is the recovery source.

High Availability QUIESCE

Purpose

Use the high availability QUIESCE methods to minimize the impact of the Quiesce operation on the transactional activity of the live database. You can use high availability Quiesce images to offload activities such as backup, recovery, certification and data warehousing.

Syntax

The following syntax elements are used by DMUTILITY and DMRECOVERY for high availability Quiesce configuration. All syntax elements are supported for single server and multiple server Quiesce methods.

DMUTILITY QUIESCE Syntax

——<quiesce option> —— QUIESCE —————|

<quiesce option>

```
——OPTIONS ——(——/1\— QTYPE=——,——CUSTOM——)——|
                   |——SPT——|
                   |——/1\— QTIMEOUT=<integer>——|
                   |——/1\— QACTION=——|
                   |——FORCE——|
                   |——ABORT——|
```

DMRECOVERY Syntax

——<recover command> ——|

<recover command>

—— FINALIZEQUIESCE ——|

<quiesce option>

The QTYPE = CUSTOM option ensures that the following actions occur:

- A QDCREC audit record is written to store information about the quiesce when this command is executed.
- Information is stored in the database control file indicating that this is a custom quiesce and that a recover FINALIZEQUIESCE action is required to complete the configuration.
- Transactional activity continues uninterrupted during the period of quiesce.
- A QDCREC audit record is written to store information about the RESUME when this command is executed
- The quiesce image cannot be finalized, rolled back, or rebuilt until the RESUME command is executed on the live database.

The QTYPE=SPT option ensures that the following actions occur:

- The DMUTILITY program waits until a natural control point occurs based on SYNCPOINT and CONTROLPOINT specifications.
- When the next control point occurs, the database system recognizes the pending QUIESCE request and automatically creates two control points to ensure the externalization of all buffers from memory to disk.
- The QUIESCE timestamp is stored in the database control file.
- The DMUTILITY program completes with the message "DATABASE QUIESCED."
- The live database remains in a quiesced state until you use the DMUTILITY RESUME command.

The QTIMEOUT = <integer> option indicates the wait time in seconds. This ensures the following actions occur:

- When the DMUTILITY program QTIMEOUT value is exceeded for the QTYPE=SPT, then a specific action is automatically taken based on the QACTION specification.
- QACTION must be specified when QTIMEOUT is specified.

In the QACTION = FORCE or QACTION = ABORT option, specifying FORCE causes the following to occur:

- After the QTIMEOUT is exceeded and the database has not been quiesced, the QUIESCE request will force transactional activity to suspend in the manner of a default quiesce request. Existing transactions will complete while new transaction requests are queued. After all active transactions are complete, two audited control points occur and the database becomes quiesced.

Specifying ABORT causes the following to occur:

- Automatic termination of the DMUTILITY QUIESCE request occurs, and database activity proceeds.

DMRECOVERY <recover command>

The FINALIZEQUIESCE command uses the audit file(s) to synchronize the quiesce database copy resulting in a database image that is both logically and physically consistent. The finalized image represents a point in time that is the most recent control point that occurred prior to the execution of the QUIESCE command.

Note: *The recovery process attempts to read audit files that are stored under the usercode of the running recovery program. If this usercode is different from the usercode of the live database, then perform one of the following*

- *Copy audit files to the desired usercode and family. If the family is different from the live database audit family, then the DMCONTROL AUDITFAMILY command should be issued for the QUIESCE database control file to record the new family.*
- *Respond to the NO FILE waiting entry of the recovery process with an FA redirection to the live database copy of the audit file.*

Example 1[QTYPE = CUSTOM, 2-Servers]

In this example, the live database is (LIVE)LIVEDB ON LIVEPK at Server 1 and a Quiesce snapshot is created as (LIVE)LIVEDB ON LIVEPK at Server 2. The point of QUIESCE will be determined by the DMRECOVERY FINALZEQUIESCE function. A list of steps required for this configuration is:

1. Quiesce the live database by entering the following command at the live system:

```
RUN *SYSTEM/DMUTILITY ("DB=(LIVE)LIVEDB
ON LIVEPK OPTIONS(QTYPE=CUSTOM)QUIESCE")
```

2. Split your mirrored disks that contain all of the database files on the live pack, LIVEPK.

3. Resume the live database at Server 1:

```
RUN *SYSTEM/DMUTILITY ("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Copy or mirror the audit files needed for a recovery to the appropriate pack as configured in the database control file at Server 2.

5. Acquire the split mirrored disks as LIVEPK at Server 2.

6. Finalize the configuration at Server 2:

```
RUN *SYSTEM/DMRECOVERY ("DB=(LIVE)LIVEDB
ON LIVEPK FINALIZEQUIESCE")
```

Example 2 [QTYPE = CUSTOM, 1 Server]

In this example, the live database is (LIVE)LIVEDB ON LIVEPK and a quiesce database copy is created as (TEST)LIVEDB ON TESTPK. The point of quiesce is determined by the DMRECOVERY FINALIZEQUIESCE function. A list of steps required for this configuration is:

1. Quiesce the live database by entering the following command:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK
OPTIONS(QTYPE=CUSTOM)QUIESCE QDC(TITLE=(TEST)
LIVEDB ON TESTPK)")
```

2. Split your mirrored disks that contain all of the database files on the live pack, LIVEPK.
3. Resume the live database:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK RESUME")
```

4. SM AUDIT CLOSE the live audit.
5. Copy the audit files needed for a recovery as

```
(TEST)LIVEDB/= on the QDC audit pack.
```

6. Enter the following command:

```
RUN *SYSTEM/DMCONTROL("DB=(LIVE)LIVEDB
ON LIVEPK CREATE QDC TITLE=(TEST)LIVEDB
ON TESTPK FAMILY LIVEPK=TESTPK")
```

The DMCONTROL CREATE QDC command must be executed from a privileged usercode or the DMCONTROL code file must be marked as a privileged code file.

7. Finalize the configuration. From the TEST usercode:

```
RUN *SYSTEM/DMRECOVERY ("DB=(TEST)LIVEDB
ON TESTPK FINALIZEQUIESCE")
```

Example 3 [QTYPE = SPT; 2-Servers]

In this example, the live database is (LIVE)LIVEDB ON LIVEPK at Server 1 and a Quiesce snapshot is created as (LIVE)LIVEDB ON LIVEPK at Server 2. The point of quiesce will be determined by the next SYNCPOINT that occurs following the execution of the QUIESCE command. A list of steps required for this configuration is:

1. Quiesce the live database by entering the following command at the live system:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON
LIVEPK OPTIONS(QTYPE=SPT) QUIESCE")
```

2. Split your mirrored disks that contain all of the database files on the live pack, LIVEPK.
3. Resume the live database at Server 1:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK RESUME")
```

4. Copy or mirror the audit files needed for a recovery to the appropriate pack as configured in the database control file at Server 2.
5. Acquire the split mirrored disks as LIVEPK at Server 2.

Example 4 [QTYPE =SPT,QTIMEOUT=5,QACTION=FORCE; 2 Servers]

In this example, the live database is (LIVE)LIVEDB ON LIVEPK at Server 1 and a Quiesce snapshot is created as (LIVE)LIVEDB ON LIVEPK at Server 2. The point of quiesce is determined by the next natural SYNCPOINT that occurs following the execution of the QUIESCE command and after all active transactions have completed. If the quiesce does not complete in less than 5 sections, the quiesce begins suspending transactions while allowing existing transactions to complete prior to the creation of the quiesce point. A list of steps required for this configuration is as follows:

1. Quiesce the live database by entering the following command at the live system. Note that if 5 seconds is exceeded and all transactions have not completed, then DMUTILITY continues with step 2:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK OPTIONS(QTYPE=SPT,QTIMEOUT=5,
QACTION=FORCE) QUIESCE")
```

2. After the database is quiesced, split the mirrored disks that contain all of the database files on the live pack.
3. Resume the live database at Server 1:

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK RESUME")
```

4. Acquire the split mirrored disks as LIVEPK at Server 2.

QUIESCE HISTORY Option of the WRITE Command

Purpose

Use the QUIESCE HISTORY option of the WRITE command to display the quiesce history information for a database. Each history record reflects information that was gathered between the execution of the QUIESCE command on the database and the execution of the RESUME command on the same database. The nine most recent history records are displayed with the following information:

- The title of the database
- The creation timestamp of the database
- The transaction wait time, reported in microseconds. This time reflects the amount of wait time until all of the active transactions are completed prior to the flush of data.
- The flush wait time, reported in microseconds. This time reflects the amount of time that was taken to flush the data from memory to disk.
- The resume wait time, reported in seconds. This time reflects the amount of time between the completion of the QUIESCE operation and the completion of the RESUME operation. This includes the time required to split mirrors.

Syntax

```
WRITE [LIST] QUIESCE HISTORY _____|
```

Example

This example illustrates the use of the QUIESCE HISTORY command.

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB
ON LIVEPK WRITE QUIESCE HISTORY")
```

CFRESTORE Command (DMUTILITY)

Purpose



Use the CFRESTORE command to restore the control file of a live database from its quiesce database copy.

Syntax

```
— CFRESTORE — FROM — QDC — ( <QDC title clause> ) —————|
```

Example

This example illustrates the use of the CFRESTORE command.

```
RUN *SYSTEM/DMUTILITY("DB=(LIVE)LIVEDB ON LIVEPK CFRESTORE FROM
QDC(TITLE = (BACKUP)LIVEDB ON BACKUPPK)")
```

Quick-Reference Information

Introduction

The information presented here is for quick-reference purposes only. For an explanation of any element of a syntax diagram, refer to the appropriate information presented earlier in this section.

QUIESCE Command

```
—QUIESCE —————|
```

RESUME Command

```
— RESUME —————|
```

QUIESCE QDC Command

```
—QUIESCE——— QDC (——<QDC title clause>——) —————|
```


Section 15

Using Database Tape Encryption

Enterprise Database Server dump files and audit files can be encrypted using database tape encryption. Data files can be encrypted when they are copied from disk to tape or from disk to disk as a part of a DMUTILITY DUMP operation. Audit files can be encrypted when they are copied from disk to tape as part of a COPYAUDIT QUICKCOPY operation.

Before you can use database tape encryption, your security administrator must establish server encryption keys by using the Security Center MMC snap-in on his or her workstation. Such a workstation can be a separate Windows-based PC or the Windows side of a ClearPath MCP server. Security Center refers to these keys as tape encryption keys. Refer to the *Security Overview and Implementation Guide* and the *Security Center Help* for additional information about configuring, exporting, and importing tape encryption keys.

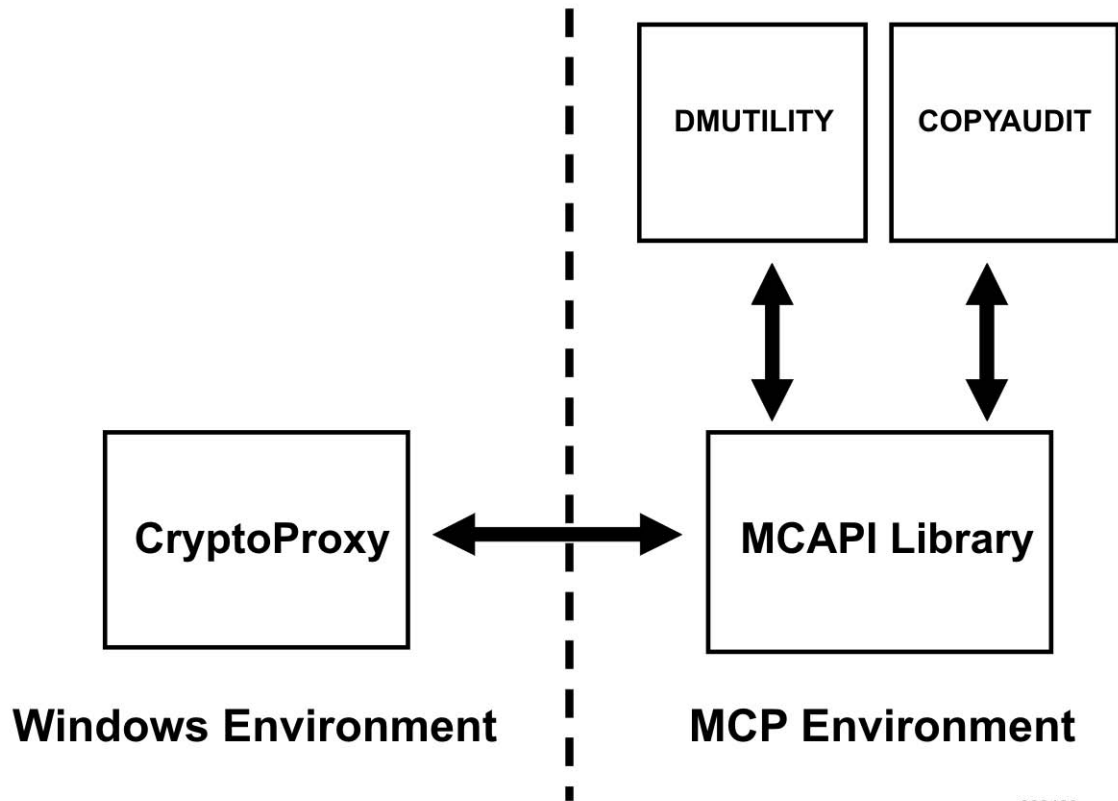
Software encryption and decryption affect the total time to transfer data and result in increased processor usage. Performance is affected by tape or disk drive throughput, the performance level of the MCP system, the performance level of the hardware doing the encryption, and competing workloads. Unisys recommends that you incrementally introduce encryption to your database environment to ensure that adequate resources are available to prevent encryption and decryption from having an adverse effect on your existing processes.

Encrypted files are automatically decrypted when copied to disk as part of DMUTILITY RECOVER, RESTORE, COPY, CLONE, and STRUCTURECLONE operations, and as part of COPYAUDIT QUICKCOPY operations.

Architecture

The following diagram shows a ClearPath MCP server environment that uses Microsoft CryptoProxy running in the Windows environment to encrypt and decrypt data needed by the DMUTILITY and COPYAUDIT software components. The MCPAPI library is the system library named MCAPISUPPORT. The code file for this library is named *SYSTEM/MCPCRYPTOAPI/SUPPORT.

ClearPath MCP Server



002480

Encryption Algorithms

The DMUTILITY and COPYAUDIT software components support the following encryption algorithms:

- Advanced Encryption Standard (AES), also known as AES256
- A variant of the Data Encryption Standard (DES) known as Triple DES (3DES or TDES)

The following table shows the Windows operating systems required to use each encryption algorithm.

Encryption Algorithm	Required Windows Operating System
AES256	Windows Server 2003
TDES	Windows Server 2003
AESGCM	Windows Server 2008 or Embedded Windows 7

DASDL Syntax

The following DASDL syntax options can be used for automatic encryption during a dump operation.

Option	Explanation
DUMPENCRYPT	This option can be set to TRUE or FALSE. The DUMPENCRYPT option can be set at the default global level for all data sets or for all sets, or the option can be specified at the structure level. The structure-level setting overrides the global-level setting.
ENCRYPTTYPE	This option controls the encryption algorithm to be used: TDES, AES256, or AESGCM. If no encryption algorithm is specified, TDES is used for encryption.

The following DASDL syntax option can be used for automatic encryption of audit files during a COPYAUDIT QUICKCOPY operation when COPYAUDIT is automatically zipped by an active database. The following option is available.

Option	Explanation
AUDITENCRYPT	This option controls the encryption algorithm to be used: TDES, AES256, or AESGCM. If no encryption algorithm is specified, TDES is used for encryption.

Refer to *Data Structure Definition Language (DASDL) Programming Reference Manual* for information about these options.

DMUTILITY Syntax

The following DMUTILITY syntax options can be used to override DASDL encryption options during a dump operation.

Option	Explanation
NOENCRYPT	This option can be set to TRUE or FALSE. The NOENCRYPT option has a global default value of FALSE. When the option is set to TRUE, the DASDL encryption specifications are overridden.
ENCRYPTTYPE	This option controls the encryption algorithm to be used and can be set to AES256, AESGCM, or TDES. TDES is used if no algorithm is defined in DMUTILITY or in DASDL.
DUMPENCRYPT	This option overrides the DUMPENCRYPT specification in DASDL.

Using Database Tape Encryption

You can specify DMUTILITY encryption options only for entire structure files.

The following dump types are not allowed:

- COPYDUMP
- DUPLICATE_DUMP
- Multidump tapes

The following syntax is not allowed:

- FAMILYINDEX=
- ROW=
- PACKNAME=
- SECTION=

Refer to [Section 6, Backing Up a Database](#), for syntax diagrams that support DMUTILITY encryption options.

COPYAUDIT Syntax

The following COPYAUDIT syntax option provides dynamic encryption during a COPYAUDIT QUICKCOPY operation.

Option	Explanation
AUDITENCRYPT	This option controls the encryption algorithm to be used: TDES, AES256 or AESGCM. If no encryption algorithm is specified, TDES is used for encryption.

Audit files that are encrypted to tape must be copied back to disk by the COPYAUDIT software before they can be processed by MCP and Enterprise Database Server software, such as DMRECOVERY, PRINTAUDIT, and others. In the process, files are automatically decrypted by the COPYAUDIT software.

Refer to [Section 9, Copying Audit Files](#), for syntax diagrams that support COPYAUDIT encryption options.

DASDL Example

The following example includes bolded encryption options. This example is used in “DMUTILITY Examples” later in this section.

```
OPTIONS
(
  AUDIT
  , INDEPENDENTTRANS
  , REAPPLYCOMPLETED
);
```



```

PARAMETERS
(
    SYNCWAIT=2
    ,ENCRYPTTYPE=AES256
);
DEFAULTS
(
    DUMPENCRYPT=FALSE
    ,PACK=DBPK
    ,DATA SET(DUMPENCRYPT=FALSE)
    ,SET(DUMPENCRYPT=FALSE)
);

AUDIT TRAIL
(
    PACKNAME=AUDITPK
    ALTERNATE IS TAPE
    ,QUICKCOPY MAXFILESPERTAPE=5 TO
    TAPE
    (
        ,AUDITENCRYPT
    )AND REMOVE
    ,BLOCKSIZE=300 WORDS
    ,AREASIZE=40
    ,AREAS=10
    ,UPDATE EOF=200
    CHECKSUM=TRUE
);

CUSTOMER DATASET%encrypt
(
    CUST-IDNUMBER(12);
    FIRSTNAMEALPHA(20);
    MIDDLENAMEALPHA(20);
    LASTNAMEALPHA(20);
    SS-NUMNUMBER(9);
)CHECKSUM=TRUE ,DUMPENCRYPT=TRUE;
CID SET OF CUSTOMER KEY IS CUST-ID% no encrypt
NO DUPLICATES DUMPENCRYPT=FALSE;

PRODUCT DATASET% no encrypt
(
    PROD-IDNUMBER(12);
    PRODNAMEALPHA(20);
)CHECKSUM=TRUE;
PID SET OF PRODUCT KEY IS PROD-ID% no encrypt
NO DUPLICATES;

```

DMUTILITY Examples

The following examples use the DASDL example described earlier.

Example 1

The following command dumps the entire database and encrypts only those structures in the DASDL specification that have the DUMPENCRYPT option set to TRUE:

```
DUMP=TO SUNDAYDUMP
```

Example 2

The following command dumps the entire database, encrypts only those structures in the DASDL specification that have the DUMPENCRYPT option set to TRUE, and overrides the ENCRYPTTYPE value of the DASDL specification:

```
OPTIONS(ENCRYPTTYPE=TDDES)DUMP=TO SUNDAYDUMP
```

Example 3

The following command dumps the entire database and encrypts every structure using the ENCRYPTTYPE value of the DASDL specification:

```
DUMP=(DUMPENCRYPT=TRUE) TO SUNDAYDUMP
```

Example 4

The following command dumps the entire database without encrypting structures and overrides all of the encryption values in the DASDL specification:

```
OPTIONS(NOENCRYPT)DUMP=TO SUNDAYDUMP
```

Example 5

The following command dumps the CUSTOMER and CID structures, encrypts both structures, and uses the ENCRYPTTYPE value in the DASDL specification:

```
DUMP PRODUCTIONDB/CUSTOMER/=(DUMPENCRYPT=TRUE) TO SUNDAYDUMP
```

Example 6

The following command dumps the PRODUCT structure and overrides the ENCRYPTTYPE value in the DASDL specification:

```
OPTIONS(ENCRYPTTYPE=TDDES)DUMP  
PRODUCTIONSDB/PRODUCT/DATA(DUMPENCRYPT=TRUE) TO SUNDAYDUMP
```

Example 7

The following command dumps the entire database by family index, encrypts all the structures, and uses the ENCRYPTTYPE value in the DASDL specification:

```
DUMP=(DUMPENCRYPT=TRUE) BY FAMILYINDEX TO SUNDAYDUMP
```

Example 8

The following command dumps the entire database to a disk file and encrypts every structure using the ENCRYPTTYPE value in the DASDL specification:

```
DUMP=(DUMPENCRYPT=TRUE) TO DISKDUMP ON DBPACK
```

Example 9

The following command verifies the encrypted dump created in example 7:

```
VERIFYDUMP SUNDAYDUMP
```

COPYAUDIT Examples

Example 1

The following command copies and encrypts audit files 1 through 3 on pack AUDITPK to a tape and uses the TDES encryption algorithm. In addition, after each audit file is encrypted and copied, the tape is repositioned and the audit file is read in the forward direction to check for correctness.

```
QUICKCOPY PRODUCTIONDB/AUDIT1 THRU  
PRODUCTIONDB/AUDIT3 ALL FROM PACK=AUDITPK  
TO TAPE (AUDITENCRYPT=TDES)  
CHECK FORWARD COMPARE
```

Example 2

The following command appends and encrypts audit files 4 and 5 to the tape created in example 1:

```
QUICKCOPY APPEND PRODUCTIONDB/AUDIT4-  
PRODUCTIONDB/AUDIT5 ALL FROM PACK=AUDITPK  
TO TAPE (DENSITY=FMTST9840, AUDITENCRYPT=TDES)  
CHECK
```

Example 3

The following command copies and encrypts audit file 1 to a tape drive with Locate Fast Access capability:

```
QUICKCOPY PRODUCTIONDB/AUDIT1 ALL FROM PACK=  
AUDITPK TO TAPESET (AUDITENCRYPT=AES256)
```

Example 4

The following command appends and encrypts audit file 2 to the tape set created in example 3:

```
QUICKCOPY PRODUCTIONDB/AUDIT2 FROM PACK=  
AUDITPK TO TAPESET 1 (AUDITENCRYPT=AES26)
```

Example 5

The following command copies and encrypts audit file 1 on pack AUDITPK to a tape. The audit is encrypted using the TDES encryption algorithm, which is the default algorithm.

```
QUICKCOPY PRODUCTIONDB/AUDIT1 ALL FROM PACK=AUDITPK TO TAPE  
(DENSITY=FMTST9840, AUDITENCRYPT)
```


Section 16

Using Permanent Directory Databases

Permanent directory databases provide a common database access without having to use common usercodes or chargecodes and provide security and control at both the macro and micro levels.

Permanent directory databases do not affect existing databases. The functionality is similar to the following directory structures:

- UNIX
- MS-DOS
- Windows

Permanent directory database names are prefixed with *DIR and can contain multiple nodes ahead of the node that identifies the database. For example, the database named MAIN-DB would have file names as follows:

```
*DIR/NODE1/NODE2/MAIN-DB/<data set name>/DATA
*DIR/NODE1/NODE2/MAIN-DB/<data set name>/<set name>
```

All DMUTILITY commands that need to locate a permanent directory database must be followed with a DATAPATH task attribute. This statement defines the location of the permanent directory database. For information on how the DB clause interacts with a permanent directory database, refer to [Appendix A, Common Syntactic Items](#). In the following example, the command indicates that the permanent directory database named SIMPLEDB is found in *DIR/DEMO/TESTENV1 on MYPK:

```
RUN $SYSTEM/DMUTILITY("DB=SIMPLEDB DUMP=TO MYUTILDUMP ON MYPK");
    DATAPATH=(MYUSERCODE)ON MYPK,*DIR/DEMO/TESTENV1 ON MYPK
```

When the DMDUMPDIR program is used, the dump directory files are placed under the usercode that initiated the task.

Creating a Permanent Directory Database

To create a permanent directory database, perform the following steps:

1. Determine whether a *DIR directory already exists on the disk on which the permanent directory is to reside.
2. If a *DIR directory does not exist, use the WFL *MKDIR* statement with a privileged usercode to explicitly create the *DIR directory on that disk. Otherwise, go to step 3.

Using Permanent Directory Databases

3. Use the WFL *MKDIR* statement with a privileged usercode to explicitly create an MCP permanent directory as the permanent directory. The directory can start at any node level. However, each higher level must either already exist or be created.

Note: A directory cannot contain both a file and a subdirectory with the same name.

4. Use the WFL *ALTER* statement to set the file attributes as follows:
 - PROPAGATESECURITYTODIRS file attribute to PROPAGATE
 - PROPAGATESECURITYTOFILES file attribute to PROPAGATE
 - ALTERNATEGROUPS file attribute to the appropriate R, W, and X values if you want to provide access for multiple group codes
 - GROUP file attribute to the appropriate R (read), W (write), and X (execute/traverse) values

Access to a file within the permanent directory namespace requires that X (traverse) permission be granted by each of the permanent directories containing the file. If the traverse permissions are granted by all containing permanent directories, then the actual file access permissions are determined by the file attributes in the same manner as for files that are not in the permanent directory namespace. If any of the traverse permissions are not granted, no access is permitted to the file.

5. Designate a datapath specification in DASDL. Refer to the control file attributes in the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for the exact syntax.
6. Compile or recompile all tailored Enterprise Database Server components.
7. For existing databases, manually copy the database structures and tailored software to the newly created permanent directory.

Examples

The following example shows how to create a permanent directory database with the GROUPS file attribute set:

```
BEGIN JOB SETUPDIR (STRING UNIQUE);
  STRING
    PATHNAME;

    PATHNAME := " *DIR/MEGAMART/ "  UNIQUE ON "MYDBPACK" ;
MYJOB (DATAPATH= ( #PATHNAME) );

MKDIR *DIR/MEGAMART;
ALTER *DIR/MEGAMART (
  PROPAGATESECURITYTODIRS =PROPAGATE,
  PROPAGATESECURITYTOFILES =PROPAGATE,
  GROUP=SALES,
  OWNERRWX =RWX,
  GROUPEX =RWX
);
END JOB;
```

The following example shows how to create a permanent directory database with the GROUP and ALTERNATEGROUPS file attributes set:

```
BEGIN JOB SETUPDIR (STRING UNIQUE);
  STRING
    PATHNAME;

    PATHNAME:= " *DIR/MEGAMART/"  UNIQUE ON "MYDBPACK";
MYJOB(DATAPATH=(#PATHNAME) );

MKDIR *DIR/MEGAMART;
ALTER *DIR/MEGAMART (
  PROPAGATESECURITYTODIRS=PROPAGATE,
  PROPAGATESECURITYTOFILES=PROPAGATE,
  GROUP=SALESMANAGEMENT,
  OWNERRWX =RWX,
  GROUPEWX =RWX,
  ALTERNATEGROUPS=
    "SALESMEN:X, SALESDISTRICT:R, SALESSUPPORT=RWX"
);
END JOB;
```

Reorganizing a Permanent Directory Database

When preparing to reorganize a permanent directory database, consider the following information:

- SYSTEM/BUILDREORG must be run to create the file DESCRIPTION/REORGANIZATION/<database name>.

This file and the REORGANIZATION program are not created in the permanent directory.

Unless the ZIP option is reset in the BUILDREORG specifications, the reorganization is compiled automatically when the BUILDREORG process is completed.

- When using the structure copy option, you must create the necessary permanent directories on the specified pack before running the reorganization.
- Do not use the INTERNAL FILES specification because the temporary files of the reorganization process for a permanent directory database are placed in the same permanent directory as the data structures. If the INTERNAL FILES specification is used, it is ignored.
- The RUN statement must include the correct DATAPATH as a task attribute. For example:

```
RUN REORGANIZATION/<database name>;
  DATAPATH=*DIR/<node-1>/node-n> ON <packname>
```

Because the DMSUPPORT code files for a permanent directory database reside in the permanent directory and their titles include the update level, no migration of the DMSUPPORT titles occur during a reorganization of the permanent directory database. In addition, the administrator must create a node for the temporary database prior to running the reorganization.

Caution

Enterprise Database Server software puts an updated or copied description file under the usercode of the task initiator. If you copy the description file to the permanent directory for easy access, be extremely cautious when doing DASDL updates and subsequent reorganizations. When you perform DASDL updates, Unisys recommends that you remove copies of the description file from the permanent directory, and that only the description files under the usercode of the database administrator be used.

By using file equation, DASDL can read the description file in the permanent directory, but the new description file is always created under the usercode of the initiator of the DASDL compilation. You can also request that the BUILDREORG utility read the description file in the permanent directory, but the DESCRIPTION/REORGANIZATION/<database name> file is still created under the usercode of the initiator. Unless the ZIP option is reset, the BUILDREORG utility also compiles the REORGANIZATION program under the usercode of the initiator.

When a reorganization is run, if there is a description file in the permanent directory as well as one under the usercode of the initiator, the reorganization process uses the description file in the permanent directory. However, the copies that the reorganization process makes of the description file and the reorganization code file are under the usercode of the initiator.

This condition also means that you cannot have a traditional database and a permanent directory database with the same database name created by the same usercode because both description files have the same title.

Working with Dumps

When using the DUMP command with a disk destination, the dump is always directed to a usercoded file. The destination cannot be a permanent directory.

Example

The following example indicates that the dump of the SIMPLEDB database is to be found in the permanent directory location *DIR/DEMO/TESTENV1 on the family named MYPK and placed in MYUTILDUMP on MYPK using the usercode associated with the initiating task:

```
RUN $SYSTEM/DMUTILITY("DB=SIMPLEDB DUMP=TO MYUTILDUMP ON MYPK");  
      DATAPATH=(MYUSERCODE)ON MYPK,*DIR/DEMO/TESTENV1 ON MYPK
```

When using the COPYDUMP and DUPLICATEDUMP commands, the copied or duplicated dump cannot be placed in a permanent directory.

Using a Quiesce Database Under a Permanent Directory

The following issues pertain to a quiesce database copy:

- A quiesce database copy must be configured to use the same level of software that the live database uses.
- A quiesce database copy can be configured to use its own database software code files by using the DMCONTROL statement with the <code file title change> command option and file-equating CF and CFOLD to the control file of the quiesce database copy.
- Once you have used the RESUME command to reactivate a quiesce database copy, it is no longer a quiesce database copy and operates as an independent non-related database. Dumps created from this resumed quiesce database copy cannot be used for recovery of the live database.
- A quiesce database copy cannot be resumed when being used as a recovery source or a copy source.
- Use the QDC title clause to register a quiesce database copy with the live database specified in the database statement. Use the DMCONTROL CREATE QDC command and specify the same QDC title clause to create the actual quiesce database copy.
- A quiesce database copy title must use the same database name as the database statement, but must use a different family name and different path name.
- The control file of the live database specified by the database statement is updated with the registration of the quiesce database copy. You can have up to 15 quiesce database copy registrations at a time.

Refer to [Section 14, Using a Quiesce Database](#), for additional information.

Section 17

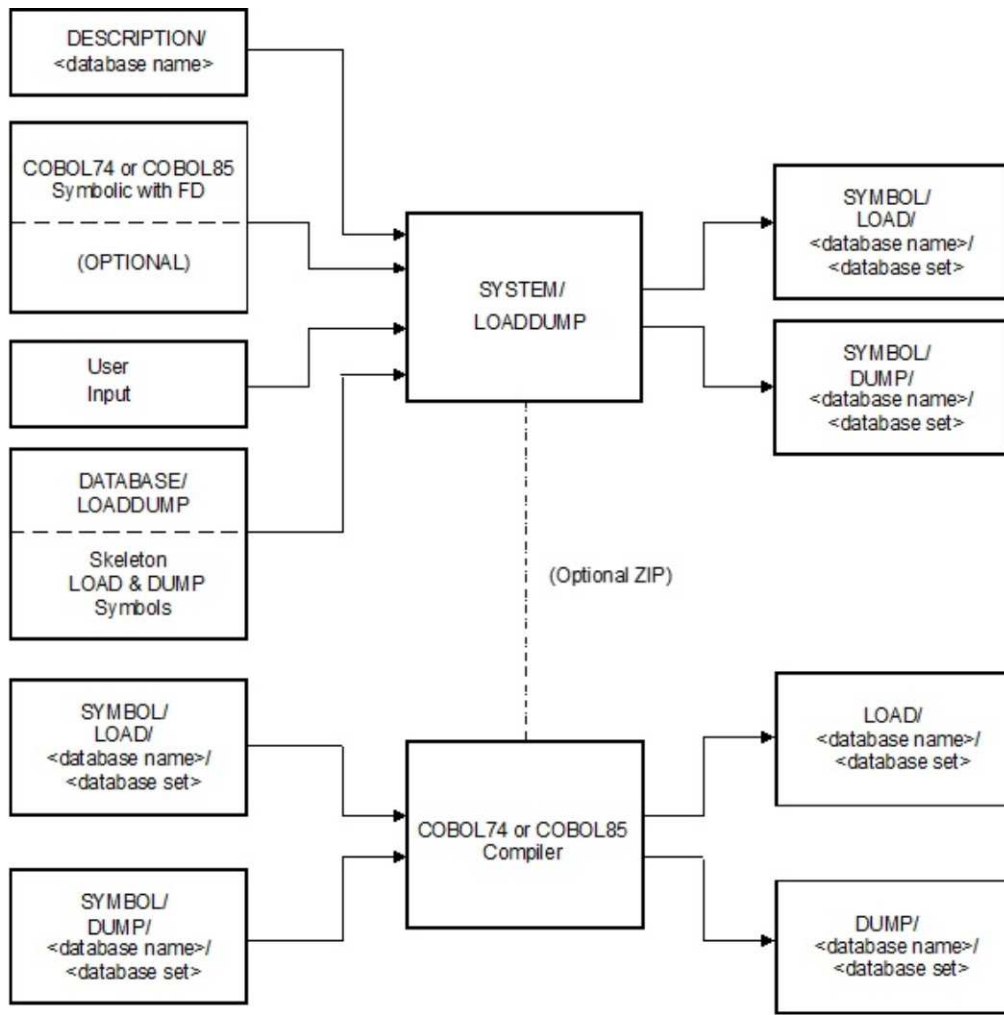
Loading and Dumping Conventional Files

SYSTEM/LOADDUMP generates programs that load a data set from a conventional file or dump a data set to a conventional file. SYSTEM/LOADDUMP requires the data set name and a file description name. Given this information, SYSTEM/LOADDUMP automatically generates and compiles COBOL74 or COBOL85 programs to perform the LOAD or DUMP. Only one copy of SYSTEM/LOADDUMP per usercode is permitted to be running at a time.

The conventional file is described using a COBOL74 or COBOL85 file description (FD). When performing a LOAD, FD items are transferred to the same named data set items. When performing a DUMP, data set items are transferred to the same named FD items.

If the names of the items in the data set and file description are not identical, they can be equated using the EQUATE statement.

The components of SYSTEM/LOADDUMP and the interrelationships among these components are illustrated in [Figure 17-1](#).



008391

Figure 17-1. LOADDUMP Components

Steps for Using LOADDUMP

Each run of SYSTEM/LOADDUMP generates LOAD and DUMP programs for one disjoint data set and a COBOL74 or COBOL85 FD. The user must do the following steps to create these programs:

1. SYSTEM/LOADDUMP must be run specifying the data set and corresponding FD. The FD must be contained in an external COBOL74 or COBOL85 symbolic or in the LOADDUMP specifications. Also, the user can specify a mapping of items that determines which data elements are transferred back and forth between the data set and the conventional file in loading and dumping.

If the LOADDUMP run is error-free, LOAD and/or DUMP symbolics are created.

2. The symbolics resulting from LOADDUMP can be compiled with the COBOL74 or COBOL85 compiler. Optionally, this compilation can be automatically initiated from

LOADDUMP. Syntax errors can result from these compilations, because LOADDUMP does not check the syntax of the COBOL74 or COBOL85 file description or the COBOL74 or COBOL85 MOVE statements resulting from the item mapping.

3. The user can edit the LOAD and/or DUMP symbolics to correct any syntax errors or add any desired features.
4. Once the LOAD and/or DUMP symbolics have been compiled, they can be run against the database. Various run-time errors, such as DATAERROR or DUPLICATES, can occur. LOADDUMP has no way of predicting such errors; however, the user can modify the generated source program and rerun it.

LOADDUMP

The code file for the LOADDUMP program is SYSTEM/LOADDUMP, and the symbol file is DATABASE/LOADDUMP. LOADDUMP generates programs that load a data set from a conventional file or dump a data set to a conventional file. The user provides the name of the data set and description of the conventional file. Given this information, SYSTEM/LOADDUMP automatically generates and compiles COBOL74 or COBOL85 programs to perform the load and dump.

LOADDUMP is not a tailored program, but instead reads and interprets the Data and Structure Definition Language (DASDL) description file at run time. The general format to run the LOADDUMP program is as follows:

```
? BEGIN JOB LOADDUMPRUN;
  RUN SYSTEM/LOADDUMP;
  DATA CARD
    .
    . <load dump specifications>
    .
? END JOB
```

The syntax for SYSTEM/LOADDUMP is illustrated and explained on the following pages.

Syntax

<load dump specifications>

—<database specification>—<file specification>—————→
 ▶┌──┐<load dump>—————┘
 └──┘
 └──┘
 └──┘

<database specification>

— DB ┌──┐ ; —————→
 └──┘
 └──┘
 └──┘
 └──┘
 └──┘

- Equation specification clause
- Load dump constant

Database Specification Clause

The following information explains the elements of the database specification clause.

Option	Explanation
<database specification>	Identifies the database and data set on which to use LOADDUMP.
DB	<p>Identifies the database or logical database that contains the data set or remap to be loaded or dumped.</p> <p>The description file associated with the database is assumed to</p> <ul style="list-style-type: none"> • Be in the same location as the database control file • Have the name DESCRIPTION/<database name> <p>If your description file is in another location or titled differently, when you run LOADDUMP, file-equate the description file title to the internal file name DASDL.</p>
DATASET	Specifies the name of the disjoint data set, global data, or remap for which LOADDUMP generates utilities. The data set, global data, or remap must be contained in the database or logical database specified in the DB clause. The RESTART data set or variable format data sets must not be specified.
EXCLUDE	<p>Informs the LOADDUMP program that the LOAD and/or DUMP programs must not transfer data into or out of the specified items. All data items listed in the EXCLUDE clause must be contained in the data set, global data, or remap specified by the DATASET clause. Only data items, virtual items, and group items can be specified in the EXCLUDE clause. Items other than data items, virtual items, or group items are implicitly excluded.</p>

File Specification Clause

The following information explains the elements of the file specification clause.

Option	Explanation
<file specification>	Defines the COBOL74 or COBOL85 description used during the LOADDUMP run.

Loading and Dumping Conventional Files

Option	Explanation
COBOL74	<p>Indicates COBOL74 as the compiler of choice for the compilation of code generated by the LOADDUMP utility. COBOL74 is the default compiler.</p> <p>Note: COBOL74 does not support DATE and TIME items. If DATE and/or TIME items are included in the LOADDUMP operations, COBOL85 must be specified for the compilation of code generated by the LOADDUMP utility.</p>
COBOL85	<p>Indicates COBOL85 as the compiler of choice for the compilation of code generated by the LOADDUMP utility.</p> <p>This option is required if DATE and/or TIME items are included in the LOADDUMP operations.</p>
<FD name>	<p>Specifies the name of the COBOL74 or COBOL85 file description. If the file description is to be extracted from COBOL74 or COBOL85 text, either included in card form with the LOADDUMP specifications or extracted from an existing COBOL74 or COBOL85 symbolic, there must be a valid file description with the specified <FD name>. If the file description is generated by LOADDUMP from the database description file, LOADDUMP assigns an <FD name> to the extracted file description.</p>
TITLE = <file title>	<p>Specifies the title of the external file to which the COBOL74 and COBOL85 programs write and read. This file title is placed in the file description program. The file title must be in control card format. If this option is specified, it overrides any title specifications in the original COBOL74 or COBOL85 text.</p>
RECORD	<p>Specifies the name of the 01 record name contained in the file description. The record contains the items that are transferred from the data set to the file and vice versa. If the file description is to be extracted from COBOL74 or COBOL85 text, either included in card form with the LOADDUMP specifications or extracted from an existing COBOL74 or COBOL85 symbol file, the record name must be a valid record name in that file description. If extraction is from the data set, LOADDUMP assigns a record name to the file description.</p>
SOURCE	<p>Specifies the location of the COBOL74 or COBOL85 text containing the file description for the specified <FD name>.</p> <p>LOADDUMP does not recognize or process COBOL74 or COBOL85 COPY statements; therefore, the user should always specify a SOURCE clause that contains the total file description.</p>

Option	Explanation
<COBOL74 text> <COBOL85 text>	<p>Consists of any COBOL74 or COBOL85 text that contains the FD and the SELECT statement for the file description to be extracted. Although COBOL74 or COBOL85 text can contain optional COBOL74 or COBOL85 source code, only the FD and the SELECT statement for the FD are extracted from the COBOL74 or COBOL85 text by LOADDUMP.</p> <p>LOADDUMP might skip certain clauses in these statements or add certain values needed for the resulting programs. In particular, clauses containing data names (instead of literals) must be deleted because LOADDUMP cannot determine what those values should be at run time. The user should verify that the resulting COBOL74 or COBOL85 declarations in the LOAD and DUMP symbolics are correct.</p>
SOURCE FOLLOWS	<p>Indicates the COBOL74 or COBOL85 source containing the FD must be specified directly in the LOADDUMP specifications as COBOL74 or COBOL85 text. LOADDUMP assumes that columns 1 through 72 contain COBOL74 or COBOL85 text until the END SOURCE statement is reached. The COBOL74 or COBOL85 text can be in free-field format using all of the columns. COBOL74 or COBOL85 sequence numbers should not be used, because they are assumed to be COBOL74 or COBOL85 format.</p>
SOURCE <source file title>	<p>Indicates the FD is contained in the external COBOL74 or COBOL85 symbolic titled <source file title>. This symbolic must contain the COBOL74 or COBOL85 text previously described and be in legitimate COBOL74 or COBOL85 format.</p>
SOURCE DATASET	<p>Informs LOADDUMP to extract a COBOL74 or COBOL85 FD from the specifications of the data set. No COBOL74 or COBOL85 text is used with this option. All data items, virtual items, and group items excluded from the data set with the EXCLUDE clause are excluded from the FD. The remaining items are included as follows:</p> <p>The order and identifiers of the items remain the same as in the DASDL specifications.</p> <p>The items are translated from DASDL item types to COBOL74 or COBOL85 item types with traditional syntax, except Boolean items, which are translated to X(1).</p>

Equation Specification Clause

The following information explains the elements of the equation specification clause.

Option	Explanation
<equation specification>	Establishes a correspondence between the items of the file and the items of the data set that do not have the same symbolic names. An equation specification can be specified only if the COBOL74 or COBOL85 file description is to be extracted from COBOL74 or COBOL85 text either included in card form with the LOADDUMP specifications or extracted from an existing COBOL74 or COBOL85 symbol file. An equation specification cannot be specified if the file description is generated from the data set.

Rules for Equation Specification

You must follow the following rules when performing an equation specification:

- Occurring items cannot be equated to nonoccurring items, and vice versa.
- Group items cannot be equated to elementary data items, and vice versa.
- The record name and data set name cannot be equated; their equation is implied.
- The <group item equation> construct must be specified to equate items contained in groups.
- The use of subscripts and COBOL74 or COBOL85 qualification is not allowed.
- Data set items excluded with the EXCLUDE clause cannot be equated.
- An Enterprise Database Server field of Boolean values is considered a group with the individual items subordinate to it.
- Items can be equated only one time.

Items can be explicitly equated using the equation specification, or implicitly equated by LOADDUMP. The LOAD and DUMP programs only move data between equated items. LOADDUMP uses the following algorithm to equate items:

- All equations specified in the equation specification are performed first.
- Implicit equation of all items not already equated is performed second.

Equation Criteria

A file item is implicitly equated to a data set item if it meets the following criteria:

- The items must have the same symbolic name.
- The items must not be equated with an equation specification.
- The data set items must not be excluded with the EXCLUDE clause.
- The following items can be equated only to items of the same type (for example, occurring items can be equated only to other occurring items):
 - Occurring items
 - Nonoccurring items

- Group items
- Elementary data items
- Both items must reside at the same level with respect to groups.
- If the items reside in groups, all possible corresponding qualifiers of the items must be either identical or equated using the EQUATE clause.

