HUD Contract H-2073-R

TCDMS UTILITIES MANUAL

(DATA MANAGEMENT SECTION)

January 1976

DEPARTMENT OF HOUSING AND URBAN DEVELOPMENT
OFFICE OF THE ASSISTANT SECRETARY FOR
POLICY, DEVELOPMENT & RESEARCH

THE INTER-REGIONAL INFORMATION SYSTEM

Regional Information Systems Department
Lane County Courthouse
Eugene, Oregon 97401

and

Data Processing Authority
4747 East Burnside
Portland, Oregon 97215

BIBLIOGRAPHIC DATA SHEET	1. Report No. USACLCG20015	2.	3. Recipient's Accession No

4. Title and Subtitle
TCDMS Utilities Manual (Data Management Section)

5. Report Date
January 30, 1976

6.

7. Author(s)

8. Performing Organization Report No.

9. Performing Organization Name and Address
Lane County Government
Lane County Courthouse
Eugene, Oregon 97401

10. Project/Task/Work Unit No.

11. Contract/Grant No.
H-2073-R

12. Sponsoring Organization Name and Address
U.S. Dept of Housing and Urban Development
Office of Policy, Development & Research
451 7th St., S.W.
Washington, D.C. 20410

13. Type of Report & Period Covered
Special Technical Report

14.

15. Supplementary Notes

16. Abstracts

This manual is from a USAC series produced by the Regional Information Systems Department of Lane County. It contains a description of the batch utilities available with the Data Management Component of the Telecommunications Data Management System.

17. Key Words and Document Analysis 17a. Descriptors
Information System
Local Government
Computer System Programs
Data Retrieval

17b. Identifiers/Open-Ended Terms
Urban Information Systems Inter-Agency Committee
Municipal Information System
Lane County
Data Management System

17c. COSATI Field/Group 5B

18. Availability Statement Released for distribution by NTIS	19. Security Class (This Report) Unclassified	21. No. of Pages 36
	20. Security Class (This Page) Unclassified	22. Price

THE CITATION OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH EQUIPMENT OR SOFTWARE.

0.1 PREFACE

In 1972, the Data Processing Authority (representing the city of Portland and Multnomah County, Oregon) and the Regional Information Systems Department of the Lane County government (representing the cities of Eugene, Springfield, Albany, Cottage Grove, and Florence; and Lane, Linn and Benton Counties, Oregon) formed an organization called the Inter-Regional Information System (IRIS). Its purpose was manifold:

> to solve some of the complex problems of public information handling through cooperative planning and development of hardware and software environments;
>
> to minimize the duplication of effort involved in writing application systems;
>
> to reduce the cost of governmental data processing; and
>
> to increase the quality of service to the taxpayer.

Since its inception, the IRIS organization has grown to represent over one hundred different city, county, state, and federal agencies serving over 70% of Oregon's population. Current projects include the Fleet Management System, the Assessment and Taxation System, and the Telecommunications Data Management System. Future involvement is anticipated in the areas of criminal justice, management analysis, human resources, geo-coding, and financial systems.

Much of the inter-regional success enjoyed by the IRIS organization has been facilitated by a cost-reimbursement contract with the Urban Information Systems Inter-Agency Committee (USAC). USAC is a consortium of ten federal agencies formed in 1968 to work together with local governments across the United States in an effort to improve urban governance through more effective use of computer-based processing systems. USAC is sponsoring several research and development projects which will result in transferable, computerized information systems available to local governments throughout the United States.

With the support of USAC, IRIS is developing the system software foundation for the application programs which control these computerized systems. This foundation is called TeleCommunications/Data Management System (TCDMS). This system contains two components which bring together the state-of-the-art features in both telecommunications and data base/data management systems.

The telecommunications component of TCDMS extends the power of the modern computer to the desk of each user. Its facilities include such features as terminal independent I/O functions, user-specified security, multiprogramming, priority scheduling, message switching, print-out spooling, on-line debugging, and remote job entry.

The data base/data management component of TCDMS optimizes the efficiency of data file construction and minimizes data redundancy by combining all files in the system into an integrated date base. Its facilities include data access flexibility, file and data element security, and application program independence from the physical file structure.

Perhaps the most important feature of TCDMS is the transferability of application systems it allows. Application programs running under TCDMS control are isolated from changes in the hardware or software configuration of the installation. This means that TCDMS-controlled application systems can be transferred between IRIS installations without the costly conversion efforts usually necessitated by such exchanges.