Examples of Equation Specification

Assume that the following COBOL74 or COBOL85 file description (FD) and Enterprise Database Server data set are to be used during a LOADDUMP run:

COBOL74 or COBOL85 FD	DMSII DATA SET
<pre> FD F-D. 01 REC-NAME. 03 A PIC X(10). 03 B-Y ... 03 C. 06 C-1 ... 06 C-2 ... 03 D ... 03 E. 06 E-1 ... 06 E-2 ... 03 F OCCURS ... 06 F-1 ... 06 F-2 ... 03 G. 06 G-1 ... 06 G-2 ... 03 H. 06 H-1 ... 06 H-2. 09 H-2-1 ... 09 H-2-2 ... 06 H-3-Y ... 03 I PIC 9(6) OCCURS ... 03 J-Y. 06 J-1 PIC X(2). 06 J-2 PIC X(6). </pre>	<pre> D-S DATA SET (A ALPHA (10); B ...; C GROUP (C-1 ...; C-2 ...;); D ...; E GROUP (E-1-X ...; E-2-X ...;); F GROUP OCCURS ... (F-1 ...; F-2 ...;); G GROUP OCCURS ... (G-1 ...; G-2 ...;); H GROUP (H-1 ...; H-2 GROUP (H-2-1 ...; H-2-2 ...;); H-3 ...;); I REAL OCCURS ...; J FIELD (J-1; J-2;);); </pre>

- If no equation specification is declared, the following implicit correspondence is assumed:

COBOL FD	DATA SET
-----	-----

Loading and Dumping Conventional Files

A	A
C	C
C-1	C-1
C-2	C-2
D	D
E	E
F	F
-1	F-1
F-2	F-2
H	H
H-1	H-1
H-2	H-2
H-2-1	H-2-1
H-2-2	H-2-2
I	I

The following associations are not made:

COBOL FD	DATA SET	EQUATE CRITERIA VIOLATED
-----	-----	-----
B-Y	B	DIFFERENT SYMBOLIC NAMES
E-1	E-1-X	DIFFERENT SYMBOLIC NAMES
E-2	E-2-X	DIFFERENT SYMBOLIC NAMES
G	G	BOTH ITEMS DO NOT OCCUR
G-1	G-1	QUALIFIERS NOT IDENTICAL OR EQUATED
G-2	G-2	QUALIFIERS NOT IDENTICAL OR EQUATED
H-3-Y	H-3	DIFFERENT SYMBOLIC NAMES
J-Y	J	DIFFERENT SYMBOLIC NAMES
J-1	J-1	QUALIFIERS NOT IDENTICAL OR EQUATED
J-2	J-2	QUALIFIERS NOT IDENTICAL OR EQUATED

- In addition to the implicit equations shown in the previous example, the following equation specification can be specified:

```

EQUATE
B-Y = B,
E   = E (
           E-1 = E-1-X,
           E-2 = E-2-X
        ),
H   = H (
           H-3-Y = H-3
        ),
J-Y = J;

```

- The following equation specification illustrates additional features and restrictions of the EQUATE clause:

```

EQUATE
A = A,           % LEGAL BUT UNNECESSARY
B-Y = B,        % NECESSARY TO MAKE THIS ASSOCIATION
C = D,          % ILLEGAL - GROUP ITEM CANNOT BE
                EQUATED TO ELEMENTARY ITEM
E = E (         % NECESSARY TO EQUATE SUBORDINATES
           E-1 = E-1-X,
           E-2 = E-2-X
        ),
F = F,          % LEGAL BUT UNNECESSARY

```

```

G = G,                % ILLEGAL - OCCURRING ITEMS CANNOT BE
                      EQUATED TO NONOCCURRING
                      ITEMS
H = H (              % NECESSARY TO EQUATE H-3-Y = H-3
    H-3-Y = H-3
),
I = D,                % ILLEGAL - OCCURRING ITEMS CANNOT BE
                      EQUATED TO NONOCCURRING
                      ITEMS
J-Y = J (            % NECESSARY TO EQUATE GROUP
    J-1 = J-1,
    J-2 = J-2
);
    
```

LOADDUMP Construct

The following information explains the elements of the LOADDUMP construct.

Option	Explanation
<load dump>	Specifies which symbolics are to be created. The LOAD and DUMP symbolics are produced only if the LOADDUMP run has been error-free.
LOAD	Specifies a COBOL74 or COBOL85 symbolic that can be compiled into a program to load a data set from a conventional file is generated. This program performs an OPEN UPDATE on the database. If INDEPENDENTTRANS has been set for a database, SYSTEM/LOADDUMP generates a load program that performs a LOCK STRUCTURE on the data set to be loaded prior to entering transaction state. LOCK STRUCTURE is required to avoid a DEADLOCK exception if more than 50,000 records must be loaded into the data set during a single program run.
DUMP	Specifies a COBOL74 or COBOL85 symbolic that can be compiled into a program to dump a data set to a conventional file is generated. If INDEPENDENTTRANS has been set for a database, SYSTEM/LOADDUMP generates a dump program that performs a LOCK STRUCTURE on the data set to be dumped.
VIA <index structure>	Causes the resultant program to dump records according to the order and WHERE condition of the <index structure> as specified in DASDL. The <index structure> construct must be a disjoint set or subset against the data set. If VIA <index structure> is not specified, records are dumped in data set physical order.

Compilation of the LOAD and DUMP symbolics is automatically initiated by default. (See "Compiler Control Options for LOADDUMP" described later in this section.)

COBOL74 or COBOL85 MOVE Algorithm

LOADDUMP is generally insensitive to item types and COBOL74 or COBOL85 MOVE rules. The resulting MOVE code should always be checked by the user.

With the possible exception of occurring items, MOVE code is generated for all equated items, whether equated implicitly or explicitly.

Data management Boolean items are dumped from a data set to a conventional file according to the following algorithm:

- If the Boolean item is TRUE, a value of 1 is stored in the corresponding file item.
- If the Boolean item is FALSE, a value of 0 is stored in the corresponding file item.

Data management Boolean items are loaded to a data set from a conventional file according to the following algorithm:

- If the file item has a value of 1, the Boolean item in the data set is set to TRUE.
- If the file item has a value other than 1, the Boolean item in the data set is set to FALSE.

Data management number items are dumped from a data set to a conventional file according to the following algorithm:

- If the number item contains null values (all F), then the corresponding BINARY item (default) in the file is given a value of 0 because no move operation is performed.
- If the number item contains any other value, then that value is moved to the corresponding file item.

Only corresponding occurrences of OCCURRING items are moved.

Example

DASDL	COBOL74 or COBOL85
A ... OCCURS 3 ...	01 A ... OCCURS 2 TO 4 ...
B ... OCCURS 10 ...	01 B ... OCCURS 5 ...

Only three occurrences of A can be moved to prevent an INVALID INDEX on the DASDL item A. Only five occurrences of B can be moved to prevent an INVALID INDEX on the COBOL74 or COBOL85 item B.

Compiler Control Options for LOADDUMP

SYSTEM/LOADDUMP has the following compiler control options that are initialized:

- LIST

When LIST is set, LOADDUMP produces a listing of the user input deck.

- ZIP

When ZIP is set, compilation of the LOAD and/or DUMP programs is automatically begun. You can assign the value RESET to ZIP and compile the LOAD and/or DUMP programs separately. The compilation deck for the LOAD and DUMP programs is as follows:

```
BEGIN JOB LOADDUMPCOMPILE;
  COMPILE LOAD/<database name>/<data set name>
    WITH <compiler name> LIBRARY;
  COMPILER FILE TAPE SOURCE =
    SYMBOL/LOAD/<database name>/<data set name>;
  COMPILER DATA CARD
000000$  MERGE

?   COMPILE DUMP/<database name>/<data set name>
    WITH <compiler name> LIBRARY;
  COMPILER FILE TAPE SOURCE =
    SYMBOL/DUMP/<database name>/<data set name>;
  COMPILER DATA CARD
000000$  MERGE

?   END JOB
```

Columns 73 through 80 are ignored by SYSTEM/LOADDUMP.

- LISTSYM

When LISTSYM is set, a listing of the symbolics for the LOAD and/or DUMP programs is produced by the COBOL74 or COBOL85 compiler after compilation.

Section 18

Compiling Software

When you need to compile Enterprise Database Server software, you can use the two Work Flow Language (WFL) job decks specified and described in the following text. It is recommended that you use the following WFL job when you are compiling the software manually:

```
DATABASE/WFL/COMPILEACR
```

The string parameter for the above WFL job cannot exceed 256 characters. If you need to exceed this character limit, use the following WFL job:

```
DATABASE/WFL/COMPILEDB
```

These two jobs function in the same way. The only difference is that you use four parameter strings in DATABASE/WFL/COMPILEDB. The first string in both statements is the same except that DATABASE/WFL/COMPILEACR can add the Enterprise Database Server software titles. The second through fourth strings in DATABASE/WFL/COMPILEDB are used to specify Enterprise Database Server software titles, one title per parameter. A null string (" ") is required for each software that is not being compiled. Currently, the software titles that can be specified are Accessroutines, DMSUPPORT, and RECONSTRUCT.

If the Data and Structure Definition Language (DASDL) ZIP option is assigned the value SET and no syntax errors are detected, the DASDL compiler automatically begins a WFL file titled DATABASE/WFL/COMPILEDB to compile the DMSUPPORT library. For audited databases, RECONSTRUCT is also compiled. No compilations are initiated when ZIP is reset.

Compilation WFL Job Parameters

The WFL job files can be started any time to compile the tailored database software. This job file takes as input a string parameter consisting of <keyword> = <value> pairs that specifies the actions to be taken. [Table 18-1](#) gives the keywords, their default values, and their meanings. The capitalized portion of each keyword is an acceptable abbreviation for it.

Table 18-1. Compilation WFL Job Parameter Keywords

Keyword	Default Value
DB	DEFAULTDB
SOurce	DISK
OBJECT	DISK
DESCription	DISK
AUDIT	RESET
INITialize	RESET
ACR	SYSTEM/ ACCESSROUTINES
COMPILE	All software

Examples

```
START DATABASE/WFL/COMPILEACR( "DB=TESTDB
    OB=DEVELPK C=RECON INIT=SET" )
```

```
START DATABASE/WFL/COMPILEDB
( "DB=TESTDB OB=DEVELPK C=DMSUPPORT,RECON
    INIT=SET" , " " , " " , " " , " " , " " , " " );
```

Either of the preceding examples causes the compilation of the RECONSTRUCT program for the database TESTDB, whose description file is DESCRIPTION/TESTDB. The created code files are placed on DEVELPK. All the database files are initialized using DMUTILITY.

If the database has more than 8000 structures and the WFL job DATABASE/WFL/COMPILEDB is initiated manually, then a value of TRUE is required for the ninth parameter. For example:

```
START DATABASE/WFL/COMPILEDB
( "DB=TESTDB OB=DEVELPK C=DMSUPPORT,RECON
    INIT=SET" , " " , " " , " " , " " , " " , " " , , TRUE ) ;
```

WFL Job Attribute Specifications

DATABASE/WFL/COMPILEACR and DATABASE/WFL/COMPILEDB do not provide any job attribute specifications. These WFL decks can be modified to add the user-specific specifications. These specifications are

- CLASS specification
- FAMILY specification
- USERCODE specification
- FETCH specification
- JOB ATTRIBUTE assignment

In addition, the SECURITY statement can also be added after a successful compilation to change the security of the compiled code.

For detailed information about JOB ATTRIBUTE assignment specifications and the SECURITY statement, refer to the *WFL Reference Manual*.

If the database software is to be compiled for a different TARGET, then a value of "LEVEL5" or "LEVEL6" may be passed for the tenth parameter. For example:

```
START DATABASE/WFL/COMPILEDB
  ("DB=TESTDB OB=DEVELPK C=DMSUPPORT,RECON",
   "","","","","","","","","","LEVEL6");
```

DMSUPPORT

The DMSUPPORT library is generative software. A separate library code file must be compiled for each database. Because the library is shared by all, there is never more than one library stack in the mix for each database.

The library symbolic is contained in the file DATABASE/DMSUPPORT, and the default code file title is DMSUPPORT/<database>.

The library is automatically compiled by the standard compilation deck, DATABASE/WFL/COMPILEDB, when the DASDL compiler control option ZIP is set. The following WFL statement causes the compilation deck to compile the library only:

```
STARTJOB DATABASE/WFL/COMPILEACR ("DB=EXAMPLEDB COMPILE DMSUPPORT")
```

Alternatively, the following WFL job shows how the DMSUPPORT library can be compiled for a database called EXAMPLEDB:

```
?BEGIN JOB COMPILE/DMSUPPORT;
TASK T;
COMPILE DMSUPPORT/EXAMPLEDB WITH DMALGOL [T] LIBRARY;
COMPILER FILE CARD (KIND = PACK, FAMILYNAME = DISK,
                   TITLE = DATABASE/DMSUPPORT);
COMPILER FILE DASDL (TITLE = DESCRIPTION/EXAMPLEDB);
IF T ISNT COMPILEDOK THEN
  ABORT "DMSUPPORT COMPILE FAILED";
?END JOB
```

RECONSTRUCT

The RECONSTRUCT program is generative software. A separate code file, named RECONSTRUCT/<database name> by default, must be compiled for each audited database. This program is used if a row reconstruction is necessary. (Refer to "DMUTILITY RECOVER Statement" in [Section 8, Recovering the Database](#), for instructions on how to reconstruct rows.) The following WFL statement causes the compilation deck to compile the RECONSTRUCT program only:

```
STARTJOB DATABASE/WFL/COMPILEACR
  ("DB=EXAMPLEDB COMPILE=RECONSTRUCT")
```

EXAMPLEDB must be an audited database. The RECONSTRUCT program is compiled automatically by the standard compilation deck, DATABASE/WFL/COMPILEDB, if the DASDL compiler control option ZIP is set and the database is audited.

DMINTERPRETER

For information on compiling the software DMINTERPRETER, refer to the *Enterprise Database Server Interpretive Interface Programming Manual*.

RMSUPPORT

The RMSUPPORT library is software that must be specially compiled for each database and for each database model.

The library symbolic is contained in the file DATABASE/RMSUPPORT, and the default code file title is RMSUPPORT/<database name>.

The library is automatically compiled by the standard compilation deck, DATABASE/WFL/COMPILEDB, when both of the following options are set:

- The DASDL compiler control option ZIP
- The DASDL option OPENOLTP

The following WFL statement causes the compilation deck to compile the RMSUPPORT library only:

```
STARTJOB DATABASE/WFL/COMPILEACR ("DB=EXAMPLEDB COMPILE RMSUPPORT")
```

Alternatively, the following WFL job shows how the RMSUPPORT library can be compiled for a database called EXAMPLEDB:

```
?BEGIN JOB COMPILE/RMSUPPORT;
TASK T;
COMPILE RMSUPPORT/EXAMPLEDB WITH DMALGOL [T] LIBRARY;
COMPILER FILE CARD (KIND = PACK, FAMILYNAME = DISK,
                    TITLE = DATABASE/RMSUPPORT);
COMPILER FILE DASDL (TITLE = DESCRIPTION/EXAMPLEDB);
IF T ISNT COMPILEDOK THEN
    ABORT "RMSUPPORT COMPILE FAILED";
?END JOB
```

Section 19

Controlling Partitioned Records

Normally all records within a structure reside in a single file. This file must be present on disk or pack whenever the structure is in use. The records of a partitioned structure do not all reside in a single file. Instead, the records are divided, or partitioned, among several files. Only those files required for the current application need be present on disk or pack, while the remaining files can be stored on some backup medium. Partitioning is used for structures that would otherwise require very large files. It reduces the amount of disk or pack space required when only a small portion of the file is used at any single time.

Only embedded structures can be partitioned. The embedded structure is partitioned based upon the value of an item in the disjoint data set that contains the embedded structure. The data item that controls the partitioning is the partition key. The partition key serves as the key of the partitioning set. The disjoint data set that contains the partition key and the partitioned structure is the partition master.

Embedded structures at any level can be partitioned. No matter how deeply embedded the partitioned structure might be, the partition master is always the disjoint data set that contains the structure.

There can be as many partitions as there are unique values of the partition key. The system automatically keeps track of the partitions that currently exist using a structure called the Partition Directory. There is a record in the Partition Directory for each partition in the database.

Partition Directory Overview

A database containing at least one partitioned structure contains one extra structure not defined by the user in the Data and Structure Definition Language (DASDL) source. This extra structure is an ORDERED data set, named PARTITIONINFO. The PARTITIONINFO data set has an access declared against it, named PARTITIONLIST. The DASDL source for this structure is contained in DATABASE/PROPERTIES, and is automatically included by the DASDL compiler for databases containing partitions. If the source text for the PARTITIONINFO data set is not found in the PROPERTIES file, an appropriate syntax error results. This situation could occur when an incorrect version of DATABASE/PROPERTIES is read.

The PARTITIONINFO data set is the Partition Directory. Although it is accessible in a normal manner by the user, it is intended to be created and maintained by the system. There is a record in the Partition Directory for each partition file belonging to the database.

Controlling Partitioned Records

This record contains three items:

- The structure number
- The partition identifier, which is the value of the partition key
- The partition number, which uniquely identifies the partition

The first time a record in a partitioned structure is referenced after the value of the partition key has changed, the system automatically looks in the Partition Directory to determine if the partition exists. If it does, the file is opened and accessed in the normal manner. If it does not, the proper file is created, and a record for it is created in the Partition Directory. Subsequent references to the same partition cause the system to seek records in the same partition file, thereby incurring very little overhead beyond that of a nonpartitioned structure. The Partition Directory is a normal database structure, and not a special data file, for the following reasons:

- The mechanism for creating, maintaining, and searching the file already exists.
- The file can be easily examined by the user; for example, through INQUIRY or userwritten programs.

Note: Do not modify the file unless absolutely necessary because incorrect modification might completely defeat the purpose of the Partition Directory.

- The file can be modified by the user if necessary. Because incorrect modification might completely defeat the purpose of the Partition Directory, this procedure is strongly discouraged.
- For audited databases, the file is fully protected by normal audit and recovery procedures.

Partition Directory Details

The DASDL source for the Partition Directory, contained in DATABASE/PROPERTIES, is as follows:

```
PARTITIONINFO                                % SYSTEM DATA SET
  ORDERED DATA SET      "PARTITION DIRECTORY FOR SYSTEM USE"
  (
    STRUCTURENUM FIELD(16);                    % SYSTEM DATA SET
    PARTITIONID  ALPHA(17);                    % SYSTEM DATA SET
    PARTITIONNUM FIELD(20);                    % SYSTEM DATA SET
  )
    POPULATION = 1000,                         % SYSTEM DATA SET
    BLOCKSIZE  = 110,                          % SYSTEM DATA SET
    CHECKSUM   = TRUE;                          % SYSTEM DATA SET
PARTITIONLIST                                % SYSTEM DATA SET
  ACCESS TO PARTITIONINFO                    % SYSTEM DATA SET
    KEY IS (STRUCTURENUM, PARTITIONID);       % SYSTEM DATA SET
```

You can alter the physical options of the PARTITIONINFO data set by including a DEFAULTS statement in the DASDL source, or by modifying its declaration in DATABASE/PROPERTIES. Changing DATABASE/PROPERTIES affects all users of the DATABASE/PROPERTIES file.

PARTITIONINFO is an ORDERED data set so that it can be searched easily and efficiently without an index set.

The keys of the access (STRUCTURENUM and PARTITIONID) are the primary structure number and a string created from the value of the partition key. These values are known by the system when a partition is required.

PARTITIONNUM is a unique integer assigned to each partition when it is created. PARTITIONNUM is used to locate a partition timestamp entry in the database control file. (Refer to [Section 2, Control File](#), for further details.) The value is assigned by incrementing a variable called PARTITIONNUMBER, which is maintained in the control file.

All references to the PARTITIONINFO data set or its access are made with the normal Accessroutines code. There is no special code to handle Partition Directory structures.

A new partition (and new entries in the PARTITIONINFO data set and control file) is not created if changes are not allowed at the time of the first reference. For example, a new partition is not created if the database is OPEN INQUIRY, or while outside transaction state for audited databases. In addition, an error exception (NOTFOUND 7) is returned to the user program. This is the same exception returned for a normal embedded structure if it has never been created.

There are occasions when a portion of the database must be initialized; for example, when new structures are added to an existing database. On these occasions, the PARTITIONINFO data set can be automatically invoked and implicitly reinitialized along with the structures that must be initialized. When this occurs, the PARTITIONINFO data set is created from the partition entries in the control file. Although it is not encouraged, reinitialization of existing structures containing partitioned structures will also work properly as far as the Partition Directory is concerned. In this case, all partitions belonging to the reinitialized master structures are purged from the Partition Directory. When the PARTITIONINFO data set is recreated from the control file, it reflects the current state of the partitions.

Audit and Recovery Considerations

The following three components of the Partition Directory require recovery after a failure:

- The PARTITIONINFO data set
- The partition timestamps in the control file
- The PARTITIONNUMBER value in the control file

These components can all be recovered properly if the database is audited.

All changes to the PARTITIONINFO data set are audited in the normal manner. Recovery of the PARTITIONINFO data set is completely standard.

The Accessroutines performs the following steps when a new partition file is created:

Controlling Partitioned Records

- Increment and update PARTITIONNUMBER in the control file.
- Write a create partition audit record that contains the partition identifier and the new value of PARTITIONNUMBER.
- Create a new record in the PARTITIONINFO data set.
- Create a null partition file with the proper title and file attributes.
- Audit all changes to the partition file, which creates the initial structure (not audited for creation of nonpartitioned structures).

It is possible to back out the creation of a partition during a recovery process. It is necessary to return the partition number to the control file, and purge the corresponding partition timestamp record. The partition file itself is purged and eliminated from the system.

If the creation of a partition is repeated by the application of afterimages such as during REBUILD, it is necessary only to create a null file with the proper name and file attributes; successive afterimages will create the proper initial structure of the file.

Section 20

Using the Audit Reader Library Interface

This section describes how to use the interface provided by the audit reader library, called DMAuditLib, to enable utility and user programs to access Enterprise Database Server audit files. This section includes the following topics:

- Audit reader library overview
- Using the ALGOL interface
- Entry points
- Error results

Audit Reader Library Overview

contains entry points The DMAuditLib library is the recommended means for accessing the Enterprise Database Server audit files for utility and user programs. Prior to the introduction of the DMAuditLib library in SSR 44.2, programs needing to access the audit trail did so by using the Direct I/O interface against the audit files.

The DMAuditLib library contains entry points that return audit records in addition to returning audit blocks. This feature enables utility and user programs either to decode the audit blocks or request the DMAuditLib library to return only audit records extracted from the audit blocks. Using the older Direct I/O interface to access the audit records within the audit file, a program performs a read operation against a physical file block of data, extracts the audit block, and then returns the information from the individual audit records within the block.

With the introduction of sectioned audit files in the XE features, all programs that directly access an audit file must be recoded to account for the following issues:

- Multiple physical files will exist where there was previously only one file.
- The number of physical files that must be read can vary between logical audit files.
- The programs must recognize the scheme used to write records and blocks into the physical files.

User programs could, of course, be modified to accommodate sectioned audit files, but changing the programs to access the audit by way of a library interface that recognizes sectioned audit files is much more efficient. Knowledge of the number of audit sections needed, the allocation of audit records and blocks within those files, and other issues are isolated and handled within the library. This approach makes converting existing programs

Using the Audit Reader Library Interface

easier for customers. In addition, use of the library reduces the impact on user programs should it be necessary to make changes in the implementation of the audit file (naming conventions, record and block allocation schemes, file locations, section limits, and so forth).

You must convert existing programs to use the DMAuditLib library by completing the following actions:

- Direct the calling programs to include the necessary entry point and other interface declarations from the DATABASE/PROPERTIES file.
- Discard all code related to physical reads of the audit files.
- Convert all calls to the discarded code so that the call is made to the appropriate DMAuditLib library entry point.

The DMAuditLib library is declared in the SYMBOL/DMAUDITLIB file as follows:

- SHARING=PRIVATE
- LIBACCESS=BYFUNCTION
- FREEZE(TEMPORARY)

The library FUNCTIONNAME attribute is DMSIIAUDITSUPPORT.

For ALGOL applications, the transfer of information from the library to the application is optimized through the use of arrays exported by the DMAuditLib library. The Audit_Buffers and AUDIT_INFO arrays return information common to all the library procedures. These procedure calls include by-name parameters, which the library uses to indicate where in the exported AUDIT_BUFFERS array the requested data can be found. Detailed status information is available in the exported AUDIT_INFO array. Specific information on procedure parameters is included later in this section under "Entry Points."

Using the ALGOL Interfaces

The ALGOL interfaces are exported from the DMAuditLib library and the ALGOL data arrays transfer the audit data from the library to the calling program.

To include the full text of the ALGOL interfaces into a calling program, use the \$INCLUDE command as shown in the following example:

```
$INCLUDE "DATABASE/PROPERTIES" 38000000 - 38999999
```

These ALGOL data arrays are exported to the calling program as part of the library declaration, for example:

```
LIBRARY AUDITLIB (
    LIBACCESS=BYFUNCTION
    , FUNCTIONNAME="DMSIIAUDITSUPPORT."
)
[ARRAY AUDIT_INFO [0] ;
  ARRAY AUDIT_BUFFERS [0, 0] (READWRITE) ;
] ;
```

ALGOL Array Reference **AUDIT_INFO [0]**

The single-dimension ALGOL array **AUDIT_INFO** is used to return static and dynamic control information about the open audit file. The following types of information are included:

- Linkage and operational
- Logical audit file
- Internal buffer
- Audit section
- Block list

You can use this information to determine

- Logical and physical file information
- Information from block 0 of the audit file
- A summary of the most recent call to the library procedures
- An interpretation of information contained within the most recent audit block or record, beyond the minimal information that is returned directly into the output parameters

The **Audit_Info** array is marked as “read only” within the calling program. If the calling program attempts to modify any information within the **Audit_Info** array, the program is terminated with faults.

The contents of this array are valid when an audit file is open. If no audit file is open, the **Audit_Info** array should not be used.

Three groups of information are repetitive:

- The internal buffer group includes one set of information for each internal buffer. (Refer to “ALGOL Array Reference **AUDIT_BUFFERS [0, 0]**” later in this section.)
- The audit section group includes one set of information for each audit section, including a copy of block 0 for each section.
- The block list group can include multiple sets of information.

To simplify the ALGOL interface for these groups of information, real variables are exported from the **DMAuditLib** library and are used by the accessing defines, which make the **Audit_Info** array look like a number of independent structures. Only the informational view as presented by these accessing defines is documented in this section. You can view the internal details in the **Database/Properties** file.

Linkage and Operational Information

The following **ALI_xxx** words describe linkage information and constantly changing operational information. These words are at fixed locations in the **Audit_Info** array for quick access.

Accessing Example

```
ALI_INFO[ALI_VERSION]
```

ALI_xxx Words

Word	Description
ALI_VERSION	<p>Indicates the version of the information being returned by the DMAuditLib library. If the value in this word does not match the value of the AUDITLIB_VERSION with which the calling program is compiled, the program should not attempt to access the library until the program has been recompiled with the correct DATABASE/PROPERTIES file. The value in this word is available immediately after the library has been linked to the calling program.</p> <p>Note: <i>This item is always available for the calling program to use, even when no audit file was opened by the library.</i></p> <p>The Boolean NEW_LIB_AVAILABLE flag is a means to notify a caller that the DMAuditLib code file has been exchanged while the library and caller were still linked. The status of NEW_LIB_AVAILABLE can be checked whenever an audit has been closed either implicitly because of an audit file switch or explicitly by calling AUDIT_CLOSE.</p> <p>In general, programs that do not require notification about the latest edition of the DMAuditLib library can safely ignore NEW_LIB_AVAILABLE. It is intended as a convenience for those applications that hold audit files open for lengthy periods of time, which might span an Enterprise Database Server software update. When changing to a new copy of DMAuditLib, be sure to save any relevant information from the exported arrays and to recheck AUDIT_VERSION after linking to the new library copy.</p>

The following words in the ALI_INFO group represent linkage information that allows the rest of the information within the AUDIT_INFO array to be found:

Word	Description
ALI_BUFFER_COUNT	<p>Indicates the number of rows in the AUDIT_BUFFERS array. This value can change when internally switching from one audit file to the next (if the FILE_SWITCH_OK parameter is TRUE) as the number of audit sections changes. The current buffer allocation scheme always allocates two buffers to each section, plus an additional two buffers.</p> <p>Note: <i>This information is always available for the calling program to use, even when no audit file was opened by the library.</i></p>

Word	Description
ALI_SECTION_COUNT	<p>Indicates the number of audit sections in the current audit file. This value can change when internally switching from one audit file to the next (if the FILE_SWITCH_OK parameter is TRUE).</p> <p>Notes:</p> <ul style="list-style-type: none"> • <i>This information is always available for the calling program to use, even when no audit file was opened by the library.</i> • <i>For SSR 44.2 and earlier releases, audit files contain only one section.</i>

The following words in the ALI_INFO group represent error summary information from the most recent procedure call to the DMAuditLib library:

Word	Description
ALI_FILE_IS_OPEN	<p>The word value is 1 (or TRUE) when the DMAuditLib library has an audit file open. Otherwise, the word value is 0 (or FALSE).</p> <p>Note: <i>This information is always available for the calling program to use, even when no audit file was opened by the library.</i></p>
ALI_ERROR_RESULT	<p>A copy of the most recent result word returned by a call to any of the DMAuditLib library procedures.</p>
ALI_ERROR_LINENUMBER	<p>The internal symbolic line number indicating where the most recently reported error stored in the ALI_ERROR_RESULT word was detected. This line number is useful for debugging the DMAuditLib library.</p> <p>Note: <i>This information is always available for the calling program to use, even when no audit file was opened by the library.</i></p>
ALI_IJORESULT	<p>A copy of the most recent I/O result word that was returned by the MCP to the DMAuditLib library at the completion of the most recent read from the audit file.</p> <p>Note: <i>This information is always available for the calling program to use, even when no audit file was opened by the library.</i></p>

Using the Audit Reader Library Interface

Word	Description
ALI_IORD_SECTION	<p>The audit section that corresponds to the ALI_IORESULT word described previously.</p> <p>Notes:</p> <ul style="list-style-type: none"> • This information is always available for the calling program to use, even when no audit file was opened by the library. • For SSR 44.2 and earlier releases, audit files contain only one section.
ALI_IORD_ADDRESS	<p>The disk record number (segment) from which the most recent buffer was read. This value is 0 (zero) if the read operation was performed from a tape.</p> <p>Note: This information is always available for the calling program to use, even when no audit file was opened by the library.</p>

The following words in the ALI_INFO group represent summary information from the most recent call to the AUDIT_NEXT_RECORD procedure:

Word	Description
ALI_REC_ABSN	The audit block serial number (ABSN) of the audit in which the most recently retrieved audit record started. This ABSN is the same value returned in the ABSN output parameter of the AUDIT_NEXT_RECORD entry point.
ALI_REC_LENGTH	The length, in words, of the most recently retrieved audit record. This information is a duplicate of the value returned in the RECORD_LENGTH output parameter of AUDIT_NEXT_RECORD.
ALI_REC_TYPE	The record type of the most recently returned record. This information duplicates the value returned in the RECORD_TYPE output parameter of AUDIT_NEXT_RECORD.
ALI_REC_STRUCTNUM	The structure number referenced by the most recently returned record. This information duplicates the value returned in the STRUCTURE_NO output parameter of AUDIT_NEXT_RECORD.
ALI_REC_PROCESSID	The process ID (stack number) of the most recent audit record.
ALI_REC_LINENUMBER	The SYMBOL/ACCESSROUTINES line number indicating where the most recent audit record was produced (if the Accessroutines was compiled with the \$SET AUDITDEBUG option).
ALI_REC_BUFFER	The row index into the AUDIT_BUFFERS array where the most recent audit record can be found. This row index duplicates the value returned in the BUFFER_INX output parameter of AUDIT_NEXT_RECORD.

Word	Description
ALI_REC_OFFSET	The word offset within the appropriate AUDIT_BUFFERS row where the most recent audit record can be found. This word offset duplicates the value returned in the RECORD_OFFSET output parameter of AUDIT_NEXT_RECORD.
ALI_REC_SHORTBY	The number of audit record words that were not retrieved from a split audit record by the AUDIT_NEXT_RECORD procedure. Normally this value is 0.

The following words in the ALI_INFO group represent summary information from the most recent call to the AUDIT_NEXT_ABSN procedure. This most recent call could have been an internal call made from within the AUDIT_RANDOM_ABSN or AUDIT_NEXT_RECORD procedure.

Word	Description
ALI_BLK_ABSN	The ABSN of the most recently read audit block.
ALI_BLK_LENGTH	The length, in words, of the most recently read audit block.
ALI_BLK_FLAGS	The flag word from the latest audit block.
ALI_BLK_DATESTAMP	The date stamp, in TIME (6) format, of the most recent audit block.
ALI_BLK_TIMESTAMP	The time stamp, in TIME (11) format, of the most recent audit block.
ALI_BLK_PREV_TIMESTAMP	The previous time stamp stored in the most recent audit block.
ALI_BLK_RECORDNUM	The number of the audit block, as counted from the beginning of the audit file. This value has no relation to the ABSN or the location of the block; it is simply a counter from the beginning of the file. The first block is numbered 1.
ALI_BLK_BUFFER	The row index into the AUDIT_BUFFERS array where the most recent audit block can be found. This row index duplicates the value returned in the BUFFER_INX output parameter of the AUDIT_NEXT_ABSN entry point.
ALI_BLK_OFFSET	The word index at the beginning of the most recent audit block within the selected row of the AUDIT_BUFFERS array.

If the PTIMES option is enabled with the Accessroutines, the following four values are captured during the construction of each audit block, are stored within that audit block, and are made available there. (See the ALF_PTIMES_ENABLED word, described later in this section.)

Word	Description
ALI_BLK_DMS_IOTIME	The Enterprise Database Server I/O time.

Using the Audit Reader Library Interface

Word	Description
ALI_BLK_NONDMS_PTIME	The non-Enterprise Database Server processor time.
ALI_BLK_UPDATE_PTIME	The Enterprise Database Server processor time related to update operations.
ALI_BLK_INQUIRY_PTIME	The Enterprise Database Server processor time related to inquiry operations.

The following words in the ALI_INFO group represent linkage information regarding the disk record location—or disk segment—of the previous, current and next audit block. For all level-6 audit files and for level-7 audit files with only one section, the xxx_SECTION words always have the value of 1

Word	Description
ALI_PREV_SECTION	The section where the previous audit block can be found.
ALI_PREV_ADDRESS	The disk location where the previous audit block can be found.
ALI_CURR_SECTION	The section where the current audit block is found.
ALI_CURR_ADDRESS	The disk location where the current audit block is found.
ALI_NEXT_SECTION	The section where the next audit block can be found.
ALI_NEXT_ADDRESS	The disk location where the next audit block can be found.

Logical Audit File Information

The following ALF_xxx words describe the logical nature of the audit file. Most information is derived directly from block 0 of the audit file, although some is from the MCP file attributes.

Accessing Examples

```
ALF_INFO[ALF_FILENUMBER]
```

```
ALF_BLOCK0[ISDUPLICATED]
```

Pointer	Description
ALF_TITLE_PTR for ALF_TITLE_LEN	The pointer to and length of the audit file title.
ALF_USER_PTR for ALF_USER_LEN	The pointer to and length of the usercode portion of the audit file title.
ALF_DBNAME_PTR for ALF_DBNAME_LEN	The pointer to and length of the database name portion of the audit file title.

Pointer	Description
ALF_PACK_PTR for ALF_PACK_LEN	The pointer to and length of the pack name portion of the audit file title.

ALF_xxx Words

Word	Description
ALF_FILENUMBER	The audit file number (AFN) of the open audit file.
ALF_AUDITLEVEL	The level of the open audit file. For audit files created before SSR 44.2, this value is 6. For audit files created in SSR 44.2 or a later release, this value is 7.
ALF_BLOCKSIZE	The maximum size, in words, of audit blocks within the current audit file.
ALF_AREASIZE	The AREASIZE file attribute for all sections of current audit file.
ALF_FIRST_ABSN	The first ABSN in the current audit file, excluding the ABSN of block 0.
ALF_LAST_ABSN	The last known ABSN in the current audit file.
ALF_AUDIT_TIMESTAMP	The time stamp from block 0 of the current audit file.
ALF_DB_TIMESTAMP	The database time stamp from block 0.
ALF_DB_UPDATELEVEL	The database update level from block 0.
ALF_ISATAPE	The value is 1 if the audit file is on tape; otherwise the value is 0.
ALF_FIRSTAW	The index within all audit blocks of where the first word of the audit record information begins. For level-6 audit files, this value is 5. For level-7 audit files, this value is 10.
ALF_PTIMES_ENABLED	The value is 1 if the PTIMES capability was enabled in the Accessroutines when this audit file was created; otherwise the value is 0.
ALF_UNKNOWN_EOF	The value is 0 if ALF_LAST_ABSN is known to be the last actual audit block in the file. The value is 1 if the last ABSN is unknown. This could occur if there is a halt/load or database failure. The value is always 1 for audit files on tape.
ALF_TAPE_UNITNO	The tape unit number, if the audit file is being read from tape.

Internal Buffer Information

The ALB_xxx words describe the current state of each row in the AUDIT_BUFFERS array being used by the DMAuditLib library. There is one set of these words for each buffer. The first set of words is for AUDIT_BUFFERS [0, *]. These words are used to debug the DMAuditLib library.

Accessing Example

```
ALB_INFO [BUFX, ALI_IO_SECTION]
```

ALB_xxx Words

Word	Description
ALB_IO_SECTION	The section from which the buffer was read.
ALB_IO_ADDRESS	The disk record number (segment) from which the buffer was read. This value is 0 if the buffer was read from tape.
ALB_IO_SIZE	The size, in words, of the last buffer read operation.
ALB_IO_IORD	The MCP I/O result word for the last read operation.
ALB_BLK_OFFSET	The current offset in the buffer that the AUDIT_NEXT_RECORD procedure is using to sequentially retrieve audit blocks.

Audit Section Information

The following ALS_xxx words describe the physical and logical nature of each audit section. The physical nature is derived from the MCP file attribute interrogation. The logical nature is derived from block 0 for that section. To imitate the numbering conventions for audit sections, the first set of words is for the true audit file using the index of 1, if only one section exists, or for the master section if there are multiple audit sections.

Accessing Example

```
ALS_INFO[1, ALS_BLOCK0_EOF]
```

```
ALS_BLOCK0 [1, ISDUPLICATED]
```

ALS_xxx Words

Word	Description
ALS_BLOCK0_EOF	The end-of-file (EOF) value for the audit section as determined from block 0.

Word	Description
ALS_FILEATT_LASTRECORD	The EOF value as determined by the LASTRECORD file.
ALS_FILEATT_CREATIONDATE	The CREATIONDATE file attribute.
ALS_FILEATT_CREATIONTIME	The CREATIONTIME file attribute.
ALS_FILEATT_VERSION	The VERSION file attribute (tape files only).
ALS_FILEATT_CYCLE	The CYCLE file attribute (tape files only).
ALS_FILEATT_DENSITY	The DENSITY file attribute (tape files only).
ALS_FILEATT_READREVERSECAPABLE	The READREVERSECAPABLE file attribute (tape files only).
ALS_FILEATT_AREAS	The AREAS file attribute (disk files only).
ALS_IO_BUFFER	The buffer index into the AUDIT_BUFFERS array of the most recent buffer that was read for this audit section.
ALS_IO_IORD	The MCP I/O result word for the most recent I/O operation completed for this audit section.
ALS_IOWAIT_CLOCKS	A summation of the elapsed I/O waiting time for this audit section.
ALS_IOWAIT_COUNT	The number of read operations for this audit section.

Block List Information

The following ALK_xxx words summarize information about audit blocks that are read while the AUDIT_NEXT_RECORD procedure is in the process of assembling a split audit record. These blocks are never directly seen by the calling program therefore, this information is retained so that the calling program can summarize it if desired. The number of entries is stored in the ALI_INFO ALI_BLKLIST_COUNT word. This value is 0 unless the most recent record read by the Audit_Next_Record procedure was split across multiple audit blocks.

Accessing Example

```
ALK_INFO[1, ALK_FILENUMBER]
```

ALK_xxx Words

Word	Description
ALK_FILENUMBER	The AFN from which this block came.
ALK_ABSN	The ABSN of the block.
ALK_LENGTH	The length, in words, of the block.

Using the Audit Reader Library Interface

Word	Description
ALK_FLAGS	The flag word from the block.
ALK_DATESTAMP	The date stamp of the block.
ALK_TIMESTAMP	The time stamp of the block.
ALK_PREV_TIMESTAMP	The previous time stamp of the block.
ALK_RECORDNUM	The ordinal record number of the block.
ALK_SECTION	The section from which the block came.
ALK_ADDRESS	The location, or segment, of the block.

For nonsectioned audits or systems upon which the XE features are not installed, ALK_SECTION is always zero.

ALGOL Array Reference **AUDIT_BUFFERS [0, 0]**

The two-dimensional ALGOL array `AUDIT_BUFFERS [0, 0]` is used to return audit block and audit record information from the open audit file. Because of ALGOL limitations, the individual rows of the `AUDIT_BUFFERS` array are not marked as “read only” within the calling program. If the calling program modifies any information within the `Audit_Buffers` array, errors or invalid information could be returned to the calling program. Nonetheless, the audit file being read by the `DMAuditLib` library is opened for input only and cannot be damaged by the library.

Calls to the `Audit_Next_ABSN`, `Audit_Random_ABSN`, and `Audit_Next_Record` entry points return the index in the output parameter `Buffer_Inx`. The index indicates the array row containing the requested information. The `Audit_Next_ABSN` and `Audit_Random_ABSN` entry points return the index within the specified row in the output parameter `Block_Offset`, which further indicates where the first word of the returned audit block is located.

In the `Audit_Next_Record` entry point, the output parameter `Record_Offset` indicates the offset in the row where the first word of the audit record is located. The explicit purpose of this “index and offset” scheme is to provide high-speed access to audit information without having to copy the data from the `DMAuditLib` library to the calling program.

Row 0 of the `AUDIT_BUFFERS` array is reserved for returning split audit records that are not fully contained within a single audit block and must be reassembled before being returned to the calling program. Row 1 of the array is reserved for assembling audit blocks that might be split because of the large block read operations performed by the `DMAuditLib` library.

Entry Points

The `DMAuditLib` library provides entry points that interface with ALGOL programs to perform the following procedures:

- The AUDIT_OPEN entry point opens the audit file.
- The AUDIT_CLOSE entry point closes the audit file.
- The AUDIT_NEXT_ABSN entry point performs sequential retrieval of audit blocks.
- The AUDIT_RANDOM_ABSN entry point performs random retrieval of audit blocks.
- The AUDIT_NEXT_RECORD entry point performs sequential retrieval of audit records.

The entry points in the DMAuditLib library also enable you to obtain error codes. These error codes provide the error type, error message, and structure on which the error occurred. For more information about these error codes, refer to “Error Results” later in this section.

AUDIT_OPEN Entry Point Parameters

This entry point runs a procedure that constructs the file title and opens the specified audit file based on the parameters passed. The calling program can indicate whether the desired file is the primary audit file, secondary audit file, or a filtered audit file, and can indicate the action to take if the specified file cannot be found. This entry point procedure automatically opens sectioned or nonsectioned audit files as appropriate.

The title of the audit file is constructed by using the DB_NAME, DB_PACK, AFN, and AF_TYPE parameters. If no such file is found, the title is assumed to be the logical name of a sectioned audit file. This procedure attempts to open all sections of the audit file.

If the audit file can be found under the specified name, the DONT_WAIT parameter is used to determine if the library waits on a No File condition or returns a No File result to the calling program.

When an audit file is successfully opened, the parameter values are saved within the library. These values can be reused by the AUDIT_NEXT_ABSN, AUDIT_RANDOM_ABSN, and the AUDIT_NEXT_RECORD entry points if they must switch audit files because the FILE_SWITCH_OK parameter is TRUE.

The ALGOL interface for this procedure has the following specifications:

```
Boolean Procedure AUDIT_OPEN (DB_NAME,  
                              DB_PACK,  
                              AFN,  
                              AF_TYPE,  
                              DONT_WAIT) ;
```

Value	DB_NAME, DB_PACK, AFN, AF_TYPE, DONT_WAIT;
String	DB_NAME, DB_PACK;
Integer	AFN, AF_TYPE;
Boolean	DONT_WAIT;

Using the Audit Reader Library Interface

Input

- **DB_NAME**
The name of the database whose audit file is to be opened.
- **DB_PACK**
The pack name on which the audit files reside.
- **AFN**
The audit file number (AFN) to be opened.
- **AF_TYPE**
The type of file to be opened.
- **DONT_WAIT**
The action to take if the specified file cannot be found.

Output

The global array AUDIT_INFO is updated.

Results

This procedure returns one of the following values:

- **FALSE**
Indicates the audit file was opened successfully.
- **TRUE**
Indicates that the file was not opened. Error values are described later in this section under "Error Results."

The AF_TYPE parameter has the following possible values:

Parameter	Description
AL_AFTYP_PRIMARY (1)	Identifies the file as a primary audit file with the title <DB_NAME>/AUDIT<nnnn>ON<DB_PACK>.
AL_AFTYP_SECONDARY (2)	Identifies the file as a secondary audit file with the title <DB_NAME>/2AUDIT<nnnn>ON<DB_PACK>.
AL_AFTYP_FILTERED (4)	Identifies the file as a filtered audit file with the title <DB_NAME>/FILTEREDAUDIT<nnnn>ON<DB_PACK>.
AL_AFTYP_USEINTNAME (16)	Designates that the name and location of the audit file is found by using the INTNAME attribute equation. To open the audit file, use the string value in the DB_NAME parameter as the INTNAME of the audit file.
AL_AFTYP_ONTAPE (32)	Specifies that the audit file is on tape.

Note: *AL_AFTYP_PRIMARY, AL_AFTYP_SECONDARY, and AL_AFTYP_FILTERED are mutually exclusive values. This means that you can only set one of these values. Otherwise, the values are added together to form the parameter value. For example, the value formed by AL_AFTYP_SECONDARY + AL_AFTYP_ONTAPE requests that the DMAuditLib library open a secondary audit file on a tape.*

The Dont_Wait parameter has the following possible values:

- FALSE
Indicates that the library is to wait on a No File condition if the requested file cannot be found.
- TRUE
Indicates that the library is to return a AEAUDITNOTFOUNDV result immediately if the requested audit file cannot be found.

This AUDIT_OPEN procedure returns one of the following values:

- FALSE
Indicates that the file was opened successfully.
- TRUE
Indicates that the file was not opened. Error values are described later in this section under "Error Results."

When the audit file is opened successfully, the exported array AUDIT_INFO contains information describing the audit file. If an error occurs and the audit file is not opened, the contents of the AUDIT_INFO array are undefined and invalid. For more information about the AUDIT_INFO array, refer to "ALGOL Array Reference Audit_Info [0]" earlier in this section.

AUDIT_CLOSE Entry Point Parameters

This entry point runs a procedure that closes the open audit file. If the open audit file is sectioned, all audit file sections are closed.

The ALGOL interface for this procedure has the following specifications:

```
Boolean Procedure AUDIT_CLOSE;
```

Input

None.

Output

The global array AUDIT_INFO is updated.

Results

The AUDIT_CLOSE procedure returns one of the following values:

Using the Audit Reader Library Interface

- FALSE
Indicates that the audit file was closed successfully.
- TRUE
Indicates that an error occurred while attempting to close the audit file. Error values are described later in this section under “Error Results.”

After this procedure returns a value, the contents of exported arrays are undefined.

AUDIT_NEXT_ABSN Entry Point Parameters

This entry point runs a procedure that retrieves the next audit block from the open audit file. If no read operations have been performed on the file, the first call to the Audit_Next_ABSN entry point attempts to return data for the first audit block in the file.

When the audit block is successfully read, the contents of the block are placed into the exported Audit_Buffers array. The index in which the information was placed is indicated by the value of the BUFFER_INX and Block_Offset parameters. The first word of the read audit block is at Audit_Buffers[BUFFER_INX, Block_Offset].

The ALGOL interface for this procedure has the following specifications:

```
Boolean Procedure AUDIT_NEXT_ABSN ( FILE_SWITCH_OK,  
                                   BLOCK_LENGTH,  
                                   BUFFER_INX,  
                                   BLOCK_OFFSET) ;  
  
Value          FILE_SWITCH_OK ;  
Boolean        FILE_SWITCH_OK ;  
Integer        BLOCK_LENGTH,  
                BUFFER_INX ;  
                Block_OFFSET ;
```

Input

- FILE_SWITCH_OK
Specifies whether it is permissible to open the next audit file, if necessary, to read the audit block.

Output

The global array AUDIT_INFO is updated. The first word of the audit block just read is at AUDIT_BUFFERS[BUFFER_INX, BLOCK_OFFSET].

- BLOCK_LENGTH
Indicates the amount of data, in words, in the data buffer.
- BUFFER_INX
Indicates the first index in the exported array AUDIT_BUFFERS into which the audit block has been placed. This value indicates the row of the AUDIT_BUFFERS array to be used.
- BLOCK_OFFSET

Indicates the second index in the exported array `AUDIT_BUFFERS` into which the audit block has been placed. This value indicates the offset into the selected row where the beginning of the audit block can be found.

Results

While it is not recommended to do so, you can call *Audit_Next_ABSN* following a call upon *AUDIT_NEXT_RECORD*. This action defines the next audit block read to be the block following the one in which the previously read record ends, and any remaining records in the current block will be ignored, however, the calling program cannot determine which audit block will be read. If you must perform a block-level operation following a call upon *Audit_Next_Record*, you should use the *Audit_Random_ABSN* procedure discussed later in this section.

The *Audit_Next_ABSN* procedure returns one of the following values:

- FALSE
Indicates that the next block was read successfully.
- TRUE
Indicates that an error occurred during the read operation. Error values are described later in this section under "Error Results."

AUDIT_RANDOM_ABSN Entry Point Parameters

This entry point runs a procedure that retrieves information from a specific block of the open audit file.

When the audit block is successfully read, the contents of the block are placed into the exported `Audit_Buffers` array. The index in which the information was placed is indicated by the value of the `Buffer_INX` and `Block_Offset` parameters. The first word of the read audit block is at `Audit_Buffers[Buffer_INX, Block_Offset]`.

Performing consecutive calls to the *Audit_Random_ABSN* entry point in which the value passed in the `ABS_N` parameter is always one greater than the previous call to *Audit_Random_ABSN* achieves the same result as performing consecutive calls to the *Audit_Next_ABSN* entry point.

The ALGOL interface for this procedure has the following specifications:

```
Boolean Procedure AUDIT_RANDOM_ABSN ( FILE_SWITCH_OK,  
                                     ABSN,  
                                     BLOCK_LENGTH,  
                                     BUFFER_INX,  
                                     BLOCK_OFFSET );  
  
Value      File_SWITCH_OK,  
           ABSN;  
Boolean    FILE_SWITCH_OK;  
Integer    BLOCK_LENGTH,  
           BUFFER_INX,  
           BLOCK_OFFSET;
```

Input

- `FILE_SWITCH_OK`
Specifies whether it is permissible to open the next audit file, if necessary, to read the audit block.
- `ABSN`
Indicates the audit block serial number of the desired audit block.

Output

The global array `AUDIT_INFO` is updated. The first word of the audit block just read is at `AUDIT_BUFFERS[BUFFER_INX, BLOCK_OFFSET]`.

- `BLOCK_LENGTH`
Indicates the amount of data, in words, in the data buffer.
- `BUFFER_INX`
Indicates the index in the exported array `AUDIT_BUFFERS` into which the audit block has been placed. This value indicates the row of the `AUDIT_BUFFERS` array to be used.
- `BLOCK_OFFSET`
Indicates the second index in the exported array `AUDIT_BUFFERS` into which the audit block has been placed. This value indicates the offset into the selected row where the beginning of the audit block can be found.

Results

The `Audit_RANDOM_ABSN` procedure returns one of the following values:

- `FALSE`
Indicates that the block was read successfully.
- `TRUE`
Indicates that an error occurred during the read operation. Error values are described later in this section under "Error Results."

In case of any error, the values of the `Buffer_INX` and `Block_Offset` parameters are undefined and should not be used.

AUDIT_NEXT_RECORD Entry Point Parameters

This entry point runs a procedure that retrieves the next audit record of the open audit file. If no reads have been performed on the file, the first call to the `Audit_Next_Record` entry point attempts to return data for the first audit record in the file. When the audit record is successfully read, the contents of the output parameters are valid. The index in which the information was placed is indicated by the value of the `Buffer_INX` and `Record_Offset` parameters. The first word of the read audit record is at `Audit_Buffers[Buffer_INX, Record_Offset]`.

If the `AUDIT_Next_Record` entry point is called after a successful call to either the `Audit_Next_ABSN` or `Audit_Random_ABSN` entry point, it returns the first audit record that begins in the current block. Partial audit records at the beginning of the block are ignored. If no record begins in the current block, subsequent audit blocks are read until the next record is located or an error, such as an end-of-file condition, is encountered.

The ALGOL interface for this procedure has the following specifications:

```
Boolean Procedure AUDIT_NEXT_RECORD (FILE_SWITCH_OK,  
                                     ABSN,  
                                     RECORD_TYPE,  
                                     STRUCTURE_NO,  
                                     RECORD_LENGTH,  
                                     BUFFER_INX,  
                                     RECORD_OFFSET);  
  
Value      FILE_SWITCH_OK;  
Boolean    FILE_SWITCH_OK;  
Integer    ABSN,  
           RECORD_TYPE  
           STRUCTURE_NO,  
           RECORD_LENGTH,  
           BUFFER_INX,  
           RECORD_OFFSET;
```

Input

- `FILE_SWITCH_OK`

Specifies whether it is permissible to open the next audit file, if necessary, to read the audit block.

Output

The global array `AUDIT_INFO` is updated. The first word of the audit block just read is at `AUDIT_BUFFERS[BUFFER_INX, BLOCK_OFFSET]`.

- `ABSN`
Indicates the audit block serial number of the audit block in which this record begins. If the audit record spans multiple audit blocks, the `ABSN` contains the audit block serial number in which the record begins.
- `RECORD_TYPE`
Indicates the audit record type of the record.
- `STRUCTURE_NO`
Indicates the structure number to which this record applies.
- `RECORD_LENGTH`
Indicates the length, in words, of the audit record, including the left and right control words.
- `BUFFER_INX`

Indicates the first index in the exported array AUDIT_BUFFERS into which the audit record has been placed. This value indicates the row of the AUDIT_BUFFERS array to be used.

- RECORD_OFFSET

Identifies the offset within AUDIT_BUFFERS[BUFFER_INX, *] at which the record begins. This offset is the index of the left control word for the audit record.

Results

The AUDIT_NEXT_RECORD procedure returns one of the following values:

- FALSE

Indicates that the audit record was read successfully.

- TRUE

Indicates that an error occurred during the read operation. Error values are described later in this section under "Error Results."

Error Results

Error results are passed from the library back to the calling program through the result value for each procedure. Error result words are consistent with the standard error result word returned by the Enterprise Database Server and have the following format:

```
DEFINE
  RSLTSN          = [47:12] #,      % structure number
  CATEGORY        = [35:08] #,      % error category
  SUBCAT          = [19:16] #,      % error subcategory
  ERROR_FLAG      = [00:01] #;      % SET if error
```

For an XL database, the error result word has the following format:

```
DEFINE
  NEW57RSLTSN    = [47:16] #,      % structure number
  NEW57CATEGORY  = [31:08] #,      % error category
  FORMATLVL      = [21:02] #,      % = 1 for XL database
  SUBCAT         = [19:16] #,      % error subcategory
  ERROR_FLAG     = [00:01] #;      % SET if error
```

The full text of the ALGOL Enterprise Database Server error result interface, including many errors not used by the DMAuditLib library, can be included into a calling program by using the \$INCLUDE command as shown in the following example:

```
$INCLUDE "DATABASE/PROPERTIES" 32000000 - 32999999
```

All the procedures in the DMAuditLib library return a Boolean error result word. This result word has the following format:

- FALSE

Indicates that the procedure completed its operation successfully.

- TRUE

Indicates that the procedure failed to complete its operation. The fields of the result word have the following values:

```

RSLTSN      = 0
CATEGORY    = AEERRORCATEGORY
SUBCAT      = One of the values described
              in Table 20-1.
ERROR_FLAG  = 1
    
```

The error results listed in the following table are defined in the DATABASE/PROPERTIES file and are returned to calling programs in the DMAuditLib library. These error subcategories, values 14 through 30, were added to the AEERRORCATEGORY (15) and return integer values in the SUBCAT field of the DM result word.

Table 20-1. Error Results

Value	Name
14	AEAUDITNOTFOUNDV
15	AESECTIIONNOTFOUNDV
16	AEAUDITNOTOPENV
17	AEENDOFFILEV
18	AEBADPARAMV
19	AEBADINTEGRITYV
20	AEINTERNALERRORV
21	AEREADERRORV
22	AECHECKSUMERRORV
23	AEABSNSEQERRORV
24	AETIMESEQERRORV
25	AEUPPERLIMITV
30	AEWRITEERRORV

When an integrity error of any kind is reported (error values 19 and 21 through 24), the calling program attempts to recover from the error by one of the following means:

- Reading a different audit block by calling AUDIT_RANDOM_ABSN
- Explicitly closing the audit file (AUDIT_CLOSE)
- Implicitly closing the audit file by opening the same or another audit file (AUDIT_OPEN)
- Disconnecting the link from the audit library

Section 21

Database Events Management

Some database exceptions and messages for a running database are captured in an event log file. These captured events are categorized under Event categories. These event messages are written to an event log file only when the Event Category is subscribed to for the database. The event categories are described next in this section under “Events Management Overview.” Event log subscription management as well as viewing events is performed through Database Operations Center. For additional information, refer to “Monitoring Database Events” in the *Database Operations Center Help*.

Events Management Overview

In database events management you can subscribe to the following categories:

- ADMINISTRATIVE – this category covers: database backup, database recovery and reorganizations.
 - Database backup – when a database backup is performed, the events generated contain information about the backup process.
 - Database recovery – contains information about database recovery when it occurs.
 - Reorganizations – contains information about reorganizations when performed on the database.
 - Online garbage collection – contains information about online garbage collect performed on the database.
- DEADLOCK – this category covers the DEADLOCK exceptions occurring in Enterprise Database Server databases. The number following the program title is the sequence number in the program where the DEADLOCK occurred. For additional information, refer to the *Enterprise Database Server Application Program Interfaces Programming Guide*, and the *Enterprise Database Server Interpretive Interface Programming Reference Manual*.

In most cases, when a Deadly Embrace subcategory of a DEADLOCK event is logged in the event log, it will be logged in the sumlog. However, there are some cases where a DEADLOCK is reported internally by Enterprise Database Server, but not logged in the sumlog. For example, if you delete an entry in a table of a set, the attempt to lock another entry of the same table by another user will get logged in the event log, but not logged in the sumlog.

- FATAL_DB_ERROR – when the database encounters errors that cause abnormal termination, these events are generated.
- SECURITY – these events are applicable to the databases protected by guard file(s)

rules. When an application program performs an operation on the database that is not authorized by the guard file rules, the exceptions events are generated.

- PUBLICIO – this category determines whether or not to initialize future event logs with the SECURITYTYPE set to PUBLIC (subscribed) or PRIVATE (unsubscribed). The subscription value of this category is considered when one of the following occurs:
 - The database needs to create a new event log because one does not exist.
 - The database needs to switch to a new event log because the previous event log is either full or corrupted.

Event Log Files

Event log files contain information about all subscribed events and are created in chronological order. The files are created in the same location as the control file of the database.

The file is named <dbname>/system/eventlog and contains textual information in the first 81 characters, and binary information in columns 82-90.

The text information can be internationalized using MLS utilities.

An event file, when full (which is currently set at 20000 records), is closed and a new file is opened. The old file is renamed to <dbname>/eventlog/yyyymmyy>/<hhmmss>. The new file becomes the current <dbname>/system/eventlog.

Currently, Enterprise Database Server does not provide a mechanism to back up these files. You are responsible for the maintenance of these files.

The events generated from the Event Management system do not include all the display information generated by the utilities. Supported analysis tools such as the LOGANALYZER and DUMPANALYZER (if applicable) can be used in addition to these events for a complete analysis.

Examples

The following are examples of a subscription and different event categories.

Every time you subscribe to an event it records a subscription event. The program title cannot exceed 46 characters in the first line, but the maximum size of a program title is 255 characters. If the program title is longer than 46 characters, the % character occurs at the 71st column, causing the remainder of text to be wrapped-around to additional lines. Similarly, any event line that is longer than 70 characters is continued on subsequent lines with the % character on the 71st column; refer to Example 4 for an example of this.

Example 1

The following is an example of a subscription:


```

EVENT:  SUBSCRIBE DEADLOCK
DATE:   1/10/2012           TIME:   8:16:06
PROGRAM NAME:
*SYSTEM/DBCENTER/SERVER ON UI1.
    
```

Example 2

The following example is of a DEADLOCK event. The * (asterisk) indicates that the stack was in a waiting state when the DEADLOCK information was retrieved.

```

EVENT CATEGORY:  DEADLOCK
SUBCATEGORY:    DEADLY EMBRACE
DATE:          1/10/2012           TIME:   9:15:52
MIX NUMBER:    7917  STRUCTURE#:    3  NAME:  D1.
7913 (0868)*   (086B) (SUBRA)OBJECT/TEST/DEADLOCK1 ON DMCP2.
(00014500)
7916 (086b)*   (086C) (SUBRA)OBJECT/TEST/DEADLOCK2 ON DMCP2.
(00014500)
7917 (086C)*   (0849) (SUBRA)OBJECT/TEST/DEADLOCK3 ON DMCP2.
(00014500)
7889 (0849)*   (084B) (SUBRA)OBJECT/TEST/DEADLOCK4 ON DMCP2.
(00014500)
7890 (084B)*   (084D) (SUBRA)OBJECT/TEST/DEADLOCK5 ON DMCP2.
(00014500)
7892 (084D)*   (0850) (SUBRA)OBJECT/TEST/DEADLOCK6 ON DMCP2.
(00014500)
7893 (0850)*   (0858) (SUBRA)OBJECT/TEST/DEADLOCK7 ON DMCP2.
(00014500)
7897 (0858)*   (0859) (SUBRA)OBJECT/TEST/DEADLOCK8 ON DMCP2.
(00014500)
7898 (0859)*   (085D) (SUBRA)OBJECT/TEST/DEADLOCK9 ON DMCP2.
(00014500)
7902 (085D)*   (085E) (SUBRA)OBJECT/TEST/DEADLOCK0 ON DMCP2.
(00014500)
    
```

Example 3

The following is an example of a reorganization event:

```

EVENT CATEGORY:  ADMIN
SUBCATEGORY:    REORG - INFORMATION
DATE:          1/10/2012           TIME:   0:21:37  1/10/2012
MIX NUMBER:    1979  STRUCTURE#:    0  NAME:
1979(0EDD)     (SUBRA)REORGANIZATION/TESTDB ON DMCP2.           (10802998)
DATABASE NAME: TESTDB
MIX      PROCESS      STATUS      START TIME      END TIME
----      -
1992     GEN/D1/3  COMPLETED (120000 RECORDS) 1/10/2012 0:20:41 1/10/2012 0:21:24
2104     FIX/S1/4  COMPLETED (10032 BLOCKS) 1/10/2012 0:21:24 1/10/2012 0:21:37
2105     FIX/S4/5  COMPLETED (48 BLOCKS) 1/10/2012 0:21:24 1/10/2012 0:21:24
    
```

Example 4

The following example is of a database backup event:

Database Events Management

EVENT CATEGORY: ADMIN
SUBCATEGORY:DMUTILITY - INFORMATION
DATE: 1/10/2012 TIME: 0:19:58 1/10/2012
MIX NUMBER: 1787 STRUCTURE#: 0 NAME:

1787(0DCD) *SYSTEM/DMUTILITY ON DEV00.
RELEASE: SSR 56.1 (56.114.0010) DATE: TUESDAY, JANUARY 10, 2012, 12:19 AM.

DB=TESTDB OFFLINE DUMP = TO TESTDBDMP1 ON DMCP2
ESTIMATED DUMP DISK SIZE PER FILE
TOTALSECTORS = 70400
DUMP OF DATABASE (SUBRA)TESTDB
RESTART DATASET:TESTDB/R/DATA ON DMCP2
USING TAPE: TESTDBDMP1 ON DMCP2.

THIS DUMP STARTED DURING AUDIT FILE NUMBER 6
THE LATEST VERSION OF THE DIRECTORY FOR THIS SET OF TAPES IS ON CYCLE 0%
1, VERSION 01
SPECIFY THE INFORMATION WHEN RECOVERINGFROM THIS SET OF TAPES.
DUMP TIMES : #ET: 6.1 PT: 0.2 IO: 1.1

Example 5

The following example is of a recovery event:

EVENT CATEGORY: ADMIN
SUBCATEGORY:RECOVERY - INFORMATION
DATE: 1/09/2012 TIME: 21:23:50 1/09/2012
MIX NUMBER: 5111 STRUCTURE#: 0 NAME:
5111(051D) *SYSTEM/DMRECOVERY ON DEV00.. (0000)
SOFTWAREVERSION:(56.114.3)
ALLOWEDCORE: 10000000, TOTALCORE: 0, MAXBUFF: 1

Example 6

The following example is of an abnormal termination of the database:

EVENT CATEGORY: FATALERROR
SUBCATEGORY: : DS OF DBS
DATE: 1/09/2012 TIME: 21:21:40 1/09/2012
NUMBER: 5083 STRUCTURE#: 0 NAME:
5083(04F5) (SUBRA)TESTDB.

Example 7

The following example is of a security event:

EVENT CATEGORY: SECURITYERROR
SUBCATEGORY: ILLEGAL TO CREATE STORE
DATE: 7/16/2010 TIME: 9:26:10
MIX NUMBER: 3527 STRUCTURE#: 0 NAME:
3527(0750) (DOCTEST)OBJECT/RAM/PRG/GUARD/CREATE/INQUIRE O% (00011302)
N DMCP.

Example 8

The following examples are of online garbage collection events:

```

EVENT CATEGORY: ADMIN
SUBCATEGORY:ONLINE GARBAGE COLLECT - INFORMATION
DATE: 7/30/2013      TIME: 13:24:42
MIX NUMBER: 4373 STRUCTURE#: 0 NAME:
      4373(OD23)      (SUBRA)EVENTDB.
GARBAGE COLLECT FOR STRUCTURE 7 (S3)  STARTED

EVENT CATEGORY: ADMIN
SUBCATEGORY:ONLINE GARBAGE COLLECT - INFORMATION
DATE: 7/30/2013      TIME: 13:24:44
MIX NUMBER: 4475 STRUCTURE#: 7 NAME:
      4475(ODA5)      (SUBRA)GARBAGE/COLLECT/S3/7.
GARBAGE COLLECT FOR STRUCTURE 7 (S3) : SWAP COMPLETED
    
```

Using the Programmatic Interface

Programmatically, you can use DMINQ call (91) to manage the event subscriptions. The following requests are currently available to perform the following two tasks:

- To obtain a list of event categories that are available in the database
- To subscribe or unsubscribe to a specific event category

The above subscription management as well as viewing the events can also be done without directly using the DMINQ call, and this is performed through Database Operations Center.

Examples of Programmatic Subscription and Interface

The following examples demonstrate how to programmatically manage event subscriptions.

Example 1: Sample Code for DMINQ Call

The following is general sample code for the DMINQ call.

```

BEGIN
DATABASE DB;
ARRAY A[0:99];
OPEN INQUIRY DB;
A[0]:=91;
A[1]:=<n>;%n=0 for unsubscribe,
      1 for subscribe,
      2 for list of event categories
A[2]:=<Event number>;
% (1 for Deadlock event); Valid only if A[1]=0 or 1
DMINQ[0] (A);
END.
    
```

Example 2: Getting a List of Event Categories

The following code obtains a list of event categories supported by the database system.

```
BEGIN
DATABASE DB;
ARRAY A[0:99];
OPEN INQUIRY DB;
A[0]:=91;
A[1]:=2;
DMINQ[0] (A);
END.
```

The resulting array A has the following format:

```
[
· Category number for 3 characters
  (value of 1 for deadlock)
· Subscription Status - 1 character
  (value of 1 to subscribe, 0 to unsubscribe)
· Length of the event name
  (the following field) - 3 characters
· Event Name - MLSable name - variable length
]
0..n times followed by a NUL character
```

Where n is the number of events supported by the database system.

The following is an example of the resulting array A:

```
0011018DEADLOCK0020010FATALERROR0030005ADMIN0040013SECURITYERROR0050008PUBLICIO?
```

The previous example of array A can be decoded as follows:

Event Category Number	Event Category Name	Subscribed?
1	DEADLOCK	Yes
2	FATALERROR	No
3	ADMIN	No
4	SECURITYERROR	No
5	PUBLICIO	No

Example 3: Subscribing to an Event Category

Based on the above example, the ADMIN category has a value of 3. The following example subscribes to the ADMIN category.

```
BEGIN
DATABASE DB;
ARRAY A[0:99];
OPEN INQUIRY DB;
A[0]:=91;
A[1]:=1;
A[2]:=3;%For ADMIN category
END.
```

Example 4: Unsubscribing to an Event Category

The following example unsubscribes to the ADMIN category.

```
BEGIN
DATABASE DB;
ARRAY A[0:99];
OPEN INQUIRY DB;
A[0]:=91;
A[1]:=0;
A[2]:=3;%For ADMIN category
END.
```


Section 22

Logging Data Access

LOGACCESS, a DASDL schema specification, allows you to perform the logging of data set access by database applications. The following configuration tasks are required to log data set access.

- Setting System Logging Options
- Set the DASDL LOGACCESS Option
- Enable the LOGACCESS Option

When data access is logged, data access information is written to the system SUMLOG. You can access and filter recorded information using any of the following interfaces:

- MCP Log analyzer
- Database Operations Center
- User-written log analysis applications
- Third-party log analysis products

Caution

For LOGACCESS-enabled structures, all structure access is logged, both update and inquiry. This can result in large increases in resource usage. For example, a single application that performs a sequential FIND operation through a set of 100,000 entries results in 100,000 SUMLOG records written. For another example, a single application that performs 100,000 DELETE operations will result in 200,000 SUMLOG records written.

[Table 22-1](#) identifies the tasks you can perform using the LOGACCESS specification and the heading in this section under which these tasks are described.

Table 22-1. Tasks Related to Using the LOGACCESS specification

To perform this task . . .	Refer to . . .
Set the DMS ACCESS system logging option	System Logging Options

Table 22–1. Tasks Related to Using the LOGACCESS specification (cont.)

To perform this task . . .	Refer to . . .
Specify the DASDL LOGACCESS option	DASDL LOGACCESS Option
Enable/Disable the LOGACCESS option	Enabling the LOGACCESS Option
Change LOGACCESS DMVERB list for structures	Changing the LOGACCESS DMVERB List
Analyze LOGACCESS SUMLOG records	LOGACCESS Analysis

System Logging Options

You can use Security Center or the LOGGING (Logging Options) system command to set the DMS ACCESS option. This action is required in order for ACCESS records to be written to the system SUMLOG.

The following is an example of a LOGGING system command:

```
LOGGING 1, 35
```

Refer to the *System Commands Reference* or *Security Center Help* for additional information about the LOGGING system command.

DASDL LOGACCESS Option

In order to log data access, a DASDL schema update is required to set the option for the selected structures. You can specify the LOGACCESS option as a global default or as a data set physical option. The LOGACCESS option is automatically and implicitly set for all associated sets and subsets of a data set which has the option set.

DASDL also provides the LOGACCESSDMVERBS option which enables logging of the DMVERBS that you select. You can specify the LOGACCESSDMVERBS option as a default or as a data set physical option. The LOGACCESSDMVERBS option is automatically and implicitly applied to all sets and subsets of a data set which has the option specified. When you specify LOGACCESS without specifying the LOGACCESSDMVERBS option, all DMVERBS are logged.

The following figure is a portion of a DASDL schema used throughout the examples of this section. LOGACCESS syntax is demonstrated in BOLD print. Access is logged for all Enterprise Database Server data sets with at LOGACCESS option set. Refer to the *DASDL Programming Reference Manual* for additional information on the LOGACCESS Option including diagrams.


```

% This schema demonstrates the various
% syntactic combinations allowed with
% the LOGACCESS option. This example
% explicitly sets LOGACCESS to TRUE
% for all structures. New LOGACCESS
% syntax is demonstrated in BOLD print
% for allsupported usages.
%
% $RESET ZIP DMCONTROL
% INITIALIZE;
% UPDATE;

DEFAULTS
(
    BUFFERS = 0 + 0 PER RANDOM USER OR 2 PER SERIAL USER,
    PACK = DBPK,
    LOGACCESS = TRUE,    % set for all structures
    LOGACCESSDMVERBS= ALL EXCEPT (DELETE) % log all dmverbs except
                                                % DELETE dmverb
    CHECKSUM = TRUE,
    DUMPSTAMP = TRUE);
    DMSUPPORT = (PROD)DMSUPPORT/TESTDB ON SOFTWARE;

OPTIONS
(
    STATISTICS, ADDRESSCHECK,
    INDEPENDENTTRANS, REAPPLYCOMPLETED,
    AUDIT
);

PARAMETERS
(
    CONTROLPOINT = 2
    ,SYNCWAIT = 2 ,SYNCPOINT = 1000
);

CONTROL FILE
(
    PACK = DBPK
    ,USERCODE = PROD
);

AUDITAREA RESTART DATA SET
(
    RS-NAME           ALPHA(30);
    RS-PNAME          ALPHA(30);
    RS-CNAME          ALPHA(30);
)
CHECKSUM,
MEMORY RESIDENT = ALL,
% LOGACCESS = TRUE,
% LOGACCESS = FALSE,
BLOCKSIZE = 9 RECORDS,
AREASIZE = 1008 SEGMENTS,
POPULATION = 800;

```

Logging Data Access

```
PRODUCT DATA SET
(
  PNAME          ALPHA(42);
  PID            NUMBER(12);
  PDEPARTMENT    ALPHA(28);
  PDATE         NUMBER(12);
) POPULATION = 300000,
  BLOCKSIZE = 100 RECORDS,
% LOGACCESS = TRUE,
  % LOGACCESS = FALSE,
AREASIZE = 10 BLOCKS,
  MEMORY RESIDENT = ALL;

COMPUTER SET OF PRODUCT KEY PID
  DUPLICATES AREASIZE = 1000 ENTRIES;

PLANT DATA SET
(
  STATE          ALPHA(27);
  CITY           ALPHA(27);
  ZIP            NUMBER(12);
) POPULATION = 300000,
  BLOCKSIZE = 100 RECORDS,
% LOGACCESS = TRUE,
  % LOGACCESS = FALSE,
  AREASIZE = 10 BLOCKS,
  MEMORY RESIDENT = ALL;
CP SET OF PLANT KEY ZIP
  DUPLICATES AREASIZE = 1000 ENTRIES;

CUSTOMER DATA SET
(
  CNAME          ALPHA(42);
  CID            NUMBER(12);
) POPULATION = 300000,
  BLOCKSIZE = 100 RECORDS,
% LOGACCESS = TRUE,
  % LOGACCESS = FALSE,
AREASIZE = 10 BLOCKS,
  MEMORY RESIDENT = ALL;

CCC SET OF CUSTOMER KEY CID
  DUPLICATES AREASIZE = 1000 ENTRIES;
PERSONNEL DATA SET
(
  NAME           ALPHA(42);
  NAME1          ALPHA(50);
  NAME2          ALPHA(50);
  NAME3          ALPHA(50);
  NAME4          ALPHA(50);
% ADDRESS       ALPHA(600);
  ID             NUMBER(12);
  ID2            NUMBER(12);
  DEPARTMENT     ALPHA(28);

% SUBPERSONNEL  SUBSET OF PERSONNEL
```

```

%           KEY IS ID,
%           ORDERED LIST;
) POPULATION = 300000,
  %REASIZE = 1000,
  BLOCKSIZE = 100 RECORDS,
% LOGACCESS = TRUE,
  % LOGACCESS = FALSE,
AREASIZE = 10 BLOCKS,
  MEMORY RESIDENT = ALL;
  SUBPERSONNEL2 SUBSET OF PERSONNEL
  WHERE ID2 > 10000
  KEY IS ID;

MANAGER SET OF PERSONNEL KEY ID
  DUPLICATES AREASIZE = 1000 ENTRIES;

```

Enabling the LOGACCESS Option

Logging begins when the LOGACCESS option is enabled on a structure. You can enable a structure through Visible DBS or Database Operations Center.

The following is sample output of structure information. It is extracted from a database control file listing following a DASDL compile to set the LOGACCESS capability, prior to enabling the option.

```

LOG ACCESS CAPABLE = SET
LOG ACCESS ENABLE = RESET

```

The following is sample output of structure information. It is extracted from a database control file listing following the enabling of the option.

```

LOG ACCESS CAPABLE = SET
LOG ACCESS ENABLE = SET

```

When a LOG ACCESS ENABLE command becomes SET, the following types of information are logged:

- When a LOG ACCESS ENABLE is SET or RESET. Information includes the structure name or * for all structures.
- When the structure is opened.
- When the structure is accessed by a selected DMVERB.

Note: LOGACCESS is implicitly set for SETS and SUBSETS that span a data set that is configured to use LOGACCESS.

[Table 22-2](#) identifies the actions that result in the logging of DMVERBS and provides information recorded for each action.

Access is recorded for the following DMVERBS:

- ASSIGN
- ASSIGNLOB

Logging Data Access

- CREATESTORE
- DELETE
- DELETEDLOB
- FIND
- FINDLOB
- FREE
- GENERATE
- INSERT
- LOCK
- LOCKSTORE
- REMOVE
- SECURE

In addition to the SUMLOG record information provided for all types of DMS records (such as program name, time-of-day, and so on), the following information is recorded for DMVERB LOGACCESS:

- Structure Name
- Structure Number
- Section Number (if sectioned)
- Record Address
- Record Serial Number (RSN) for XE structures
- DMEXCEPTION
- DMVERB (from the list above)

Most of the DMVERBS generate one log entry, although some generate two. Update verbs also have audit file and ABSN numbers logged.

The following table provides information logged for DMVERB access.

Table 22-2. DMVERB Access

DMVERB	Information Logged
ASSIGN <data set> TO <link>	Two log records: <ul style="list-style-type: none">• The <data set> which contains the <link> item• The <data set> assigned to the <link> item The audit file and ABSN numbers in both log entries point to the audit record for the ASSIGN operation.

Table 22-2. DMVERB Access (cont.)

DMVERB	Information Logged
ASSIGN <link1> TO <link2>	Two log records: <ul style="list-style-type: none"> • The data set> which contains the <link2> item • The <data set> assigned to the <link1> item The audit file and ABSN numbers in both log entries point to the audit record for the ASSIGN operation.
ASSIGN LOB/FIND LOB/DELETE LOB	The <data set> which contains the LOB item
ASSIGN NULL	The <data set> which contains the <link> item
DELETE	Two log records: <ul style="list-style-type: none"> • LOCK verb with the <data set> • DELETE verb with <data set> information
DELETE <set selection expression>	Two log records: <ul style="list-style-type: none"> • LOCK verb with the <data set> • DELETE verb with <set> information along with its associated <data set> information.
FREE <data set>	<data set> information
GENERATE <subset>	<subset> information only
INSERT <data set> INTO <subset>	Two log records: <ul style="list-style-type: none"> • <subset> information only • <data set> information The audit file and ABSN numbers in both log entries point to the audit record for the INSERT operation.
LOCK/FIND/SECURE <data set>	<data set> information
LOCK/FIND/SECURE<set>	<set> information along with its associated <data set> information. For FIND KEY OF <set>, only <set> information is present.
REMOVE <data set> From <subset>	Two log records: <ul style="list-style-type: none"> • <subset > information only • <data set> information The audit file and ABSN numbers in both log entries point to the audit record for the REMOVE operation.

Notes:

- Data accessed through REMAP is logged as access through a physical structure.
- An Enterprise Database Server exception word is recorded in the SUMLOG when an exception occurs. No record (record address, RSN and section), and audit (audit file number and ABSN) information is included.

- *The structure number in the DMS ACCESS log record is the structure number on which the DMVERB is performed and can be different from the structure number in the exception word when an exception occurs. The structure number in the exception word is the structure that causes the exception during the DMVERB operation. For example: For DMVERB on an ACCESS structure, the exception word contains the structure number of the data set that the ACCESS structure spans. A DMVERB on a SET can result in an exception on the data set that the set spans; the reverse is also true.*

Changing the LOGACCESS DMVERB List

You can change the DMVERB list for structures using the DMCONTROL LOGACCESS command. The changes take effect the next time the database is opened.

Example 1

The following is sample output of structure D information from a control file listing. The DMVERB list for structure D is ALL (which is the default).

```
LOG ACCESS DMVERB      = ALL
```

Example 2

The following DMCONTROL statement changes the DMVERB list of structure D.

```
RUN $SYSTEM/DMCONTROL ("DB=TESTLOBDB LOGACCESS STRUCTURE D
                        DMVERBS = (CREATESTORE, DELETE, LOCK)")
```

Refer to the DMCONTROL LOGACCESS statement in [Section 5, Initializing and Maintaining](#), for additional information on changing the DMVERB list for structures.

LOGACCESS Examples

The following examples demonstrate enabling the LOGACCESS option with Visible DBS, using the DMCONTROL LOGACCESS command and performing LOGACCESS analysis.

Enabling LOGACCESS Option with Visible DBS

For syntax and command descriptions, refer to the STRUCTURE CHANGE command in [Section 12, Communicating with the Database](#).

Example 1

This example is of a Visible DBS command used to enable all LOGACCESS capable structures:

```
<Visible DBS number> SM STRUCTURE *(LOGACCESS SET)
```

Example 2

This example is of a Visible DBS command used to disable the LOGACCESS option for the PLANT data set:

```
<Visible DBS number> SM STRUCTURE PLANT
(LOGACCESS RESET)
```

Using the DMCONTROL LOGACCESS Command

For syntax and command descriptions, refer to the DMCONTROL statement in [Section 5, Initializing and Maintaining](#), .

Example 3

This example is of a DMCONTROL command used to disable all LOGACCESS-capable structures:

```
DB = <database name> LOGACCESS RESET
```

Example 4

This example is of a DMCONTROL command used to change the DMVERB list for structures D1, D2, and D3. The DMVERB list for structures D1 and D2 is changed to LOCK DMVERB only. The DMVERB list for structure D3 is changed to include all DMVERBS.

```
DB = <database name> LOGACCESS
STRUCTURE D1, D2 DMVERBS = (LOCK),
STRUCTURE D3 DMVERBS = ALL
```

Database Operations Center provides the necessary configuration forms for the DMCONTROL LOGACCESS command. Refer to the Database Operations Center product for more information.

LOGACCESS Analysis**Loganalyzer**

Examples 5 and 6 assume that the following required configuration tasks were successfully executed:

- Setting the system log option for Major Type 1, Minor Type 35 (as described in "System Logging Options" earlier in this section).
- Setting the DASDL schema data set specification for structure-based LOGACCESS capability (following example 1 earlier in this section).
- Enabling the structure-based LOGACCESS option (following example 1 earlier in this section).

Example 5

This example extracts DMS ACCESS records written when the structures were enabled:

Logging Data Access

```
Input:
LOG "LOGACCESS/SUMLOG ON RDBCP" DMS (ACCESS) FIND "ENABLE"
Output:
14:35:51 DMS 7643 (PROD)TESTDB.
      DATABASE ACCESS : ENABLE LOGACCESS
      USERCODE: PROD. ACCESSCODE: PROD.
      USERCODE PRIVILEGES : PU
      DATABASE NAME : (PROD)TESTDB.(MIX 7643, STACK 0C4C
      )
      STRUCTURE NAME : *
```

Example 6

This example extracts DMS ACCESS records written when the structure PLANT was accessed.

```
Input:
LOG "LOGACCESS/SUMLOG ON RDBCP" DMS (ACCESS) FIND "PLANT"
Output:
14:35:53 DMS 7642 (PROD)CANDE/CODE6550 ON RDBCP.
      DATABASE ACCESS : ACCESS RECORD
      USERCODE: PROD. ACCESSCODE: PROD.
      USERCODE PRIVILEGES : PU
      DMVERB : CREATESTORE
      DATABASE NAME : (PROD)TESTDB. (MIX 7643, STACK 0C4C
      )
      STRUCTURE NAME : PLANT, STRUCTURE # 5
      RECORD ADDRESS : (000000000064)
      AUDIT FILE NUMBER : 1
      ABSN : 15
14:35:56 DMS 7642 (PROD)CANDE/CODE6550 ON RDBCP.
      DATABASE ACCESS : ACCESS RECORD
      USERCODE: PROD. ACCESSCODE: PROD.
      USERCODE PRIVILEGES : PU
      DMVERB : CREATESTORE
      DATABASE NAME : (PROD)TESTDB. (MIX 7643, STACK 0C4C
      )
      STRUCTURE NAME : PLANT, STRUCTURE # 5
      RECORD ADDRESS : (00000000006E)
      AUDIT FILE NUMBER : 1
      ABSN : 16
14:35:59 DMS 7642 (PROD)CANDE/CODE6550 ON RDBCP.
      DATABASE ACCESS : ACCESS RECORD
      USERCODE: PROD. ACCESSCODE: PROD.
      USERCODE PRIVILEGES : PU
      DMVERB : CREATESTORE
      DATABASE NAME : (PROD)TESTDB. (MIX 7643, STACK 0C4C
      )
      STRUCTURE NAME : PLANT, STRUCTURE # 5
      RECORD ADDRESS : (000000000078)
      AUDIT FILE NUMBER : 1
      ABSN : 17
14:36:02 DMS 7642 (PROD)CANDE/CODE6550 ON RDBCP.
      DATABASE ACCESS : ACCESS RECORD
      USERCODE: PROD. ACCESSCODE: PROD.
      USERCODE PRIVILEGES : PU
```



```
DMVERB : CREATESTORE
DATABASE NAME : (PROD)TESTDB. (MIX 7643, STACK 0C4C
)
STRUCTURE NAME : PLANT, STRUCTURE # 5
RECORD ADDRESS : (000000000082)
AUDIT FILE NUMBER : 1
ABSN : 18
```

Database Operations Center

Database Operations Center provides the interfaces that enable comprehensive filtered queries for LOGACCESS analysis. Refer to the *Database Operations Center Help* for additional information.

User-Written Applications

You can write applications that enable comprehensive filtered queries for LOGACCESS analysis. Methods to accomplish this are detailed in the section titled, "Writing Log Analysis Programs" in the *System Log Programming Reference Manual*.

Section 23

Database Encryption

Database encryption provides customers with the ability to transparently apply encryption within the database making the data unreadable to those who do not have the proper key to decode the data. Recompile is required to handle record format/size changes and you should review the restrictions section to determine if any application modifications are necessary.

The Enterprise Database Server for ClearPath MCP supports the use of two levels of database encryption: field level encryption (FLE) and structure level encryption (SLE).

Field Level Encryption (FLE)

Note: *The terms field and item are used interchangeably.*

The DATAENCRYPT option enables you to encrypt an individual field, multiple fields, or all Alpha, Number, and Real fields in a data set. An existing database can be encrypted without changing any existing applications. FLE requires record format change reorganization. After the database is encrypted, user programs must be recompiled.

Structure Level Encryption (SLE)

The STRENCRYPT option enables you to encrypt an entire file: a data set and all spanning sets and subsets of that data set. An existing database can be encrypted without changing any existing applications. SLE requires file record format change reorganization. After the database is encrypted, user programs do not need to be recompiled.

Before using the database encryption feature, you need to install the Unisys MCP Cryptographic Services product. You must also install Security Center and the MCP security products, both of which are installed as part of the MCP software installation.

Database Encryption Components and Interdependencies

The following figures illustrate the configuration of the Enterprise Database Server and how it interfaces with Security Center for key management and MCP Cryptographic Services for database encryption and decryption. MCP Cryptographic Services work as an agent for the Enterprise Database Server to interface with other components.

[Figure 23-1](#) illustrates the relationship between the DASDL compiler, MCAPISUPPORT library, key manager, and the Security Center database during the DASDL compilation.