TCDMS may be implemented on any IBM System 360/370 computer having 252K bytes or more of storage capacity. It will support IBM System 360/370 BAL, FORTRAN, COBOL, and DL/1 user languages. TCDMS will run in real core under the control of IBM OS or VS operating systems. The modular construction of TCDMS makes it hardware independent; it can operate with any IBM terminal hardware configuration.

The joint software development and maintenance by means of the regional and inter-regional cooperation of IRIS and the integrated data base/data communications approach of TCDMS are becoming an increasingly popular solution to the problems of information handling in the public domain.

TABLE OF CONTENTS

0.2 INTRODUCTION

This manual completes the TCDMS Utilities Manuals. It extends the descriptions of utility programs in the TCS Utility Manual to include DMS utility programs. This manual is a reference guide for data base administrators and others responsible for the creation and maintenance of a TCDMS data base. It contains information describing the TCDMS utility programs which are used in this process.

This manual is divided into two main sections. They are:

> OVERVIEW OF DMS - There is a general overview at the beginning of the manual which describes the data management component of TCDMS.
>
> DATA MANAGEMENT UTILITIES - This section contains descriptions of the TCDMS batch utility programs for the data management component. It describes the options available with these programs and illustrates sample job control statements.

The material presented in this manual is of a technical nature. It is intended for experienced computer programmers and system analysts. The user of this manual should be familiar with the physical structure of TCDMS data bases. In addition, knowledge of the retrieval and update logic of the TCDMS data base access may be useful.

For other information regarding any aspect of the data management component of TCDMS, the reader is referred to the following documents.

INTRODUCTION Continued

TCDMS DETAIL DESIGN AND DEVELOPMENT ITCR - A summary of the interrelation of DMS modules within major functional areas and the interrelation of these major areas during DMS processing. In addition, the manual includes a description of each module and its purpose.

TCDMS SYSTEM PROGRAMMER'S MANUAL - A summary of the data base definition and DBCB generation process.

TCDMS APPLICATION PROGRAMMER'S MANUAL - A description of the TCDMS data base access functions available to application programmers.

For further information about TCDMS and its documentation products please contact the Regional Information Systems Department, Lane County Courthouse, Eugene, Oregon 97401.

1.0 OVERVIEW OF DMS

1.0 OVERVIEW

The data management component of TCDMS provides to an installation both a hierarchically structured data base with extensive capabilities for inter-relating data, and a data base access system which permits application programmers to retrieve, insert, delete, or change data in the data base.

Each installation defines the structure of its data base using the data base definition and DBCB generation modules. The data files are formatted and loaded onto direct access storage devices using the DMS file load utility programs.

Data base access requests from an application program are initially handled by the request manager modules. These modules determine the nature of the request, convert the request to an appropriate format for the access method modules, and handle conversion of data from its stored format to the format desired by the user. In addition the request manager modules maintain positioning within the data base for direct or sequential access requests. The request manager modules handle data elements. Within the data base, data elements are stored in segments. The request manager modules call the segment processor modules to handle data segments. The segment processor modules manage the segment work area (SWA) where segments are constructed from data elements (for insertion) or held during data element retrieval. The segment processor also determines whether segments are pointers (which inter-relate different data base files) and performs special processing for them. The segment processor modules call the access method modules for actual data base retrievals, insertions, or deletions.

The access method modules map the complex data structures to and from their physical representations on direct access storage. The access method groups segments into data families and stores each family on one physical block on disk (if possible). The access method is designed so that the physical structuring of segments into families is independent of the technique used to access families. This means that a family may be accessed in several different ways. The access method index handling routines provide direct or sequential access by index. A family may be accessed via a pointer from another file. The access method modules also manage any buffers needed during data base physical access. They also perform the actual read and write operations to the data files on disk.

The data base integrity modules provide data base restoration facilities. They are used if a system failure has occurred. DMS also provides utility programs to write off the data base files for reorganization and reload.

2.0 DATA MANAGEMENT UTILITIES

2.1 DUAMLDDR - TCDMS FILE LOAD UTILITY

DUAMLDDR is a TCDMS utility used to create a TCDMS file and relate it logically to the rest of the data base. This may involve updating existing related files or loading entirely new related files. DUAMLDDR runs as a batch job step independent of TCDMS.

Input to DUAMLDDR consists of operating system job control language statements and a utility control statement data set. Output consists of a message data set, which includes statistical information about each of the files loaded and error messages where appropriate.