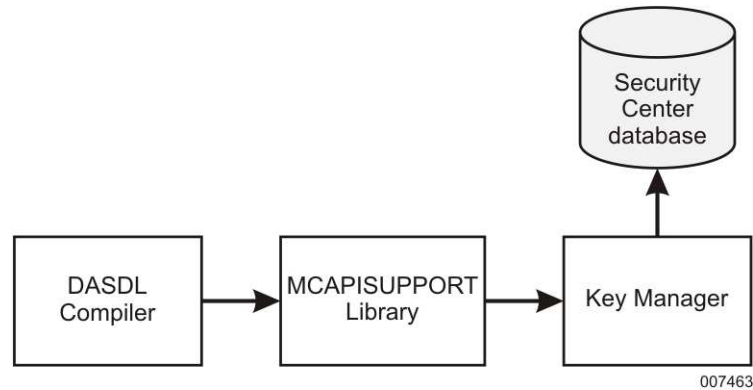
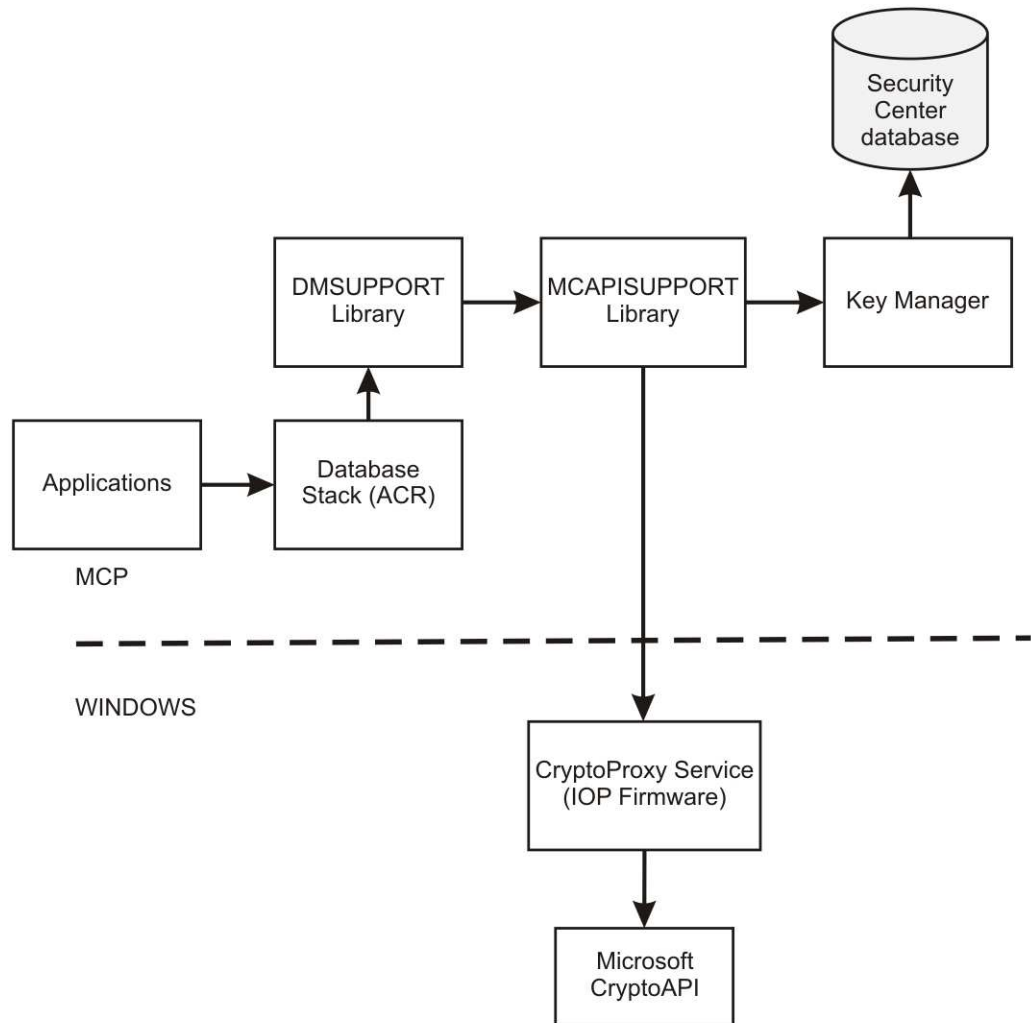


Figure 23-1. Compile-Time Database Encryption Configuration

When you specify the DATAENCRYPT option in the DASDL source, the DASDL compiler calls the key manager indirectly, through the MCAPISUPPORT library, to create a key set. There is one unique key set for a database. Key sets are stored in the Security Center database. The Security Center database manages encryption keys for all Unisys products including the Enterprise Database Server DATAENCRYPT key sets.

[Figure 23-2](#) illustrates the relationship between the Accessroutines, DMSUPPORT library and the MCAPISUPPORT library when applications are accessing a database encryption-capable database.



007464

Figure 23-2. Run-Time Database Encryption Configuration

When the first user opens an encrypted database, the DMSUPPORT library links to the MCAPISUPPORT library and opens the unique key set. Once the key set is opened, the DMSUPPORT library calls routines in the MCAPISUPPORT library to encrypt or decrypt a record.

Refer to Sections 9 and 10 of the *Security Overview and Implementation Guide* and the *Security Center Help* for information about the Security Center and Crypto Service components

Components

The following information describes the database encryption components:

DASDL compiler – calls the key manager indirectly through the MCAPISUPPORT library to create a key set.

MCAPISUPPORT library – interfaces with the Enterprise Database Server for all of the encryption and decryption functions, and interfaces with Security Center for tasks related to key management. For additional information refer to the *Security Overview and Implementation Guide*.

Key Manager – creates the key set and sends the key set to the Security Center database and the MCAPISUPPORT library.

Security Center – enables the security administrator to configure and manage several MCP security modules through a single graphical user interface (GUI). Security Center database is enhanced to manage Enterprise Database Server database encryption key sets. It is extremely important that you back-up the key sets because the encrypted data will *not* be seen unless the key sets are stored in the Security Center database. Refer to the BACKUPKEYSET option in the *DASDL Programming Operations Reference Manual* for information about backing up key sets.

CryptoProxy Service – runs on the Windows environment of the Clearpath MCP server and is invoked by the MCAPISUPPORT library. It calls the Microsoft CryptoAPI to manage keys and certificates, and uses it for the cryptography functions used by user applications.

DMSUPPORT Library – links to the MCAPISUPPORT library and opens the unique key set.

Database Stack (ACR) – contains all the information necessary for the Enterprise Database Server Accessroutines to manage a database.

Cryptographic services - provided through the following programs and services in an MCP environment:

- CryptoProxy service
- MCAPISUPPORT library

These programs and services, in conjunction with TLS and Kerberos Security, provide data protection and strong user authentication within an MCP environment. For detailed information, refer to the *Security Overview and Implementation Guide*.

Using Database Encryption

Encryption Key Set

When you specify the DATAENCRYPT option or the STRENCRYPT option in the DASDL source, a key set is created. This key set is composed of multiple keys, and the DASDL compiler assigns a key in a key set for a data set that contains one or more encrypted items.

In order to use FLE or SLE, a DASDL source is required to enable the setting of several options.

Refer to the *Enterprise Database Server for ClearPath MCP Data and Structure Definition Language (DASDL) Programming Reference Manual* for additional information about the DATAENCRYPT option, STRENCRYPT option, and other database encryption syntaxes.

Set the DATAENCRYPTKEYSET option under the defaults specification so that a crypto key set can be used for the encryption process. A syntax error occurs if the DATAENCRYPTKEYSET option is not specified and the database contains encrypted data. For a database that does not have a key set associated with it, this option instructs the DASDL compiler to call the MCAPI SUPPORT library which then calls the Key Management library to create a key set for the database. A key set contains the database encryption keys and an RSA Master Key. The database encryption keys are the AES-256 keys that are used to encrypt data in the database. There are 16 database encryption keys in a key set. RSA is the most widely used public-key crypto-system today. The 4096-bit RSA Master key is considered to be a very large and secure key by today's standards. This RSA Master key is used to protect the database encryption keys. A unique key set is created for each database that uses encryption. If this option is specified for a database that already has a key set associated with it, then the existing key set is used for the encryption process. The name of the key set is stored in the control file and can be listed using the LIST or WRITE statement in DMUTILITY. Users are responsible for backing up the key set manually using Security Center.

For security purposes, users can change the existing key set by using either the NEWKEYSET or the REPLACEKEYS option in a subsequent DASDL update.

- NEWKEYSET Option

The NEWKEYSET option allows users to generate new database encryption keys and a new RSA Master Key. This change requires that a database reorganization be performed to decrypt the existing data with the old key set and then encrypt it again with the new key set. User programs do not need to be recompiled. However, if a set or subset contains an FLE key, or the structure is SLE, then the sets or subsets must be generated from the data set. Note that a reorganization cannot generate a manual subset with encrypted keys. There is a maximum of 300 structures in one reorganization at a time.

- REPLACEKEYS Option

The REPLACEKEYS option allows users to generate a new RSA Master Key and re-encrypt all the existing database encryption keys. Since the database encryption keys remain the same, there is no need to perform a reorganization to decrypt and re-encrypt the data. This change only requires a control file update to reflect the change; the applications do not need to be recompiled. Once the RSA Master Key has been changed, it can no longer be used to decrypt an old database backup. Therefore, it is strongly recommended that a full database backup be taken after the key set is changed.

- COPYKEYSET Option

A database can be copied from one usercode to another usercode, and can also be retitled as a different database name. Since the database name includes the usercode as part of the key set name, copying the database under a new usercode is considered a separate database from the original one and requires a different key set name. The option COPYKEYSET is used to create a new key set from an existing key set. A new RSA Master Key is created during this process. All the database encryption keys from the original key set are copied to the new key set and encrypted with the new RSA Master Key. The COPYKEYSET option is only performed if a different usercode is specified under the CONTROL FILE specification in DASDL. This change only requires a control file update to reflect the change.

Set the BACKUPKEYSET option under the Options Specification to specify that the key set needs to be manually backed-up before the database can be accessed. By default, the BACKUPKEYSET option is SET. If the BACKUPKEYSET option is SET and the key set is not manually backed-up, an OPEN ERROR 120 is returned when the database is opened. This option can be changed by a subsequent DASDL update which also requires an update to the control file.

It is recommended that the BACKUPKEYSET option is always SET, since a database cannot be recovered if the keys are lost. For information on how to back up the key set, refer to the *Security Center Help*.

If Remote Database Backup (RDB) is enabled for the database, or if you plan to copy the database to another host (for example, using the QUIESCE command), you must use Security Center to export the key set from the primary host and then import the key set on the secondary host.

Encryption Algorithm

Set the DATAENCRYPTTYPE option under the parameters specification to specify the encryption algorithm. Two encryption algorithms are supported: AESGCM and AESHMAC. Only one algorithm is permitted per database. Use the AESGCM algorithm for structure level encryption. You can use either AESGCM or AESHMAC algorithms for field level encryption; however, it is recommended that you use the AESGCM algorithm as it provides faster performance. The AESHMAC algorithm is provided as a back-up in the event that the AESGCM algorithm becomes weakened during an attack. If no algorithm is specified, AESGCM is used as the default algorithm to encrypt the data. The encryption algorithm can be changed with a subsequent DASDL update. These two encryption algorithms require different "padding" to the data which makes the record size different if the algorithm is changed. Therefore, if the algorithm is changed, a reorganization and recompilation of the applications is required.

DATAENCRYPT Option

The DATAENCRYPT option enables you to encrypt specific data items in a data set within an Enterprise Database Server database. Only alpha, numeric, real, or group data item types are valid for FLE. The items inside a group cannot specify the DATAENCRYPT option. An item in a remap must have the same DATAENCRYPT option with its associated data set item.

You can set the DATAENCRYPT option at the global database level or data set level. If this option is set at the data set level, all alpha, numeric, real, and group items in the data set inherit the DATAENCRYPT option. Additionally, you can set the option at the structure level in the physical options for data sets, or for selected items in a data set. The DATAENCRYPT option cannot be used for restart data sets, internal structures, or partitioned data sets.

If a record contains an encrypted item, the data is encrypted on both memory and disk. The record size is increased based on the encryption algorithm being used. To optimize runtime performance and storage usage, all encrypted items are automatically moved to the end of a record. If you change the DATAENCRYPT option for an item, a record format

change reorganization is required and user programs must be recompiled. For more information about using the DATAENCRYPT option, refer to the *Enterprise Database Server for ClearPath MCP Data and Structure Definition Language (DASDL) Programming Reference Manual*.

The following example is of a DASDL source using data encryption for Social Security and driver's license numbers:

```
OPTIONS
(
  BACKUPKEYSET
);
PARAMETERS
(
  DATAENCRYPTTYPE = AESGCM
);
DEFAULTS
(
  DATAENCRYPTKEYSET
);
EMPLOYEE DATA SET
(
  EMPLOYEEID NUMBER(8) DATAENCRYPT = TRUE;
  FIRST NAME ALPHA (30);
  LAST NAME ALPHA (30);
  SSNUMBER NUMBER (9) DATAENCRYPT = TRUE;
  ADDRESS GROUP
  (
    STREETNO NUMBER (5);
    STREETNAME ALPHA (30);
    CITY ALPHA (10);
    STATE ALPHA (2);
    ZIPCODE NUMBER (6);
  );
  HIREDATE DATE;
  SERVICEYEAR REAL;
  PHONENO REAL;
  DRIVELICENSE ALPHA (10) DATAENCRYPT;
);
PHONESET SET OF EMPLOYEE KEY IS PHONENO,
INDEX SEQUENTIAL;
```

Using Encrypted Key Items

When a key item has the DATAENCRYPT option set, that key will be hashed into a 32-byte hash value. As a result, the size of the table (TABLESIZE or BLOCKSIZE) needs to be extended. The same key value will have the same hashed value as long as it is in the same set or subset. Keys in different sets or subsets will have different hashed values even if they contain the same data. Once a key item is hashed, it cannot be undone; this creates the some restrictions for querying. Refer to the topic "Restrictions" in this section for more information.

Database Encryption

The following example is of a DASDL source using data encryption for the employee ID, social security number, address and driver's license number. These encrypted items can be used as key items of the sets.

```
OPTIONS
(
  BACKUPKEYSET
);

PARAMETERS
(
  DATAENCRYPTTYPE = AESGCM
);

DEFAULT
(
  DATAENCRYPTKEYSET
);

EMPLOYEE DATA SET
(
  EMPLOYEEID    NUMBER(8)          DATAENCRYPT = TRUE
  FIRST NAME    ALPHA (30);
  LAST NAME     ALPHA (30);
  SSNUMBER      NUMBER (10)        DATAENCRYPT = TRUE;
  % (or) DATAENCRYPT = FALSE
  ADDRESS GROUP
  (
    STREETNO    NUMBER (6);
    STREETNAME  ALPHA (30);
    CITY        ALPHA (10);
    STATE       ALPHA (2);
    ZIPCODE     NUMBER (6);
  ) DATAENCRYPT ;
  HIREDATE     DATE;
  SERVICEYEAR  REAL;
  PHONENO      REAL DATAENCRYPT;
  DRIVELICENSE ALPHA (10) DATAENCRYPT;
);

PHONESET SET OF EMPLOYEE KEY IS PHONENO,
INDEX SEQUENTIAL;
ID-SSN-SET SET OF EMPLOYEE KEY IS
  (EMPLOYEEID, SSNUMBER);
ID-ADDRESS-SET SET OF EMPLOYEE KEY IS
  (EMPLOYEEID, STREETNO, ZIPCODE)
INDEX SEQUENTIAL;
ID-SSN-SET SET OF EMPLOYEE KEY IS
  (EMPLOYEEID, SSNUMBER);
ID-ADDRESS-SET SET OF EMPLOYEE KEY IS
  (EMPLOYEEID, STREETNO, ZIPCODE);
```

Restrictions

The following restrictions apply to the items specified with the DATAENCRYPT option:

- Only alpha, numeric, real, and group data item types are valid for field level encryption.
- The DATAENCRYPT option cannot be used for restart data sets, internal structures, or partitioned data sets.

The following restrictions apply to key items specified with the DATAENCRYPT option:

- Only FIND equal is supported.
- Linear search is not supported if
 - a user selection expression contains an encrypted key item and a non-encrypted signed numeric or real item.
 - a user selection expression contains an encrypted REAL type key item.
- The size of an encrypted numeric (NUMBER) key item must be in multiples of bytes. That is, the encrypted numeric item must be an even number of digits.
- An encrypted group cannot be used as a key of a set or subset. To use a group item as one of the keys in the set or subset, you must declare the individual items of the group as the keys rather than declaring the group itself.
- The ASCENDING and DESCENDING options cannot be specified for an index sequential set or subset containing encrypted key items. The ordering of the set or subset is based on the order of the hashed key values and not on the order of the unencrypted (clear text) key values.
- An Access of a random, direct, or ordered data set cannot contain encrypted key items.
- An item specified with the DATAENCRYPT option cannot be used as key data.
- An Index Random set cannot contain encrypted key items.
- An item specified with the DATAENCRYPT option cannot be used as a part of SELECT, VERIFY, or WHERE clause.
- Only OFFLINE and USEREORGDB reorganizations are supported if an index sequential set or subset contains any encrypted key items. ONLINE reorganization and online garbage collection are not supported if an index sequential set contains any encrypted key items.
- A reorganization of a set or subset with encrypted keys must be generated from its data set. Reorganization cannot generate a manual subset with encrypted keys.

Additionally, for database reorganization, there is a restriction of up to 300 structures in one reorganization at a time. It is recommended that you use REPLACEKEYS instead of the NEWKEYSET option when changing the key set.

STRENCRYPT Option

The STRENCRYPT option encrypts the entire selected data set within an Enterprise Database Server database. Additionally, all spanning sets and subsets of the database that belong to the data set are automatically encrypted. This feature is also called structure level encryption (SLE).

Database Encryption

You can set the STRENCRYPT option at the global default or at the structure level in the physical options for data sets. If a database has the STRENCRYPT option defined as a global default, all of the disjoint standard fixed format data sets and their index sequential spanning sets and subsets are encrypted.

The STRENCRYPT and DATAENCRYPT options are mutually exclusive. The CHECKSUM option must be set for the structure with STRENCRYPT option set. When a structure has the STRENCRYPT option set, the block size of the structure is increased according to the encryption algorithm; the encrypted data must be a multiple of 16 bytes in length and requires an additional 30 bytes for the encryption algorithm. If the VSS2OPTIMIZE or VSS3OPTIMIZE option is set in addition to a specification of BLOCKSZ for the structure with the STRENCRYPT option set, you must adjust the block size of the structure to maintain the VSS2OPTIMIZE or VSS3OPTIMIZE setting.

If you change the STRENCRYPT option for a data set, a file format change reorganization is required. A reorganization of a set or subset must be generated from its data set. Note that reorganization cannot generate a manual subset with encrypted keys. User programs do not need to be recompiled.

For more information about using the STRENCRYPT option, refer to the *Enterprise Database Server for ClearPath MCP Data and Structure Definition Language (DASDL) Programming Reference Manual*.

The following example of a DASDL source shows the STRENCRYPT option set at the data set physical option of the EMPLOYEE data set. This setting encrypts the EMPLOYEE, PHONESET, ID-SSN-SET, and ID-ADDRESS-SET physical files.

```
OPTIONS
(
    BACKUPKEYSET
);
PARAMETERS
(
    DATAENCRYPTTYPE = AESGCM
);
DEFAULTS
(
    DATAENCRYPTKEYSET
);
EMPLOYEE DATA SET
(
    EMPLOYEEID  NUMBER(8);
    FIRST NAME  ALPHA (30);
    LAST NAME   ALPHA (30);
    SSNUMBER    NUMBER (9);
    ADDRESS GROUP
    (
        STREETNO NUMBER (5);
        STREETNAME ALPHA (30);
        CITY ALPHA (10);
        STATE ALPHA (2);
        ZIPCODE NUMBER (5);
    );
    HIREDATE    DATE;
```

```

SERVICEYEAR REAL;
PHONENO REAL;
DRIVELICENSE ALPHA(10);
)STRENCRYPT = TRUE;

PHONESET SET OF EMPLOYEE KEY IS PHONENO,
INDEX SEQUENTIAL;
ID-SSN-SET SET OF EMPLOYEE KEY IS
(EMPLOYEEID, SSNUMBER);
ID-ADDRESS-SET SET OF EMPLOYEE KEY IS
(EMPLOYEEID, STREETNO, ZIPCODE);
INDEX SEQUENTIAL;
ID-SSN-SET SET OF EMPLOYEE KEY IS
(EMPLOYEEID, SSNUMBER);
ID-ADDRESS-SET SET OF EMPLOYEE KEY IS
(EMPLOYEEID, STREETNO, ZIPCODE);

```

Compiler-DMSII Compatibility Matrix for Encrypted Items

Compiler	Enterprise Database Server Release					
Release	57.1		58.1		59.1	
	Data Item	Key Item	Data Item	Key Item	Data Item	Key Item
57.1			X		X	
58.1			X		X	
59.1			X		X	X

Note: The compilers in the compatibility matrix are for ALGOL and COBOL85 only.

DMSII-MCP Compatibility Matrix for SLE

DMSII	MCP		
	58.1	59.1	60.0 or later
60.0	Only IC MCP-058.1A.67 or later	Only IC MCP-059.1A.37 or later	X

Error Handling

The MCAPI support library opens the key set and performs the encryption and decryption process. If an error is returned from the MCAPI support library, the error message text from the MCAPI support library is displayed as well as an error message from the Enterprise Database Server.

System Configuration Errors

If the MCAPI support library is installed, and encryption is enabled on a system that does not have the firmware to support it, the DASDL compiler generates the following error:

```
LINKAGE FAILED DUE TO MISSING LIBRARY OBJECT
```

Normally, this error is related to a problem in the system configuration. To verify your system configuration, do the following:

Note: The following procedure assumes that the halt/load unit is DISK.

1. On the ODT, enter **PD *SYSTEM/SECURITYCENTERDB/DMSIIRUNTIME ON DISK**.
 - If the SYSTEM/SECURITYCENTERDB/DMSIIRUNTIME file does not exist, the Security Center software is not installed.
To install Security Center and initialize the Security Center database, contact the Unisys Support Center.
 - If the SYSTEM/SECURITYCENTERDB/DMSIIRUNTIME file exists but the *DESCRIPTION/SECURITYCENTERDB/INSTALLED file is not present, the Security Center software is installed; however, the Security Center database is not initialized. To initialize the Security Center database, do the following:
 - a. Connect to Security Center from the client workstation.
 - b. Click **MCP Cryptographic Services Manager**.
Note: A warning displays if the database is not initialized.
 - c. Complete the initialization using the database installation wizard.

Note: For more information on initializing the Security Center database, see the topic "Installing Security Center" in Section 7, "Security Configuration" of the MCP Security Overview and Implementation Guide.

2. On the ODT, enter **PD *SECURITYCENTERDB/= ON DISK**.
The ODT returns the SECURITYCENTERDB database files.
3. Enter **SL KMAPISUPPORT** to verify that the KMAPISUPPORT library is established.
If the KMAPISUPPORT support library is established, the ODT returns the following message:

```
SL KMAPISUPPORT = *SYSTEM/KMAPI/SUPPORT ON DISK
```
4. Enter **SL MCAPISUPPORT** to verify that the MCAPISUPPORT support library is established.
If the MCAPISUPPORT support library is established, the ODT returns the following message:

```
SL MCAPISUPPORT = *SYSTEM/MCPCRYPTOAPI/SUPPORT ON DISK
```
5. If KMAPISUPPORT and MCAPISUPPORT were not automatically installed, enter the following commands to install them manually:

SL KMAPISUPPORT = *SYSTEM/KMAPI/SUPPORT ON DISK: ONEONLY, TRUSTED, LINKCLASS=1

SL MCAPISTATUS = *SYSTEM/MCPCRYPTOAPI/SUPPORT ON DISK: ONEONLY, TRUSTED, LINKCLASS=1

6. Enter **NA MCAPI STATUS**.

The ODT displays the status of each CryptoProxy service.

If the Cryptoproxy or Cryptoproxy CPMCryptoProxy statuses are unavailable, the ODT displays a reason. For example

```
Database Encryption Not Available. CPM is too old
or
```

Database Encryption Not Available. CRYPTOFILE Feature Key Not Installed

Note: If either CryptoProxy service is unavailable, contact Unisys Support.

7. Enter **DBS**.

The ODT displays a list of all active database stacks.

- If the SECURITYCENTERDB is open, it appears in the list. For example, the following DBS output identifies SECURITYCENTERDB by mix number 5921.

```
--Mix-Pri--Usr----- 2 ACTIVE DATABASES -----
5921  50  1 Job SECURITYCENTERDB
7654  50  2 Job (USER1) (USER1)EMPLOYEEEDB
```

- If the SECURITYCENTERDB is not open, it does not appear in the list. Do the following to open the SECURITYCENTERDB:

- a. Enter **LIBS NAME =KMAPI=** and identify the KMAPISUPPORT mix number.

For example, the following LIBS output identifies KMAPI by mix number 6071.

```
-Mix--Frz--Shr--Usr-- 1 FROZEN LIBRARY (ALL) =KMAPI= --
6071 Conn Priv Job *SYSTEM/KMAPI/SUPPORT
```

- b. Enter **<KMAPI MIX number> AX OPEN DATABASE** , where <KMAPI MIX number> is the mix number that you identified in this step.

For example, **6071 AX OPEN DATABASE**.

The SECURITYCENTERDB opens.

Encryption and Decryption Errors

For an encrypted database, the key set needs to be opened at the time the database is opened.

If the key set cannot be opened, the following open error 120 is returned:

```
OPEN DATAENCRYPT KEY SET FAILED OR KEY SET IS NOT BACKED UP
```

Database Encryption

If a database contains structure level encryption and you use an MCP that does not support structure level encryption, you received error 123:

```
CURRENT MCP LEVEL DOES NOT SUPPORT STRUCTURE ENCRYPTION
```

During the encryption and decryption process, one of two types of errors may occur. If the database integrity cannot be maintained, an error occurs and the database shuts down. The second type of error is returned if the database integrity can be maintained, and in this case, the database does not shut down. Typically, if the error occurs during the encryption process, it will be SYSTEMERROR 7 and the database will be terminated. If it occurs during the decryption process, it most likely will be INTEGRITYERROR 10, and the database will not terminate.

Note: If the CryptoProxy Service is down during the encryption and decryption process, a fatal error will occur and database will shut down.

Example 1

The following example is of an error during a FIND Operation. The database does not shutdown.

```
#7950 DISPLAY:(USER1)TESTDB DATA ENCRYPT/DECRYPT FAILED
@ 40734500 IN ACR STRUCTURE #3 D1.
#7950 DISPLAY:(USER1)TESTDB: NON-FATAL
PROGRAMDUMP BY 7905/7950.
#7950 INTEGRITYERROR 10#3 @ (00020250)
#E-DS @ 00020250, 00021340.
Program Dump:
(USER1)PDUMP/OBJECT/TEST/141114/003344/07950
ON DUMPS.
#ET=0.7 PT=0.3 IO=0.3
```

Example 2

The following example is of an error during a STORE Operation. In this case SYSTEMERROR 7 is returned, and the database is terminated.

```
#7910 DISPLAY:(USER1)TESTDB DATA ENCRYPT/DECRYPT FAILED
@ 41053600 FAULT IN ACR STRUCTURE #3 D1.
#7910 DISPLAY:(USER1)TESTDB: FATAL ERROR IN DATABASE.
#7910 SYSTEMERROR 7#3 @ (00018600)
#E-DS @ 00018600, 00021340.
Program Dump:
(USER1)PDUMP/OBJECT/TEST/141114/001010/07910
ON DUMPS.
#ET=2.0 PT=0.2 IO=0.6
```

Performance Impact and Best Practices

Even though the Enterprise Database Server allows you to specify the DATAENCRYPT option at the global default, global data set, physical data set, or at the item levels, note that encryption affects the overhead storage and performance of the database.

For performance consideration, it is recommended that you use the AESGCM algorithm for encryption and decryption.

Since overhead is involved when performing database encryption, encryption should only be performed on the necessary items.

When a structure contains encrypted items, the record size increases. All of the encrypted items are moved to the end of a record. The item offsets can be different from a data set with the same item descriptions, but contains no encrypted items.

If you change the DATAENCRYPT option of an item, you must perform a reorganization and recompile the user programs.

When a structure has the STRENCRYPT option set, the block size of the structure is increased according to the encryption algorithm; the encrypted data must be a multiple of 16 bytes in length and requires an additional 30 bytes for the encryption algorithm.

If the VSS2OPTIMIZE or VSS3OPTIMIZE option is set in addition to a specification of BLOCKSZ for the structure with the STRENCRYPT option set, you must adjust the block size of the structure to maintain the VSS2OPTIMIZE or VSS3OPTIMIZE setting.

If you change the STRENCRYPT option for a data set, a file format change reorganization is required. User programs do not need to be recompiled.

Always set the BACKUPKEYSET option and manually back up the key set before accessing the database. Without a backup key set, a database cannot be recovered if the key set is lost.

If you compile a new database with SLE in addition to setting INITIALIZE and ZIP, DMUTILITY is not initiated to initialize the database files because the user needs to back up the key set before accessing the database. A warning message is written to the DASDL compiler listing.

Section 24

Troubleshooting

The following information provides information when you experience Enterprise Database Server problems.

Events That Cause Halt/Load Recoveries to Fail

When a halt/load recovery fails, the Accessroutines sends diagnostic information to the printer. The following table lists the causes for an unsuccessful halt/load recovery and the task or manual recovery operation you need to perform to bring the recovery to a successful conclusion.

Cause	Remedy
The halt/load recovery process is discontinued.	Perform a manual halt/load recovery.
The system halt/loads during the halt/load recovery process.	Perform a manual halt/load recovery.
The audit file necessary to the halt/load process is corrupt or unavailable.	Retry on a duplicate audit file. If the retry fails, perform a rebuild recovery.
The RECOVERYINFO file becomes corrupted or unavailable.	Perform a manual halt/load recovery.
The ROWLOCKOUTAUDIT file becomes corrupted or unavailable.	Perform a manual halt/load recovery.

Handling I/O Errors During a Halt/Load Recovery

Like any other task, a halt/load recovery might encounter I/O errors when performing reads or writes. If I/O errors do occur, SYSTEM/DMRECOVERY locks the area or areas of the database file or files where the errors occurred.

If the REAPPLYCOMPLETED option is not set, the locked area or areas can be reconstructed using the RECOVER ROWS capability once the recovery is complete and the database has been initiated.

If the REAPPLYCOMPLETED option is set and SYSTEM/DMRECOVERY encounters I/O errors while reprocessing completed transactions, the relevant area or areas are locked as before, but the recovery process fails with a fatal error. In either case, the locked area or areas can be reconstructed using the RECOVER ROWS capability.

Note: When the REAPPLYCOMPLETED option is set, the reconstruction is unusual because it has to be initiated while the database is down. The normal recovery process or Tracker in Remote Database Backup is initiated and in turn starts the row reconstruction process. The database remains locked until both processes complete. If, ultimately, the database cannot be recovered, a full database rebuild is required.

Enterprise Database Server Errors During a Halt/Load Recovery

When a halt/load recovery updates the restart data set for each active application, occasionally the Enterprise Database Server software issues an error. Frequent errors are DUPLICATES or LIMITERROR.

The message the halt/load recovery displays at the ODT is

```
<job#> DISPLAY: RESTART DS ERR : CAT nn SUBCAT nnn
```

The category and subcategory values enable you to identify the error in a list of Enterprise Database Server exceptions and errors.

DATAENCRYPT Option Error during a DASDL Compilation

Installing an incorrect version of the Security Center database or Key Manager library may be the cause of the following message:

```
KMAPI ERR: DATABASE NOT OPEN
```

The Key Manager library, *SYSTEM/KMAPI/SUPPORT, handles the creation of database encryption keys for the Enterprise Database Server. The Key Manager library uses the Security Center database and must be upgraded at the same time that the Security Center server is upgraded.

Refer the *Security Overview and Implementation Guide* for information about the Security Center database and the Key Manager library.

REQUIRES *PK DISK Error during a Reorganization

Running a reorganization without INTERNAL FILES specified, or with INTERNAL FILES=DISK specified on a system which does not have a pack called DISK, results in the following message:

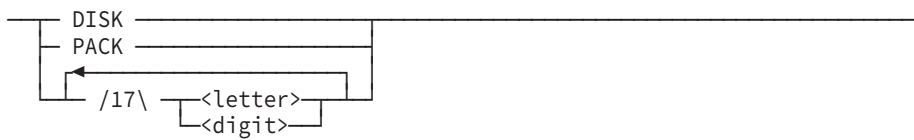
```
REQUIRES *PK DISK
```

To resolve this error, include an INTERNAL FILES specification in the BUILDREORG specs.

<family name>

The <family name> identifies a disk or pack family.

Syntax



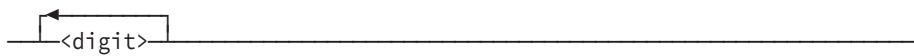
Examples

DISK
PACK
DBPACK
SYSPACK

<digit>

One of the decimal digits from 0 through 9.

Syntax



<integer>

The <integer> construct is used to represent an unsigned whole value.

<dump name>

The <dump name> construct identifies a particular dump on a multidump backup tape produced by DMUTILITY.

Syntax



Explanation

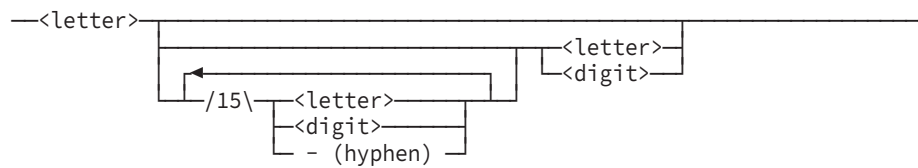
The <dump name> construct is an identifier or series of identifiers that represent an Enterprise Database Server file name. It is used in conjunction with multidump tapes.

For example, if MYDB960501 is specified as a dump name, the dump is created with the dump name MYDB960501 on a tape with the name <tape name>. When this construct is used, <tape name> must be a single node.

<identifier>

The <identifier> constructs have no intrinsic meaning. They are used to represent symbolic names of structures within an Enterprise Database Server database.

Syntax



Explanation

An identifier is composed of from 1 to 17 letters, digits, and hyphens. The first character must be a letter. The last character must not be a hyphen.

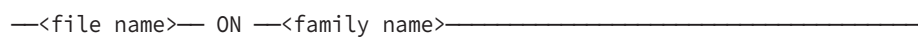
Examples

- A
- EMPLOYEE
- ACCOUNTS-PAYABLE
- B-1

<file title>

The <file title> construct consists of a <file name> and a <family name>.

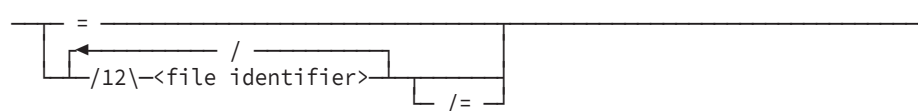
Syntax



<file name>

The <file name> construct is an identifier, or series of identifiers separated by a slash (/), which represents the name of a ClearPath MCP file.

Syntax



Explanation

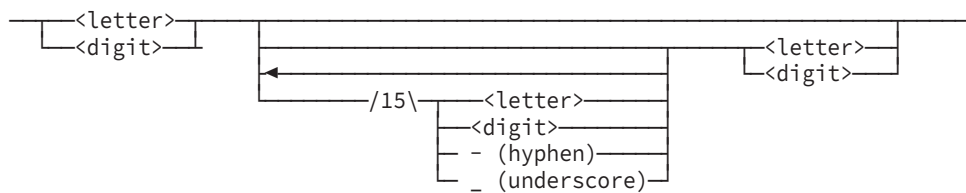
The slash equal sign (/=) can be used to represent a family of files. The equal sign (=) alone designates all files in the database.

<file identifier>

The <file identifier> construct is a single node of a ClearPath MCP file name.

Common Syntactic Items

Syntax



Explanation

A file identifier is composed of 1 to 17 letters, digits, hyphens, or underscores. The first and last character must be a letter or digit.

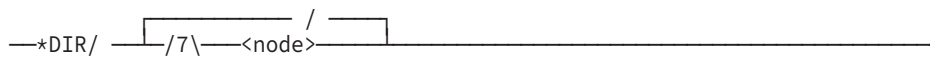
Examples

```
A5
EMPLOYEE
ACCOUNTS-PAYABLE
B_1
```

<path name>

A path name identifies the location of a permanent directory within the file system. A path name cannot exceed 7 nodes and it must start with the *DIR node to identify the permanent directory.

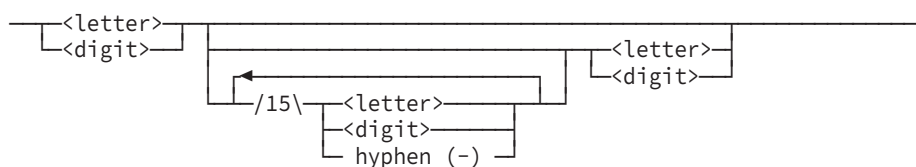
Syntax



<tape name>

The <tape name> construct identifies a dump tape produced by DMUTILITY.

Syntax



Explanation

If a multinode tape name is specified, DMUTILITY accepts the full tape name. However, only the first node and the last node are retained by the system. When a multinode tape name is specified for a single dump tape, DMUTILITY accepts the full tape name. However, only the first node and the last node are retained by the system.

For multidump tape specifications, the tape name must consist of a single node. The reason for this requirement is that files on tape can have at most, two nodes. The first node is the tape name; the second node is the file name. When you use multidump tapes, you can explicitly specify the tape name by using the "TAPE =" syntax and specify the dump node name with the <name> variable that follows the TO or FROM syntax component.

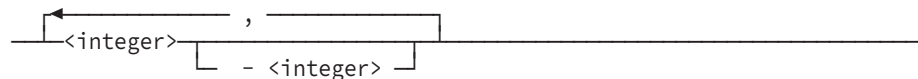
For example, if MYDB/960501/DUMP1 is specified as a dump tape name, the tape is created with the tape name MYDB/DUMP1. You can verify the tape name by initiating a PER MT command from the ODT.

If you require unique tape names in your processes, be sure to make the combination of the first and last node unique. Use the SERIALNO option to uniquely identify tapes.

<range>

The <range> construct identifies specific family indexes and rows to be on by DMUTILITY.

Syntax

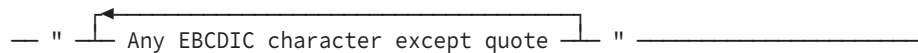


Explanation

A contiguous group of family indexes or rows can be declared by specifying two integers separated by a hyphen. If a contiguous group is specified, the second integer must be larger than the first.

<string>

Syntax



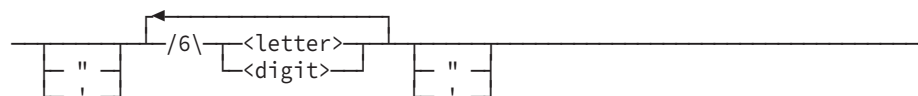
Explanation

Strings must be contained within double quotation marks (" "). A string can contain, at most, 255 characters.

<string6>

The <string6> is used in DMUTILITY to designate dump tape serial number.

Syntax



Explanation

The <string6> construct is a group of between one and six letters and digits, optionally enclosed within single (' ') or double quotation (" ") marks.

When DMUTILITY is initiated by CANDE, double quotation marks cannot be used to delimit tape serial numbers.

Appendix B

Interpreting Database Statistics

The STATISTICS option of the Data and Structure Definition Language (DASDL) gathers and prints statistical information about the efficiency of the database. This information includes statistics on input/output (I/O), buffer use, database use, audit files, and transactions.

The Visible DBS *STATISTICS RESTART* command initializes all statistics information, and clears the timers and totals. When the last user closes the database, the statistics are printed if the DASDL option list of the database includes the STATISTICS option. You can also print the statistics at any time by using the Visible DBS *STATISTICS* command.

The Visible DBS *STATISTICS* command is described in [Section 12, Communicating with the Database](#). The STATISTICS option is described in the *Data and Structure Definition Language (DASDL) Programming Reference Manual*.

The following paragraphs describe the parts of a statistics report in the order that they appear in the report. For example, the header appears at the beginning of the report and the transactions statistics appear at the end.

These paragraphs also describe some general guidelines on using statistics. These guidelines are not specific because each database environment is unique. Consequently, recommendations cannot fit all situations.

Note: In addition to using the Visible DBS commands described in this appendix, you can use the Visible Recovery *STATUS*, *STATISTICS*, and *STATISTICS CLEAR* commands to monitor the status of database recovery operations. The Visible Recovery commands are described in [Section 8, Recovering the Database](#).

Header

The header, as shown in the following example, describes the dates and times that the statistics were gathered.

```
Unisys Enterprise Database Server for ClearPath MCP SYSTEM/DMUTILITY
      <release number> <cycle number>
      <day of week>, <month>, <day>, <year> <time>
SYSTEM: <name> SERIAL NUMBER: <number>
```

DATABASE	DATE	TIME
OPENED	3/31/06	9:15:19
CLOSED	3/31/06	10:05:17
INTERVAL		49:58

Interpreting Database Statistics

```
STATISTICS          DATE          TIME
  STARTED          3/31/06          9:15:19
    ENDED          3/31/06          10:05:17
  INTERVAL                          49:58
```

The header contains the information described in the following table.

Field Name	Description
Header	Identifies the machine type, version number of the Enterprise Database Server release, and the system serial number.
Database date and time	Identifies the date and time when the database was first opened, when it was closed by the last user, and the interval between both times. The text (** DATA BASE STILL OPEN **) in this area shows that the statistics report was printed with the Visible DBS <i>STATISTICS</i> command. For more information on this command, see Section 12, "Communicating with the Database," of this guide.
Statistics date and time	Identifies the date and time during which the statistics apply. By default, statistics are generated from the time a user first opens the database until the last user successfully closes the database. If a halt/load abnormally closes the database, statistics are not generated.

To reset the time, use the Visible DBS command `STATISTICS RESTART`. For more information on this command, see [Section 12, Communicating with the Database](#), of this guide.

Buffer Statistics

The buffer statistics provide information about the `ALLOWEDCORE` and buffer specifications. The following example shows these statistics.

```
BUFFER STATISTICS

ALLOWEDCORE                100000
MAXIMUM BUFFER STORAGE USED 101924
NUMBER OF BUFFERS USED      568
OVERLAY GOAL                1.000
OVERLAY RATE                0.000
NUMBER OF FORCED OVERLAYS   516
NUMBER OF NORMAL OVERLAYS  1477
MEMORY RESIDENT LIMIT       100000
MAXIMUM MEMORY RESIDENT USED 0
```

The buffer statistics are described in the following table.

Field Name	Description
ALLOWEDCORE	<p>Indicates the number of words of main memory that the Accessroutines is allowed to use for database buffers. When buffer space exceeds the ALLOWEDCORE value, the Accessroutines uses an overlay algorithm to write modified buffers back to disk and return the space until the amount of memory used by the buffers is less than the ALLOWEDCORE value. To control this value, use the DASDL parameter ALLOWEDCORE or the ALLOWEDCORE option in the Visible DBS command.</p> <p>The ALLOWEDCORE parameter is described in the <i>Data and Structure Definition Language (DASDL) Programming Reference Manual</i>. The Visible DBS command is described in Section 12, Communicating with the Database, of this guide.</p>
Maximum Buffer Storage Used	Identifies the maximum number of words of main memory that Accessroutines has used since the database opened or the since the STATISTIC option was reset for database buffers and data control blocks. This value can exceed the ALLOWEDCORE value.
Number of Buffers Used	Identifies the total number of buffers used for all structures.
Overlay Goal	<p>Controls the rate at which buffers are overlaid to the disk. The unsigned integer value for OVERLAYGOAL must be a number or decimal value in the range 0 through 100. For databases using the XE features, the default value is 1 percent of the allowed core for each minute and applies only if the OVERLAYGOAL parameter is not specified. Otherwise, the default value is 5 percent of the allowed core for each minute.</p> <p>OVERLAYGOAL is a dynamic database parameter that can be changed by a DASDL update without recompilation of the Enterprise Database Server software.</p> <p>If the OVERLAYGOAL parameter is not 0 (zero), the Enterprise Database Server dynamically allocates buffers to a structure as long as the total core usage does not exceed the ALLOWEDCORE value. When OVERLAYGOAL is not zero, the BUFFERS specifications for structures are ignored. To manually set the number of buffers for each structure, set the OVERLAYGOAL parameter to 0 (zero). Next, designate the number of buffers by using either the DEFAULTS option or the attributes of the individual structures.</p> <p>The value for the OVERLAYGOAL option can also be changed by using a Visible DBS command. For instructions on using Visible DBS commands, refer to Section 11, Checking Integrity and Performance.</p>
Overlay Rate	Identifies the current, dynamic run-time rate of overlaid memory expressed as a percentage of the ALLOWEDCORE value per minute. If the statistics are printed at the final closing of the database, all the dynamic variables used to calculate the overlay rate have been cleared, so the overlay rate is 0. To get the current overlay rate in the database statistics, use the Visible DBS command SM STATISTICS RESTART.