2.1.1 CONTROL STATEMENTS

A description of the required job control and utility control statements follows:

DUAMLDDR is invoked with an operating system EXEC statement with a PARM field of 'FILE=n', where n is the one to five digit decimal TCDMS file number of the primary file being loaded.

DD STATEMENTS

DUAMLDDR requires several operating system DD statements. They are listed below by ddname.

DDNAME DCTn

Ddname DCTn defines a control statement data set for TCDMS file n, where n is the five-digit TCDMS file number. One such data set is required for the primary file and for each related file being loaded. None is required for existing related files.

Each DCTn data set consists of 80 byte records containing one utility control statement per record. A control statement consists of an eight-byte keyword beginning in column one, followed by an '=' sign and an appropriate value. The control statements may appear in any order. The valid keywords are listed below.

HICHAIN#=n where n is a decimal number greater than zero and less than 256 , represents the decimal equivalent of the highest segment identifier on the file that can represent a chain segment. Any data segments on the file must have higher segment identifiers. If you do not code the HICHAIN# keyword, DUAMLDDR uses a value of 127.

INDXFREE=n where n is a one to five digit decimal number, represents the approximate amount of free space to be left in each index block. The minimum value you may specify is five plus the file's key length (as specified by the KEYLENTH keyword). The maximum is the block size minus five minus the key length. If you do not code the INDXFREE keyword, DUAMLDDR uses a value equal to ten percent of the block size. DUAMLDDR ignores this keyword if you code INDEXOPT=NO.

INDEXOPT=option where 'option' may be either 'YES' or 'NO', specifies whether an index structure is to be created. DUAMLDDR will create an index structure for the file if you code 'INDEXOPT=YES' or if you do not code this keyword at all. If you code 'NO', no index structure is built.

INTRACKS=n   where n is a decimal number, represents the number of tracks in the file that DUAMLDDR will reserve for the index structure. This value must be less than the primary space allocation for the file as specified in the DD statement used to create the file. The value you code must be large enough to accomodate the index structure when it is created. Any unused space will be reserved for expansion of the index structure due to the addition of new root segments. You must code this keyword unless you specified INDEXOPT=NO.

KEYLENTH=n   where n is a decimal number, specifies the length of the root segment for the file. It must be greater than zero and less than sixty-five. You must code this keyword.

MAXPFREE=n   where n is a decimal number, specifies the maximum amount of space that DUAMLDDR is to leave in each data block. TCDMS can use this free space during normal processing of the file to expand data families in the block. The value you specify for MAXPFREE must be less than the block size and greater than the file's key length plus twenty plus the value of MINPFREE. If you do not code the MAXPFREE keyword, DUAMLDDR uses a value equal to ten percent of the block size.

MAXSEGLN=n   where n is a decimal number, specifies the length of the largest possible segment for the file. DUAMLDDR uses this value to reserve adequate main storage for buffers. The value must be greater than zero and less than 32,768. If you do not code this keyword, DUAMLDDR uses a value of 256.

MINPFREE=n   where n is a decimal number, specifies the minimum amount of free space that DUAMLDDR will leave in each data block. TCDMS can use this free space during normal processing of the file to expand data families in the block. The value you specify for MINPFREE must be equal to or greater than zero and less than the value of MAXPFREE minus the file's key length minus 20. If you do not code the MINPFREE keyword, DUAMLDDR uses a value equal to five percent of the block size.

Performance hint: You can use MAXPFREE and MINPFREE to limit the number of data families which are split across two physical blocks. If the typical data family size is considerably less than block size, you may set MAXPFREE to a value somewhat larger than MINPFREE plus typical data family size. This will keep DUAMLDDR from splitting a data family of typical size across two blocks.

OVTRACKS=n   where n is a decimal number, specifies the number of overflow tracks that DUAMLDDR will reserve for later expansion of existing families. TCDMS uses this area during normal processing to expand existing data families when there is insufficient free space in the blocks where the families were originally loaded or created. You must code a value for this keyword. It must be equal to or greater than zero.

PRTRACKS=n   where n is a decimal number, specifies the number of tracks that DUAMLDDR will reserve for the later addition of new data families to the file. You must code a value for this keyword. It must be equal to or greater than zero.

DDNAME DLDn

Ddname DLDn defines a data set which will become TCDMS file number n, where n is a five-digit decimal number. You need to code one such DD statement for the primary file and one for each related file that will be loaded in the same job step. You do not need DD statements for existing related files. You may allocate space for these files before running DUAMLDDR, or you may allocate it in the same job step by coding 'NEW' in the DISP parameter of the DD statement. The SPACE paramenter must include a primary allocation greater than the number of tracks specified in the INTRACKS keyword of the control data set for the file. You may specify secondary allocations; DUAMLDDR will force secondary allocations until the space requirements specified in the PRTRACKS and OVTRACKS keywords are satisfied.