Interpreting Database Statistics

Field Name	Description
Number of Forced Overlays	<p>Identifies the number of buffers flushed when the buffer space exceeds the ALLOWEDCORE value and the number of all buffers for all users is less than the buffer specifications in the DASDL source. When the OVERLAYGOAL parameter is used, it controls the rate at which these buffers are overlaid.</p> <p>A high forced overlay rate can severely slow down the system. When the Maximum Buffer Storage Used value exceeds the ALLOWEDCORE value, the system uses forced overlays to reduce the number of buffers stored until the amount is less than the ALLOWEDCORE value.</p> <p>Perform the following steps to reduce the number of forced overlays:</p> <ol style="list-style-type: none">1. Increase the ALLOWEDCORE value.2. Decrease the buffer specifications.3. Decrease the buffer sizes.4. Decrease the audit block size. <p>Try to bring the number of forced overlays as close to 0 (zero) as possible—for example, 1 or 2 per minute.</p>
Number of Normal Overlays	<p>Indicates the number of buffers flushed when the database structures close normally. This value is also the number caused by user state changes such as NORMAL, READAHEAD, and REBLOCK.</p> <p>This value should be as close to 0 (zero) as possible—for example, 1 or 2 per minute.</p>
Memory Resident Limit	<p>Limits the amount of memory used for MEMORY RESIDENT buffers. When the RESIDENT LIMIT parameter is exceeded, no more MEMORY RESIDENT buffers can be added. The value of the RESIDENT LIMIT parameter cannot exceed the value of the ALLOWEDCORE parameter. The default value is one half the value of the ALLOWEDCORE parameter.</p>
Maximum Memory Resident Used	<p>Identifies the maximum number of words of resident memory that Accessroutines has used since the database opened. This value can exceed the Memory Resident Limit value.</p>

For more information on these values and specifications, see the *Data and Structure Definition Language (DASDL) Programming Reference Manual* or the explanation of the Visible DBS command in [Section 12, Communicating with the Database](#).

Input/Output (I/O) Statistics

Input/output (I/O) statistics contain information about user access to a database—that is, about the process of writing data to and reading data from a database. With these statistics and the transaction statistics, you can determine whether

- Too many read and write operations are taking place.
- Too much time is spent waiting for channel and control resources.
- Too much time is spent waiting for data transfer, which suggests that the blocks are too large.

Each line of the I/O statistics describes the database structures that were accessed. The I/O statistics also include Access statistics because an Access is part of a data set. The following example shows these statistics.

Interpreting Database Statistics

INPUT/OUTPUT STATISTICS										
(1) I/O TIME (SECONDS) (2) NUMBER OF READS (NORMAL PLUS READ AHEAD) (3) NUMBER OF READ AHEADS (4) WAIT TIME FOR READS (SECONDS) (5) AVERAGE WAIT TIME FOR READS (MILLISECONDS) (6) NUMBER OF WRITES (NORMAL PLUS WRITE AHEAD) (7) NUMBER OF WRITE AHEADS (8) WAIT TIME FOR WRITES (SECONDS) (9) AVERAGE WAIT TIME FOR WRITES (MILLISECONDS)										
		READS					WRITES			
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
1	ORGDB	0.0	0	0	0.0	0.0	0	0	0.0	0.0
2	RST	0.1	1	0	0.0	44.3	1	0	0.0	29.2
3	PERSON	68.9	111593	0	24.1	0.2	1887	121	67.8	35.9
4	PERSON-SET	53.2	2367	0	0.3	0.1	2497	51	57.4	23.0
5	EMPLOYEE	51.2	2445	0	0.7	0.3	2402	44	54.5	22.7
6	PREVIOUS-EMP	0.6	7	0	0.1	19.4	19	0	0.7	35.1
7	MANAGER	0.3	21	0	0.1	5.5	7	0	0.2	29.3
8	PROJECT-EMP	43.3	2738	0	0.4	0.1	2065	41	45.7	22.1
9	EMP-MGR	38.9	1826	0	0.4	0.2	1972	40	40.8	20.7
10	FAMILY	0.1	1	0	0.0	31.4	1	0	0.0	35.4
11	FAMILY-SET	0.1	1	0	0.0	40.4	1	0	0.0	33.6
12	EDUCATION	0.1	1	0	0.0	36.5	1	0	0.0	37.4
13	EDUCATION-SET	0.1	1	0	0.0	43.9	1	0	0.0	29.5
14	INTERIM-MANAGER	0.2	2	0	0.1	42.5	2	0	0.1	41.3
15	INTERIM-HISTORY	0.2	2	0	0.1	50.4	2	0	0.1	29.1
16	INTERIM-SET	0.2	3	0	0.1	37.3	2	0	0.1	41.4
17	PROJECT	23.5	23945	0	0.4	0.0	997	0	28.3	28.4
18	PROJECT-SET	26.2	17196	0	0.2	0.0	926	3	27.0	29.2
19	SUPER-PROJECTS	32.3	1133	0	0.5	0.5	1253	30	32.2	25.7
20	DEPARTMENT	0.4	12	0	0.1	10.6	6	0	0.2	37.4
21	DEPARTMENT-SET	0.3	21	0	0.1	4.3	5	0	0.1	28.0
22	ASSIGNMENT	15.9	220	0	1.4	6.5	559	37	16.7	29.9
23	ASSIGNMENT-SET	7.6	19	0	0.4	21.1	280	46	4.4	15.6
24	STAFF-ASSIGNED	59.9	3283	0	3.0	0.9	2732	133	59.6	21.8
25	PROJASSIGN	78.4	3167	0	3.8	1.2	3553	198	76.8	21.6
26	PROJECT-PERSON	58.2	40338	0	2.3	0.1	2517	24	61.0	24.2
27	PROJPERSON-SET	97.5	5946	0	2.4	0.4	3584	123	106.7	29.8
28	PROJECT-TEAM	89.3	3344	0	3.7	1.1	3563	185	84.6	23.7
	TOTAL	746.7	219633	0	45.0	0.2	30835	1076	764.8	24.8

008390

Each line of the I/O statistics begins with a structure number and a name. Nine fields of information are printed for each structure. These nine fields are described in the following table.

Field Name	Description
I/O Time	Indicates the time spent by the MCP, in seconds, writing to and reading from the structure. This machine-dependent value excludes the time necessary to search for and transfer data, and excludes the delay between those times.
Number of Reads	Indicates the number of times the structure was physically read. This value includes readahead operations
Number of Readaheds	<p>Indicates the number of read operations that occur during readaheds. This value includes reads of small blocks and large blocks (reblock), normal block reads, and storage control reads.</p> <p>You activate readahead operations when you designate serial buffers in your DASDL source and when the Accessroutines detects sequential physical access and sequential access through the set. The Accessroutines reads a structure in blocks.</p> <p>The readahead facility allows the Accessroutines to read blocks before they are needed. This reduces the time required to access records serially by decreasing the buffer wait time. For data sets, this means that the next physical block of data is retrieved. When serially accessing records through a set or subset, the following records are retrieved and placed in the data buffers of the data set:</p> <ul style="list-style-type: none"> • The next set or subset • The data set records pointed to by entries in the set or subset table <p>Reblocking causes the Accessroutines to use a larger block size when reading data blocks. Using a larger block size increases the amount of serially accessed records being put into the buffers for each physical read. Reblocking is not valid for index sequential structures. Both optimization methods cause a decrease in elapsed time for data retrieval.</p> <p>When the DASDL REBLOCK option is turned on and the REBLOCKFACTOR is set to a value greater than 1, readahead logic is turned off for data set accessing. Since readahead logic only reads one block ahead at a time, setting the REBLOCKFACTOR option to a value of 2 or higher means that more information can be read in one physical I/O operation than if readahead logic is used to read smaller blocks of information. You cannot use readahead logic to read large blocks. When the REBLOCKFACTOR option is set to a value greater than 1 for a data set, readahead logic is still used for accessing associated sets serially.</p>
Wait Time for Reads	Indicates the time, in seconds, that the program spent waiting for read operations to finish. This value does not include housekeeping while waiting to read. The Wait Time for Reads value is a key indicator of the structures that need tuning

Interpreting Database Statistics

Field Name	Description
Average Wait Time for Reads	<p>Indicates the Wait Time for Reads value divided by the Number of Reads value, in milliseconds. Normally, this value should range from 10 to 20. For heavily loaded systems, the value can range from 20 to 30. Values over 30 can be excessive.</p> <p>To reduce the waiting time for read operations and to increase overall performance, perform the following steps:</p> <ul style="list-style-type: none"> • Add more channel capacity. • Add more I/O resources. • Add more packs. • Distribute the database among different packs. <p>You might also need to use System Management Facility II (SMFII) to determine if high values are the result of inefficient processor use or too many reads or writes. See the <i>System Management Facility II (SMFII) Query Operations Guide</i> for more information.</p>
Number of Writes	<p>Indicates the number of times the structure was physically written, including normal write operations, writeahead operations, and storage control writes</p>
Number of Writeaheads	<p>Indicates the number of writeahead operations. The system writes the oldest buffer while waiting for a read operation to finish. The system writes the serial buffer depending on the audit specifications, as soon as it accesses the next buffer</p>
Wait Time for Writes	<p>Indicates the time, in seconds, that the program spent waiting for write operations. The time measurement begins after all possible housekeeping is finished and ends when I/O is finished</p>
Average Wait Time for Writes	<p>Indicates the Wait Time for Writes value divided by the Number of Writes value, in milliseconds. The value should range from 10 to 30. Values over 40 milliseconds are too high and can be caused by</p> <ul style="list-style-type: none"> • Too many buffers • Storage control writes • Forced overlays • Heavy sequential update <p>To reduce the waiting time for write operations, perform the following steps:</p> <ul style="list-style-type: none"> • Decrease the number of buffers. • Defer physical addition and deletion. • Increase the ALLOWEDCORE value. • Increase the number of writeahead operations.

VSS2 Optimization

VSS2 optimization lists the database structures that are not optimized for VSS-2 devices. Structures that are not optimized for VSS-2 devices are not aligned on block boundaries that are multiples of 60 words.

There is one line for each structure that is not optimized. The first field is the structure number. The second field is the structure name. The total number of structures that are not optimized is at the end of the list.

VSS3 Optimization

The following information is valid only if VSS3OPTIMIZE is specified in the database.

VSS3 optimization lists the database structures that are not optimized for VSS-3 devices. Structures that are not optimized for VSS-3 devices are not aligned on block boundaries that are multiples of 660 words.

There is one line for each structure that is not optimized. The first field is the structure number. The second field is the structure name. The total number of structures that are not optimized is at the end of the list.

Note that only structures that are opened during the report time frame are listed.

Control file and audit file are also listed after the structure list if they are not optimized for VSS3 blocking.

Database Usage Statistics

The database usage statistics describe activity on the database. Each line of these statistics describes the number and name of the accessed structure. Five fields of data are listed for each structure. All fields include both successful and unsuccessful operations. Because an Access is part of a data set, the Access statistics are included with the database usage statistics. The following example shows the database usage statistics.

DATABASE USAGE STATISTICS

	(1)	(2)	(3)	(4)	(5)	(6)
(1) FIND DATA RECORD/FIND ENTRY IN SET						
(2) STORE AFTER CREATE/INSERT KEY INTO SET						
(3) STORE AFTER LOCK/CHANGE DATA IN KEY						
(4) DELETE DATA RECORD/DELETE ENTRY FROM SET						
(5) CHANGE A CONTROL FIELD OF DATA RECORD						
(6) USER FIND DATA RECORD						
1 ORDERSDB	0	0	0	0	7	0
2 RSTART-INFO	0	0	0	0	0	0
3 RS-SET	4	0	0	0		

Interpreting Database Statistics

```

.
.
24 ORDERS          49    0    0    0    0    0
25 ORD-SET        56    0    0    0
26 ORD-STATUS-SET  0    0    0    0
27 ORD-DATE-SET   0    0    0    0
.
.
49 MAJOR-ASSEMBLY 3296 11111 3290    0    0    0
50 MASMBLY-SET    3297 11111    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
50 MASMBLY-SET     0    0    0    0
SUBTOTAL          3297 11111    0    0
.
.
56 PART-DESC      2169    0    0 1084    0 1084
56 PART-DESC      2169    0    0 1084    0 1084
56 PART-DESC      2169    0    0 1084    0 1084
SUBTOTAL          6507    0    0 3252    0 3252
57 PDESC-SET      3255    0    0 3252
57 PDESC-SET       0    0    0    0
57 PDESC-SET       0    0    0    0
57 PDESC-SET       0    0    0    0
57 PDESC-SET       0    0    0    0
57 PDESC-SET       0    0    0    0
57 PDESC-SET       0    0    0    0
57 PDESC-SET       0    0    0    0
57 PDESC-SET       0    0    0    0
SUBTOTAL          3255    0    0 3252

```

Field Name	Description
Find Data Record/Find Entry in Set	Indicates for a data set, the number of FIND operations on a record. For a data set, the number of FIND operations can include internal FIND operations of deleted records. For a set, the number of logical FIND operations using a set and a FIND KEY OF clause. This value does not include internal FIND operations such as those required to insert a record into a set
Store After Create/Insert Key into Set	Indicates for a data set, the number of records stored after the records were created. For a set, the number of keys inserted into a set
Store After Lock/Change Data in Key	Indicates for a data set, the number of times that records were stored after the records were locked. For a set, the number of times data in a key was changed

Field Name	Description
Delete Data Record/Delete Entry from Set	Indicates for a data set, the number of records deleted. For a set, the number of entries deleted
Change a Control Field of Data Record	Indicates the number of times a control field, such as POPULATION or COUNT, was changed
User Find Data Record	Indicates the number of times application programs executed statements resulting in a FIND operation being performed for a data set. It does not include internal FINDs performed by the ACCESSROUTINES such as those that occur when deleted records are encountered as the result of processing records serially through the data set.

Database usage statistics are most valuable when compared with data in I/O statistics. You can detect linear search problems by comparing the ratio of logical reads to the ratio of physical reads.

Structure Lock Statistics

The structure lock statistics provide locking information on various locks of structures. These statistics are printed only when the LOCKSTATISTICS option is turned on. The following example shows the structure lock statistics.

STRUCTURE LOCK STATISTICS

```

(1) NUMBER OF TIMES LOCK HELD
(2) TOTAL LOCK HOLD TIME (SECONDS)
(3) NUMBER OF LOCK WAITS
(4) TOTAL LOCK WAIT TIME (SECONDS)
(5) AVERAGE LOCK HOLD TIME (MS)
(6) AVERAGE LOCK WAIT TIME (MS)

```

	(1)	(2)	(3)	(4)	(5)	(6)
2 AUDITAREA						
BLOCKLOCK	11	0.0	0	0.0	3.915	0.000
STORELOCK	3	0.0	0	0.0	11.229	0.000
3 GLB-CRITIC						
BLOCKLOCK	3414111	46.0	0	0.0	0.013	0.000
STORELOCK	2	0.0	0	0.0	10.908	0.000
RSNLOCK	1	0.0	0	0.0	0.000	0.000
4 GLB-PATH						
BLOCKLOCK	3	0.0	0	0.0	3.946	0.000
STORELOCK	2	0.0	0	0.0	5.922	0.000
PATHLOCK	1	0.0	0	0.0	0.000	0.000
5 GLB-PATH-2						
BLOCKLOCK	3	0.0	0	0.0	7.161	0.000
STORELOCK	2	0.0	0	0.0	11.717	0.000
PATHLOCK	1	0.0	0	0.0	0.000	0.000

Interpreting Database Statistics

The structure lock statistics consists of six fields which are described in the following table.

Field Name	Description
Number of Times Lock Held	Indicates the number of times the lock was held.
Total Lock Hold Time	Indicates the total hold time, in seconds, on the lock.
Number of Lock Waits	Indicates the number of times the lock was waited for.
Total Lock Wait Time	Indicates the total wait time, in seconds, on the lock.
Average Lock Hold Time	Indicates the Total Lock Hold Time value divided by the Number of Times Lock Held value, in milliseconds.
Average Lock Wait Time	Indicates the Total Lock Wait Time value divided by the Number of Lock Waits value, in milliseconds.

Audit Statistics (First Part)

Audit statistics provide information on audit file performance. These statistics are printed only for audited databases. The following example shows the first part of the statistics.

```
AUDIT STATISTICS

STARTING   AUDIT FILE NUMBER           1
ENDING     AUDIT FILE NUMBER           1
STARTING   AUDIT BLOCK SERIAL NUMBER   18000
ENDING     AUDIT BLOCK SERIAL NUMBER   99851
AVERAGE   AUDIT BLOCK SIZE (WORDS)    52 OF 900
                                                008397
```

The first part of the audit statistics consists of five fields, which are described in the following table.

Field Name	Description
Starting Audit File Number	Indicates the number of the audit file when statistics were started
Ending Audit File Number	Indicates the number of the audit file when statistics were printed
Starting Audit Block Serial Number	Indicates the serial number of the audit file when statistics were started

Field Name	Description
Ending Audit Block Serial Number	Indicates the serial number of the audit file when statistics were printed
Average Audit Block Size	Indicator of excessively large audit block sizes or syncpoint values. An Average Audit Block Size value that is less than the physical audit BLOCKSIZE value declared in the DASDL source might indicate a low syncpoint value

Audit Statistics (Second Part)

The second part of the audit statistics consists of the fields shown in the following example, which describe audit wait times.

	TOTAL WAIT TIME (SEC)			AVERAGE WAIT TIME (MS)		AVERAGE BLOCK SIZE
	COUNT	PRIMARY	SECONDARY	PRIMARY	SECONDARY	
FORCE AUDIT AT SYNC POINT TIME	1444	20.1	0.0	13.9	0.0	836
FORCE AUDIT TO UNCONSTRAIN BUFFERS	8	4.9	0.0	607.2	0.0	885
FORCE AUDIT FOR REAPPLYCOMPLETED	0	0.0	0.0	0.0	0.0	0
FORCE AUDIT FOR NEWTOP BUFFER	0	658.9	0.0	0.0	0.0	0
FORCE AUDIT FOR CATCHUP SERVER	0	0.0	0.0	0.0	0.0	0
FORCE AUDIT FOR FILE SWITCH	1	26.5	0.0	26537.2	0.0	30
FORCE AUDIT FOR REORG ETR	0	0.0	0.0	0.0	0.0	0
FORCE AUDIT FOR WRITE AUDIT BLOCK	0	0.0	0.0	0.0	0.0	0
FORCE AUDIT (OTHER)	75	2.2	0.0	29.3	0.0	458
NORMAL AUDIT I/O	17313	0.0	0.0	0.0	0.0	1731
TOTALS	18841	712.6	0.0	37.8	0.0	3940

008398

These audit statistics describe audit wait times and counts for the fields shown in the following table.

Field Name	Description
Force Audit at Syncpoint Time	Indicates the time in milliseconds that the program spent waiting for the audit buffers to flush during syncpoints. Values over 80 milliseconds might be excessive depending on the system configuration.
Force Audit to Unconstrain Buffers	When reusing or overlaying a data buffer, those audit buffers recording the updates of that data buffer have to be flushed first. This field indicates the time in milliseconds that the program spent waiting for those audit buffers to flush. Values over 80 milliseconds might be excessive depending on the system configuration.

Interpreting Database Statistics

Field Name	Description
Force Audit for REAPPLYCOMPLETED	Indicates the time in milliseconds that the program spent waiting for the audit buffers to flush at the end of the transaction. Values over 80 milliseconds might be excessive depending on the system configuration.
Force Audit for New Top Buffer	Indicates the time in milliseconds that the program spent waiting for the top audit buffers to flush because all the buffers were full
Force Audit for CatchupServer	Indicates the time in milliseconds that the Remote Database Backup Catchup-server task spent waiting for the audit buffers to flush at the end of a Catchup process. Values over 80 milliseconds might be excessive depending on the system configuration.
Force Audit for FILE SWITCH	Indicates the time in milliseconds that the program spent waiting for the audit buffers to flush during the audit switch
Force Audit for REORG ETR	Indicates the time in milliseconds that reorganization tasks spent waiting for the audit buffers to flush at the end of the transaction. Values over 80 milliseconds might be excessive depending on the system configuration.
Force Audit for WRITE AUDIT BLOCK	Indicates the time in milliseconds that the program spent waiting for the audit buffers to flush at the completion of an Enterprise Database Server statement such as CREATE, STORE, or DELETE
Force Audit (Other)	Indicates the time in milliseconds that the program spent waiting for the audit buffers to flush at other miscellaneous situations.
Normal Audit I/O	Indicates the time in milliseconds that the program spent waiting for the audit buffers to flush when the audit buffers were full. The values for total waiting times should range from 1 to 5 milliseconds. The values for average waiting time should range from 3 to 8 milliseconds for a tape and about 15 milliseconds for a pack.

High audit wait times are caused by

- Small audit block sizes
- Updates to large data set records
- Too many syncpoints
- Heavy overlay (unconstrained buffers)
- Audits to a busy pack

An audit block size that is too small or logical audit records that are too large can cause an audit block to fill before the I/O operation finishes for another audit block. To prevent audit blocks from filling too quickly, increase the audit BLOCKSIZE value.

The audit BLOCKSIZE value you designate in the DASDL source determines the block size. Shorter blocks can be written if needed.

Transaction Statistics

Transaction statistics provide statistics about transactions, syncpoints, and controlpoints. The following example shows these statistics.

TRANSACTION STATISTICS	
TOTAL TRANSACTION COUNT	80877
TOTAL SYNCPOINT COUNT	404
TOTAL CONTROLPOINT (CP) COUNT	404
WAIT FOR SYNCPOINT/CONTROLPOINT COUNT	0
AVERAGE WAIT FOR SYNCPOINT/CONTROLPOINT	0.000 SECONDS
AVERAGE TIME TO TAKE CONTROLPOINT	0.252 SECONDS
AVERAGE NUMBER OF BUFFERS FLUSHED AT CP	3.8
PERCENT OF MODIFIED BUFFERS FLUSHED AT CP	13.4 %

008396

Transaction statistics consist of the information in the following table.

Field Name	Description
Total Transaction Count	Indicates the number of successful BEGINTRANSACTION statements
Total Syncpoint Count	Indicates the number of syncpoint operations
Total Controlpoint Count	Indicates the number of controlpoint operations. A syncpoint frequency of 100 and a controlpoint frequency of 2 produces two syncpoints and one controlpoint for every 200 transactions
Wait for Syncpoint/Controlpoint Count	Indicates the number of times the programs waited at a BEGINTRANSACTION statement for syncpoint and controlpoint operations
Average Wait for Syncpoint/Control	Indicates the average time the program spent waiting at a BEGINTRANSACTION statement for a syncpoint or a controlpoint operation. This value should range from 1 to 2 seconds. The value is biased by the controlpoint overhead. To reduce this value, perform one or all of the following steps: <ul style="list-style-type: none"> • Increase the syncpoint or controlpoint values. • Reduce the controlpoint overhead. • Reduce the time the program is in transaction state
Average Time to Take Controlpoint	Indicates the average time spent taking a controlpoint. This value should be as close to the number 0 as possible. High values can be caused by too many buffers.

Interpreting Database Statistics

Field Name	Description
Average Number of Buffers Flushed at Controlpoint	Indicates the average number of buffers that are flushed at controlpoints
Percent of Modified Buffers Flushed at Controlpoint	Indicates the overhead for data buffer flushing (not including storage control writes). This value is typically 10 to 20 percent of the modified buffers that are flushed and, in the worst cases, can go as high as 50 percent. Too many buffers cause higher values.

Global Lock Statistics

The global lock statistics provide locking information on various locks in the database. These statistics are printed only when the LOCKSTATISTICS option is turned on. The global lock statistics consists of six fields. the following example shows these statistics:

GLOBAL LOCK STATISTICS

- (1) NUMBER OF TIMES LOCK HELD
- (2) TOTAL LOCK HOLD TIME (SECONDS)
- (3) NUMBER OF LOCK WAITS
- (4) TOTAL LOCK WAIT TIME (SECONDS)
- (5) AVERAGE LOCK HOLD TIME (MS)
- (6) AVERAGE LOCK WAIT TIME (MS)

	(1)	(2)	(3)	(4)	(5)	(6)
DCBALLOC	212	0.0	0	0.0	0.001	0.000
OLAYLOCK	1	0.0	0	0.0	0.000	0.000
MEMLOCK	109	0.0	0	0.0	0.001	0.000
DBLOCK	2	0.0	0	0.0	0.000	0.000
GBLOCK	5	0.0	0	0.0	0.679	0.000
DBOPENLOCK	3	0.0	0	0.0	12.164	0.000
CONTROLLOCK	4	0.0	0	0.0	3.565	0.000
VDBSLOCK	1	0.0	0	0.0	0.000	0.000
CNTLIOLOCK	3	0.0	0	0.0	4.237	0.000
POPZIPLOCK	1	0.0	0	0.0	0.000	0.000
TRANSTAMPLOCK	8	0.0	0	0.0	0.008	0.000
GCLOCK	1	0.0	0	0.0	0.000	0.000
VSS2WARNLOCK	1	0.0	0	0.0	0.000	0.000
RSLOCK	3	0.0	0	0.0	0.001	0.000
SPACE_LOCK	11	0.0	0	0.0	0.004	0.000
CPT_LOCK	1	0.0	0	0.0	0.002	0.000
AUDITLOCK	18	0.0	0	0.0	1.501	0.000
SYNCL	9	0.0	0	0.0	0.002	0.000
PTIMELOCK	5	0.0	0	0.0	0.004	0.000
AUDITIOLOCK	9	0.0	0	0.0	2.968	0.000
AUDITQUEUELOCK	18	0.0	0	0.0	1.382	0.000
AUDITCLOSELOCK	3	0.0	0	0.0	8.272	0.000
AUDITWRITELOCK	8	0.0	0	0.0	3.146	0.000
AUDITACCESSLOCK	25	0.0	0	0.0	0.003	0.000
AUDITWAITLOCK	11	0.0	0	0.0	0.002	0.000

The global lock statistics consists of six fields which are described in the following table.

Field Name	Description
Number of Times Lock Held	Indicates the number of times the lock was held.
Total Lock Hold Time	Indicates the total hold time, in seconds, on the lock.
Number of Lock Waits	Indicates the number of times the lock was waited for.
Total Lock Wait Time	Indicates the total wait time, in seconds, on the lock.
Average Lock Hold Time	Indicates the Total Lock Hold Time value divided by the Number of Times Lock Held value, in milliseconds.
Average Lock Wait Time	Indicates the Total Lock Wait Time value divided by the Number of Lock Waits value, in milliseconds.

Control Point Buffer Statistics

The Control Point Buffer Statistics provide buffer information between and at control points for every structure.

Statistics are printed only when the CPSTATS option is turned on. The statistics consists of five fields.

The following example shows these statistics.

```

CONTROLPOINT COUNT                                14

(1) NUMBER OF WRITEAHEADS
(2) NUMBER OF AUDIT CONSTRAINED BUFFERS
(3) NUMBER OF I/O PENDING BUFFERS
(4) NUMBER OF CP FLUSHED BUFFERS
(5) NUMBER OF CP MODIFIED BUFFERS

          (1)    (2)    (3)    (4)    (5)
2 RESTART-DS      0      0      0      0      0
3 DS             4286   317   548    1    464
4 DS-XE/0        266    0    185    1    301
4 DS-XE/1        261    0    201    1    302
4 DS-XE/2        252    1    145    1    266
   SUBTOTAL      779    1    531    3    869
5 DS-SET         39     0    10     1     22
6 DS-XE-SET     103    116    1     1     52
    
```

The control point buffer statistics consists of five fields which are described in the following table.

Interpreting Database Statistics

Field Name	Description
Number of writeaheads	Indicates the number of writeaheads completed between control points.
Number of audit constrained buffers	Indicates the number of buffers the writeahead process skipped between control points due to audit constraints.
Number of I/O pending buffers	Indicates the number of buffers the writeahead process skipped between control points due to I/O pending.
Number of CP flushed buffers	Indicates the number of buffers flushed at the control point.
Number of CP modified buffers	Indicates the number of buffers modified (and not flushed) at the control point.

Using Statistics

Some general guidelines to using the statistics include the following:

- Begin with a test version of a database. Run your programs on this version, and generate statistics to discover the effect of the programs on the database. When you are satisfied with the results, use a production version of the database and generate statistics on that version.
- If you adjust a statistics parameter, study the statistics again to ensure that you did not adversely affect another statistic.

Note: Using the *ON* parameter with the *LOCKSTATISTICS* command can adversely impact performance. It is recommended that you use the *LOCKSTATISTICS ON* in short, controlled time intervals. Use this setting for diagnostic purposes only.

Appendix C

COPYAUDIT Error Messages

Introduction

All COPYAUDIT messages can be categorized as follows.

Category	Caused by . . .
I/O error	Difficulties with reading or writing audit block information I/O errors usually occur under either of the following circumstances: <ul style="list-style-type: none">• Physical problems with the I/O devices being used to transfer the data• Attempting to read or write more or less data than expected For more detailed information on any I/O error messages, refer to the <i>File Attributes Programming Reference Manual</i> .
Data error	Problems with the actual data being transferred
File attribute error	Problems with the file to which or from which data is being transferred
Syntax error	An incorrect COPYAUDIT statement

This appendix provides tables of the COPYAUDIT error numbers you might receive when running the COPYAUDIT program.

Error Message Format

The general format of a fatal error message is

```
<mix number> DISPLAY:  
  ERROR <error number>: <audit file name>.  
<mix number> DISPLAY:  
  <error message text>.
```

The general format of a nonfatal error message is

```
<mix number> DISPLAY:  
  NONFATAL ERROR <error number>: <audit file name>.  
<mix number> DISPLAY: <error message text>.
```

The audit file name construct provides the following information:

COPYAUDIT Error Messages

- The file title, including the pack name for disk files and the tape drive number (MT#) for tape files
- Either SOURCE AUDIT FILE, DEST AUDIT FILE, or NEXT AUDIT FILE to identify which audit file has the problem

If the error is an I/O error, then the following information is appended to the last line of the error message:

```
IOERRORTYPE = <number>: <I/O error message text>.
```

COPYAUDIT Errors

The following information provides an explanation of the COPYAUDIT error messages that can occur during a COPYAUDIT run. Errors are listed in fatal and nonfatal categories and in numeric order under each category.

For a nonfatal problem, COPYAUDIT attempts to find a solution. If a fatal error occurs, you must fix the cause of the problem and then try rerunning the COPYAUDIT program. If a syntax error occurs, the last token scanned before the error was detected is included in the error message.

Some of the COPYAUDIT errors are also accompanied by an IOERRORTYPE error message. Refer to the *File Attributes Programming Reference Manual* for details of these error messages.

Lists of fatal and nonfatal errors are as follows:

COPYAUDIT Fatal Errors

ERROR 1:	READ AUDIT BLOCK# <audit block number> An I/O error occurred while reading the input audit file. Refer to the IOERRORTYPE error message accompanying this message for more details. This error occurs if the problem is with the input audit file. Error 10 is returned when the problem occurs with the first copy of the audit file. Error 11 is returned when the problem occurs with the second copy of the audit file.
ERROR 2:	READ AUDIT BLOCK# <audit block number>: <words> WORDS BUT COULD ONLY READ <words> The audit block being read was not of the expected size. This error occurs if the problem is with the input audit file. Error 12 is returned when the problem occurs with the first copy of the audit file. Error 13 is returned when the problem occurs with the second copy of the audit file.
ERROR 3:	AT AUDIT BLOCK# <audit block number>: ABSN = <absn> SHOULD = <absn>

	<p>The ABSN of the block being read does not have the expected value.</p> <p>This error occurs if the problem is with the input audit file. Error 14 is returned when the problem occurs with the first copy of the audit file. Error 15 is returned when the problem occurs with the second copy of the audit file.</p>
ERROR 4:	AUDIT BLOCK# <audit block number> TIME STAMP MISMATCH FROM PREVIOUS BLOCK
	<p>The previous block timestamp field of the current audit block does not match the timestamp of the previous audit block.</p> <p>This error occurs if the problem is with the input audit file. Error 16 is returned when the problem occurs with the first copy of the audit file. Error 17 is returned when the problem occurs with the second copy of the audit file.</p>
ERROR 5:	AUDIT BLOCK# <audit block number> CHECKSUM ERROR
	<p>A checksum error occurred on an audit block.</p> <p>This error occurs if the problem is with the input audit file. Error 18 is returned when the problem occurs with the first copy of the audit file. Error 19 is returned when the problem occurs with the second copy of the audit file.</p>
ERROR 6:	WRITE AUDIT BLOCK# <audit block number>
	<p>An I/O error occurred while trying to write an audit block. Refer to the IOERRORTYPE error message accompanying this message for more details.</p> <p>There are several reasons that an I/O error is identified as DESCRIPTOR ERROR. This error occurs if the maximum allowed I/O length for an I/O tape on the system has been exceeded. For example, some tape subsystems might have a maximum I/O size as small as 65,535 bytes. To ensure that audit files copy to any tape device on those systems, the declared block size of the audit file must be less than or equal to 65, 535 bytes (10,922 words).</p> <p>Ensure that the audit block size declared in DASDL does not exceed the maximum I/O length for this particular tape device when the COPYAUDIT program runs. Enter the following to find the maximum I/O block size allowed:</p> <pre>OL MT <device type> <unit number></pre> <p>For more information on this command, refer to the <i>System Commands Operations Reference Manual</i>.</p> <p>This error occurs if this is the first copy of the audit file. Error 7 is returned when the problem occurs with the second copy of the audit file.</p>
ERROR 7:	WRITE AUDIT BLOCK# <audit block number>
	<p>An I/O error occurred while trying to write an audit block. Refer to the IOERRORTYPE error message accompanying this message for more details.</p> <p>This error occurs if this is the second copy of the audit file. Error 6 is returned when the problem occurs with the first copy of the audit file.</p>

COPYAUDIT Error Messages

ERROR 8:	AT AUDIT BLOCK# <audit block number> : <words> WORDS TO WRITE BUT WROTE <words>
	<p>An error occurred while trying to write an audit block. This problem usually occurs because the audit block is smaller than expected.</p> <p>This error occurs if this is the first copy of the audit file. Error 9 is returned when the problem occurs with the second copy of the audit file.</p>
ERROR 9:	AT AUDIT BLOCK# <audit block number> : <words> WORDS TO WRITE BUT WROTE <words>
	<p>An error occurred while trying to write an audit block. This problem usually occurs because the audit block is smaller than expected.</p> <p>This error occurs if this is the second copy of the audit file. Error 8 is returned when the problem occurs with the first copy of the audit file.</p>
ERROR 10:	READ AUDIT BLOCK# <audit block number>
	<p>An I/O error occurred while trying to <i>check</i> the first copy of the audit file. Refer to the IOERRORTYPE error message accompanying this message for more details.</p> <p>This error occurs if this is the first copy of the audit file. Error 11 is returned when the problem occurs with the second copy of the audit file. Error 1 is returned when the problem occurs while reading the input audit file.</p>
ERROR 11:	READ AUDIT BLOCK# <audit block number>
	<p>An I/O error occurred while trying to <i>check</i> the second copy of the audit file. Refer to the IOERRORTYPE error message accompanying this message for more details.</p> <p>This error occurs if this is the second copy of the audit file. Error 10 is returned when the problem occurs with the first copy of the audit file. Error 1 is returned when the problem occurs while reading the input audit file.</p>
ERROR 12:	READ AUDIT BLOCK# <audit block number> : <words> WORDS BUT COULD ONLY READ <words>
	<p>The audit block was not of the expected size; the error occurred while checking the first copy of the audit file.</p> <p>This error occurs if this is the first copy of the audit file. Error 13 is returned when the problem occurs with the second copy of the audit file. Error 2 is returned when the problem occurs while reading the input audit file.</p>
ERROR 13:	READ AUDIT BLOCK# <audit block number> : <words> WORDS BUT COULD ONLY READ <words>:
	<p>The audit block was not of the expected size; the error occurred while checking the second copy of the audit file.</p> <p>This error occurs if this is the second copy of the audit file. Error 12 is returned when the problem occurs with the first copy of the audit file. Error 2 is returned when the problem occurs while reading the input audit file.</p>

ERROR 14:	AT AUDIT BLOCK# <audit block number> : ABSN = <absn> SHOULD = <absn>
	The ABSN of the current audit block does not have the expected value. This error occurs if this is the first copy of the audit file. Error 15 is returned when the problem occurs with the second copy of the audit file. Error 3 is returned when the problem occurs while reading the input audit file.
ERROR 15:	AT AUDIT BLOCK# <audit block number> : ABSN = <absn> SHOULD = <absn>
	The ABSN of the current audit block does not have the expected value. This error occurs if this is the second copy of the audit file. Error 14 is returned when the problem occurs with the first copy of the audit file. Error 3 is returned when the problem occurs while reading the input audit file.
ERROR 16:	AUDIT BLOCK# <audit block number> TIMESTAMP MISMATCH FROM PREVIOUS BLOCK
	The previous block timestamp field of the current audit block does not match the timestamp of the previous audit block. This error occurs if this is the first copy of the audit file. Error 17 is returned when the problem occurs with the second copy of the audit file. Error 4 is returned when the problem occurs while reading the input audit file.
ERROR 17:	AUDIT BLOCK# <audit block number> TIMESTAMP MISMATCH FROM PREVIOUS BLOCK
	The previous block timestamp field of the current audit block does not match the timestamp of the previous audit block. This error occurs if this is the second copy of the audit file. Error 16 is returned when the problem occurs with the first copy of the audit file. Error 4 is returned when the problem occurs while reading the input audit file.
ERROR 18:	AUDIT BLOCK# <audit block number> CHECKSUM ERROR
	A checksum error occurred on an audit block. This error occurs if this is the first copy of the audit file. Error 19 is returned when the problem occurs with the second copy of the audit file. Error 5 is returned when the problem occurs while reading the input audit file.
ERROR 19:	AUDIT BLOCK# <audit block number> CHECKSUM ERROR
	A checksum error occurred on an audit block. This error occurs if this is the second copy of the audit file. Error 18 is returned when the problem occurs with the first copy of the audit file. Error 5 is returned when the problem occurs while reading the input audit file.
ERROR 20:	SYNTAX <token>: “)” EXPECTED

COPYAUDIT Error Messages

	<p>A syntax error exists in your COPYAUDIT statement. A closing parenthesis appears to be missing.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 21:	SYNTAX <token>: DECIMAL INTEGER EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. Check that integer values have been supplied.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 22:	SYNTAX <number>: DECIMAL INTEGER> 12 DIGITS
	<p>A syntax error exists in your COPYAUDIT statement. A value you have supplied is larger than 12 digits.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 23:	SYNTAX <token>: PRIMARY OR SECONDARY EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. Either the keyword SECONDARY or PRIMARY is misspelled, or you have provided an incorrect phrase.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 24:	SYNTAX <token>: "=" EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. The equal sign (=) is missing from a clause of the COPYAUDIT statement.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams for information on the AS PRIMARY and AS SECONDARY options.</p>
ERROR 25:	SYNTAX <token>: "1" OR "2" EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. You have designated an alphanumeric, a negative, or a noninteger value in the COPIES clause. You can either omit the COPIES clause, or request COPIES=1 or COPIES=2. Assigning a value of 1 in the COPIES clause has the same effect as not providing a COPIES clause. The value you supply in the COPIES clause must be an unsigned integer. Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 26:	SYNTAX <number>: "1" OR "2" EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. In the COPIES clause you have assigned an integer other than 1 or 2. You can either omit the COPIES clause, or request COPIES=1 or COPIES=2. Assigning a value of 1 in the COPIES clause has the same effect as not providing a COPIES clause.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 27:	SYNTAX <token>: END OF INPUT EXPECTED

	<p>A syntax error exists in your COPYAUDIT statement. Check the end of the statement; more input was found than expected.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 28:	This error number is not currently used.
ERROR 29:	SYNTAX <token>: UNRECOGNIZED REQUEST
	<p>A syntax error exists in your COPYAUDIT statement. There is an unrecognized request or command. Check the first keyword in the statement.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 30:	SYNTAX <token>: "TO" EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. The TO keyword is missing.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 31:	SYNTAX <token>: "FROM" EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. The FROM keyword is missing.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 32:	SYNTAX <token>: DATABASE NAME EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. Check for any of the following conditions:</p> <ul style="list-style-type: none"> • The first node of the audit file name is not a valid database name. • You have provided an audit file name that begins with a slash (/). • You have used lowercase letters. <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 33:	SYNTAX <token>: "AUDIT" OR "2AUDIT" OR "QCAUDIT" OR "QC2AUDIT" EXPECTED

COPYAUDIT Error Messages

	<p>A syntax error exists in the audit file name you provided in your COPYAUDIT statement. The correct format for audit file names is as follows:</p> <ul style="list-style-type: none"> • <database name>/AUDIT<integer> for primary audit files • <database name>/2AUDIT<integer> for secondary audit files • <database name>/QCAUDIT<integer> for primary audit files copied using the QUICKCOPY command <p><database name>/QC2AUDIT<integer> for secondary audit files copied using the QUICKCOPY command</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 34:	SYNTAX <token>: AUDIT FILE NUMBER EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. The last node of the audit file name does not end with an integer in the range 1 through 9999.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 35:	SYNTAX <number>: AUDIT FILE NUMBER > 4 DIGITS
	<p>A syntax error exists in your COPYAUDIT statement. The last node of the audit file name is an integer greater than 9999.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 36:	SYNTAX <token>: PACK, DISK, or TAPE EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. Check that you have designated DISK, PACK, or TAPE.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 37:	SYNTAX <token>: PACKNAME NOT VALID IDENTIFIER
	<p>A syntax error exists in your COPYAUDIT statement. Check that you have designated a valid pack name. Be sure to use uppercase letters.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 38:	READ AUDIT BLOCK 0
	<p>An I/O error occurred while trying to read audit block 0 (zero).</p> <p>Refer to the IOERRORTYPE error message accompanying this message for more details.</p>
ERROR 39:	This error number is not currently used.
ERROR 40:	AUDIT BLOCK 0: CHECKSUM ERROR
	The checksum verification on audit block 0 (zero) of the input file failed.
ERROR 41:	AUDIT BLOCK 0: ABSN = <absn> NEQ <absn>

	The starting ABSN specified in the COPYAUDIT statement does not match the actual ABSN of the first audit block in the audit file. Check the ABSN range you specified.
ERROR 42:	ENDING ABSN = <absn> < (AUDIT BLOCK 0: ABSN + LASTCNTR)
	The ending ABSN specified in the COPYAUDIT statement does not match the actual ABSN of the last audit block in the audit file. Check the ABSN range you specified.
ERROR 43:	READ AUDIT BLOCK# 0
	An I/O error occurred while trying to read and rewrite block 0 of the destination audit file as part of copying an audit file to disk. Refer to the IOERRORTYPE error message accompanying this message for more details.
ERROR 44:	READ AUDIT BLOCK# 0: <words> WORDS BUT COULD ONLY READ <words>
	The audit block being read was not of the expected size. This error occurs when trying to read and rewrite block 0 of the destination audit file as part of copying an audit file to disk. This error occurs when the audit file is on disk. Error 50 occurs when the audit file is on tape.
ERROR 45:	AUDIT BLOCK# 0: ABSN = <absn> SHOULD = <absn>
	The ABSN of audit block 0 (zero) does not have the expected value. This error occurs when trying to read and rewrite block 0 of the destination audit file as part of copying an audit file to disk.
ERROR 46:	AUDIT BLOCK# 0: CHECKSUM ERROR
	A checksum error occurred in audit block 0 (zero) under either of the following circumstances: <ul style="list-style-type: none"> • While reading the next audit file in order to verify the starting ABSN of that audit file • While reading the output file in order to write block 0 when an audit file is being copied to disk
ERROR 47:	WRITE AUDIT BLOCK# 0: <words>
	An I/O error occurred while trying to rewrite block 0 when copying an audit file to disk. Refer to the IOERRORTYPE error message accompanying this message for more details.
ERROR 48:	This error number is currently not used.
ERROR 49:	READ AUDIT BLOCK# 0: <words>
	An I/O error occurred while trying to read the next audit file in order to verify the starting ABSN of that audit file. Refer to the IOERRORTYPE error message accompanying this message for more details.