DDNAME DMSCATLG

Ddname DMSCATLG defines the TCDMS catalog for the data base. You must specify the DISP parameter as SHR.

DDNAME DMSIN

Ddname DMSIN defines the primary load input data set.

DDNAME DMSCHAIN

Ddname DMSCHAIN defines a temporary sequential data set. It is used to collect chain segments to be resolved in the primary file. Its format is fixed blocked. Each logical record is 22 bytes long. You must specify the block size in the DCB parameter. It must be a multiple of 22 and should be an appropriate block size for the I/O device. You should allocate enough space to contain one record for each chain segment loaded on the primary file. You may specify secondary allocations in the SPACE parameter.

DDNAME DMSOUT

Ddname DMSOUT defines a temporary sequential data set. DUAMLDDR uses it to collect segments to be put on related files. Its format is variable blocked. You must specify logical record length and block size in the DCB parameter. Logical record length may be any value less than the block size in this data set and greater than the maximum segment length plus 206. Block size should be appropriate for the I/O device. You should allocate enough space to contain one record for each segment in the load input which will be put on related files. You may specify secondary allocations in the space parameter.

DDNAME DMSPRINT

Ddname DMSPRINT defines a sequential message data set. You do not need to code DCB parameters for it.

DDNAME JOBLIB or STEPLIB

Ddname JOBLIB or STEPLIB must be included unless DUAMLDDR and all the modules it links to are on the system library. These include DUAMCHPS, DUAMCHSP, DUAMLOAD and SORT. SORT must be an alias for a sort program which accepts input in the standard form used by most IBM-developed sorts.

DDNAMES FOR SORT

Ddnames SORTLIB, SORTWK01, SORTWK02, SORTWK03, SORTWK04, SORTWK05, SORTWK06, and SYSOUT define data sets required by SORT. For information on how to specify them, consult the documentation of the SORT program you are using. You should allocate adequate work space (SORTWK01, etc.) to sort a data set the size of the DMSOUT data set.

2.1.2 DUAMLDDR EXAMPLE

In this example, three TCDMS files are loaded in one job step:

```
//LOAD35 JOB
// EXEC PGM=DUAMLDDR PARM='FILE=35'
//STEPLIB DD DSN=TCDMS.LOADLIB,DISP=SHR
//DMSCATLG DD DSN=DMS.CATALOG,DISP=SHR
//DMSIN DD DSN=LOADTAPE,UNIT=2400,VOL=SER=001937,DISP=OLD
//DMSCHAIN DD UNIT=SYSDA,SPACE=(2200,(1000,100)),
//             DCB=BLKSIZE=2200
//DMSOUT DD UNIT=SYSDA,SPACE=(2400,(4000,500)),
//             DCB=(LRECL=500,BLKSIZE=2400)
//DMSPRINT DD SYOUT=A
//DLD00035 DD UNIT=3330,VOL=SER=333001,DSN=DMS.ACCT1,
//       DISP=(,KEEP),SPACE=(TRK,(440,40))
//DCT00035 DD *
INTRACKS=25
PRTRACKS=100
OVTRACKS=40
INDXFREE=300
MINPFREE=75
MAXPFREE=200
//DLD00020  DD UNIT=3330,VOL=SER=333003,DSN=DMS.NAME,
//       DISP=(,KEEP),SPACE=(TRK,(80,10))
//DCT00020 DD *
INTRACKS=12
PRTRACKS=20
OVTRACKS=10
//DLD00021 DD UNIT=3330,VOL=SER=333013,DSN=DMS.ADDRESS,
//       DISP=OLD
//DCT00021 DD *
INTRACKS=15
PRTRACKS=5
OVTRACKS=5
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD UNIT=SYSDA,SPACE=(TRK,100)
//SORTWK02 DD UNIT=SYSKA,SPACE=(TRK,100)
//SORTWK03 DD UNIT=SYSKA,SPACE=(TRK,100)
//SORTWK04 DD UNIT=SYSKA,SPACE=(TRK,100)
//SORTWK05 DD UNIT=SYSKA,SPACE=(TRK,100)
//SORTWK06 DD UNIT=SYSKA,SPACE=(TRK,100)
//SYSOUT DD SYSOUT=A
```

EXPLANATION:

The EXEC statement specifies the execution of DUAMLDDR. The

PARM field specifies the primary file as number 35.

The STEPLIB DD statement specifies the load library containing DUAMLDDR and its submodules.

The DMSCATLG DD statement specifies the installation's DMS catalog data set.

The DMSIN DD statement specifies a data set containing the actual data to be loaded on the files.

The DMSCHAIN DD statement specifies a work data set for pointer segments.

The DMSOUT DD statement specifies a work data set for secondary data segments and pointer segments.

The DMSPRINT DD statement specifies a message data set.

The DLD00035 DD statement specifies the primary file to be loaded, file 35. In this example, space for it is to be allocated at the beginning of the job step.

The DCT00035 DD statement defines the control data set for file 35. The control statements specify certain physical characteristics of the file:

INTRACKS=25 specifies that 25 tracks are to be reserved for index area; any unused portion will be available for expansion.

PRTRACKS=100 specifies that 100 tracks are to be reserved for the expansion of prime data with new data families after the load.

OVTRACKS=40 specifies that 40 tracks are to be reserved for an overflow area. Overflow is used for the expansion of existing families.

INDXFREE=300 specifies that 300 bytes are to be left for expansion in each index block.

MINPFREE=75 and MAXPFREE=200 specifies that between 75 and 200 bytes are to be left in each data block for expansion.

DLD00020 DD and DLD00021 DD specify TCDMS files 20 and 21, respectively, as the secondary files to be loaded.

DCT00020 and DCT00021 define the control statement data set for files 20 and 21, respectively.

Note that existing files which have a secondary relation to file 35 need not have DLDn and DCTn DD statements.

SORTLIB DD, SORTLIBnn DD, and SYSOUT DD statements define the data sets required by the SORT program, which is invoked by DUAMLDDR.

2.2 DUAMRGDR - TCDMS FILE REORGANIZATION UTILITY

DUAMRGDR is the TCDMS file reorganization utility. It is used to physically restructure a file. This may be needed because more space is required for the file, or because extensive modification of the file has caused performance to deteriorate. DUAMRGDR must also be used if you wish to move a file to a different volume or to change the physical attributes of the file which were assigned when it was loaded, such as the amount of free space to be left in a data or index block, or the amount of space to be reserved for index, prime data, or overflow areas.

Input to DUAMRGDR consists of operating system job control language statements and a utility control statement data set. Output consists of a message data set, which includes statistical information about the reorganized file and error messages where appropriate.

2.2.1 CONTROL STATEMENTS

A description of the required job control and utility control statements follows.

DUAMRGDR is invoked with an operating system EXEC statement with a PARM field of 'FILE=n', where n is the one to five digit decimal TCDMS file number of the file being reorganized.

DD STATEMENTS

DUAMRGDR requires several operating system DD statements. They are listed below by ddname.

DDNAME DCTN

Ddname DCTn defines a control statement data set for TCDMS file n, where n is the five-digit TCDMS file number. One such data set is required for the primary file and for each related file being loaded. None is required for existing related files.

Each DCTn data set consists of 80 byte records containing one utility control statement per record. A control statement consists of an eight-byte keyword beginning in column one, followed by an '=' sign and an appropriate value. The control statements may appear in any order. The valid keywords are listed below.

HICHAIN#=n where n is a decimal number greater than zero and less than 256, represents the decimal equivalent of the highest segment identifier on the file that can represent a chain segment. Any data segments on the file must have higher segment identifiers. If you do not code the HICHAIN# keyword, DUAMRGDR uses a value of 127.

INDXFREE=n where n is a one to five digit decimal number, represents the approximate amount of free space to be left in each index block. The minimum value you may specify is five plus the file's key length (as specified by the KEYLENTH keyword). The maximum is the block size minus five minus the key length. If you do not code the INDXFREE keyword, DUAMRGDR uses a value equal to ten percent of the block size. DUAMRGDR ignores this keyword if you code INDEXOPT=NO.

INDEXOPT=option where 'option' may be either 'YES' or 'NO', specifies whether an index structure is to be created. DUAMRGDR creates an index structure if you code 'INDEXOPT=YES' or if you do not code this keyword at all. If you code 'NO', no index structure is built.

INTRACKS=n where n is a decimal number, represents the number of tracks in the file that DUAMRGDR will reserve for the index structure. This value must be less than the primary space allocation for the file as specified in the DD statement used to create the file. The value you code must be large enough to accomodate the index structure when it is created. Any unused space will be reserved for expansion of the index structure due to the addition of new root segments. You must code this keyword unless you specified INDEXOPT=NO.