COPYAUDIT Error Messages

ERROR 50:	READ AUDIT BLOCK# 0: <words> WORDS BUT COULD ONLY READ <words>
	<p>The audit block being read was not of the expected size. This error occurs while trying to read the next audit file in order to verify the starting ABSN of that audit file.</p> <p>This error occurs when the audit file is on tape. Error 44 occurs when the audit file is on disk.</p>
ERROR 51:	ATTRIBUTE ERROR(S)
	<p>A file attribute that the COPYAUDIT program tried to associate with the destination audit file does not have a valid value. Ensure that the DIRECTION file attribute is valid for the destination device. For more information on file attributes, refer to the <i>File Attributes Reference Manual</i>.</p>
ERROR 52:	CHECKSUM ERROR ON FILE SEPARATOR AFTER TAPE AUDIT FILE # <audit number>
	<p>A checksum error exists on the file separator after the tape audit file.</p> <p>This error is fatal if the problem occurs while appending audit files to tape, but is nonfatal if the problem occurs while copying audit files from tape.</p>
ERROR 53:	AUDIT BLOCK# 0: ABSN = <absn> < STARTING ABSN = <absn>
	<p>The ABSN in block 0 (zero) is less than the expected value.</p>
ERROR 54:	ATTRIBUTE ERROR(S)
	<p>A file attribute that the COPYAUDIT program tried to associate with the next audit file does not have a valid value. Ensure that the TITLE file attribute is valid. For more information on file attributes, refer to the <i>File Attributes Reference Manual</i>.</p>
ERROR 55:	FILE ATTRIBUTE: AREAS OR AREASIZE NOT LABEL EQUATED
	<p>The AREAS file attribute or the AREASIZE file attribute of the destination disk audit file was not specified or fileequated, and the value is not stored on the tape audit file.</p>
ERROR 56:	This error number is currently not used.
ERROR 57:	ABSN = <absn> DISAGREES WITH STARTING ABSN = <absn>
	<p>The ABSN of the first block copied is not the same as the expected ABSN.</p>
ERROR 58:	SYNTAX <token>: DECIMAL INTEGER OR (<hex integer>) OR ALL EXPECTED

	<p>A syntax error exists in your COPYAUDIT statement. The value supplied for the ABSN range is invalid. Use the ALL keyword or designate the correct audit range using the ABSN values.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 59:	SOME BLOCKS NOT COPIED (BLOCK 0 OF NEXT AUDIT FILE GIVES LIMIT)
	<p>Some audit blocks were not copied because the audit file was shorter than expected. This error is not fatal if the OVERRIDE option was included in the COPYAUDIT command syntax.</p>
ERROR 60:	SYNTAX <token>: HEX INTEGER EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. A hexadecimal integer was expected and not found.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 61:	SYNTAX <hex number>: HEX INTEGER> 12 DIGITS
	<p>A syntax error exists in your COPYAUDIT statement. A hexadecimal integer is too large. Hexadecimal integers must be no more than 12 characters in length.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 62:	SYNTAX <token>: BEGINNING ABSN = <absn> < 1
	<p>A syntax error exists in your COPYAUDIT statement. You have designated an invalid starting ABSN. The ABSN value cannot be less than 1.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 63:	SYNTAX <token>: ENDING ABSN = <absn> < BEGINNING ABSN = <absn>
	<p>A syntax error exists in your COPYAUDIT statement. You have provided an invalid ABSN range. The ending ABSN value in the ABSN range is less than the starting value.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 64:	SYNTAX <token>: "/" EXPECTED
	<p>A syntax error exists in the audit file you provided in your COPYAUDIT statement. A slash (/) was expected.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 65:	SYNTAX <name>: DATABASE NAME> 17 CHARACTERS

COPYAUDIT Error Messages

	<p>A syntax error exists in your COPYAUDIT statement. The first portion of the audit file name, which identifies the database, is greater than the allowed 17 characters.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 66:	SYNTAX <token>: AUDIT FILE NUMBER = 0
	<p>A syntax error exists in your COPYAUDIT statement. The number 0 was supplied as the audit file number. Audit file numbers must be in the range 1 through 9999.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 67:	SYNTAX <token>: <USERCODE> EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. A usercode was expected as the first part of the audit file name.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 68:	FILE ATTRIBUTE ERRORS
	<p>A file attribute that the COPYAUDIT program tried to associate with the source audit file does not have a valid value. Ensure that the TITLE file attribute is valid. For more information on file attributes, refer to the <i>File Attributes Reference Manual</i>.</p>
ERROR 69:	SYNTAX <token>: "ON" EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. If you are verifying an audit file, ensure that the ON <medium> designation is included correctly in the VERIFY statement.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 70:	DB TIMESTAMP NEQ TO DB TIMESTAMP OF <timestamp>
	<p>The database timestamp stored in the audit file being copied does not match the database timestamp of the next audit file. If you are appending an audit file, this error can also occur if the database timestamp of the audit file being copied does not match the database timestamp of the previous audit file.</p>
ERROR 71:	UPDATE LEVEL > UPDATE LEVEL OF <number>
	<p>The database update level stored in the audit file being copied is greater than the database update level stored in the next audit file.</p>
ERROR 72:	FAILED TO SET LASTRECORD FILE ATTRIBUTE
	<p>The COPYAUDIT program was unable to set the last record file attribute for the disk file.</p>
ERROR 74:	SYNTAX <token>: "COMPARE" EXPECTED

	<p>A syntax error exists in your COPYAUDIT statement. The keyword FORWARD is included in the statement, but the keyword COMPARE is missing.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 75:	TIME OUT ON IO
	A timeout error occurred while trying to perform an I/O operation.
ERROR 76:	COPIES=2 IS NOT ALLOWED WHEN COPY TO DISK
	<p>A syntax error exists in your COPYAUDIT statement. You cannot request two copies of the audit file when copying the audit file to disk.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 77:	AUDIT BLOCK# 0 : ABSN = <absn> DISAGREES WITH LAST ABSN = <absn>
	The ABSN of the last audit block of the audit file you want to copy does not match the block 0 ABSN of the next audit file.
ERROR 78:	SYNTAX <token>: VALID TAPE DENSITY MNEMONIC EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. Check that you have designated a valid tape density or mnemonic.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 79:	SYNTAX <token>: <density> or COMPRESSION OPTION EXPECTED
	<p>A syntax error exists in your COPYAUDIT statement. A density specification or a compression option was expected.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 80:	AUDIT BLOCK# <number> : FIXED ABSN = <absn> SHOULD = <absn>
	<p>The audit block does not have the expected tape block number and a sequence error occurred. The problem occurred while checking the copy of the audit file during a quickcopy operation.</p> <p>This error occurs if this is the first copy of the audit file. Error 81 is returned when the problem occurs with the second copy of the audit file.</p>
ERROR 81:	AUDIT BLOCK# <number> : FIXED ABSN = <absn> SHOULD = <absn>

COPYAUDIT Error Messages

	<p>The audit block does not have the expected tape block number and a sequence error occurred. The problem occurred while checking the copy of the audit file during a quickcopy operation.</p> <p>This error occurs if this is the second copy of the audit file. Error 80 is returned when the problem occurs with the first copy of the audit file.</p>
ERROR 82:	AUDIT BLOCK# <number> : FIXED ABSN = <absn> SHOULD = <absn>
	<p>The audit block does not have the expected tape block number and a sequence error occurred. The problem occurred while reading the audit file during a quickcopy operation.</p>
ERROR 83:	UNEXPECTED END OF TAPE
	<p>An endoftape error occurred unexpectedly while writing to a new reel during a quickcopy operation.</p>
ERROR 84:	FIXED BLOCK SEQUENCE ERROR DURING REEL SWITCH TO CYCLE <cycle number>
	<p>A tapeblocksequence error occurred during a reel switch while performing a quickcopy operation.</p>
ERROR 85:	TIMESTAMP OF LAST BLOCK OF CYCLE <cycle number> DOES NOT MATCH PREVIOUS BLOCK TIMESTAMP OF FIRST BLOCK OF CYCLE <cycle number>
	<p>A timestamp mismatch occurred during a reel switch while performing a quickcopy operation.</p>
ERROR 86:	SYNTAX <token>: QUICKCOPY VALID ONLY FROM DISK TO TAPE OR FROM TAPE TO DISK
	<p>A syntax error exists in your COPYAUDIT statement. You cannot use the QUICKCOPY command to copy audit files from tape to tape or from disk to disk. Instead, use the COPY command.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 87:	SYNTAX <token>: QUICKCOPY APPEND VALID ONLY TO TAPE
	<p>A syntax error exists in your COPYAUDIT statement. You cannot use the QUICKCOPY APPEND command to copy audit files to disk; the command can be used only to append audit files to tape.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 88:	SYNTAX <token>: SCRATCHPOOL ATTRIBUTE VALID ONLY FOR TAPE
	<p>A syntax error exists in your COPYAUDIT statement. You cannot include a scratch pool statement with disk files.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 89:	SYNTAX <token>: SCRATCHPOOL NAME NOT IDENTIFIER

	<p>A syntax error exists in your COPYAUDIT statement. You have designated an invalid scratch pool name.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 90:	SYNTAX <name>: SCRATCHPOOL NAME EXCEEDS 17 CHARACTERS
	<p>A syntax error exists in your COPYAUDIT statement. Scratch pool names must be less than 18 characters.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 91:	THE CONSTRUCT <token> WILL BE DEIMPLEMENTED ON SSR <ssr number>
	<p>In your COPYAUDIT statement, you have included a tape mnemonic that is no longer supported.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p> <p>This error message indicates a nonfatal problem.</p>
ERROR 92:	This error number is not currently used.
ERROR 93:	SYNTAX <token>: COMPRESSION OPTION ALREADY SPECIFIED
	<p>A syntax error exists in your COPYAUDIT statement. The COMPRESSED option was already specified.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 94:	SYNTAX <token>: COMPRESSION OPTION ALLOWED ONLY FOR QUICKCOPY
	<p>A syntax error exists in your COPYAUDIT statement. You cannot use the COMPRESSED and NONCOMPRESSED options with the COPY command.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 95:	ERROR OPENING PREVIOUS FILE FOR QUICKCOPY APPEND
	<p>During an APPEND request, an error occurred while trying to open the last file copied to the tape.</p>
ERROR 96:	CLOSE ERROR# <error number> ON QUICKCOPY TAPE FILE
	<p>An error occurred while closing a tape file during a quickcopy operation.</p>
ERROR 97:	OPEN ERROR# <error number> ON QUICKCOPY TAPE FILE
	<p>An error occurred while opening a tape file during a quickcopy operation.</p>
ERROR 98:	SYNTAX <token>: SPECIFIED AUDIT FILE RANGE FIRST NOT LOWER THAN RANGE LAST



COPYAUDIT Error Messages

	<p>You have specified an invalid audit file number range in the COPYAUDIT statement. The first audit file number must always be less than the second audit file number.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 99:	SYNTAX <token>: AUDIT FILE #1 CANNOT BE SPECIFIED FOR QUICKCOPY APPEND
	<p>You cannot append audit file number 1 to an existing tape. The first audit file of a database and the first audit file after an audit file number rollover occurs must always be copied to a new tape.</p>
ERROR 100:	AUDIT BLOCK# 0 : ABSN = <absn> DISAGREES WITH LAST ABSN OF PREVIOUS FILE= <audit file number>
	<p>The ABSN value of block 0 (zero) for the file you want to copy or append does not match the last ABSN of the last audit file currently on the tape.</p>
ERROR 101:	FILE SEPARATOR AFTER QCAUDIT# <audit file number> ON UNIT# <unit number> DOES NOT MATCH FIRST COPY ON UNIT# <unit number>
	<p>An error exists in the file separator after the audit file. The file separator is used for data checks when appending audit files and when copying audit files from tape.</p> <p>In this instance, you requested an append operation with the COPIES=2 option, and the tape you supplied for the first copy does not match the tape you supplied for the second copy.</p>
ERROR 102:	SYNTAX <token>: APPEND OPTION NOT VALID FOR SPECIFIED TAPE DEVICE
	<p>A syntax error exists in your COPYAUDIT statement.</p>
ERROR 103:	READ ERROR <number> ON FILE SEPARATOR AFTER AUDIT FILE # <audit number>
	<p>A read error occurred while trying to read the file separator after the audit file.</p> <p>The file separator is used for data checks when appending audit files and when copying audit files from tape. File separators are used only on tapes created using a QUICKCOPY command.</p> <p>This error is fatal if the problem occurred while appending audit files to tape, but is nonfatal if the problem occurred while copying audit files from tape.</p>
ERROR 104:	DB TIMESTAMP MISMATCH WITH FILE SEPARATOR AFTER QC AUDIT # <audit file number>; WILL CHECK WITH NEXT AUDIT FILE


	<p>You are trying to copy from a tape that is probably corrupted and the database timestamp information in the file separator after the last audit file does not match the audit file database timestamp.</p> <p>The database timestamp information in the next audit file is checked instead.</p> <p>This error message indicates a nonfatal problem.</p>
ERROR 105:	LAST ABSN MISMATCH WITH FILE SEPARATOR AFTER QC AUDIT # <audit file number>; WILL CHECK WITH NEXT AUDIT FILE
	<p>You are trying to copy from a tape that is probably corrupted and the ABSN information in the file separator after the last audit file does not match the audit file ABSN information. The ABSN information in the next audit file is checked instead.</p> <p>This error message indicates a nonfatal problem.</p>
ERROR 107:	FILE SEPARATOR NOT FOUND AFTER AUDIT FILE # <audit file number>
	<p>No file separator exists after the last audit file. You can append audit files only to tapes that are created using the QUICKCOPY command.</p> <p>The file separator is used for data checks when appending audit files and when copying audit files from tape.</p> <p>This error message indicates a nonfatal problem.</p>
ERROR 108:	CHECK OPTION NOT VALID FOR MULTI-FILE COPY TO SPECIFIED TAPE DEVICE
	A syntax error exists in your COPYAUDIT statement.
ERROR 109:	BLOCKSIZE STORED IN BLOCK#0 DOES NOT MATCH TAPE BLOCKSIZE
	The block size of the data does not match the block size noted in the tape file header information.
ERROR 111:	SYNTAX <token>: INVALID AUDIT FILE SPECIFIED IN RANGE
	<p>A syntax error exists in your COPYAUDIT statement. You have provided an invalid audit file range. Either the database name or the type of audit file (primary or secondary) does not match the first audit file specified or the last audit file specified.</p> <p>Refer to Section 9, Copying Audit Files, for the COPYAUDIT syntax diagrams.</p>
ERROR 112:	DIRECTORY COMMAND VALID ONLY FOR QUICKCOPY AUDIT TAPES
	You cannot use the DIRECTORY command with tapes generated using the COPY command. The DIRECTORY command is valid only with tapes created using the QUICKCOPY command.
ERROR 115:	SYNTAX <token>: MAXFILESPERTAPE MUST BE BETWEEN 1 AND 9999

COPYAUDIT Error Messages

	You have designated an invalid value in the MAXFILESPERTAPE clause. A valid setting for the MAXFILESPERTAPE clause is any integer in the range 1 to 9999.
ERROR 117:	SYNTAX <token>: REMOVE OPTION NOT ALLOWED WITH OVERRIDE CLAUSE
	If you use the OVERRIDE option in a COPYAUDIT command, you cannot use both the CHECK option and the REMOVE option in the same COPYAUDIT command.
ERROR 118:	TAPE CLOSE ERROR WHILE ATTEMPTING TO LABEL BADTAPE ON MT #
	The COPYAUDIT program encountered a close error while attempting to label the designated tape as BADTAPE.
ERROR 119:	NOT READ REVERSE CAPABLE, USE QUICKCOPY
	The tape you supplied to copy the audit file does not have read reverse capability. Use the QUICKCOPY command to copy the audit file instead.
ERROR 120:	AUDIT LEVEL OF <audit file> NOT SUPPORTED BY THIS VERSION OF COPYAUDIT
	You used a version of COPYAUDIT with an audit level that is different from that of the audit file created by the Enterprise Database Server software.
ERROR 123:	QCAUDITS CAN ONLY BE COPIED FROM TAPE TO DISK
	You can use the QUICKCOPY command to copy audit files from disk to tape or from tape to disk. You cannot use the QUICKCOPY command to copy audit files from disk to disk or from tape to tape.
ERROR 124:	SYNTAX <token>: TAPESET NUMBER MUST BE BETWEEN 1 AND 9999
	The valid tapeset numbers range from 1 to 9999.
ERROR 125:	SYNTAX <token>: TAPESET NUMBER ONLY ALLOWED FOR VERIFY OR QUICKCOPY COMMAND
	Tapeset number can only be specified for the VERIFY or QUICKCOPY command.
ERROR 126:	SYNTAX <token>: TAPESET SPECIFICATION NOT ALLOWED
	Tapeset specification is only allowed for the VERIFY or the QUICKCOPY command.
ERROR 127:	SYNTAX <token>: TAPESET NUMBER REQUIRED
	Tapeset number is missing in the tapeset specification.
ERROR 128:	SYNTAX <token>: MCP DOES NOT SUPPORT LOCATE FAST ACCESS. TAPESET SPECIFICATION NOT ALLOWED
	You cannot use the tapeset specification since MCP does not support locate fast access.

ERROR 129:	SYNTAX <token>: MCP DOES NOT SUPPORT ASSOCIATEDFILENAME. TAPESET SPECIFICATION NOT ALLOWED
	You cannot use the tapeset specification since MCP does not support ASSOCIATEDFILENAME tape attribute.
ERROR 130:	AUDIT FILE#<1> ENDED AT ABSN AT ABSN <2> (PRIOR TO USER SPECIFIED ABSN = <3>)
	The ending ABSN value specified in the COPYAUDIT command was greater than the actual ending ABSN value of the audit file copied. If the ABSN value specified in the command is correct, this error might indicate a fault in the audit file.
ERROR 131:	TAPESET NUMBER MUST BE LESS THAN OR EQUAL TO AUDIT FILE NUMBER
	The tapeset number specified is greater than the audit file number specified for the copy operation.
ERROR 132:	TAPESET NUMBER MUST BE LESS THAN AUDIT FILE NUMBER FOR QUICKCOPY APPEND COMMAND
	The tapeset number specified is greater than or equal to the audit file number specified for the append operation.
ERROR 200:	AUDIT BLOCK# <1> READ ERROR
	A read error occurred while reading the audit file.
ERROR 201:	AUDIT BLOCK# <1> UNKNOWN ERROR RETURNED BY AUDITLIB
	An audit library error occurred while reading the audit file.
Errors 202, 203 and 204 are related to the XE features.	
ERROR 202:	THE QUICKCOPY COMMAND MUST BE USED TO COPY A SECTIONED AUDIT FILE TO TAPE
	When copying a sectioned audit file to tape, use the QUICKCOPY command.
ERROR 203:	AUDIT BLOCK # <1> WRITE ERROR
	A write error occurred while writing the audit file.
ERROR 204:	ONE OR MORE AUDIT SECTIONS MISSING

COPYAUDIT Error Messages

	<p>If the audit file is sectioned, all the sections of the audit file must be present.</p>
<p>ERROR 205:</p>	<p>AUDIT FILE# <audit file number> IS INCOMPLETE. QUICKCOPY CAN'T BE USED TO COPY THIS AUDIT FILE. PLEASE USE THE COPY COMMAND.</p>
	<p>The audit file is not complete. This may be caused by a reclone of the database at the Remote Database Backup secondary host or the database may need a Halt/Load recovery. You can use the COPYAUDIT COPY command to back-up the audit file. Once the partial audit is resolved, you may use the QUICKCOPY command to back-up the audit file.</p>
<p>ERROR 206:</p>	<p>AUDIT FILE# <audit file number> IS INCOMPLETE. COPYAUDIT CANNOT BE USED TO COPY THIS AUDIT FILE. PLEASE USE LIBRARY MAINTENANCE.</p>
	<p>The sectioned audit files are not complete. This may be caused by a reclone of the database at the Remote Database Backup secondary host or the database may need a Halt/Load recovery. You can only use the Library Maintenance COPY command to back-up the audit file. Once the partial audit is resolved, you may use the QUICKCOPY command to back up the audit file.</p>
<p>ERROR 207:</p>	<p>AUDIT FILE# <audit file number> IS INCOMPLETE. QUICKCOPY CAN'T BE USED TO COPY THESE AUDIT FILES (<audit file number> - <audit file number>). PLEASE USE THE COPY COMMAND FOR NON-SECTIONED AUDITS OR LIBRARY/MAINTENANCE FOR SECTIONED AUDITS.</p>
	<p>This range of audit files cannot be copied using the QUICKCOPY command, since one of the audit files is not complete. This may be caused by a reclone of the database at the Remote Database Backup secondary host or the database may need a Halt/Load recovery. You can use the COPYAUDIT COPY command to back-up non-sectioned audits or use the LIBRARY/MAINTENANCE COPY command to back-up the sectioned audit files. Once the partial audit is resolved, you can use the QUICKCOPY command to back-up the audit files.</p> <p>Errors 208, 209, 210, and 211 are internal DMAuditLib errors passed to COPYAUDIT.</p>
<p>ERROR 208:</p>	<p>AUDITLIB ERROR: AUDIT FILE IS NOT OPEN</p>
	<p>The DMAuditLib library found the audit file not opened. Do not perform the AUDIT_OPEN call before the other functions.</p>
<p>ERROR 209:</p>	<p>AUDITLIB ERROR: BAD PARAMETER PASSED TO AUDIT LIB</p>
	<p>A bad parameter was passed to the audit reader library.</p>
<p>ERROR 210:</p>	<p>AUDIT BLOCK# <audit block number> :INTEGRITY ERROR IN AUDIT BLOCK/RECORD</p>

	An integrity error occurred while processing an audit block.
ERROR 211:	AUDIT BLOCK# <audit block number> :INTERNAL ERROR IN AUDITLIB
	An internal error occurred while processing an audit block.
ERROR 212:	QUICKCOPY TAPE – TAPE BLOCK SEQUENCE ERROR AT BLOCK # <block number>
	The QUICKCOPY tape has a block sequence error.
ERROR 213:	QUICKCOPY TAPE – TAPE BLOCK TIMESTAMP SEQUENCE ERROR AT BLOCK # <block number>
	The QUICKCOPY tape has a timestamp sequence error at the specified tape block.
ERROR 214:	QUICKCOPY TAPE – TAPE BLOCK CHECKSUM ERROR AT BLOCK # <block number>
	The QUICKCOPY tape has a block checksum error at the specified tape block.
ERROR 219:	AUDIT FILE # <audit file number> NOT INT THE TAPESET <tape set number>
	The specified tape set does not contain the audit file that is specified.
ERROR 220:	INVALID BLOCK ID IN TAPESET DIRECTORY <directory file title>
	The tape set directory file contains an invalid block identifier.
ERROR 221:	SPECIFIED AUDIT FILE RANGE NOT IN THE TAPESET
	The audit file range specified is not in the tape set.
ERROR 222:	OPEN ERROR# <error result> ON TAPESET VOLUME MARKER FILE
	An error occurred while opening the volume marker file on the tape set.
ERROR 233:	SYNTAX “<token>”: “AUDIT” OR “2AUDIT” OR “QCAUDIT” OR “QC2AUDIT” OR “TAPESET” OR “2TAPESET” EXPECTED
	A syntax error exists in your COPYAUDIT directory statement.
ERROR 235:	SYNTAX <token>: INVALID ENCRYPT TYPE
	A syntax error exists in your COPYAUDIT statement. An invalid algorithm type was designated.
ERROR 237:	SYNTAX <token>: AUDITENCRYPT OPTION ALLOWED ONLY FOR QUICKCOPY TAPE
	A syntax error exists in your COPYAUDIT statement. You must use QUICKCOPY when specifying AUDITENCRYPT.
ERROR 238:	ERROR IN TAPE ENCRYPTION
	An error occurred during the tape encryption.

COPYAUDIT Error Messages

ERROR 239:	ERROR CODES TRANSLATION FAILED
	An error occurred during the translation of the encryption/decryption error.
ERROR 240:	ERROR IN TAPE DECRYPTION
	An error occurred during the tape decryption.
ERROR 243:	FEATURE NOT AVAILABLE
	The DATACOMPRESSION feature key is not installed.
ERROR 244:	COMPRESSED LENGTH IS GREATER THAN INPUT LENGTH
	The result of the compression indicates that the size of the output data is greater than the input length.
ERROR 245:	ONE OF THE ARRAY PARAMETERS IS SEGMENTED
	One of the specified arrays passed to the COMPRESSION procedure is segmented.
ERROR 246:	NOT AESGCM CAPABLE
	The installed MCPCRYPTOAPI does not support AESGCM.
ERROR 247:	MCPCRYPTCAPABILITIES DOES NOT EXIST
	The installed MCPCRYPTOAPI does not support MCPCRYPTCAPABILITIES.
ERROR 248:	ERROR IN DATACOMPRESSION
	An error occurred during the data compression.

COPYAUDIT Nonfatal Errors

ERROR 106:	AUDIT FILE # <audit file number> NOT FOUND ON TAPE
	The audit file you want to copy is not on the mounted tape. Load the correct tape. This error message indicates a nonfatal problem.
ERROR 110:	DISK AUDIT "LOCKEDFILE" ATTRIBUTE DOES NOT MATCH TAPE; AUDIT# <audit file number> TO BE COPIED TO A NEW TAPE
	The setting of the LOCKEDFILE attribute of the audit file being copied or appended does not match the setting of the LOCKEDFILE attribute for the tape. Enter AX OK to copy the audit file to a new or different tape. Enter AX QUIT to end this COPYAUDIT run. This error message indicates a nonfatal problem.

ERROR 113:	<p>TAPE COMPRESSION STATUS INCOMPATIBLE WITH COMPRESSION REQUEST; AUDIT# <audit file number> TO BE COPIED TO A NEW TAPE</p> <p>During an APPEND request, either you included the COMPRESSED option in the COPYAUDIT syntax and the mounted tape does not have compression set, or you included the NONCOMPRESSED option in the COPYAUDIT syntax and the mounted tape has compression set.</p> <p>Enter AX OK to have the audit file copied to a different tape.</p> <p>Enter AX QUIT to end this COPYAUDIT run.</p> <p>This error message indicates a nonfatal problem.</p>
ERROR 114:	<p>SPECIFIED MAXIMUM FILES PER TAPE (number) EXCEEDED; AUDIT# <audit file number> TO BE COPIED TO A NEW TAPE</p> <p>The number of files you requested to be copied to a tape exceeds the current MAXFILESPERTAPE setting.</p> <p>Enter AX OK to have some of the files copied to a new tape.</p> <p>Enter AX QUIT to terminate this COPYAUDIT run.</p> <p>This error message indicates a nonfatal problem.</p>
ERROR 121:	<p>READPOSITION ERROR <error number> ON QUICKCOPY TAPE FILE</p> <p>An error occurred while getting the position on the tape.</p> <p>COPYAUDIT can still position the tape by doing rewinds.</p>
ERROR 122:	<p>LOCATEBLOCK ERROR <error number> ON QUICKCOPY TAPE FILE</p> <p>An error occurred while locating a block on the QUICKCOPY tape.</p>
ERROR 215:	<p>WARNING: SOME OF THE FILES SPECIFIED DO NOT EXIST IN THIS TAPESET. COPYAUDIT CAN ONLY <copy or verify> FROM AUDIT FILE <audit file number> TO AUDIT FILE <audit file number></p> <p>Some of the files specified in the file range for the COPY or VERIFY command do not exist in the tape set. These files are ignored for this process.</p>
ERROR 216:	<p>WARNING: TAPESET DIRECTORY FILE <directory file title> NOT FOUND</p> <p>The directory file for the specified tape set could not be located.</p>
ERROR 217:	<p>NO ASSOCIATED FILE ON <database name> TAPESET <tape set number></p> <p>There is no associated file on the tape that is labeled <database name>/TAPESET <tape set number></p>
ERROR 218:	<p>WARNING: UNIT <tape unit numbers> DOES NOT HAVE LOCATE FAST ACCESS CAPABILITIES</p>

COPYAUDIT Error Messages

	The tape drive does not have the Locate Fast Access capability. The COPYAUDIT run does not benefit from a tape set specification.
ERROR 223:	READ ERRIR <error result> ON DIRECTORY FILE AUDIT FILE ENTRY
	An error occurred while reading the audit file entry from the TAPESET directory file.
ERROR 224:	WRITE ERROR <error result> ON DIRECTORY FILE AUDIT FILE ENTRY
	An error occurred while writing the audit file entry to the TAPESET directory file.
ERROR 225:	READ ERROR <error result> ON DIRECTORY FILE VOLUME ENTRY
	An error occurred while reading the volume entry from the TAPESET directory file.
ERROR 226:	WRITE ERROR <error result> ON DIRECTORY FILE VOLUME FILE ENTRY
	An error occurred while writing the volume entry to the TAPESET directory file.
ERROR 227:	READ ERROR <error result> ON DIRECTORY FILE CONTROL ENTRY
	An error occurred while reading the control entry from the TAPESET directory file.
ERROR 228:	WRITE ERROR <error result> ON DIRECTORY FILE CONTROL ENTRY
	An error occurred while writing the control entry to the TAPESET directory file.
ERROR 229:	CHECKSUM ERROR ON DIRECTORY FILE <record type> RECORD
	A checksum error occurred on a TAPESET directory record.
ERROR 230:	NO VOLUME MARKER FILE ON <database> TAPESET <tape set number>
	There is no volume marker file on the designated tape.
ERROR 231:	OPEN ERROR# <error result> ON DIRECTORY FILE
	An error occurred while opening the TAPESET directory file.
ERROR 232:	WARNING: TAPESET DIRECTORY FILE <directory file title> CONTAINS ERRORS AND IS NOT USABLE
	The TAPESET directory file is corrupted and cannot be used.
ERROR 241:	WARNING: EXCLUSIVE OPTION IGNORED FOR TAPE AUDIT
	This error message indicates a nonfatal problem.

ERROR 242:	WRITING MARKER FILE HIT EOT; AUDIT <audit file number> TO BE COPIED TO A NEW TAPE/TAPESET
	<p>COPYAUDIT hits an EOT error while writing a marker file on the QUICKCOPY tape, the following messages are displayed:</p> <p>NONFATAL ERROR 242: <audit file title> (SOURCE AUDIT FILE). WRITING MARKER FILE HIT EOT; AUDIT <audit file number> TO BE COPIED TO A NEW TAPE. PLEASE ENTER "OK" TO START NEW TAPE OR "QUIT" TO QUIT.</p> <p>COPYAUDIT copies the next audit file to a new tape when "OK" is entered. COPYAUDIT quits when "QUIT" is entered.</p>
ERROR 249:	WARNING: COMPRESSED AND DATACOMPRESSION SPECIFIED AT THE SAME TIME RESULTS IN DOUBLE COMPRESSION, MAY NOT BE ADVANTAGEOUS
	<p>Using the COMPRESSED option and DATACOMPRESSION at the same time will not be beneficial and may increase processor time.</p>

Appendix D

Using Mirrored Disks for Disaster Recovery

You can use mirrored disks to provide a remote copy of an audited Enterprise Database Server database for disaster recovery backup. This disaster recovery solution is an alternative to using Remote Database Backup or quiesced databases. Follow the information in this appendix when using mirrored disks with Enterprise Database Server database software so that you can optimize data recovery and minimize database downtime if a switch over to the remote system becomes necessary.

Which Enterprise Database Server Files to Mirror?

The following list identifies the Enterprise Database Server files you need to copy, or mirror, and provides information about whether the files must be placed on mirrored packs. If the files are not placed on mirrored packs, it is essential that identical copies of these files are maintained on the local and remote hosts. To expedite a switch over to the remote system, these files should be placed on packs on the remote host with the same name as the packs on the local host.

- Database data files
The physical database files—such as data sets, sets, and subsets—must be placed on mirrored packs.
- Control file
The database control file must be placed on a mirrored pack.
- Audits
The database audit files must be placed on mirrored packs.
- Enterprise Database Server tailored files and software
Enterprise Database Server tailored files and software—such as the description file, DMSUPPORT library, and the RECONSTRUCT program—must be placed on mirrored packs. Although these files are generally static files, locating them on mirrored packs is essential for recovery if a switch over to the remote system is necessary during a reorganization.
- Guard files
Guard files associated with the database can be placed on mirrored packs or copies can be maintained on nonmirrored packs on the remote host.
- Reorganization files

The generated reorganization code file must be placed on a mirrored pack. The BUILDREORG internal files must identify this mirrored pack. If SORT or COPY specifications are used, they also must identify the mirrored pack. These requirements exist so that these temporary files are available on the remote system if a switch over to the remote system is necessary during a reorganization.

- User programs and Enterprise Database Server system software

User program and Enterprise Database Server system software files can be placed on mirrored packs or copies can be maintained on nonmirrored packs on the remote host. The Enterprise Database Server software must be at the same release level on the remote host as on the local host.

Environment Considerations

The USERDATA file on the remote system must contain the same usercode declaration as the local host for all mirrored databases.

Backup Procedures

It is imperative that the remote system be able to fully recover the database, independent of the local host. Because database dumps can be performed only at the local host, existing disaster backup procedures must remain in place to ensure a recent copy of a DMUTILITY database dump is available at the remote site.

It is also imperative that audit files be available at the remote site. Many recovery scenarios—including those for single abort, abort, reconstruct, rollback, and rebuild—can require more than one audit. You need to determine the appropriate audit history required should one of these recovery scenarios need to occur soon after a switch over to the remote host.

If you normally set the REMOVE option as part of the audit trail specification in your DASDL source, COPYAUDIT removes mirrored audits from both the local pack and the remote mirror. This removal is done once copying of the audit file is complete on the local host. When using URDF, the Enterprise Database Server is not active on the remote host, and there is no facility for copying audits from the remote host. One possible means of transferring these audits is to modify the DATABASE/WFL/COPYAUDIT file to have library maintenance transfer a copy of the audit files to a nonmirrored pack on the remote host before the audit files are removed from disk.

All audits copied using library maintenance should be verified or backed up at the remote host using COPYAUDIT. This action ensures the integrity of the audit files should it become necessary to recover the database on the remote system.

Recovery Procedures

In most cases, when a switch over to a remote system occurs, halt/load recovery runs on the remote system, the database recovers successfully, and normal processing resumes. But, there are abnormal situations where critical Enterprise Database Server files on the remote packs might not contain up-to-the-minute information. These situations cause halt/load recovery to fail.

An example of an abnormal situation might be concurrent hardware problems with the remote disk causing some remote packs to be out of synchronization with the local packs when a switch over occurs or when a sudden disconnect of the communication lines between the local and remote disk occurs.

In a worse-case scenario, a full database rebuild might be required.

Control File Integrity

Before attempting to bring up the database on the remote host, check the control file on the remote pack to be sure it is not out of synchronization with the database audit file. The audit file number (AFN) in the control file must match the current audit file number. The audit block serial number (ABSN) in the control file must be one greater than the last block serial number of the current audit file.

Use the following procedure to determine if the audit file and the control file of the database are synchronized:

1. Use the SYSTEM/DMUTILITY LIST or WRITE function to examine the control file information.
2. Use SYSTEM/PRINTAUDIT to print out block 0 and the last block of the current audit.
3. If the information in the control file is not synchronized with the current audit file, use SYSTEM/DMCONTROL to perform a RECOVER UPDATE operation before opening the database. Otherwise the audit file can be corrupted.

Data File Integrity

If halt/load recovery fails because of data corruption, you should be able to reconstruct the bad rows to recover the database. Examples of such errors resulting from data corruption include file integrity errors, and key and data mismatches. Messages from the failed halt/load recovery indicate the parts of the database that require repair.

Audit File Integrity

If halt/load recovery fails with an AUDIT TIME-STAMP MISMATCH error, then the audit is incomplete. You must choose the appropriate action from the following list. Each of these actions allows halt/load recovery to complete successfully.

Using Mirrored Disks for Disaster Recovery

- If running with duplicated audits and one audit is corrupt, perform the recommended COPYAUDIT procedures to replace the corrupted audit with a copy of the uncorrupted audit.
- If running with duplicated audits and both audits are corrupt, use the COPYAUDIT VERIFY command to determine which audit file contains the most audit information. As a precaution, the most complete of the two files should be backed up using library maintenance. Then repair the most complete audit file by using the COPYAUDIT COPY function with the ALL OVERRIDE option to copy the audit to tape or disk. This action repairs audit files by creating a valid end of audit. The copied (or repaired) audit file should then be substituted for the corrupted audit file.
- If not running with duplicated audits, back up the corrupted audit file. Then repair the audit file by using the COPYAUDIT COPY function with the ALL OVERRIDE option to copy the audit to tape or disk. This action creates a valid end of audit. The copied audit file should be substituted for the corrupted audit file.

Appendix E

Understanding Railroad Diagrams

This appendix explains railroad diagrams, including the following concepts:

- Paths of a railroad diagram
- Constants and variables
- Constraints

The text describes the elements of the diagrams and provides examples.

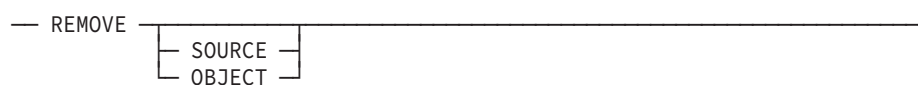
Railroad Diagram Concepts

Railroad diagrams are diagrams that show you the standards for combining words and symbols into commands and statements. These diagrams consist of a series of paths that show the allowable structures of the command or statement.

Paths

Paths show the order in which the command or statement is constructed and are represented by horizontal and vertical lines. Many commands and statements have a number of options so the railroad diagram has a number of different paths you can take.

The following example has three paths:



The three paths in the previous example show the following three possible commands:

- REMOVE
- REMOVE SOURCE
- REMOVE OBJECT

A railroad diagram is as complex as a command or statement requires. Regardless of the level of complexity, all railroad diagrams are visual representations of commands and statements.

Railroad diagrams are intended to show

- Mandatory items
- User-selected items

Understanding Railroad Diagrams

- Order in which the items must appear
- Number of times an item can be repeated
- Necessary punctuation

Follow the railroad diagrams to understand the correct syntax for commands and statements. The diagrams serve as quick references to the commands and statements.

The following table introduces the elements of a railroad diagram:

The diagram element . . .	Indicates an item that . . .
Constant	Must be entered in full or as a specific abbreviation
Variable	Represents data
Constraint	Controls progression through the diagram path

Bold Faced Words



A boldfaced word in a railroad diagram indicates an XE feature.

Constants and Variables

A constant is an item that must be entered as it appears in the diagram, either in full or as an allowable abbreviation. If part of a constant appears in boldface, you can abbreviate the constant by

- Entering only the boldfaced letters
- Entering the boldfaced letters plus any of the remaining letters

If no part of the constant appears in boldface, the constant cannot be abbreviated.

Constants are never enclosed in angle brackets (< >) and are in uppercase letters.

A variable is an item that represents data. You can replace the variable with data that meets the requirements of the particular command or statement. When replacing a variable with data, you must follow the rules defined for the particular command or statement.

In railroad diagrams, variables are enclosed in angle brackets.

In the following example, BEGIN and END are constants, whereas <statement list> is a variable. The constant BEGIN can be abbreviated, since part of it appears in boldface.

— BEGIN —<statement list>— END —————|

Valid abbreviations for BEGIN are

- BE
- BEG
- BEGI

Constraints

Constraints are used in a railroad diagram to control progression through the diagram. Constraints consist of symbols and unique railroad diagram line paths. They include

- Vertical bars
- Percent signs
- Right arrows
- Required items
- User-selected items
- Loops
- Bridges

A description of each item follows.

Vertical Bar

The vertical bar symbol (|) represents the end of a railroad diagram and indicates the command or statement can be followed by another command or statement.

— SECONDWORD — (—<arithmetic expression>—) —————|

Percent Sign

The percent sign (%) represents the end of a railroad diagram and indicates the command or statement must be on a line by itself.

— STOP —————%

Right Arrow

The right arrow symbol (>)

- Is used when the railroad diagram is too long to fit on one line and must continue on the next
- Appears at the end of the first line, and again at the beginning of the next line

Understanding Railroad Diagrams

— SCALERIGHT — (—<arithmetic expression>— , —————→
▶<arithmetic expression>—) —————|

Required Item

A required item can be

- A constant
- A variable
- Punctuation

If the path you are following contains a required item, you must enter the item in the command or statement; the required item cannot be omitted.

A required item appears on a horizontal line as a single entry or with other items. Required items can also exist on horizontal lines within alternate paths, or nested (lowerlevel) diagrams.

In the following example, the word EVENT is a required constant and <identifier> is a required variable:

— EVENT —<identifier>—————|

User-Selected Item

A user-selected item can be

- A constant
- A variable
- Punctuation

User-selected items appear one below the other in a vertical list. You can choose any one of the items from the list. If the list also contains an empty path (solid line) above the other items, none of the choices are required.

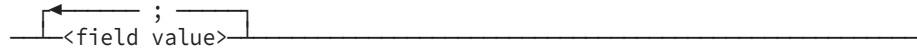
In the following railroad diagram, either the plus sign (+) or the minus sign (–) can be entered before the required variable <arithmetic expression>, or the symbols can be disregarded because the diagram also contains an empty path.

— [+] —<arithmetic expression>—————|
— [–] —

Loop

A loop represents an item or a group of items that you can repeat. A loop can span all or part of a railroad diagram. It always consists of at least two horizontal lines, one below the other, connected on both sides by vertical lines. The top line is a right-to-left path that contains information about repeating the loop.

Some loops include a return character. A return character is a character—often a comma (,) or semicolon (;)—that is required before each repetition of a loop. If no return character is included, the items must be separated by one or more spaces.



Bridge

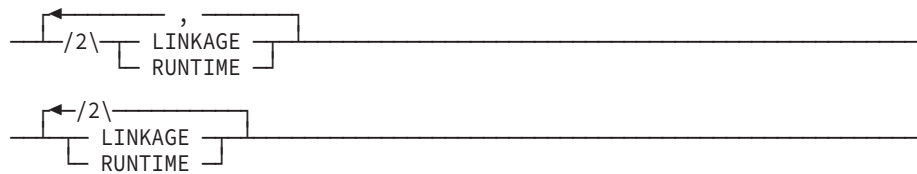
A loop can also include a bridge. A bridge is an integer enclosed in sloping lines (/ \) that

- Shows the maximum number of times the loop can be repeated
- Indicates the number of times you can cross that point in the diagram

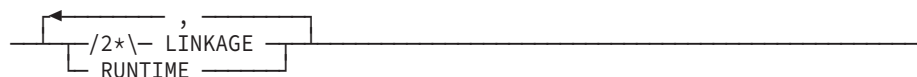
The bridge can precede both the contents of the loop and the return character (if any) on the upper line of the loop.

Not all loops have bridges. Those that do not can be repeated any number of times until all valid entries have been used.

In the first bridge example, you can enter LINKAGE or RUNTIME no more than two times. In the second bridge example, you can enter LINKAGE or RUNTIME no more than three times.



In some bridges an asterisk (*) follows the number. The asterisk means that you must cross that point in the diagram at least once. The maximum number of times that you can cross that point is indicated by the number in the bridge.

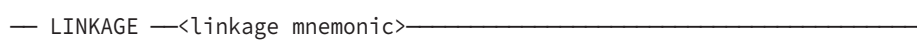


In the previous bridge example, you must enter LINKAGE at least once but no more than twice, and you can enter RUNTIME any number of times.

Following the Paths of a Railroad Diagram

The paths of a railroad diagram lead you through the command or statement from beginning to end. Some railroad diagrams have only one path; others have several alternate paths that provide choices in the commands or statements.

The following railroad diagram indicates only one path that requires the constant LINKAGE and the variable <linkage mnemonic>:



Alternate paths are provided by

Understanding Railroad Diagrams

- Loops
- User-selected items
- A combination of loops and user-selected items

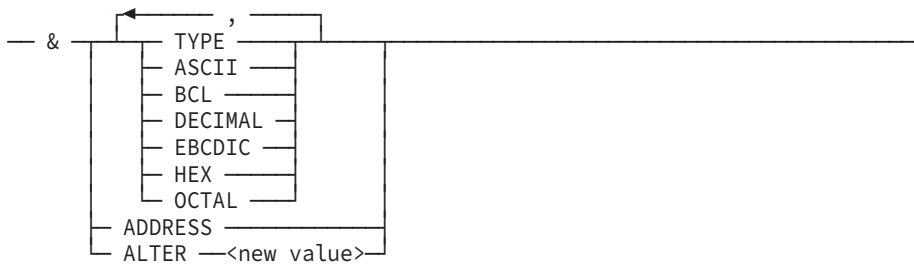
More complex railroad diagrams can consist of many alternate paths, or nested (lowerlevel) diagrams, that show a further level of detail.

For example, the following railroad diagram consists of a top path and two alternate paths. The top path includes

- An ampersand (&)
- Constants that are user-selected items

These constants are within a loop that can be repeated any number of times until all options have been selected.