KEYLENTH-n where n is a decimal number, specifies the length of the root segment for the file. It must be greater than zero and less than sixty-five. You must code this keyword.

MAXPFREE=n where n is a decimal number, specifies the maximum amount of space that DUAMRGDR is to leave in each data block. TCDMS can use this free space during normal processing of the file to expand data families in the block. The value you specify for MAXPFREE must be less than the block size and greater than the file's key length plus twenty plus the value of MINPFREE. If you do not code the MAXPFREE keyword, DUAMRGDR uses a value equal to ten percent of the block size.

MAXSEGLN=n where n is a decimal number, specifies the length of the largest possible segment for the file. DUAMRGDR uses this value to reserve adequate main storage for buffers. The value must be greater than zero and less than 32,768. If you do not code this keyword, DUAMRGDR uses a value of 256.

MINPFREE=m where n is a decimal number, specifies the minimum amount of free space that DUAMRGDR will leave in each data block. TCDMS can use this free space during normal processing of the file to expand data families in the block. The value you specify for MINPFREE must be equal to or greater than zero and less than the value of MAXPFREE minus the file's key length minus 20. If you do not code the MINPFREE keyword, DUAMRGDR uses a value equal to five percent of the block size.

Performance hint: You can use MAXPFREE and MINPFREE to limit the number of data families which are split across two physical blocks. If the typical data family size is considerably less than block size, you may set MAXPFREE to a value somewhat larger than MINPFREE plus typical data family size. This will keep DUAMRGDR from splitting a data family of typical size across two blocks

OVTRACKS=n where n is a decimal number, specifies the number of overflow tracks that DUAMRGDR will reserve for later expansion of existing families. TCDMS uses this area during normal processing to expand existing data families when there is insufficient free space in the blocks where the families were originally loaded or created. You must code a value for this keyword. It must be equal to or greater than zero.

PRTRACKS=n where n is a decimal number, specifies the number of tracks that DUAMRGDR will reserve for the later addition of new data families to the file. You must code a value for this keyword. It must be equal to or greater than zero.

DDNAME DLDn

Ddname DLDn defines a data set which will become the reorganized TCDMS file n where n is the five-digit decimal file number of the file being reorganized. You may specify the existing file n in this DD statement, or you may specify a different data set. In the latter case, you may allocate space for this file before running DUAMRGDR, or you may allocate it in the same job step by coding 'NEW' in the DISP parameter of the DD statement. The SPACE parameter must include a primary allocation greater than the number of tracks specified in the INTRACKS keyword of the control data set for the file. You may specify secondary allocations; DUAMRGDR will force secondary allocations until the space requirements specified in the PRTRACKS and OVTRACKS keywords are satisfied.

DDNAME DMSCATLG

Ddname DMSCATLG defines the TCDMS catalog for the data base. You must specify the DISP parameter as SHR.

DDNAME DMSCHAIN

Ddname DMSCHAIN defines a temporary sequential data set. It is used to collect chain segments to be resolved in the file being reorganized. Its format is fixed blocked. Each logical record is 22 bytes long. You must specify the block size in the DCB parameter. It must be a multiple of 22 and should be an appropriate block size for the I/O device. You should allocate enough space to contain one record for each chain segment on the file being reorganized. You may specify a secondary allocation.

DDNAME DMSOUT

Ddname DMSOUT defines a temporary sequential data set. DUAMRGDR uses it to collect segments to be put on related files. Its format is variable blocked. You must specify logical record length and block size in the DCB parameter. Logical record length may be any value less than the block size of this data set and greater than the maximum segment length plus 206. Block size should be appropriate for the I/O device. You should allocate enough space to contain one record for each chain segment in the file being reorganized. You may specify secondary allocations in the SPACE parameter.

DDNAME DMSPRINT

Ddname DMSPRINT defines a sequential message data set. You do not need to code DCB parameters for it.

DDNAME JOBLIB or STEPLIB

Ddname JOBLIB or STEPLIB must be included unless DUAMRGDR and all the modules it links to are on the system library. These include DUAMCHPS, DUAMCHSP, DUAMLOAD, DUAMSGWO and SORT. SORT must be an alias for a sort program which accepts input in the standard form used by most IBM-developed sorts.

DDNAMES FOR SORT

Ddnames SORTLIB, SORTWK01, SORTWK02, SORTWK03, SORTWK04, SORTWK05, SORTWK06, and SYSOUT define data sets required by SORT. For information on how to specify them, consult the documentation of the SORT program you use using. You should allocate adequate work space (SORTWK01, etc.) to sort a data set the size of the DMSOUT data set.