The first alternative path requires the ampersand and the required constant ADDRESS. The second alternative path requires the ampersand followed by the required constant ALTER and the required variable <new value>.



Railroad Diagram Examples with Sample Input

The following examples show five railroad diagrams and possible command and statement constructions based on the paths of these diagrams.

Example 1

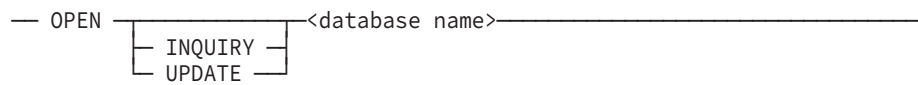
<lock statement>

— LOCK — (— <file identifier> —)

Sample Input	Explanation
LOCK (FILE4)	<p>LOCK is a constant and cannot be altered. Because no part of the word appears in boldface, the entire word must be entered.</p> <p>The parentheses are required punctuation, and FILE4 is a sample file identifier.</p>

Example 2

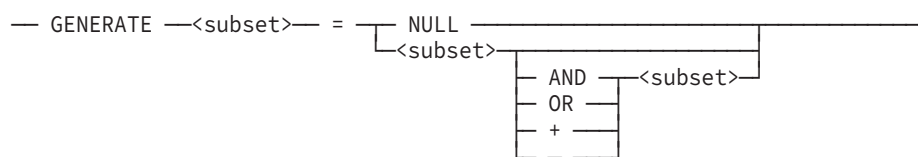
<open statement>



Sample Input	Explanation
OPEN DATABASE1	The constant OPEN is followed by the variable DATABASE1, which is a database name. The railroad diagram shows two user-selected items, INQUIRY and UPDATE. However, because an empty path (solid line) is included, these entries are not required.
OPEN INQUIRY DATABASE1	The constant OPEN is followed by the user-selected constant INQUIRY and the variable DATABASE1.
OPEN UPDATE DATABASE1	The constant OPEN is followed by the user-selected constant UPDATE and the variable DATABASE1.

Example 3

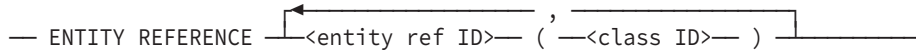
<generate statement>



Sample Input	Explanation
GENERATE Z = NULL	The GENERATE constant is followed by the variable Z, an equal sign (=), and the user-selected constant NULL.
GENERATE Z = X	The GENERATE constant is followed by the variable Z, an equal sign, and the user-selected variable X.
GENERATE Z = X AND B	The GENERATE constant is followed by the variable Z, an equal sign, the user-selected variable X, the AND command (from the list of user-selected items in the nested path), and a third variable, B.
GENERATE Z = X + B	The GENERATE constant is followed by the variable Z, an equal sign, the user-selected variable X, the plus sign (from the list of user-selected items in the nested path), and a third variable, B.

Example 4

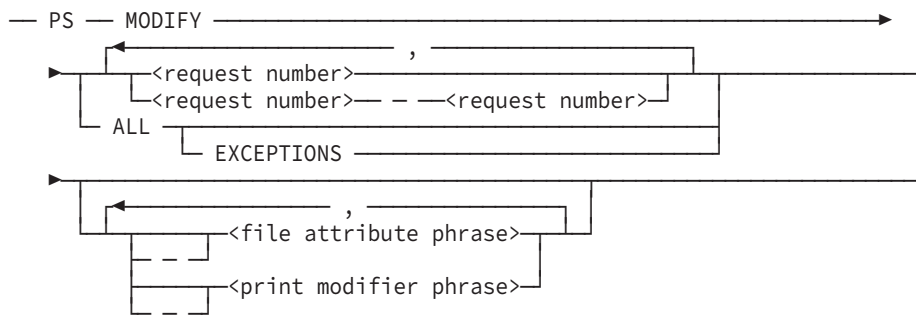
<entity reference declaration>



Sample Input	Explanation
ENTITY REFERENCE ADVISOR1 (INSTRUCTOR)	The required item ENTITY REFERENCE is followed by the variable ADVISOR1 and the variable INSTRUCTOR. The parentheses are required.
ENTITY REFERENCE ADVISOR1 (INSTRUCTOR), ADVISOR2 (ASST_INSTRUCTOR)	Because the diagram contains a loop, the pair of variables can be repeated any number of times.

Example 5

<PS MODIFY command>



Sample Input	Explanation
PS MODIFY 11159	The constants PS and MODIFY are followed by the variable 11159, which is a request number.
PS MODIFY 11159,11160,11163	Because the diagram contains a loop, the variable 11159 can be followed by a comma, the variable 11160, another comma, and the final variable 11163.
PS MOD 11159–11161 DESTINATION = "LP7"	The constants PS and MODIFY are followed by the userselected variables 11159–11161, which are request numbers, and the user-selected variable DESTINATION = "LP7", which is a file attribute phrase. Note that the constant MODIFY has been abbreviated to its minimum allowable form.
PS MOD ALL EXCEPTIONS	The constants PS and MODIFY are followed by the userselected constants ALL and EXCEPTIONS.

Index

A

ABORT recovery program, overview, 2–11

ABSN, 2–10

ABSN command

DMCONTROL, 5–8

access to the database

during reorganization, 7–53

INQUIRYONLY option, 7–54

OFFLINE option, 7–56

during backup process, 6–4

EXCLUSIVE option, 7–55

PREVERIFY option, 7–55

Accessroutines

buffers, B–2

interface with the control file, 2–10

ALGOL interfaces

Audit_Info array, 20–3

Database/Properties file, 20–3

exported from the DMAuditLib library,
20–2

using the \$INCLUDE command, 20–2

\$INCLUDE command, 20–2

ALI_INFO group

AUDIT_INFO array, 20–4

AUDIT_NEXT_ABSN procedure, 20–7

AUDIT_NEXT_RECORD procedure, 20–6

linkage information, 20–8

linkage information ALI_INFO group, 20–8

ALI_xxx words

AUDIT_INFO array, 20–3

ALL option

COPYAUDIT program, 9–16

Database Certification, 11–11

DMUTILITY INITIALIZE statement, 5–33

Database Certification, 11–8

ALLOWEDCORE

Database Certification option, 11–9

DBS CHANGE command, 12–5

phrase in reorganization, 7–49

FAMILYNAME option, Database
Certification, 11–9

Visible Recovery command, 8–43

ALTERNATE AUDITFAMILY command

DMCONTROL, 5–24

ALTERNATE SECAUDITFAMILY command

DMCONTROL, 5–24

APPEND command

multidump tape specification, 6–31

syntax, 6–9

APPEND option, COPYAUDIT program

limiting, 9–14

appending audit files

changing audit file type, 9–17

identifying the file names, 9–15

limitations, 9–12

making two copies, 9–22

syntax, 9–11

syntax, examples, 9–19, 9–22

data compression, 9–20

designating tape density, 9–19

encrypting, 9–20

FORWARD COMPARE option, 9–22

limiting files on a tape, 9–14

MAXFILESPERTAPE clause, 9–14

primary audit files, copying as secondary
audit files, 9–17

QUICKCOPY command, using, 9–6

SCRATCHPOOL option, 9–21

assigned partition number, last, 2–4

attributes of audit file, 2–4

audit block serial number (ABSN)

rollover, 2–10

audit files

accessing, 20–1

appending to tapes, 9–11

checking integrity, 9–27

copying, 9–17

data compression, 9–8, 9–10

deleting after copying, 9–22

effect of LOCKEDFILE attribute, 9–9

information in the control file, 2–3

making two copies, 9–22

maximizing tape usage, 9–7

primary audit files as secondary audit files,
9–17

- printing, 10-1
- secondary audit files as primary audit files, 9-17
- statistics, B-12
- syntax examples, 9-19, 9-22
- tapes, creating a directory, 9-26
- attributes, 2-4
- viewing, 10-1
- with COPY command, 9-23
- control information, 2-8
- copying, 9-1
- extracting audit information, 10-1
- handling discontinuities, 2-9
- integrity with remote database copy, D-3
- naming convention, 9-10
- PRINTAUDIT program, 10-1
- railroad diagrams, explanation of, E-1
- selecting for analysis, 10-4
- VERIFY command, COPYAUDIT program, 9-27
- viewing, 10-1
- viewing audit files, 10-1
- audit records
 - ADSS audit record type, 10-22
 - AGCI audit record type, 10-22
 - AINT audit record type, 10-22
 - AIO audit record type, 10-22
 - AIRE audit record type, 10-22
 - AISE2 audit record type, 10-22
 - ALN audit record type, 10-22
 - ARNE audit record type, 10-22
 - RUODSS, 10-25
 - type mnemonics, 10-22, 10-23 , 10-24 , 10-25
 - type name control records, 10-22
 - type name data change records, 10-22
 - type numbers, 10-22
 - AUDTB2 audit record type, 10-22
 - B4ROOT audit record type, 10-23
 - BCP audit record type, 10-23
 - BIO audit record type, 10-23
 - BLKIMG audit record type, 10-23
 - BTR audit record type, 10-23
 - BVEOF audit record type, 10-23
 - C audit record type, 10-22
 - CCD audit record type, 10-23
 - CDI audit record type, 10-23
 - CIRE audit record type, 10-23
 - CISE audit record type, 10-23
 - CPID audit record type, 10-23
 - CPNT audit record type, 10-23
 - CUOL audit record type, 10-23
 - D audit record type, 10-22
 - DBSI audit record type, 10-23
 - DBST audit record type, 10-23
 - DDCD audit record type, 10-23
 - DDSEOF audit record type, 10-23
 - DDSIEF audit record type, 10-23
 - DDSIEOF audit record type, 10-23
 - DIRE audit record type, 10-23
 - DISE2 audit record type, 10-23
 - DISE audit record type, 10-23
 - DLIRT audit record type, 10-23
 - DLKBLK audit record type, 10-23
 - DLLIST audit record type, 10-23
 - DRNE audit record type, 10-23
 - DSC audit record type, 10-23
 - DSD audit record type, 10-23
 - DSM audit record type, 10-23
 - DSSD audit record type, 10-24
 - ECP audit record type, 10-24
 - ETR audit record type, 10-24
 - FGTBLK audit record type, 10-24
 - FILEDC audit record type, 10-24
 - GCB audit record type, 10-24
 - GETBLK audit record type, 10-24
 - GIST audit record type, 10-24
 - GRNS audit record type, 10-24
 - GRWIST audit record type, 10-24
 - GTRL audit record type, 10-24
 - IDSBLK audit record type, 10-24
 - INSBLK audit record type, 10-24
 - ISPT audit record type, 10-24
 - KIOA audit record type, 10-24
 - LGRA audit record type, 10-24
 - LGRR audit record type, 10-24
 - LTAR audit record type, 10-24
 - MIDTR audit record type, 10-24
 - mnemonics for audit records, 10-23, 10-24 , 10-25
 - ODDSC audit record type, 10-24
 - ODEOF audit record type, 10-24
 - ODRTBL audit record type, 10-24
 - ODSFDN audit record type, 10-25
 - ODSFUP audit record type, 10-25
 - ODSPBL audit record type, 10-25
 - OINZP audit record type, 10-25
 - PFIX audit record type, 10-25
 - PNT audit record type, 10-25
 - RCB audit record type, 10-25
 - RDERR audit record type, 10-25
 - RDSC audit record type, 10-25

- RDSO audit record type, 10–25
 - RECOV audit record type, 10–25
 - RFIX audit record type, 10–25
 - RFLUSH audit record type, 10–25
 - RFMT audit record type, 10–25
 - RIST audit record type, 10–25
 - RLOCK audit record type, 10–25
 - RMOVE audit record type, 10–25
 - RPATH audit record type, 10–25
 - RRNS audit record type, 10–25
 - RSTATE audit record type, 10–25
 - RUODSS audit record type, 10–25
 - SAA audit record type, 10–25
 - SAAG audit record type, 10–25
 - SAC audit record type, 10–25
 - SAD audit record type, 10–25
 - SAI audit record type, 10–25
 - SAM audit record type, 10–25
 - SAR audit record type, 10–25
 - SDSEOF audit record type, 10–25
 - SIBC audit record type, 10–26
 - SIBO audit record type, 10–26
 - SPIRT audit record type, 10–26
 - SPLIST audit record type, 10–26
 - SPT audit record type, 10–26
 - STRDC audit record type, 10–26
 - SVPT audit record type, 10–26
 - TABAI audit record type, 10–26
 - TABBA audit record type, 10–26
 - TABBI audit record type, 10–26
 - TBLXCH audit record type, 10–26
 - audited databases
 - availability during dump, 6–4
 - statistics, B–12
 - AUDITFAMILY command
 - DMCONTROL, 5–23
 - AUDIT_INFO array
 - ALI_INFO group, 20–4
 - Audit_Info array
 - ALI_xxx words, 20–3
 - AUDIT_NEXT_ABSN procedure
 - ALI_INFO group, 20–7
 - AUDIT_NEXT_RECORD procedure
 - ALI_INFO group, 20–6
- B**
- backing up a database
 - APPEND command, 6–9
 - creating compound dump lists, 6–26
 - database control file, 6–5
 - disk and tape output, 6–43
 - effect of DASDL LOCKEDFILE option, 6–5
 - locked control file, resolving, 6–17
 - main directory file, 6–58
 - multidump tapes, 6–37
 - number of tapes for output, 6–27
 - process overview, 6–59
 - recovering, 6–69
 - requesting data compression, 6–28
 - tools, for, 6–58
 - VERIFYDUMP command, 6–44
 - while reorganizing the database, 6–17
 - BUILDDUMPDIRECTORY command, 6–68
 - cataloging information, 6–57
 - choosing a dump media, 6–4
 - compound dump lists, creating, 6–26
 - COMPRESSED option, using in DUMP command, 6–28
 - COPYDUMP command, 6–46
 - deleted dump directory, recovering, 6–69
 - density of tapes, specifying for backups, 6–29
 - designating output disk requirements, 6–31
 - designating output tape requirements, 6–27
 - disk output, 6–40
 - DMDUMPDIR program, 6–60
 - DUMP command, 6–9
 - dump directory file, 6–58
 - DUPLICATEDUMP command, 6–50
 - grouping data on the output media, 6–26
 - identifying what to back up, 6–22
 - limiting the data, 6–25
 - lost dump directory, recovering, 6–69
 - mismatch errors during database backup, 6–5
 - multiple backup dumps, definition, 6–4
 - NONCOMPRESSED option, DUMP command, 6–28
 - offline, 6–4
 - online, 6–4
 - partial dumps, selecting, 6–4
 - process, 6–3
 - recovering dump directories, 6–69
 - regenerating dump directories, 6–69
 - restoring dump directories, 6–69
 - running the DMDUMPDIR program, 6–60
 - selecting parts of the database, 6–4

- storing more than one backup dump on the same tape, 6–31
- syntax, 6–60
- tape density specification, 6–29
- tape density, specifying for backups, 6–29
- tape output, 6–34
- TAPEDIRECTORY command, 6–53
- TAPESET DIRECTORY command, 6–8
- timestamp mismatch errors, 6–5
- updating the dump tape directory, 6–59
- VERIFYDUMP command, 6–44
- BLOCKSIZE clause
 - DUMP command, 6–32
 - COPYDUMP command, 6–48
 - DUMP command, 6–29
 - file title clause, DUMP command, 6–34
- BUILDREORG utility
 - ALLOWEDCORE phrase, 7–49
 - AUTOSWAP option, 7–64
 - CAPTUREUPDATETRAN phrase, 7–43
 - central data set sequence statement, 7–51
 - combinations, 7–57
 - compiler control options, 7–22
 - EXCLUSIVE option, 7–55
 - EXTRACT phrase, 7–39
 - generating a data set, 7–30
 - generating from a data set, 7–33
 - generating from index structures, 7–34
 - index control option, 7–37
 - INQUIRYONLY option, 7–54
 - alias name, 7–28
 - LOADFACTOR phrase, 7–38
 - MAXUPDATERS phrase, 7–43
 - OFFLINE option, 7–56
 - open options, 7–53
 - ORDERBYCORE phrase, 7–50
 - PREVERIFY option, 7–55
 - procedure sequence option, 7–45
 - resource control, 7–37
 - STRCOPYCORE phrase, 7–44, 7–65
 - structure COPY TO option, 7–40
 - syntax, 7–24
 - TASKLIMIT phrase, 7–51
 - TOTALCOPYCORE phrase, 7–44, 7–65
 - UPDATE option, 7–27
 - USEREORGDB option, 7–61
 - AUTOSWAP option, BUILDREORG utility, 7–64
 - central data set control options, 7–46
 - central data set control options in reorganization, 7–46
 - Central Data Set GENERATE statement, 7–29
 - COPY TO option, 7–41
 - deadlock error during reorganization, 7–55
 - EXCLUSIVE option, BUILDREORG utility, 7–55
 - explicitly generating structures, 7–37
 - EXTRACT phrase, using to control reorganization resources, 7–39
 - GENERATE statement, 7–29
 - index structure list option and reorganization, 7–33
 - inquiry-only access to database during reorganization, 7–54
 - INQUIRYONLY option, BUILDREORG utility, 7–54
 - INTERNAL FILES phrase, 7–47
 - LIST compiler control option, using with reorganization, 7–22, 7–32
 - LOADFACTOR phrase, using to control reorganization resources, 7–38
 - MAXUPDATERS phrase, BUILDREORG utility, 7–43
 - open options, BUILDREORG utility, 7–53
 - order of processes, 7–51
 - ORDERBYCORE phrase in reorganization, 7–50
 - PREVERIFY option, BUILDREORG utility, 7–55
 - purpose, 7–20
 - reorg global control statement, 7–53
 - reorganization order, 7–45
 - REORGBALLOWEDCORE option, 7–66
 - resources, controlling during reorganization, 7–37
 - SORT phrase, 7–39
 - SORT phrase, BUILDREORG utility, 7–39
 - SORT phrase, using to control reorganization resources, 7–39
 - STRCOPYCORE phrase, BUILDREORG utility, 7–44, 7–65
 - ZIP compiler control option, using with reorganization, 7–22
- BYCYCLE option
 - COPY statement, 8–55
 - RECOVER statement, 8–9

C

- CANCEL statement, DMUTILITY
 - overview, 2–14
 - syntax, 5–29
- cataloging backup information
 - main directory file, 6–58
 - process overview, 6–59
 - recovering, 6–69
 - running the DMDUMPDIR program, 6–60
 - tools, for, 6–58
 - updating the dump tape directory, 6–59
 - BUILDDUMPDIRECTORY command, 6–68
 - dump directory file, 6–58
 - syntax, 6–60
- CERTIFY
 - command, Database Certification, 11–10
 - options, 11–11
 - options for sets and subsets, 11–22
 - options for structure types, 11–14
 - options for data sets, 11–14
- checking
 - compatibility, 2–8
 - file compatibility, 2–8
- COBOL74
 - SYSTEM/LOADDUMP, 17–12
 - text in LOADDUMP output, 17–7
 - SOURCE <source file title> clause, LOADDUMP, 17–7
- code file family change command
 - DMCONTROL, 5–24
 - security file title change command, 5–25
- code file title change command
 - DMCONTROL, 5–25
- commands, host language
 - OPEN INQUIRY, 2–11
 - OPEN UPDATE, 2–11
- compact data sets
 - crosscheck certification options for, 11–15
 - internal certification options for, 11–14
 - AVAILABLE SPACE option, Database Certification, 11–14
 - crosscheck certification, 11–15
 - LINK option, Database Certification, 11–15
- compatibility checking
 - between control file and software, 2–8
 - between files in database, 2–8
- compiler control options
 - LIST/ZIP/LISTSYM, LOADDUMP utility, 17–12
- compiling Enterprise Database Server software
 - DMINTERPRETER, 18–4
 - DMSUPPORT, 18–3
 - RECONSTRUCT program, 18–3
 - RMSUPPORT library, 18–4
 - WFL job parameters, 18–1
 - RMSUPPORT library, compiling Enterprise Database Server software, 18–4
- compressed tapes
 - audit files, using with, 9–10
 - audit files, using with, 9–8
 - requesting for backups, 6–28
- control file
 - Accessroutines, 2–10
 - audit file attributes, 2–4
 - audit information, 2–3
 - CANCEL function for aborted DBCERT run, 11–4
 - CANCEL statement, 2–14
 - CANCEL statement, syntax, 5–29
 - database update level, 2–3
 - deadlock, 2–7
 - directory descriptor, 2–3
 - DMUTILITY program, 2–13
 - dumping, 6–5
 - dumping the database and, 2–15
 - dynamic database parameters, 2–3
 - exclusive, 2–7
 - family changes, 5–28
 - file compatibility checking, 2–8
 - format level, 2–2
 - identifiers, 2–6
 - initializing, 5–1
 - interfaces, 2–10
 - last number assigned, 2–4
 - LIST/WRITE statements, 2–15
 - locked, resolving, 6–17
 - overwriting, 5–7
 - printing, overview, 2–15
 - procedure, 5–28
 - procedure for partitioned databases, 5–28
 - recovery, 5–27
 - recovery flag, 2–7
 - running DMCONTROL, 5–1
 - state variables, 2–3
 - structure, 2–2
 - structure directory descriptor, 2–3
 - structure numbers, 2–6
 - syntax, 5–3
 - table of contents, 2–2

- tailored software compatibility, 2–8
- tape directory flag, 2–4
- text directory descriptor, 2–4
- TPS synchronized recovery, 2–8
- writing overview, 2–15
- conflicts in TPS environment, 5–26
- contents, 2–2
- creating, 5–7
- DASDL compiler, 2–10
- database recovery, 2–11
- database timestamp, 2–3
- functions, 2–6
- guard file directory descriptor, 2–4
- integrity of remote database copy, D–3
- interlock control, 2–7
- listing, overview, 2–15
- maintaining, 5–1
- overview, 2–1
- overwriting an existing control file, 5–7
- procedure for nonpartitioned databases, 5–28
- purpose, 2–1
- records, 2–6
- recovering from failure, 5–30
- syntax, 5–3
- text directory, 2–4
- TPS information, 2–4
- unlocking, 6–17
- updating, 5–9
- updating control file, 5–9
- WRITE statement, overview, 2–15
- control information for audits, 2–8
- convention for audit file names, copied using COPY command, 9–10
- convention for audit file names copied using QUICKCOPY command, 9–8
- COPY command, COPYAUDIT program
 - audit file naming convention, 9–10
 - data compression, 9–10
 - database recovery, 9–10
 - overview, 9–9
 - DASDL LOCKEDFILE attribute, 9–10
 - noncompressed tapes and audit files, 9–10
 - uncompressed tapes and audit files, 9–10
- COPY statement in DMUTILITY
 - copy to, 8–59
 - overview, 2–14
 - BYCYCLE option, 8–55
 - copy as, 8–57
 - copy onto, 8–59
 - QDCVERIFY option, 8–56
 - QDCWORKERS option, 8–56
 - recovery, 8–51
 - WORKERS option, 8–55
- COPYAUDIT program
 - ALL option, 9–16
 - APPEND option, 9–12
 - AS PRIMARY option, 9–17
 - audit block serial number clause, 9–17
 - audit file name clause, 9–24
 - audit file range clause, 9–15
 - AUDITENCRYPT option, 9–20
 - checking results, 9–5
 - COMPRESSED option, 9–20
 - COPIES option, 9–22
 - data compression, 9–8
 - database recovery, 9–10
 - database recovery limitations, 9–9
 - DEBUG option, 9–4
 - DIRECTORY command, 9–26
 - error messages, C–1
 - examples, 9–19, 9–22
 - file naming convention, 9–8, 9–10
 - FORWARD COMPARE option, 9–22, 9–25
 - identifying destination location, 9–17
 - identifying source files, 9–17
 - limiting files on a tape, 9–14
 - LOCKEDFILE attribute, 9–9
 - MAXFILESPEPTAPE clause, 9–14
 - methods for copying files, 9–6
 - NONCOMPRESSED option, 9–20
 - OVERRIDE option, 9–16
 - overview, 9–1, 9–6, 9–9
 - APPEND option, COPYAUDIT program, 9–12
 - quick reference, 9–29
 - QUICKCOPY command, 9–11
 - REMOVE option, 9–22
 - running, 9–3
 - running, manually, 9–4
 - sample WFL job, 9–5
 - AS SECONDARY option, 9–17
 - SCRATCHPOOL option, 9–21
 - syntax, 9–4, 9–11, 9–23
 - tape density specification, 9–19
 - TAPESET specification, 9–17
 - tasks, 9–6
 - audit block serial number (ABSN) clause, 9–17
 - audit file name clause, 9–15

- audit file name clause, COPY command, 9–24
- COPY command, 9–23
- data compression, 9–10
- density specification, COPY command, 9–25
- EXCLUSIVE option, 9–17
- facilities, 9–3
- integrity of audit files, checking, 9–27
- LOCKEDFILE attribute, 9–10
- maximizing tape usage, 9–7
- medium option, COPY command, 9–24
- medium options, 9–24
- noncompressed tapes and audit files, 9–8
- overriding errors during audit file copy, 9–16
- QC audit file name convention, 9–8
- range clause, 9–15
- running, automatically, 9–3
- sectioned audits copying with COPY command, 9–23
- sectioned audits verifying with VERIFY command, 9–28
- syntax option for tape encryption, 15–4
- tape density specification, 9–25
- task values for COPYAUDIT program, 9–5
- uncompressed tapes and audit files, 9–8
- VERIFY command, 9–27
- COPYDUMP command
 - BLOCKSIZE clause, 6–48
 - destination dump disk clause, 6–49
 - destination dump tape clause, 6–48
 - examples, 6–49
 - options clause, 6–47
 - source dump disk clause, 6–48
 - source dump tape clause, 6–48
 - syntax, 6–46
- copying audit files
 - ALL option, 9–16
 - APPEND option, 9–12
 - AS PRIMARY option, 9–17
 - AS SECONDARY option, 9–17
 - audit block serial number clause, 9–17
 - audit file name clause, 9–15, 9–24
 - audit file range clause, 9–15
 - AUDITENCRYPT option, 9–20
 - COMPRESSED option, 9–20
 - COPIES option, 9–22
 - COPY command, 9–23
 - copying primary audit files as secondary audit files, 9–17
 - copying secondary audit files as primary audit files, 9–17
 - data compression, 9–8, 9–10
 - database recovery, 9–10
 - database recovery limitations, 9–9
 - examples, 9–22
 - EXCLUSIVE option, 9–17
 - file naming convention, 9–10
 - FORWARD COMPARE option, 9–22, 9–25
 - identifying destination location, 9–17
 - identifying source files, 9–17
 - limiting files on a tape, 9–14
 - LOCKEDFILE attribute, 9–9, 9–10
 - making two copies, 9–22
 - MAXFILESPERTAPE clause, 9–14
 - maximizing tape usage, 9–7
 - medium options, 9–24
 - one at a time, 9–9
 - one or more at a time, 9–6
 - options, 9–6
 - OVERRIDE option, 9–16
 - overview, 9–1, 9–6, 9–9
 - purpose, 9–2
 - quick reference, 9–29
 - QUICKCOPY command, 9–11
 - REMOVE option, 9–22
 - sample WFL job, 9–5
 - SCRATCHPOOL option, 9–21
 - syntax, 9–4, 9–11, 9–23
 - tape density specification, 9–19, 9–25
 - tasks, 9–6
 - checking results, 9–5
 - DEBUG option, 9–4
 - examples, 9–19
 - file naming convention, 9–8
 - NONCOMPRESSED option, 9–20
 - QUICKCOPY command, COPYAUDIT program, 9–11
- copying backup dumps
 - COPYDUMP command, 6–46
 - destination dump disk clause, 6–49
 - destination dump tape clause, 6–48
 - examples, 6–49
 - options clause, 6–47
 - source dump disk clause, 6–48
 - source dump tape clause, 6–48
 - syntax, 6–46
 - DUPLICATEDUMP command, 6–46
- creation timestamp for structures, 2–5
- customizing the PRINTAUDIT program
 - compiling, 10–27

- file equations, 10–27
 - stopper pattern information, 10–29
 - syntax, 10–31
 - variables, 10–28
 - defining the interval, 10–31
 - developing the code, 10–28
 - examples, 10–35
 - explanation, 10–31
 - introduction, 10–31
 - options, 10–32
 - PRINTIT variable, 10–28
 - USERPROCEDURE procedure, 10–28
 - USERWRAPUP procedure, 10–28
- D**
- DASDL, 2–10
 - Data and Structure Definition Language (DASDL)
 - interface with the control file, 2–10
 - syntax for tape encryption, 15–3
 - data compression, using with
 - audit files, 9–8
 - COPYAUDIT COPY command, 9–10
 - COPYAUDIT QUICKCOPY command, 9–8
 - database backups, 6–28
 - data errors
 - reorganization, 7–84
 - data path change command
 - DMCONTROL, 5–25
 - data sets
 - direct preallocating records, 5–33
 - large object tank, 13–2
 - analyzing large object tank data sets, 13–2
 - random, 11–18
 - standard (fixed-format), 11–19
 - unordered, 11–21
 - CERTIFY options for structure types in, 11–14
 - compact, 11–14
 - direct, 11–16
 - direct data sets, 11–16
 - large object tank data sets, report of disk usage, 13–2
 - LOBANALYZE command, DMUTILITY, 13–2
 - ordered, 11–16
 - ordered data sets, 11–16
 - preallocating records for direct data sets, 5–33
 - standard (variable-format), 11–20
 - standard (variable-format) data sets, 11–20
 - Database Certification
 - ALL option, 11–11
 - AVAILABLE SPACE option, 11–15
 - examples, 11–13
 - LINK option, 11–12, 11–15
 - ALL, 11–8
 - ALL option, 11–11
 - ALLOWEDCORE option, 11–9
 - ordered list sets, CERTIFY option, 11–24
 - REMOTEOUT option, 11–2
 - STRUCTURE CHECK option, 11–13
 - structure number, 11–11
 - AVAILABLE SPACE option, 11–12
 - AVAILABLE SPACE option, Database Certification, 11–12
 - batch mode, 11–3
 - bit vector sets, 11–22
 - bit vector sets, CERTIFY option, 11–22
 - BYE command, 11–10
 - certification options, 11–1
 - CERTIFY ALL option, Database Certification, 11–11
 - coarse table key, 11–23
 - command, 11–10
 - command variables, 11–7
 - compact data sets, 11–14
 - compact data sets, CERTIFY option, 11–14
 - CONTENTS option, 11–12
 - CONTENTS option, Database Certification, 11–12
 - control words, checking, 11–13
 - controlling output, 11–8
 - COUNT option, 11–12
 - COUNT option, Database Certification, 11–12
 - data control word check, 11–13
 - Database Certification, 11–1
 - direct data sets, CERTIFY option, 11–16
 - discontinuing, 2–17
 - DMSUPPORT library, 3–1
 - DUMPBLOCK, 11–8
 - DUMPBLOCK option, 11–8
 - DUMPBLOCKS option, Database Certification, 11–8
 - END command, 11–10
 - error indicator, 11–3
 - failure, recovering, 2–14
 - FAMILYNAME option, 11–9
 - fine tables, 11–23

- HELP CERTIFY command, Database Certification, 11–7
- HELP command, 11–6
- index random sets, 11–23
- index random sets, CERTIFY option, 11–23
- index sequential sets, 11–23
- index sequential sets, CERTIFY option, 11–23
- interactive mode, 11–2
- INTERNAL FILES command, 11–7
- INTERNAL FILES command, Database Certification, 11–7
- LINK option, Database Certification, 11–12
- LOWERCASE command, 11–9
- LOWERCASE command, Database Certification, 11–9
- ONLINE command, 11–6
- ONLINE command, Database Certification, 11–6
- options, 11–11
- OPTIONS command, 11–8
- OPTIONS command, Database Certification, 11–8
- options for structure types, 11–14
- ordered data sets, CERTIFY option, 11–16
- ordered list sets, 11–24
- output description, 11–4
- OWNER option, 11–12
- OWNER option, Database Certification, 11–12
- partition identifier, 11–11
- QUIT command, 11–10
- QUIT command, Database Certification, 11–10
- random data sets, 11–18
- random data sets, CERTIFY option, 11–18
- READONLY option, 11–12
- READONLY option, Database Certification, 11–12
- RECORD PLACEMENT option, 11–12
- RECORD PLACEMENT option, Database Certification, 11–12
- REMOTEOUT, 11–8
- REMOTEOUT option, 11–8
- REMOTEOUT option, Database Certification, 11–2, 11–8
- RESET, 11–8
- RESET option, Database Certification, 11–8
- restart, 11–4
- restart data sets, 11–18
- restart data sets, CERTIFY option, 11–18
- sample report, 11–4
- SEGMENTS option, 11–9
- SEGMENTS option, Database Certification, 11–9
- sets and subsets, CERTIFY option, 11–22
- sets and subsets, Database Certification, 11–22
- SETS option, 11–12
- SETS option, Database Certification, 11–12
- SORT command, 11–8
- SORT command, Database Certification, 11–8
- SORT USING option, 11–9
- SORT USING option, Database Certification, 11–9
- standard (fixed-format) data sets, 11–19
- standard (variable-format) data sets, 11–20
- STOP command, 11–10
- STOP command, Database Certification program, 11–10
- STRUCTURE CHECK option, Database Certification, 11–13
- structure names, 11–11
- TAPES option, 11–9
- TAPES option, Database Certification, 11–9
- TASKVALUE, 11–3
- unordered data sets, 11–21
- unordered data sets, CERTIFY option, 11–21
- unordered list sets, 11–25
- unordered list sets, CERTIFY option, 11–25
- UPPERCASE command, 11–9
- VERIFYSTORE option, 11–13
- VERIFYSTORE option, Database Certification, 11–13
- database control file, 2–1
- database encryption
 - components and interdependencies, 23–1
 - DATAENCRYPT option, 23–4
 - error handling, 23–11
 - example, 23–6
 - performance and best practices, 23–15
 - troubleshooting, 24–1
- database parameters, dynamic, 2–3, 2–8
- database recovery
 - audit files, using, 9–10

- QUICKCOPY audit files, using, 9–9
- RECOVER statement, 2–15
 - overview, 8–1
 - Visible Recovery commands, 8–1
- database reorganization
 - access to the database, 7–53
 - compiler control options, 7–22
 - GENERATE statement, 7–29
 - generating from index structures, 7–34
 - algorithm, 7–15
 - amount of fixup core, 7–49
 - amount of sort core, 7–49
 - SORT phrase, 7–39
 - backing up files, 7–15
 - BUILDREORG utility, 7–20
 - central data set control options, 7–46
 - Central Data Set GENERATE statement, 7–29
 - COPY TO option, BUILDREORG utility, 7–41
 - data errors, 7–84
 - disk storage requirements, 7–89
 - DMSUPPORT library file name changes, 7–7
 - effect of TPS, 7–69
 - EXCLUSIVE option, 7–55
 - explicitly generated structures, 7–37
 - EXTRACT phrase, 7–39
 - file format conversion, 7–3
 - garbage collection, 7–2
 - generating a data set, 7–30
 - generating from a data set, 7–33
 - identifying the location of internal files, 7–47
 - index control option, 7–37
 - index control option, BUILDREORG utility, 7–37
 - INQUIRYONLY option, 7–54
 - introduction, 7–1
 - I/O errors, 7–83
 - limitations, 7–91
 - limiting the number of processes, 7–51
 - LOADFACTOR phrase, 7–38
 - OFFLINE option, 7–56
 - OFFLINE reorganization, 7–16
 - ONLINE option, 7–15
 - order of processes, 7–51
 - ORDERBYCORE phrase, 7–50
 - overview, 2–16
 - performance enhancement, 7–87
 - phases, 7–67
 - preparation tasks, 7–67
 - PREVERIFY option, 7–55
 - process, 7–5
 - rebuild recoveries, 7–80
 - record format conversion, 7–4
 - reorganization order, 7–45
 - REORGDB control COPY TO option, 7–41
 - REORGDBALLOWEDCORE option, 7–66
 - REORGDBALLOWEDCORE option,
 - BUILDREORG utility, 7–66
 - resource control, 7–37
 - restarting, 7–79
 - rollback recoveries, 7–83
 - running REORGANIZATION program,
 - 7–66
 - starting the REORGANIZATION program,
 - 7–68
 - status, 7–86
 - structure availability, 7–70
 - structure COPY TO option, 7–40
 - syntax, 7–24
 - types, 7–2
 - UPDATE option, 7–27
- databases
 - allowing access during backup, 6–4
 - allowing access during reorganization,
 - 7–53
 - displaying file contents, 11–30
 - file name clause in DUMP command,
 - 6–23
 - initializing, 5–30
 - large objects, maintaining, 13–1
 - nonusercoded, reorganizing, 7–69
 - partitioned, recovering control files, 5–28
 - usage statistics, B–9
 - VSS2 optimization, B–9
 - backing up, 6–1
 - communicating with Visible DBS
 - interface, 12–1
 - directory database, permanent, 16–1
 - initializing files, syntax, 5–32
 - LIST statement in DMUTILITY, 11–30
 - listing file contents, 11–30
 - partial recovery, 8–6
 - permanent directory, 16–1
 - permanent directory databases, 16–1
 - status of files and rows, 11–25
- DBDIRECTORY statement
 - DMUTILITY, 11–25
 - overview, 2–14

- DBS CHANGE command
 - AUDIT BUFFERS option, 12-6
 - AUDIT SECTIONS option, 12-6
 - ALLOWEDCORE option, 12-5
 - SYNCWAIT option, 12-8
 - TRACKERQPFACOR option, 12-8
 - AUDIT ANALYZE AFN command, 12-9
 - AUDIT BLOCKSIZE, DBS CHANGE command, 12-5
 - AUDIT BLOCKSIZE option, 12-5
 - AUDIT BUFFERS, in DBS CHANGE command, 12-6
 - audit generation rates, reporting, 12-9
 - AUDIT SECTIONS, in DBS CHANGE command, 12-6
 - Visible DBS program,, 12-5
 - MAXUPDATEPERTR, in DBS CHANGE command, 12-7
 - MAXUPDATEPERTR option, 12-7
 - OVERLAYGOAL, in DBS CHANGE command, 12-7
 - OVERLAYGOAL option, 12-7
 - RESIDENT LIMIT, in DBS CHANGE command, 12-7
 - RESIDENT LIMIT option, 12-7
 - SETFAMILYINDEX, in DBS CHANGE command, 12-7
 - SETFAMILYINDEX option, 12-7
 - SYNCPOINT, in DBS CHANGE command, 12-8
 - SYNCPOINT option, 12-8
 - SYNCWAIT, in DBS CHANGE command, 12-8
 - TRACKERFLUSHDB, in DBS CHANGE command, 12-8
 - TRACKERFLUSHDB option, 12-8
 - TRACKERQPFACOR option, 12-8
- deadlock situations, preventing, 2-7
- deleting
 - audit files after a copy job, 9-22
 - deleted large object items of an existing structure, 13-3
 - large objects of a deleted structure, 13-3
- directories, 2-4
 - creating for audit file tapes, 9-26
 - guard file, 2-5
 - tape flag, 2-4
 - structure, 2-5
 - tape flag, 2-4
- DIRECTORY command
 - COPYAUDIT program, 9-26
- disaster recovery
 - using mirrored disks, D-1
 - mirrored disks, for disaster recovery, D-1
 - mirroring files, D-1
- discontinuities, handling, 2-9
- disk dumps
 - contrasting with tape dumps, 4-5
 - directory, 6-57
 - input/output errors, 4-12
 - omitting tape options, 4-6
 - online disadvantages, 4-7
 - operator interface, 4-8
 - resource contention, 4-7
 - restarting, 4-8
 - tape incompatibility, 4-7
 - usercode, 4-7
 - verifying, 6-45
 - cataloging backup information, 6-57
 - limitations, 4-6
 - limitations; disk dumps, 4-6
 - warnings, 4-16
- DMAuditLib library
 - ALI_INFO ALI_BLKLIST_COUNT word, 20-11
 - AUDIT_BUFFERS array, 20-10, 20-12
 - converting existing programs, 20-2
 - database encryption, 23-1
 - debug words, 20-10
 - declaring in the SYMBOL/DMAUDITLIB file, 20-2
 - entry points, 20-1, 20-12
 - error results, 20-20
 - exporting the ALGOL interfaces, 20-2
 - SYMBOL/DMAUDITLIB file, 20-2
- DMCONTROL
 - ABSN command, 5-8
 - ALTERNATE AUDITFAMILY command, 5-24
 - ALTERNATE SECAUDITFAMILY command, 5-24
 - AUDITFAMILY command, 5-23
 - code file family change command, 5-24
 - CREATE QDC command, 14-12
 - data path change command, 5-25
 - DONTOVERWRITE command, 5-8
 - FAMILY command, 5-23
 - INITIALIZE command, 5-7
 - LOCKEDFILE command, 5-10
 - MAXUPDATEPERTR command, 5-17
 - OVERRIDE AUDITBUFFERS command, 5-19

- OVERWRITE command, 5–8
- QUIESCEDBRESET command, 5–26
 - recovering control files with family changes, 5–28
- RESTORE FROM QDC command, 14–19
 - DMCONTROL, 14–19
 - running, 5–1
 - security file title change command, 5–25
 - syntax, 5–3
 - SYSTEM/DMCONTROL, 5–1
- DMDATARECOVERY program, overview, 2–11
- DMDUMPDIR program
 - disabling, 6–62
 - enabling, 6–61
 - examples, 6–65
 - fatal error, 8–32
 - running, 6–60
 - log of dumps, 6–57
 - using to maintain a dump directory, 6–57
 - using with duplicates of dumps, 6–51
- DMSUPPORT library
 - entry points, 3–1
 - file name changes during reorganization, 7–7
 - compiling Enterprise Database Server software, 18–3
 - declaring, 3–2
 - DMEXCEPTIONNAME, 3–1
 - DMEXCEPTIONTEXT, 3–2
 - DMSTRUCTURENAME, 3–1
 - example programs showing entry points, 3–3
- DMUTILITY
 - CANCEL statement, syntax, 5–29
 - CFRESTORE command, 14–31
 - continuing, 4–10
 - COPY statement, recovery, 8–51
 - database disk I/O errors, 4–13
 - DBDIRECTORY statement, 11–25
 - disk dumps, limitations, 4–6
 - disk dumps, operator interface, 4–8
 - dumps, disk input/output errors, 4–12
 - dumps, tape input/output errors, 4–11
 - error handling, 4–11
 - INITIALIZE statement, syntax, 5–32
 - loads, disk input/output errors, 4–13
 - LOBANALYZE command, 13–2
 - LOBCLEANUP command, 13–3
 - LOBSQUASH command, 13–3
 - operator interface, tape dumps, 4–8
 - QUIESCE command, 14–2
 - QUIESCE QDC command, 14–6
 - RECOVER statement, 8–1
 - restarting disk dumps, 4–8
- OVERWRITE AUDITSECTIONS command, 5–20
- OVERWRITE DATAPATH command, 5–22
- OVERWRITE FAMILY command, 5–20
- OVERWRITE GUARDFILETITLE command, 5–21
- OVERWRITE HL command, 5–20
- OVERWRITE LOCKEDFILE command, 5–22
- OVERWRITE POPULATIONINCR command, 5–21
- OVERWRITE POPULATIONWARN command, 5–21
- OVERWRITE SECURITYADMIN command, 5–22
- OVERWRITE SENSITIVEDATA command, 5–22
- OVERWRITE USEREORGDB command, 5–22
- RECOVER INITIALIZE command, 5–19
- RECOVER PARTITIONS command, 5–18
- RECOVER UPDATE command, 5–17
- SECAUDITFAMILY command, 5–24
- SENSITIVEDATA command, 5–11
- STRUCTURE command, 5–23
- UPDATE command, 5–9
 - creating incremental/accumulated dumps
 - from a quiesce database, 14–21
 - data file family change, 14–14
 - description file, file-equating in
 - DMCONTROL, 5–1
 - family change bit, 5–2
 - file-equating description file in
 - DMCONTROL, 5–1
 - file-equating the description file, 5–1
- OVERWRITE DATAPATH command
 - DMCONTROL, 5–22
- OVERWRITE LOCKEDFILE command
 - DMCONTROL, 5–22
- OVERWRITE POPULATIONINCR command
 - DMCONTROL, 5–21
- OVERWRITE SECURITYADMIN command
 - DMCONTROL, 5–22
- OVERWRITE SENSITIVEDATA command
 - DMCONTROL, 5–22
- OVERWRITE USEREORGDB command
 - DMCONTROL, 5–22

- restarting tape dumps, 4–8
- RESUME command, 14–5
- RETRYIO error messages, examples, 4–15
- tape encryption syntax, 15–3
- task attribute task values, 4–4
- warnings during disk dumps, 4–16
- warnings during tape dumps, 4–16
 - canceling offline dumps, 5–30
- CHECKSUM errors, 4–13
- contrasting dump media, 4–5
- control file interface, 2–13
- database disk I/O errors, 4–13
- deadlock situations, preventing, 2–7
- debugging information, printing
 - unexpectedly, 4–10
- deleted large objects items of an existing structure, deleting, 13–3
- disabling the database, 11–28
- DMUTILITY commands, 4–1
- enabling a database, 11–28
- large objects of a deleted structure, deleting, 13–3
- LIST statement, 11–30
- loads, tape I/O errors, 4–12
- LOBCLEANUP command, DMUTILITY, 13–3
- LOBCOMBINE command, 13–3
- operator interface, for disk dumps, 4–8
 - purpose, 4–1
- QUIESCE command, DMUTILITY, 14–2
- read error messages, 4–13
- restoring original configuration of quiesce database copy, 14–6
- RESUME command, DMUTILITY, 14–5
- retry error messages, 4–13
- RETRYIO error messages, 4–13
- running, 4–3
- tape encryption, 15–1
- task values for DMUTILITY, 4–4
- write error messages, 4–13
- WRITE statement, 11–30
- DONTOVERWRITE command
 - DMCONTROL, 5–8
- DUMP command
 - ACCUMULATED option, 6–20
 - BLOCKSIZE clause, 6–29, 6–32
 - ACCUMULATED option, DUMP command, 6–20
 - COMPRESSED option, 6–28
 - CORRECTREADERROR option, 6–15
 - DATACompression option, 6–15
 - disk and tape output, 6–43
 - disk output, 6–40
 - dump clause, 6–20
 - dump list clause, 6–22
 - dump selector clause, 6–24
 - DUMPDISKSIZE option, 6–15
 - exclude list clause, 6–24
 - file name clause, 6–23
 - file title clause, 6–34
 - FORWARD COMPARE option, 6–13
 - INCREMENTAL option, 6–18
 - multidump tape specification, 6–31
 - NONCOMPRESSED option, 6–28
 - overview, 2–15
 - OVERWRITEDISK option, 6–16
 - portion selector clause, 6–25
 - SCRATCHPOOL option, 6–30
 - serial number specification, 6–30
 - tape density specification, 6–29
 - tape output, 6–34
 - tape serial number specification, 6–30
 - TAPES clause, 6–27
 - BY FAMILYINDEX option, 6–26
 - BY FAMILYINDEX option, DUMP command, 6–26
 - CORRECTREADERROR option, DUMP command, 6–15
 - DATACompression option, DUMP command, 6–15
 - density specification, 6–29
 - disk requirements for database backups, 6–31
 - dump disk specification, 6–31
 - dump list clause, DUMP command, 6–22
 - dump selector clause, DUMP command, 6–24
 - dump tape specification, 6–27
 - DUMPDISKSIZE option, DUMP command, 6–15
 - ENCRYPTTYPE option, 6–15
 - ENCRYPTTYPE option, DUMP command, 6–15
 - exclude list clause, DUMP command, 6–24
 - INCREMENTAL option, DUMP command, 6–18
 - NOCOMPARE option, 6–14
 - NOENCRYPT option, 6–15
 - NOENCRYPT option, DUMP command, 6–15

- OFFLINE option, 6–16
- OVERWRITEDISK option, DUMP
 - command, 6–16
- portion selector clause, DUMP command, 6–25
- syntax, 6–9
- tape requirements for database backups, 6–27
- WAITTIME option, 6–16
- WAITTIME option, DUMP command, 6–16
- WORKERS option, 6–12
- dump directory
 - adding entries, 6–62
 - adding entries for duplicates of dumps, 6–51
 - building, 6–68
 - ADD command, 6–62
 - adding entries to a dump directory, 6–62, 6–65
 - disabling, 6–62
 - dump information, 6–63
 - dump information for a family index, 6–64
 - enabling use of, 6–61
 - identifying dumps, 6–65
 - limiting the information stored, 6–62
 - listing, 6–53
 - locked row information, 6–64
 - main directory, 6–64
 - maintaining, 6–57
 - program, 6–60
 - program syntax, 6–60
 - rows with read errors, 6–64
- BUILDDUMPDIRECTORY command, 6–68
- DELETE command, 6–62
 - deleting entries, 6–63
- DISABLE command, 6–62
 - dump information, 6–63
- ENABLE command, 6–61
 - inserting entries in a dump directory, 6–62
 - locked row information, 6–64
 - locked row information, generating, 6–64
 - main directory, 6–58, 6–64
 - PACKNAME option, 6–62
 - read error information, generating, 6–64
 - recovering, 6–69
 - SYSTEM/DMDUMPDIR program, 6–57
- TAPEDIRECTORY command, 6–53
- DUMP statement, 2–15
- DUMPDIR option
 - DUPLICATEDUMP command, 6–51
- dumping databases
 - limitations, 4–6
 - warnings, 4–16
 - input/output errors, 4–12
 - media selection, 4–5
 - omitting tape options, 4–6
 - overview, 2–15
 - restarting, 4–8
 - tape input/output errors, 4–11
- dumps
 - copying, 6–45
 - duplicating, 6–45
 - listing information, 6–63
 - retaining information about, 6–62
 - verifying, 6–44
 - COPYDUMP command, 6–46
 - copying backup dumps, 6–45
 - duplicating backup dumps, 6–45
 - offline, recovering from failure, 5–30
 - permanent directory databases, 16–4
 - printing dump information, 6–63
- DUPLICATEDUMP command
 - destination dump tape clause, 6–51
 - source dump tape clause, 6–51
 - destination dump disk clause, 6–52
 - DUMPDIR option, 6–51
 - examples, 6–52
 - FORWARD COMPARE option, 6–51
 - options clause, 6–51
 - source dump disk clause, 6–52
 - syntax, 6–50
 - WORKERS option, 6–51
- duplicating backup dumps
 - destination dump tape clause, 6–51
 - DUPLICATEDUMP command, 6–46
 - source dump tape clause, 6–51
 - destination dump disk clause, 6–52
 - DUMPDIR option, 6–51
 - examples, 6–52
 - FORWARD COMPARE option, 6–51
 - options clause, 6–51
 - source dump disk clause, 6–52
 - storing information in dump directory, 6–51
 - syntax, 6–50
 - WORKERS option, 6–51
- dynamic database parameters, 2–3, 2–8

E

encryption algorithms
 AES, 15–2
 TDES, 15–2

END command
 BYE command, Database Certification program, 11–10
 Database Certification, 11–10

Enterprise Database Server
 errors during halt/load recovery, 24–2

entry point parameters
 AUDIT_CLOSE, 20–15
 AUDIT_CLOSE entry point parameters, 20–15
 AUDIT_NEXT_ABSN, 20–16
 AUDIT_NEXT_ABSN entry point parameters, 20–16
 AUDIT_NEXT_RECORD, 20–18
 AUDIT_NEXT_RECORD entry point parameters, 20–18
 AUDIT_OPEN, 20–13
 AUDIT_OPEN entry point parameters, 20–13
 AUDIT_RANDOM_ABSN, 20–17
 AUDIT_RANDOM_ABSN entry point parameters, 20–17

entry points
 declaring, 3–2
 DMEXCEPTIONNAME, 3–1
 DMEXCEPTIONTEXT, 3–2
 DMSTRUCTURENAME, 3–1
 ALGOL entry point example, 3–4
 DMAuditLib library, 20–1, 20–12
 DMSUPPORT library, 3–1
 example programs, 3–3

equation specification
 criteria for equation, 17–8
 in LOADDUMP, 17–8
 mapping, 17–9
 rules for, 17–8

errors
 during disk stream loads, 4–13
 during halt/load recovery, 24–2
 during tape loads, 4–12
 effect on copying audit files, 9–16
 messages for COPYAUDIT program, C–1
 and warnings in Visible DBS, 12–2
 RETRYIO messages, 4–13
 returning structure name, using DMSTRUCTURENAME entry point, 3–1

COPYAUDIT program, 9–16
 during disk dumps, 4–12
 during halt/load recovery, 24–1
 during tape dumps, 4–11
 handling in DMUTILITY, 4–11
 results in DMAuditLib library, 20–20
 returning exception category, using DMEXCEPTIONNAME entry point, 3–1
 returning text, using DMEXCEPTIONTEXT entry point, 3–2

exception
 DMEXCEPTIONNAME entry point, 3–1
 DMEXCEPTIONNAME entry point, 3–1
 DMEXCEPTIONTEXT entry point, 3–2
 DMSTRUCTURENAME entry point, 3–1
 for COPYAUDIT program, C–1
 returning the structure name by using the DMSTRUCTURENAME entry point, 3–1

exclusive
 access to database during reorganization, 7–55

exclusive functions, 2–7

F

failures, during
 disk dumps, 4–12
 tape dumps, 4–11
 disk loads, 4–13
 I/O errors disk dumps, 4–12
 I/O errors tape dumps, 4–11
 tape loads, 4–12

families
 assigning by using DMCONTROL, 5–2
 altering families by using DMCONTROL, 5–2
 resetting the change bit, 5–2
 changes and control file recovery, 5–28
 changing families by using DMCONTROL, 5–2
 family assignments, changing by using DMCONTROL, 5–2
 LOCKEDFILE file attribute, 5–3
 name for structures, 2–6

FAMILY command
 DMCONTROL, 5–23

field audit selection
 syntax, 10–20
 introduction, 10–20

- record type audit selection, 10–21
- file attributes for audit files, 2–4
- files
 - compatibility, 2–8
 - guard, 2–4
 - initializing, 5–30
 - listing contents, 11–30
 - partition files, row recovery, 8–18
 - partition, recovering rows, 8–18
 - WRITE statement in DMUTILITY, 11–30
- flags
 - recovery, 2–7
 - structure state, 2–5
 - tape directory, 2–4
- format
 - level in control file, 2–2
 - timestamp for structures, 2–5
 - WRITE command, 11–31
 - block range, specifying, 11–31
 - hex block address, specifying, 11–32
 - LIST command, 11–31
 - Visible DBS program, 12–1
- FORWARD COMPARE option
 - COPY command, 9–25
 - DUMP command, 6–13
 - DUPLICATEDUMP command, 6–51
 - QUICKCOPY command, 9–22

G

- global transactions, using, 1–5
- guard files, 2–4, 2–5

H

- halt/load recovery
 - messages, 24–2
 - remedies for failed, 24–1
- halt/load recovery program, overview, 2–11
- HELP command, designating in
 - Database Certification, 11–6
- host language commands
 - OPEN INQUIRY, 2–11
 - OPEN UPDATE, 2–11

I

- identifier for partitions, 2–6
- indicating
 - number of tape drives during a copy, 8–55

- number of tape drives used during a recovery, 8–8
- number of tasks during verification process of a recovery, 8–10
- INITIALIZE command
 - DMCONTROL, 5–7
- INITIALIZE statement
 - DMUTILITY ALL option, 5–33
 - DMUTILITY = option, 5–33
 - DMUTILITY designating structures, 5–33
 - DMUTILITY syntax, 5–32
- failure recovery, 2–14, 5–30
 - options, 5–33
 - backing up a database, 6–1
 - dumping databases, 6–1
 - MIGRATEDB Command, 5–36
 - REDISTRIBUTE command, 5–34
- initializing
 - structures, syntax, 5–33
- inquiring, OPEN INQUIRY host language command, 2–11
- interfaces, control file, 2–10
 - Accessroutines, 2–10
 - DASDL compiler, 2–10
- interlock control, 2–7
- INTERNAL FILES
 - option, BUILDREOGRDG utility, 7–47
- internationalizing messages, 1–4
- intervals for PRINTAUDIT output
 - relative block syntax, 10–14
 - serial number syntax, 10–13
 - time syntax, 10–12
 - relative block, 10–14
 - relative block intervals for PRINTAUDIT output, 10–14
 - serial number, 10–13
 - serial number intervals for PRINTAUDIT output, 10–13
- in-use bit
 - effect of opening the database, for inquiry, 2–11
 - effect of opening the database, for update, 2–11
- I/O errors
 - database disk, 4–13
 - databases, 4–13
 - disk stream loads, 4–13
 - tape loads, 4–12
 - disk stream loads, I/O errors, 4–13
 - disks, I/O errors, 4–13
 - loading information, I/O errors, 4–13

reorganization, 7–83

L

large objects

consolidating space of deleted objects,
13–3

LOBSQUASH command, DMUTILITY,
13–3

maintaining databases with, 13–1
consolidating, 13–3

consolidating large object tank data sets,
13–3

LOBCOMBINE command, DMUTILITY,
13–3

quiesced database, using, 14–1

last assigned partition number, control file,
2–4

limitations

reorganization processes, 7–51

dump numbers, 6–62

reorganization, 7–91

TASKLIMIT phrase in reorganization, 7–51

LIST statement in DMUTILITY

overview, 2–15

LOADDUMP

COBOL74 MOVE algorithm, with
examples, 17–12

file specification, 17–5

load dump specification, 17–11

compiler control options, 17–12

database specification, 17–5

database specification in LOADDUMP,
17–5

DATASET clause, LOADDUMP, 17–5

DB clause, LOADDUMP, 17–5

DUMP option, LOADDUMP, 17–11

equation specification, 17–8

EXCLUDE clause, LOADDUMP, 17–5

FD name for LOADDUMP file, 17–6

file specification, LOADDUMP, 17–5

load dump specifications, 17–11

LOAD option, LOADDUMP, 17–11

RECORD clause, LOADDUMP, 17–6

RUN statement, 17–3

SOURCE clause, LOADDUMP, 17–6

steps for using LOADDUMP, 17–2

VIA <index structure> construct,
LOADDUMP, 17–11

localizing messages, 1–4

locked control files

unlocking after a dump, 6–17

unlocking, 2–14

unlocking control files, syntax, 5–29

unlocking syntax, 5–29

LOCKEDFILE command

DMCONTROL, 5–10

LOCKEDFILE file attribute

database backup, 6–5

DMUTILITY program, 5–31

COPY command, 9–10

copying audit files, 9–9

QUICKCOPY command, 9–9

M

main directory

listing, 6–64

listing locked row information, 6–64

listing rows with read errors, 6–64

printing locked row information, 6–64

printing rows with read errors, 6–64

writing, 6–64

writing locked row information, 6–64

printing, 6–64

writing rows with read errors, 6–64

MAXUPDATEPERTR command

DMCONTROL, 5–17

MEMORY RESIDENT

option in STRUCTURE CHANGE

command, 12–33

STATUS HISTORY command, Visible DBS
program, 12–34

messages

exception category by using
DMEXCEPTIONNAME entry
point, 3–1

for COPYAUDIT program, C–1

structure name by using the
DMSTRUCTURENAME entry
point, 3–1

translating, 1–4

common syntactic items, A–1

database statement, syntax, A–1

db statement, syntax, A–1

digit, syntax, A–2

dump name, syntax, A–2

family name, syntax, A–2

file name, syntax, A–3

halt/load recovery, 24–2

identifier syntax, A–3

- integer, syntax, A-2
- path name, A-4
- range, syntax, A-5
- reports, statistics, B-1
- string6, syntax, A-5
- string, syntax, A-5
- tape name, syntax, A-4
- text by using DMEXCEPTIONTEXT entry
 - point, 3-2
- time, B-1
- multidump tapes
 - creating and accessing, 4-5
 - listing, 6-54
 - recreating fast access directory file, 6-54

N

- naming convention for audit file names
 - COPY command, 9-10
 - QUICKCOPY command, 9-8
- No File condition
 - RoboHost units, effect on PRINTAUDIT program, 8-42, 10-5
 - running PRINTAUDIT program, 8-42, 10-5
 - TAPESERVER system option, effect on PRINTAUDIT program, 8-42, 10-5
- NOZIP option
 - partial database recovery, 8-6
 - recovering the database, 8-6
 - row recovery, 8-6
 - whole database recovery, 8-18
- number of
 - forced overlays in buffer statistics, B-4
 - last assigned partition, 2-4
 - I/O statistics, B-4
 - partition structures, 2-6

O

- offline
 - certification failure, recovering from, 5-30
 - copy failure, recovering from, 5-30
 - recovering from failure, 5-30
 - storing the last audit file, 6-17
 - canceling offline certification, 5-30
 - canceling offline copies, 5-30
 - certification, offline, recovering from failure, 5-30
 - copy offline, recovering from failure, 5-30
 - selecting, 6-4

- OFFLINE option
 - BUILDREORG utility, 7-56
 - DUMP command, 6-16
- OFFLINE reorganization
 - OFFLINE FIXUP Exchange error, 7-86
 - status of reorganization, displaying, 7-86
- online dump
 - disadvantages for disk dumps, 4-7
 - selecting, 6-4
 - process, 6-17
- Open Distributed Transaction Processing
 - overview, 1-5
- OPEN INQUIRY, host language command, 2-11
- open options, BUILDREORG utility
 - combinations, 7-57
- OPEN UPDATE, host language command, 2-11
- opening databases
 - during backups, 6-4
 - during reorganizations, 7-53
 - inquiry only, 7-54
 - no access allowed, 7-55
 - OPEN INQUIRY host language command, 2-11
 - OPEN UPDATE host language command, 2-11
 - unaudited databases, availability during dump, 6-4
- ordered data sets
 - crosscheck certification options for, 11-17
 - internal certification options for, 11-16
 - NUMSUBBLOCKS field, 11-17
- output
 - disk requirements for database backups, 6-31
 - for dumps, choosing, 6-4
 - tape requirements for database backups, 6-27
 - description of Database Certification, 11-4
 - disk dumps, 6-4
 - dump media, selecting, 6-4
 - tape dumps, 6-4
- output parameter
 - BLOCK_OFFSET, 20-12
 - Block_Offset output parameter, 20-12
 - RECORD_OFFSET, 20-12
 - Record_Offset output parameter, 20-12
- OVERRIDE AUDITBUFFERS comamnd
 - DMCONTROL, 5-19

OVERRIDE AUDITSECTIONS command
 DMCONTROL, 5–20
 OVERRIDE FAMILY command
 DMCONTROL, 5–20
 OVERRIDE GUARDFILETITLE command
 DMCONTROL, 5–21
 OVERRIDE HL command
 DMCONTROL, 5–20
 OVERRIDE POPULATIONWARN command
 DMCONTROL, 5–21
 OVERWRITE command
 DMCONTROL, 5–8
 modifying the control file, 5–9

P

pack name for structures, 2–6
 parameters
 dynamic, 2–3
 dynamic database, 2–8
 partial database recovery
 FLUSHDB option, 8–8
 partition files, 8–18
 recovery specification, 8–13
 FLUSHDB option, 8–8
 in an RDB environment, 8–7
 NOZIP option, 8–7, 8–8
 Quickfix process, 8–12
 recover specification, 8–13
 steps to avoid extra processing, 8–7
 using the NOZIP option, 8–6
 WORKERS option, 8–8
 partition directory descriptor, 2–3
 partitioned records
 audit and recovery considerations, 19–3
 directory overview, 19–1
 in the control file, 2–6
 audit and recovery considerations for
 partitioned records, 19–3
 audit reader library, 20–1
 controlling, 19–1
 directory details, 19–2
 DMAuditLib library, 20–1
 partitions
 identifier in the control file, 2–6
 identifying in Database Certification,
 11–11
 last assigned number, 2–4
 structure number in the control file, 2–6
 permanent directory database
 replication, 8–26
 permanent directory databases
 creating, 16–1
 LOADDUMP, 17–1
 reorganizing, 16–3
 working with dumps, 16–4
 PRINTAUDIT program
 block zero information, 10–9
 by block number, 10–18
 by field, 10–20
 by mix number, 10–16
 by program identifier, 10–17
 by record type, 10–21
 command syntax, 10–32
 customized PRINTAUDIT program, 10–35
 designating intervals, 10–9
 developing the code, 10–28
 directing output, 10–7
 examples, 10–32
 formatting output, 10–9
 identifying audit files for analysis, 10–4
 initiating, 10–2
 initiating batch mode, 10–3
 initiating interactively, 10–3
 introduction, 10–26
 overview, 10–1, 10–2
 PRINTIT variable, 10–28
 quick reference, 10–39
 relative block intervals, 10–14
 RoboHost units, 8–42, 10–5
 SELECT command, 10–31
 selecting audit data, 10–15
 serial number intervals, 10–13
 stopper pattern information, 10–29
 stopping, 10–8
 syntax, 10–6
 USERPROCEDURE procedure, 10–28
 USERWRAPUP procedure, 10–28
 by stack number, 10–15
 by structure name, 10–18
 by structure number, 10–18
 command syntax, 10–34
 compiling, 10–27
 extra output, 10–9
 file equations, 10–27
 identifying DASDL source files, 10–5
 partial audit files, 10–9
 TAPESERVER system option, 8–42, 10–5
 time intervals, 10–12
 variables, 10–28
 printing audit files
 batch mode, 10–3

- block zero information, 10–9
- by block number, 10–18
- by field, 10–20
- by mix number, 10–16
- by program identifier, 10–17
- by record type, 10–21
- by stack number, 10–15
- by structure name, 10–18
- by structure number, 10–18
- compiling, 10–27
- customizing the PRINTAUDIT program, 10–26
- designating intervals, 10–9
- developing the code, 10–28
- directing output, 10–7
- ending the PRINTAUDIT program, 10–8
- examples, customized PRINTAUDIT program, 10–35
- examples, PRINTAUDIT commands, 10–32
- extra output, 10–9
- file equations, 10–27
- formatting output, 10–9
- identifying audit files for analysis, 10–4
- initiating the PRINTAUDIT program, 10–2
- interactive mode, 10–3
- alphanumeric audit file information, generating, 10–9
- overview, 10–1, 10–2
- partial audit records, 10–9
- PRINTIT variable, 10–28
- quick reference, 10–39
- relative block intervals, 10–14
- RoboHost units, 8–42, 10–5
- SELECT command, 10–31
- selecting audit data, 10–15
- serial number, 10–13
- serial number intervals, 10–13
- stopper pattern information, 10–29
- syntax, 10–6
- tape directory, 9–26
- TAPESERVER system option, 8–42, 10–5
- time intervals, 10–12
- USERPROCEDURE procedure, 10–28
- USERWRAPUP procedure, 10–28
- variables, 10–28
- block number audit selection, 10–18
- block zero information, printing, 10–9
- hexadecimal audit file information, generating, 10–9
- identifying DASDL source files, 10–5

- mix number audit selection introduction, 10–16
- mix number audit selection syntax, 10–17
- mnemonics for audit records, 10–22
- monitoring commands, Visible Recovery, 8–42
- partial audit records, printing, 10–9
- PRINTAUDIT program, 10–1
- program identifier audit selection, 10–17
- selecting audit data for PRINTAUDIT output, 10–15
- stack number audit selection, introduction, 10–15
- stack number audit selection, syntax, 10–16
- stopper pattern information, 10–29
- tuning commands, Visible Recovery, 8–42
- USERPROCEDURE procedure, using to customize PRINTAUDIT, 10–28
- USERWRAPUP procedure in PRINTAUDIT program, 10–28
- variables available for PRINTAUDIT program, 10–28
- Visible Recovery commands, 8–42
- printing dump information for a family index, 6–64
- processes
 - for backing up a database, 6–3
 - rollback recovery, 2–11
- programs
 - ABORT recovery, 2–11
 - Database Certification, discontinuing, 2–17
 - DMCONTROL, running, 5–1
 - DMDATARECOVERY, 2–11
 - halt/load recovery, 2–11
 - REORGANIZATION, overview, 2–16
 - DMUTILITY, control file interface, 2–13
 - example illustrating entry points, 3–3

Q

- QDCVERIFY option
 - COPY statement, 8–56
 - RECOVER statement, 8–10
- QDCWORKERS option
 - COPY statement, 8–56
 - RECOVER statement, 8–10
 - FAMILYINDEX specification, COPY statement, 8–57

- QUICKCOPY command, COPYAUDIT
 - program
 - overview, 9–6
 - quick-reference information
 - COPYAUDIT statement, 9–29
 - PRINTAUDIT statement, 10–39
 - quiesce database
 - creating incremental/accumulated
 - dumpse, 14–21
 - quiesce database copy
 - restoring original configuration, 14–6
 - restoring the control file to the live
 - environment, 14–31
 - as a copy source, 14–22
 - as a recovery source, 14–22
 - CFRESTORE command, DMUTILITY,
 - 14–31
 - CREATE QDC command, DMCONTROL,
 - 14–12
 - QDCVERIFY option for copy, 8–56
 - QDCVERIFY option for recovery, 8–10
 - QDCWORKERS option, 8–56
 - QDCWORKERS option for recovery, 8–10
 - QUIESCE HISTORY option, DMUTILITY
 - WRITE command, 14–30
 - QUIESCE QDC command, DMUTILITY,
 - 14–6
 - restoring control file of live database using
 - quiesce database copy, 14–31
 - quiesce database copy recovery source
 - QDCVERIFY option, 8–10
 - QDCWORKERS option, 8–10
 - QUIESCE DB recovery sources
 - VERIFY option, 8–9
 - QUIESCEDBRESET command
 - DMCONTROL, 5–26
- R**
- random data sets
 - internal certification options for, 11–18
 - REBLOCK
 - option in STRUCTURE CHANGE
 - command, 12–28
 - REBLOCK RESET
 - option in STRUCTURE CHANGE
 - command, 12–28
 - REBLOCK SET
 - option in STRUCTURE CHANGE
 - command, 12–28
 - REBLOCKFACTOR
 - BUFFERS option, STRUCTURE CHANGE
 - command, 12–32
 - option in STRUCTURE CHANGE
 - command, 12–28
 - POPULATIONINCR option, in
 - STRUCTURE CHANGE command,
 - 12–28
 - POPULATIONWARN option, STRUCTURE
 - CHANGE command, 12–31
 - REBUILD process
 - continuing, 4–10
 - and reorganizations, 7–80
 - whole database recovery, 8–18
 - time point, for recovery, 8–24
 - whole database recovery, 8–18
 - RECONSTRUCT program
 - compiling Enterprise Database Server
 - software, 18–3
 - continuing, 4–10
 - DMINTERPRETER, compiling Enterprise
 - Database Server software, 18–4
 - records
 - creation timestamp, 2–5
 - family name, 2–6
 - partition, 2–6
 - preallocating for direct data sets, 5–33
 - state flag, 2–5
 - version timestamp, 2–5
 - format timestamp, 2–5
 - guard file, 2–5
 - partitioned, 19–1
 - structure, 2–5
 - RECOVER INITIALIZE command
 - DMCONTROL, 5–19
 - RECOVER PARTITIONS command
 - DMCONTROL, 5–18
 - RECOVER statement
 - BYCYCLE option, 8–9
 - DMUTILITY, 8–1
 - FLUSHDB option, 8–8
 - overview, 2–15
 - QDCVERIFY option, 8–10
 - VERIFY option, 8–9
 - WORKERS option, 8–8
 - examples, 8–30
 - QDCWORKERS option, 8–10
 - VERIFYTASKS option, 8–10
 - VERIFYTASKS option, RECOVER
 - statement, 8–10

- RECOVER UPDATE command
 - DMCONTROL, 5–17
- RECOVER UPDATE statement
 - potential problems, 5–26
 - RECOVERTPSINFO option, problems, 5–26
- recovery
 - ABORT program, 2–11
 - audit files, using, 9–10
 - control file, family changes, 5–28
 - control file, procedure, 5–28
 - database, 2–11
 - DMDATARECOVERY program, 2–11
 - flag, 2–7
 - halt/load program, 2–11
 - partial database recovery of partition files, 8–18
 - RECOVER statement, 2–15
 - restrictions for tape dumps, 8–28
 - row recovery, 8–6
 - row recovery using the NOZIP option, 8–6
 - whole database, 8–18
 - continuing, 4–10
 - control file, 5–27
 - database, D–3
 - deadlock situations, preventing, 2–7
 - DMUTILITY COPY statement, 8–51
 - DMUTILITY RECOVER statement, 8–1
 - partial database recovery, 8–6
 - QUICKCOPY audit files, using, 9–9
 - RESTORE option recovery, 8–6
 - rollback process, 2–11
 - row recovery using the NOZIP option, 8–6
 - running, 8–40
 - synchronized TPS, 2–8
- relative block intervals for PRINTAUDIT
 - output
 - syntax, 10–14
- reloading control file, 2–14
- reloading database, 2–14
- Remote Database Backup facility
 - overview, 1–5
- reorganization
 - OFFLINE option, 7–56
 - improving performance, 7–87
 - permanent directory databases, 16–3
- REORGANIZATION program
 - overview, 2–16
 - SORT errors, 7–85
 - certifying the database, discontinuing, 2–17
 - finishing, 7–76
 - nonusercoded databases, reorganizing, 7–69
 - phases, 7–67
 - preparation tasks, 7–67
 - RECESS directive, 7–78
 - reorganization status report, 7–77
 - running, 7–66
 - starting, 7–68
- reorganizing the database
 - compiler control options, 7–22
 - explicitly generating structures, 7–37
 - GENERATE statement, 7–29
 - generating from index structures, 7–34
 - INTERNAL FILES phrase, 7–47
 - algorithm, 7–15
 - ALLOWEDCORE phrase, 7–49
 - OFFLINE option, 7–16
 - record format conversion, 7–3, 7–17
 - REORGDB option, 7–17
 - SORT phrase, 7–39
 - availability of structures during reorganization, 7–70
 - backing up files, 7–15
 - BUILDREORG utility, 7–20
 - central data set control options, 7–46
 - Central Data Set GENERATE statement, 7–29
 - central data set sequence statement, 7–51
 - central data set sequence statement in reorganization, 7–51
 - data errors, 7–84
 - disk storage requirements, 7–89
 - disk storage requirements for reorganization, 7–89
 - DMSUPPORT library file name changes, 7–7
 - effect of TPS, 7–69
 - EXCLUSIVE option, 7–55
 - EXTRACT phrase, 7–39
 - file format conversion, 7–3
 - garbage collection, 7–2
 - generating a data set, 7–30
 - generating from a data set, 7–33
 - index control option, 7–37
 - INQUIRYONLY option, 7–54
 - introduction, 7–1
 - I/O errors, 7–83
 - limitations, 7–91
 - LOADFACTOR phrase, 7–38

- MAXWAIT, 7-71
- nonusercoded databases, 7-69
- OFFLINE option, 7-56
- ONLINE option, 7-15
- ONLINE reorganization, 7-15
- open options, 7-53
- order of processes, 7-51
- ORDERBYCORE phrase, 7-50
- performance enhancement, 7-87
- preparation tasks, 7-67
- PREVERIFY option, 7-55
- procedure sequence option, 7-45
- process, 7-5
- rebuild recoveries, 7-80
- record format conversion, 7-4, 7-5
- reorg global control statement, 7-53
- reorganization order, 7-45
- REORGANIZATION program, 2-16
- REORGANIZATION program, running, 7-66
- REORGDB memory usage, 7-19
- resource control, 7-37
- restarting, 7-79
- rollback recoveries, 7-83
- running, 7-66
- SORT errors during reorganization, 7-85
- space requirements for reorganization, 7-89
- starting, 7-68
- status, 7-86
- storage requirements for reorganization, 7-89
- structure availability, 7-70
- structure COPY TO option, 7-40
- swap phase, database reorganization, 7-73
- syntax, 7-24
- TASKLIMIT phrase, 7-51
- types of reorganization, 7-2
- UPDATE option, 7-27
- USEREORGDB option, 7-17
- REORGDB control
 - CAPTUREUPDATETRAN phrase, BUILDREORG utility, 7-43
 - COPY TO option, 7-41
- REORGDB reorganization
 - recessing, 7-78
 - AUTOSWAP option, 7-64
 - CAPTUREUPDATETRAN phrase, 7-43
 - MAXUPDATERS phrase, 7-43
 - REORGDB control COPY TO option, 7-41
 - REORGBALLOWEDCORE option, 7-66
 - REORGBTITLE phrase, 7-63
 - STRCOPYCORE phrase, 7-44, 7-65
 - terminating, 7-78
 - TOTALCOPYCORE phrase, 7-44, 7-65
 - TOTALCOPYCORE phrase, BUILDREORG utility, 7-44, 7-65
 - USEREORGDB option, 7-61
 - USEREORGDB option, BUILDREORG utility, 7-61
- replication
 - permanent directory database, 8-26
- requesting
 - number of processes for a quiesce database copy copy, 8-56
 - number of processes for a quiesce database copy recovery, 8-10
 - preverification of data during a recovery, 8-9
 - tasks for each cycle during a copy, 8-55
 - tasks for each cycle during a recovery, 8-9
 - verification for a quiesce database copy copy, 8-56
 - verification for a quiesce database copy recovery, 8-10
- restart data sets
 - internal certification options for, 11-19
 - crosscheck certification option for, 11-19
- restarting
 - DMUTILITY tape dumps, 4-8
 - Database Certification, 11-4
 - DMUTILITY disk dumps, 4-8
 - reorganizations, 7-79
- restrictions
 - recovery, 8-28
 - certifying the database, restrictions, 11-13
 - disadvantages of online disk dumps, 4-7
 - incompatibility problems for disk dumps, 4-7
 - online disadvantages, 4-7
 - resource contention, 4-7
 - resource contention for disk dumps, 4-7
 - tape incompatibility, 4-7
 - tape incompatibility for disk dumps, 4-7
 - use of Database Certification, 11-13
 - usercode, 4-7
 - usercode limits, for disk dumps, 4-7
- RETRYIO error messages
 - examples, 4-15
- rollback process
 - continuing, 4-10

- and reorganizations, 7–83
- recovery process, 2–11
- whole database recovery, 8–18
- rollover of the ABSN, 2–10
- row recovery
 - EXTRACT option, filtering, 8–11
 - FLUSHDB option, 8–8
 - partition files, 8–18
 - Quickfix process, 8–12
 - recover specification, 8–13
 - recovering the database, 8–6
 - recovering the database using the NOZIP option, 8–6
 - EXTRACT option, filter option, 8–11
 - Quickfix process, example of row recovery, 8–12
 - Quickfix process, row recovery, 8–12
 - RESTORE option recovery, 8–13
 - WORKERS option, 8–8

S

- SCRATCHPOOL option
 - DUMP command, 6–30
- SCRATCHPOOL option, requesting for copying audit files, 9–21
- SECAUDITFAMILY command
 - DMCONTROL, 5–24
 - <security file title change>, 5–25
- secondary audit files
 - copying as primary audit files, 9–17
- security file title change command
 - DMCONTROL, 5–25
- SELECT command in PRINTAUDIT program
 - syntax, 10–31
 - using to customize PRINTAUDIT program, 10–31
 - defining the interval, 10–31
 - options, 10–32
- selecting audit data for PRINTAUDIT output
 - by block number, 10–18
 - by field, 10–20
 - by mix number, 10–16
 - by program identifier, 10–17
 - by record type, 10–21
 - by stack number, 10–15
 - by structure name, 10–18
 - by structure number, 10–18
- SENSITIVEDATA command
 - DMCONTROL, 5–11
- state flag for structures, 2–5

- statistics
 - audit, B–13
 - database usage, B–9
 - transaction, B–15
 - audit, B–12
 - audit statistics, B–12, B–13
 - buffer, B–2
 - buffer statistics, B–2
 - global lock, B–16
 - global lock statistics, B–16
 - guidelines for use, B–18
 - header, B–1
 - interpreting database, B–1
 - interpreting database statistics, B–1
 - I/O, B–4
 - reports, statistics, 8–50
 - structure lock, B–11
 - structure lock statistics, B–11
 - transaction statistics, B–15
 - Visible Recovery statistics report, 8–50
 - VSS2 optimization, B–9
- STRUCTURE command
 - DMCONTROL, 5–23
- structure directory descriptor, 2–3
- structures
 - attributes kept in the control file, 2–6
 - availability during reorganization, 7–70
 - access statistics, B–5
 - copying during reorganizations, 7–40
 - creation timestamp, 2–5
 - directory description, 2–5
 - directory of structures, 2–5
 - explicitly generating, 7–37
 - family name, 2–6
 - format timestamp, 2–5
 - name audit selection, introduction, 10–18
 - name, specifying, Database Certification, 11–11
 - numbers for partitions, 2–6
 - records, 2–5
 - recovering from failure, 5–30
 - returning the name in an exception, 3–1
 - specifying, WRITE command, 11–31
 - state flag, 2–5
 - syntax, 5–33
 - version timestamp, 2–5
 - control file, 2–2
 - copying during reorganization, 7–41
 - initializing, 5–30
 - number audit selection introduction, 10–18

- numbers, specifying, Database Certification, 11–11
 - partition name, specifying, 11–31
 - section specifying, LIST command, 11–31
 - section specifying, WRITE command, 11–31
 - specifying, LIST command, 11–31
 - usage statistics, B–9
 - VSS2 optimization, B–9
 - SYMBOL/DMAUDITLIB file
 - declaring the DMAuditLib library, 20–2
 - synchronized recovery, TPS, 2–8
 - SYSTEM/DMDUMPDIR program
 - disabling, 6–62
 - examples, 6–65
 - fatal error, 8–16
 - disabling use of a dump directory, 6–62
 - enabling, 6–61
 - enabling use of a dump directory, 6–61
- T**
- table of contents, control file, 2–2
 - tailored software, compatibility checking, 2–8
 - tailoring the PRINTAUDIT program
 - examples, 10–35
 - tape directory flag, 2–4
 - tape dumps
 - contrasting with disk dumps, 4–5
 - forward comparison, 6–13
 - input/output errors, 4–11
 - no comparison, 6–14
 - number of tape workers, 6–12
 - warnings, 4–16
 - directory, 6–57
 - media selection for database dumps, 4–5
 - multidump tapes; specification, 6–31
 - multiple dumps to a single backup tape, 6–31
 - NOCOMPARE option, DUMP command, 6–14
 - operator interface, 4–8
 - restarting, 4–8
 - verifying, 6–45
 - tape encryption
 - algorithms, 15–2
 - architecture, 15–1
 - DASDL syntax, 15–3
 - DMUTILITY syntax, 15–3
 - Advanced Encryption Standard (AES), 15–2
 - AES (Advanced Encryption Standard), 15–2
 - COPYAUDIT syntax, 15–4
 - Data Encryption Standard (DES), 15–2
 - DES (Data Encryption Standard), 15–2
 - examples, 15–5
 - tape serial number specification
 - DUMP command, 6–30
 - tape serial number, requesting for backups, 6–30
 - TAPEDIRECTORY command
 - overview, 2–14
 - syntax, 6–53
 - syntax, examples, 6–54
 - fast access directory file, 6–54
 - TAPESET DIRECTORY command, 6–54
 - TAPEDIRECTORY statement, 2–14
 - tapes
 - appending audit files, 9–11
 - multiple-reel, using with audit files, 9–6
 - compressed audit files, using with, 9–8
 - maximizing audit tape usage, 9–7
 - maximizing usage for audit files, 9–7
 - multiple audit file copies, creating, 9–6
 - multiple-reel tapes, using with audit files, 9–6
 - TAPESET numbers
 - COPYAUDIT program, 9–15
 - TAPESET specification
 - QUICKCOPY command, 9–17
 - text directory, control file, 2–4
 - text directory descriptor, 2–4
 - time intervals for PRINTAUDIT output
 - syntax, 10–12
 - timestamp of the database, 2–3
 - timestamps
 - control file, universal version, 2–8
 - control file, using to verify file compatibility, 2–9
 - mismatch error during database backup, 6–5
 - structure creation, 2–5
 - structure format, 2–5
 - structure version, 2–5
 - listing, 2–14
 - TPS, 2–4
 - Transaction Processing System (TPS)
 - effect on database reorganization, 7–69
 - conflicts with RECOVER UPDATE statement, 5–26
 - overview, 2–4

- synchronized recovery, 2–8
- transaction, using global, 1–5
- translating messages, 1–4

U

- unlocking control files, 2–14
- unordered data sets
 - internal certification options for, 11–21
 - crosscheck certification options for, 11–22
- UPDATE command
 - DMCONTROL, 5–9
- update level in the control file, 2–3
- updating OPEN UPDATE host language
 - command, 2–11
- UPPERCASE
 - command, Database Certification, 11–9
- utilities overview, 1–2
 - controlling databases, 1–3
 - initializing databases, 1–3
 - maintaining databases, 1–3
 - running, 1–3
 - verifying databases, 1–4

V

- variables
 - state variables in the control file, 2–3
- VERIFYDUMP command
 - designating a disk dump, 6–45
 - designating a tape dump, 6–45
 - WORKERS option, 6–45
 - replicating backup dumps, 6–45
- verifying
 - data during a quiesce database copy copy, 8–56
 - data during a quiesce database copy recovery, 8–10
 - data before recovery, 8–9
 - VERIFY option, RECOVER statement, 8–9
- version timestamp for structures, 2–5
- Visible DBS program
 - DBS STATUS command, 12–2
 - errors and warnings, 12–2
 - GARBAGE COLLECT command, 7–2, 12–17, 12–33
 - LOCKSTATISTICS command, 12–21
 - SNAPSHOT command, 12–21
 - STATISTICS command, 12–23
 - STATUS MIX command, 12–39
 - STRUCTURE CHANGE command, 12–27

- SUPERCP RESTOREDBFILES command, 12–44
- AUDIT CLOSE command, 12–11
- AUDIT CLOSE command Visible DBS program, 12–11
- AUDIT PROCESSOR TIMES command, 12–10
- AUDIT PROCESSOR TIMES command, Visible DBS program, 12–10
- AUDIT QUICKCOPY MAXFILESPERTAPE command, 12–14
- AUDIT QUICKCOPY MAXFILESPERTAPE command, Visible DBS program, 12–14
- AUDIT SCRATCHPOOL command, 12–13
- AUDIT SCRATCHPOOL command, Visible DBS program, 12–13
- USEREORGDB TERMINATE command, 12–45
- DBS CHANGE command, 12–5
- DBS STATUS command, Visible DBS program,, 12–2
- DONTFORCE, 12–13
- GARBAGE COLLECT command, Visible DBS program, 12–17, 12–33
- garbage collection, overview, 7–2
- LOCKSTATISTICS command, Visible DBS program, 12–21
- MAXFILESPERTAPE value, changing setting, 12–14
- online garbage collection, 12–17, 12–18, 12–33
- SNAPSHOT command, Visible DBS program, 12–21
- STATISTICS command, Visible DBS program, 12–23
- STATUS HISTORY command, 12–34
- STATUS MIX command, Visible DBS program, 12–39
- STATUS RDB command, 12–42
- STATUS RDB command in Visible DBS program, 12–42
- STATUS REORG command, 12–43
- STATUS REORG command in Visible DBS program, 12–43
- STATUS STRUCTURE command, 12–24
- STATUS STRUCTURE command, Visible DBS program, 12–24
- STRUCTURE CHANGE command, Visible DBS program, 12–27

SUPERCP RESTOREDBFILES command
 in VISIBLE DBS program, 12–44

USEREORGDB DISCARD command,
 12–45

USEREORGDB DISCARD command in
 VISIBLE DBS program, 12–45

USEREORGDB TERMINATE command in
 VISIBLE DBS program, 12–45

Visible Recovery commands

ALLOWEDCORE = <integer> command,
 8–43

OVERLAYGOAL = <decimal value>
 command, 8–44

STATISTICS CLEAR command, 8–49

STATISTICS CLEAR command, Visible
 Recovery program, 8–49

STATISTICS command, 8–49

STATISTICS command, Visible Recovery
 program, 8–49

STATUS command, 8–46

STATUS command, Visible Recovery
 program, 8–46

tuning commands, 8–42

WRITEDELAYFACTOR = <decimal value>
 command, 8–44

W

warnings

- using entry points to return information,
 3–1
- during dumps, 4–16

WFL jobs

- certifying the database, 11–3

DATABASE/WFL/COMPILEACR, 18–1

- initiating COPYAUDIT, 9–5
- initiating PRINTAUDIT, 10–4
- compiling Enterprise Database Server
 software, 18–1

DATABASE/WFL/COMPILEACR, 18–1

DATABASE/WFL/COMPILEADB, 18–1

- parameters for compiling Enterprise
 Database Server software, 18–1
- task values for Database Certification,
 11–3

whole database recovery

- NOZIP option, 8–18
- REBUILD process, 8–18
- ROLLBACK process, 8–18

WORKERS option

- COPY statement, 8–55
- DUMP command, 6–12
- DUPLICATEDUMP command, 6–51
- RECOVER statement, 8–8
- VERIFYDUMP command, 6–45
- controlling restart intervals for a recovery,
 8–8

X

XE features

- generating structures explicitly, 7–37
- migrating to, performance
 recommendations for set
 generation, 7–36
- performance recommendations for set
 generation, migrating to XE
 features, 7–36