2.2.2 DUAMRGDR EXAMPLE

In this example, a TCDMS file is reorganized without moving it from its original location:

```
//REORG35 JOB
// EXEC PGM=DUAMRGDR,PARM='FILE=35'
//STEPLIB DD DSN=TCDMS.LOADLIB,DISP=SHR
//DMSCATLG DD DSN=DMS.CATALOG,DISP=SHR
//DMSWO DD UNIT=TAPE,VOL=SER=001756,
//       DCB=(RECFM=VB,LRECL=500,BLKSIZE=7200)
//DMSCHAIN DD UNIT=SYSDA,SPACE=(2200,(1000,100)),
//       DCB=BLKSIZE=2200
//DMSOUT DD UNIT=SYSDA,SPACE=(2400,(4000,500)),
//       DCB=(LRECL=500,BLKSIZE=2400)
//DMSPRINT DD SYSOUT=A
//DLD00035 DD UNIT=3330,VOL=SER=333001,DSN=DMS.ACCT1,DISP=OLD
//DCT00035 DD *
INTRACKS=30
PRTRACKS=80
OVTRACKS=40
INDXFREE=300
MINPFREE=75
MAXPFREE=200
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD UNIT=SYSDA,SPACE=(TRK,100)
//SORTWK02 DD UNIT=SYSDA,SPACE=(TRK,100)
//SORTWK03 DD UNIT=SYSDA,SPACE=(TRK,100)
//SORTWK04 DD UNIT=SYSDA,SPACE=(TRK,100)
//SORTWK05 DD UNIT=SYSDA,SPACE=(TRK,100)
//SORTWK06 DD UNIT=SYSDA,SPACE=(TRK,100)
//SYSOUT DD SYSOUT=A
```

EXPLANATION:

The EXEC statement specifies the execution of DUAMRGDR. The PARM field indicates that file 35 is to be reorganized.

The STEPLIB DD statement specifies the load library containing DUAMRGDR and its submodules.

The DMSCATLG DD statement specifies the installation's DMS catalog data set.

The DMSWO DD statement specifies the work data set which will contain the file in unloaded format.

The DMSCHAIN DD statement specifies a work data set for pointer segments.

The DMSOUT DD statement specifies a work data set for secondary data segments and pointer segments.

The DMSPRINT DD statement specifies a message data set.

The DLD00035 DD statement specifies the location of the reorganized data set. In this case the data set will remain in its old location, so this DD statement describes the existing file.

The DCT00035 DD statement defines the control data set for file 35. The control statement specify certain physical characteristics of the file.

INTRACKS=25   specifies that 25 tracks are to be reserved for index area; any unused portion will be available for expansion.

PRTRACKS=100   specifies that 100 tracks are to be reserved for expansion of prime data with new data families after the load.

OVTRACKS=40   specifies that 40 tracks are to be reserved for an overflow area. Overflow is used for the expansion of existing families.

INDXFREE=300 specifies that 300 bytes are to be left for expansion in each index block.

MINPFREE=75 and MAXPFREE=200 specifies that between 75 and 200 bytes are to be left in each data block for expansion.

SORTLIB DD, SORTLIBnn DD, and SYSOUT DD define the data sets required by the SORT program, which is invoked by DUAMRGDR.

2.3 DUFRCDR - TCDMS DATA BASE BACKUP AND RECOVERY UTILITY

DUFRCDR is a TCDMS utility used to back up a data base and to restore it in the event of unrecoverable problems. The write-off function may be used to create back up copies of a file or group of logically related files. Then, in the event that hardware or software problems make the files unusable, the reload and catch up functions may be used to restore the files to their condition as of a given point in time. The reload function recreates the files from the back up copy. Then the catch up function uses the TCDMS capture file to reapply to the files the transactions which occurred between the time of the write off and the time at which the files became unusable. DUFRCDR runs as a batch job step independent of TCDMS.

Input to DUFRCDR consists of operating system job control language statements and a utility control statement data set. Output consists of a message data set, which includes statistical information about each of the files involved and error messages where appropriate.

2.3.1 CONTROL STATEMENTS

A description of the required job control and utility control statements follows.

DUFRCDR is invoked with an operating system EXEC statement. The PARM parameter indicates the function to be performed:

WO	for write-off
RL	for reload
CT	for catch-up
RLCT	for reload and catch-up

DD STATEMENTS

DUFRCDR requires several operating system DD statements. They are listed below by ddname.

DDNAME DMSCATLG

Ddname DMSCATLG defines the TCDMS catalog for the data base. You must specify the DISP parameter as SHR.

DDNAME DMSIN

Ddname DMSIN defines a sequential control statement data set. You use it to specify the files involved in the operation, and, in the case that a catch-up is requested, to specify the ddnames of the TCDMS capture files containing the transactions to be reapplied to the files.

This data set consists of 80 byte records. Each record may contain one or more parameters, which must begin in column one and be separated by commas. DUFRCDR ignores any data after the first blank in a record, so you may include comments if you wish. You may have any number of control statement records. You may specify the files involved in the operations by listing their numbers as parameters. To specify all the files in the data base, you may either code 'ALL' as a parameter, or simply code no file numbers as parameters.

The integrity of the data base is preserved through a write-off and reload operation only if all related files are written off and reloaded at the same time. All files related through intermediate files must also be included.

If you specify the catch-up or reload and catch-up operation, you must code as parameters the ddnames of DD statements defining the TCDMS capture files containing the transactions to be reapplied. The ddname parameters must follow the parameters specifying the files involved in the operation.

DDNAME DMSPRINT

Ddname DMSPRINT defines a sequential message data set. You do not need to code DCB parameters for it.

DDNAME DMSRL

Ddname DMSRL is required only for the reload and reload with catch-up operations, It defines a data set created by DUFRCDR in a previous write off operation. This data set contains the back-up copies of TCDMS files. See the description of DMSWO below.

DDNAME DMSWO

Ddname DMSWO is required only for the write-off operation. It defines a sequential data set to which DUFRCDR will output the back-up copies of the TCDMS files specified in the control statements. The data set has fixed blocked format. You may specify logical record length and block size if you wish. If you do not, DUFRCDR uses a logical record length equal to the block size of the TCDMS files, and a block size equal to the logical record length times the number of blocks per track of the TCDMS files.

DDNAME JOBLIB or STEPLIB

Ddname JOBLIB or STEPLIB must be coded unless your installation's system load library includes DUFRCDR.

2.3.2 DUFRCDR EXAMPLES

DUFRCDR Example 1

In this example, back-up copies of several TCDMS files are written to tape:

```
//WOFILES JOB
//WO EXEC PGM=DUFRCDR,PARM=WO
//STEPLIB DD DSN=TCDMS.LOADLIB,DISP=SHR
//DMSCATLG DD DSN=DMS.CATALOG,DISP=SHR
//DMSPRINT DD SYSOUT=A
//DMSWO DD UNIT=TAPE,DISP=(NEW,KEEP),VOL=SER=001351,
//             DSN=WO.THURS
//DMSIN DD *
17,18,19,20         WRITE OFF
21,35                ALL RELATED FILES
//
```

EXPLANATION:

The EXEC statement specifies the execution of DUFRCDR. The PARM field specifies the write-off function.

The STEPLIB DD statement specifies the load library containing DUFRCDR.

The DMSCATLG DD statement specifies the installation's DMS catalog data set.

The DMSPRINT DD statement specifies a message data set.

The DMSWO DD statement specifies the output tape which is to contain the back-up copies of the data sets.

The DMSIN DD statement specifies the control data set. The control statements in it specify that files 17, 18, 19, 20, 21, and 35 are to be written to tape.

DUFRCDR Example 2

In this example, several TCDMS files a reloaded from tape and caught up using transactions from a capture file.

```
//RLCT JOB
// EXEC PGM=DUFRCDR,PARM=RLCT
//STEPLIB DD DSN=TCDMS.LOADLIB,DISP=SHR
//DMSCATLG DD DSN=DMS.CATALOG,DISP=SHR
//DMSPRINT DD SYSOUT=A
//DMSRL DD UNIT=TAPE,DISP=OLD,VOL=SER=001351,
//             DSN=WO.THURS
//CAPTAPE DD UNIT=TAPE,DISP=OLD,VOL=SER=010751,
//             DSN=TCDMS.CAPTURE
//DMSIN DD *
20,21,35,CAPTAPE
//
```

EXPLANATION:

The DMSRL DD statement describes the input tape containing the data sets (in unloaded format) to be reloaded and caught up. Note that it might contain data sets not involved in the operation.

The DMSIN DD statement describes the control data set. The control statement specifies that files 20, 21, and 35 are to be reloaded and caught up, and that the CAPTAPE DD statement describes the TCDMS capture file containing the transactions to be reapplied.